# PIC17C7XX

## High-Performance 8-bit CMOS EPROM Microcontrollers with 10-bit A/D

### Microcontroller Core Features:

- Only 58 single word instructions to learn
- All single cycle instructions (121 ns), except for program branches and table reads/writes which are two-cycle
- Operating speed:
  - DC - 33 MHz clock input
  - DC - 121 ns instruction cycle
- 8 x 8 Single-Cycle Hardware Multiplier
- Interrupt capability
- 16 level deep hardware stack
- Direct, indirect, and relative addressing modes
- Internal/external program memory execution, capable of addressing 64 K x 16 program memory space

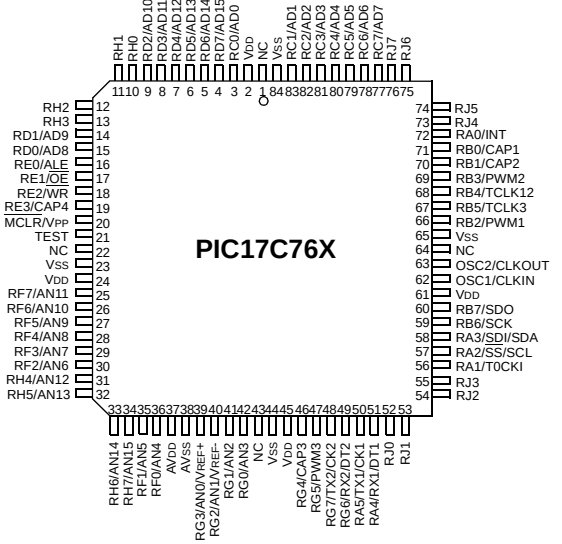
| Device | Memory | |
|---|---|---|
| | Program (x16) | Data (x8) |
| PIC17C752 | 8 K | 678 |
| PIC17C756A | 16 K | 902 |
| PIC17C762 | 8 K | 678 |
| PIC17C766 | 16 K | 902 |

### Peripheral Features:

- Up to 66 I/O pins with individual direction control
- 10-bit, multi-channel Analog-to-Digital converter
- High current sink/source for direct LED drive
- Four capture input pins
  - Captures are 16-bit, max resolution 121 ns
- Three PWM outputs (resolution is 1 to 10-bits)
- TMR0: 16-bit timer/counter with 8-bit programmable prescaler
- TMR1: 8-bit timer/counter
- TMR2: 8-bit timer/counter
- TMR3: 16-bit timer/counter
- Two Universal Synchronous Asynchronous Receiver Transmitters (USART/SCI) with independent baud rate generators
- Synchronous Serial Port (SSP) with SPI™ and $I^2C$™ modes (including $I^2C$ Master mode)

### Pin Diagrams

**84 PLCC**




### Special Microcontroller Features:

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

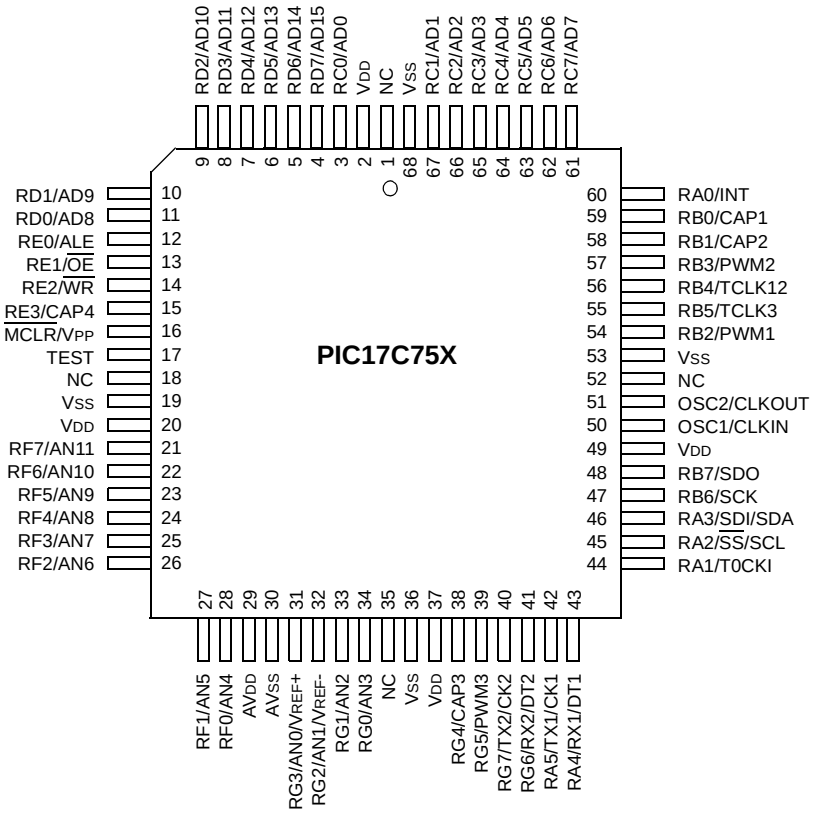
### CMOS Technology:

- Low power, high speed CMOS EPROM technology
- Fully static design
- Wide operating voltage range (3.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low power consumption
  - < 5 mA @ 5V, 4 MHz
  - 100 µA typical @ 4.5V, 32 kHz
  - < 1 µA typical standby current @ 5V

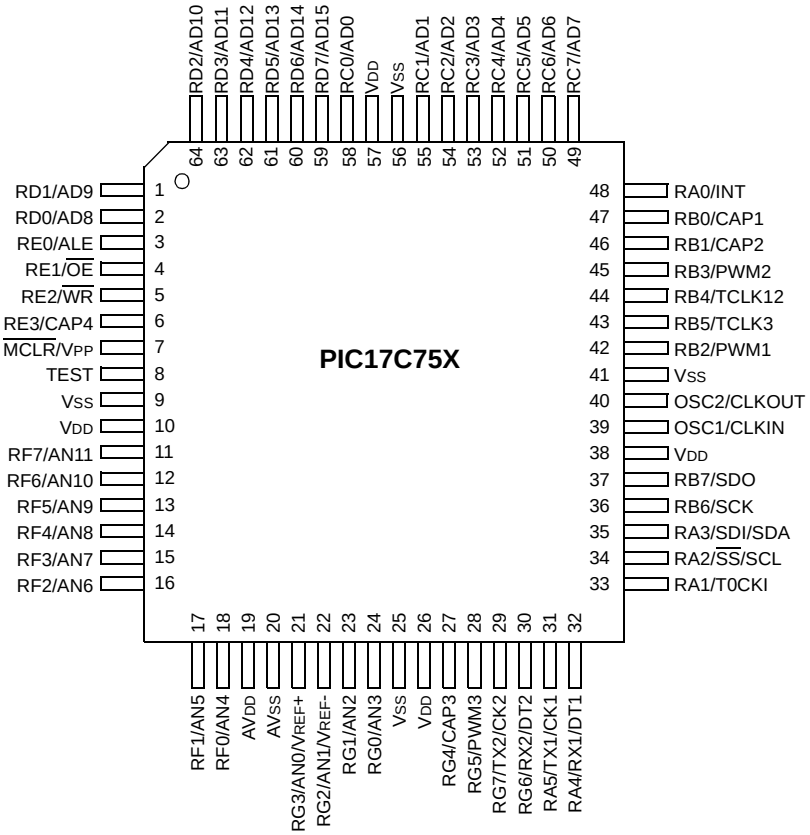
© 2000 Microchip Technology Inc.

## Pin Diagrams cont.'d

### 68-Pin PLCC




### 64-Pin TQFP

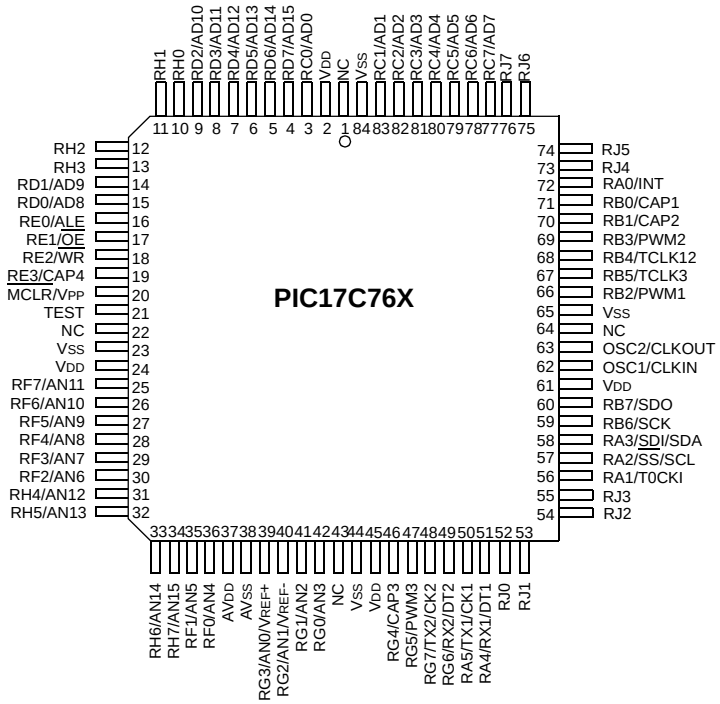




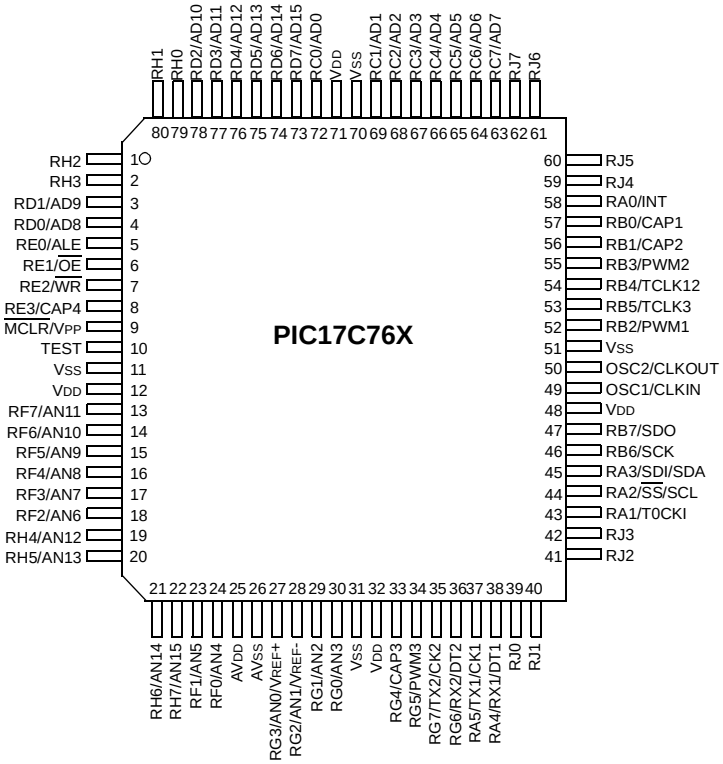
© 2000 Microchip Technology Inc.

## Pin Diagrams cont.'d

### 84-pin PLCC




### 80-Pin TQFP



## Table of Contents




© 2000 Microchip Technology Inc.

## TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@mail.microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

### Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

### Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

### Customer Notification System

Register on our web site at **www.microchip.com/cn** to receive the most current information on all of our products.

**NOTES:**


© 2000 Microchip Technology Inc.

## 1.0 OVERVIEW

This data sheet covers the PIC17C7XX group of the PIC17CXXX family of microcontrollers. The following devices are discussed in this data sheet:

- PIC17C752
- PIC17C756A
- PIC17C762
- PIC17C766

The PIC17C7XX devices are 68/84-pin, EPROM based members of the versatile PIC17CXXX family of low cost, high performance, CMOS, fully static, 8-bit microcontrollers.

All PICmicro® microcontrollers employ an advanced RISC architecture. The PIC17CXXX has enhanced core features, 16-level deep stack, and multiple internal and external interrupt sources. The separate instruction and data buses of the Harvard architecture allow a 16-bit wide instruction word with a separate 8-bit wide data path. The two stage instruction pipeline allows all instructions to execute in a single cycle, except for program branches (which require two cycles). A total of 58 instructions (reduced instruction set) are available. Additionally, a large register set gives some of the architectural innovations used to achieve a very high performance. For mathematical intensive applications, all devices have a single cycle 8 x 8 Hardware Multiplier.

PIC17CXXX microcontrollers typically achieve a 2:1 code compression and a 4:1 speed improvement over other 8-bit microcontrollers in their class.

PIC17C7XX devices have up to 902 bytes of RAM and 66 I/O pins. In addition, the PIC17C7XX adds several peripheral features, useful in many high performance applications, including:

- Four timer/counters
- Four capture inputs
- Three PWM outputs
- Two independent Universal Synchronous Asynchronous Receiver Transmitters (USARTs)
- An A/D converter (multi-channel, 10-bit resolution)
- A Synchronous Serial Port (SPI and $I^2C$ w/ Master mode)

These special features reduce external components, thus reducing cost, enhancing system reliability and reducing power consumption.

There are four oscillator options, of which the single pin RC oscillator provides a low cost solution, the LF oscillator is for low frequency crystals and minimizes power consumption, XT is a standard crystal and the EC is for external clock input.

The SLEEP (power-down) mode offers additional power saving. Wake-up from SLEEP can occur through several external and internal interrupts and device RESETS.

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

There are four configuration options for the device operational mode:

- Microprocessor
- Microcontroller
- Extended microcontroller
- Protected microcontroller

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

The device also has Brown-out Reset circuitry. This allows a device RESET to occur if the device $V_{DD}$ falls below the Brown-out voltage trip point ($BV_{DD}$). The chip will remain in Brown-out Reset until $V_{DD}$ rises above $BV_{DD}$.

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

The PIC17C7XX fits perfectly in applications that require extremely fast execution of complex software programs. These include applications ranging from precise motor control and industrial process control to automotive, instrumentation, and telecom applications.

The EPROM technology makes customization of application programs (with unique security codes, combinations, model numbers, parameter storage, etc.) fast and convenient. Small footprint package options (including die sales) make the PIC17C7XX ideal for applications with space limitations that require high performance.

High speed execution, powerful peripheral features, flexible I/O, and low power consumption all at low cost make the PIC17C7XX ideal for a wide range of embedded control applications.

### 1.1 Family and Upward Compatibility

The PIC17CXXX family of microcontrollers have architectural enhancements over the PIC16C5X and PIC16CXX families. These enhancements allow the device to be more efficient in software and hardware requirements. Refer to Appendix A for a detailed list of enhancements and modifications. Code written for PIC16C5X or PIC16CXX can be easily ported to PIC17CXXX devices (Appendix B).

### 1.2 Development Support

The PIC17CXXX family is supported by a full featured macro assembler, a software simulator, an in-circuit emulator, a universal programmer, a “C” compiler and fuzzy logic support tools. For additional information, see Section 19.0.

**TABLE 1-1: PIC17CXXX FAMILY OF DEVICES**

| Features | | PIC17C42A | PIC17C43 | PIC17C44 | PIC17C752 | PIC17C756A | PIC17C762 | PIC17C766 |
|---|---|---|---|---|---|---|---|---|
| Maximum Frequency of Operation | | 33 MHz | 33 MHz | 33 MHz | 33 MHz | 33 MHz | 33 MHz | 33 MHz |
| Operating Voltage Range | | 2.5 - 6.0V | 2.5 - 6.0V | 2.5 - 6.0V | 3.0 - 5.5V | 3.0 - 5.5V | 3.0 - 5.5V | 3.0 - 5.5V |
| Program Memory ( x16) | (EPROM) | 2 K | 4 K | 8 K | 8 K | 16 K | 8 K | 16 K |
| | (ROM) | — | — | — | — | — | — | — |
| Data Memory (bytes) | | 232 | 454 | 454 | 678 | 902 | 678 | 902 |
| Hardware Multiplier (8 x 8) | | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Timer0 (16-bit + 8-bit postscaler) | | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Timer1 (8-bit) | | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Timer2 (8-bit) | | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Timer3 (16-bit) | | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Capture inputs (16-bit) | | 2 | 2 | 2 | 4 | 4 | 4 | 4 |
| PWM outputs (up to 10-bit) | | 2 | 2 | 2 | 3 | 3 | 3 | 3 |
| USART/SCI | | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
| A/D channels (10-bit) | | — | — | — | 12 | 12 | 16 | 16 |
| SSP (SPI/$I^2C$ w/Master mode) | | — | — | — | Yes | Yes | Yes | Yes |
| Power-on Reset | | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Watchdog Timer | | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| External Interrupts | | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Interrupt Sources | | 11 | 11 | 11 | 18 | 18 | 18 | 18 |
| Code Protect | | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| Brown-out Reset | | — | — | — | Yes | Yes | Yes | Yes |
| In-Circuit Serial Programming | | — | — | — | Yes | Yes | Yes | Yes |
| I/O Pins | | 33 | 33 | 33 | 50 | 50 | 66 | 66 |
| I/O High Current Capability | Source | 25 mA | 25 mA | 25 mA | 25 mA | 25 mA | 25 mA | 25 mA |
| | Sink | 25 mA(1) | 25 mA(1) | 25 mA(1) | 25 mA(1) | 25 mA(1) | 25 mA(1) | 25 mA(1) |
| Package Types | | 40-pin DIP<br>44-pin PLCC<br>44-pin MQFP<br>44-pin TQFP | 40-pin DIP<br>44-pin PLCC<br>44-pin MQFP<br>44-pin TQFP | 40-pin DIP<br>44-pin PLCC<br>44-pin MQFP<br>44-pin TQFP | 64-pin TQFP<br>68-pin PLCC | 64-pin TQFP<br>68-pin PLCC | 80-pin TQFP<br>84-pin PLCC | 80-pin TQFP<br>84-pin PLCC |

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


© 2000 Microchip Technology Inc.

## 2.0 DEVICE VARIETIES

Each device has a variety of frequency ranges and packaging options. Depending on application and production requirements, the proper device option can be selected using the information in the PIC17C7XX Product Selection System section at the end of this data sheet. When placing orders, please use the "PIC17C7XX Product Identification System" at the back of this data sheet to specify the correct part number. When discussing the functionality of the device, memory technology and voltage range does not matter.

There are two memory type options. These are specified in the middle characters of the part number.

1. **C**, as in PIC17**C**756A. These devices have EPROM type memory.
2. **CR**, as in PIC17**CR**756A. These devices have ROM type memory.

All these devices operate over the standard voltage range. Devices are also offered which operate over an extended voltage range (and reduced frequency range). Table 2-1 shows all possible memory types and voltage range designators for a particular device. These designators are in **bold** typeface.

**TABLE 2-1: DEVICE MEMORY VARIETIES**

| Memory Type | Voltage Range | |
|---|---|---|
| | **Standard** | **Extended** |
| EPROM | PIC17**C**XXX | PIC17**LC**XXX |
| ROM | PIC17**CR**XXX | PIC17**LCR**XXX |

**Note:** Not all memory technologies are available for a particular device.

### 2.1 UV Erasable Devices

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

The UV erasable version can be erased and reprogrammed to any of the configuration modes. Third party programmers also are available; refer to the *Third Party Guide* for a list of sources.

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

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

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

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

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

### 2.4 Serialized Quick-Turnaround Production (SQTP$^{sm}$) 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.

### 2.5 Read Only Memory (ROM) Devices

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

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

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

**Note:** Presently, NO ROM versions of the PIC17C7XX devices are available.

**NOTES:**


© 2000 Microchip Technology Inc.

## 3.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC17CXXX can be attributed to a number of architectural features, commonly found in RISC microprocessors. To begin with, the PIC17CXXX uses a modified Harvard architecture. This architecture has the program and data accessed from separate memories. So, the device has a program memory bus and a data memory bus. This improves bandwidth over traditional von Neumann architecture, where program and data are fetched from the same memory (accesses over the same bus). Separating program and data memory further allows instructions to be sized differently than the 8-bit wide data word. PIC17CXXX opcodes are 16-bits wide, enabling single word instructions. The full 16-bit wide program memory bus fetches a 16-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions. Consequently, all instructions execute in a single cycle (121 ns @ 33 MHz), except for program branches and two special instructions that transfer data between program and data memory.

The PIC17CXXX can address up to 64K x 16 of program memory space.

The **PIC17C752** and **PIC17C762** integrate 8K x 16 of EPROM program memory on-chip.

The **PIC17C756A** and **PIC17C766** integrate 16K x 16 EPROM program memory on-chip.

A simplified block diagram is shown in Figure 3-1. The descriptions of the device pins are listed in Table 3-1.

Program execution can be internal only (Microcontroller or Protected Microcontroller mode), external only (Microprocessor mode), or both (Extended Microcontroller mode). Extended Microcontroller mode does not allow code protection.

The PIC17CXXX can directly or indirectly address its register files or data memory. All special function registers, including the Program Counter (PC) and Working Register (WREG), are mapped in data memory. The PIC17CXXX has an orthogonal (symmetrical) instruction set that makes it possible to carry out any operation on any register using any addressing mode. This symmetrical nature and lack of 'special optimal situations' make programming with the PIC17CXXX simple, yet efficient. In addition, the learning curve is reduced significantly.

One of the PIC17CXXX family architectural enhancements from the PIC16CXX family, allows two file registers to be used in some two operand instructions. This allows data to be moved directly between two registers without going through the WREG register, thus increasing performance and decreasing program memory usage.

The PIC17CXXX devices contain an 8-bit ALU and working register. The ALU is a general purpose arithmetic unit. It performs arithmetic and Boolean functions between data in the working register and any register file.

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

All PIC17CXXX devices have an 8 x 8 hardware multiplier. This multiplier generates a 16-bit result in a single cycle.

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.

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

Signed arithmetic is comprised of a magnitude and a sign bit. The overflow bit indicates if the magnitude overflows and causes the sign bit to change state. That is, if the result of 8-bit signed operations is greater than 127 (7Fh), or less than -128 (80h).

Signed math can have greater than 7-bit values (magnitude), if more than one byte is used. The overflow bit only operates on bit6 (MSb of magnitude) and bit7 (sign bit) of each byte value in the ALU. That is, the overflow bit is not useful if trying to implement signed math where the magnitude, for example, is 11-bits.

If the signed math values are greater than 7-bits (such as 15-, 24-, or 31-bit), the algorithm must ensure that the low order bytes of the signed value ignore the overflow status bit.

Example 3-1 shows two cases of doing signed arithmetic. The Carry (C) bit and the Overflow (OV) bit are the most important status bits for signed math operations.

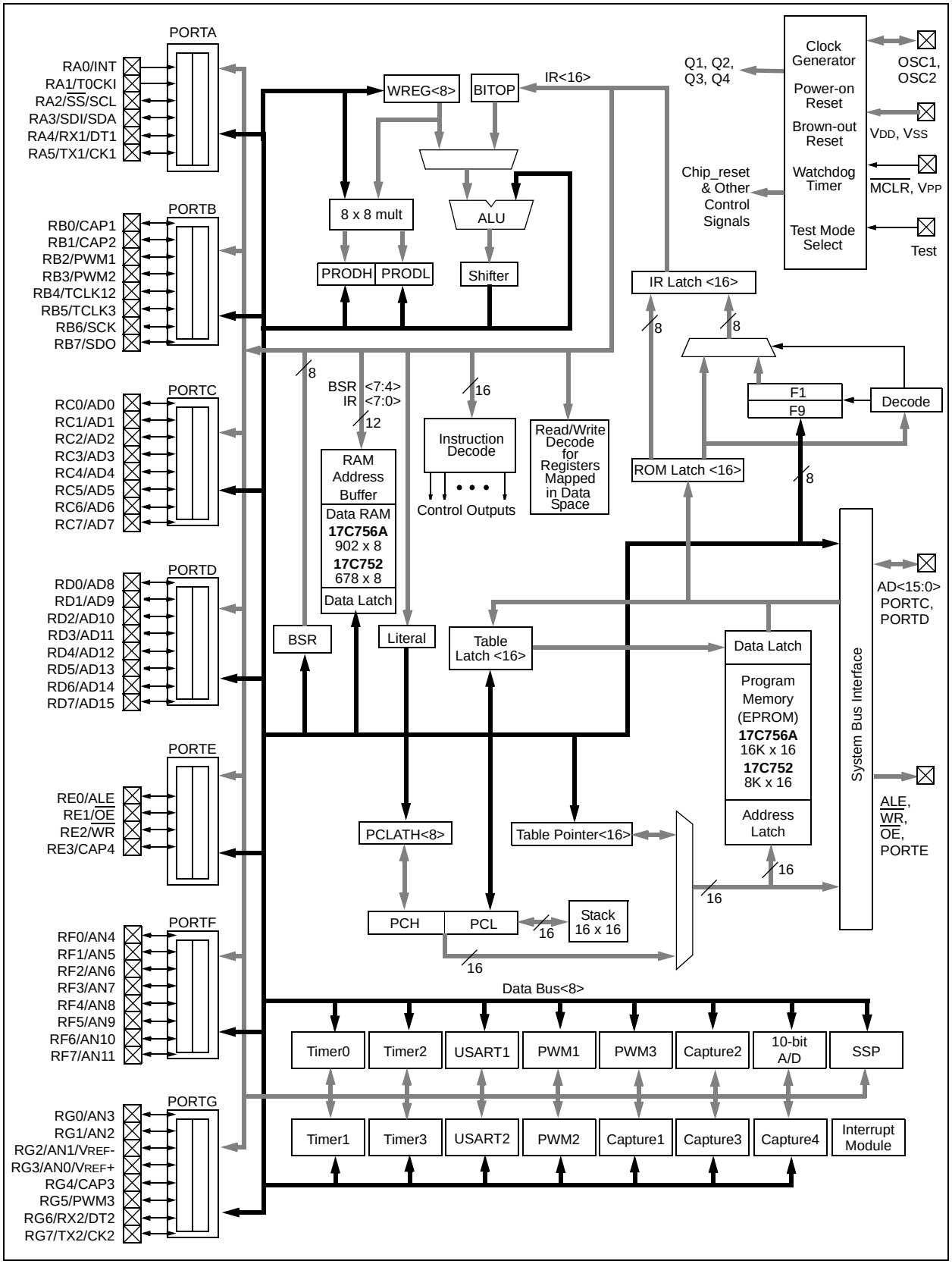
**EXAMPLE 3-1: 8-BIT MATH ADDITION**

| Hex Value | Signed Values | Unsigned Values |
|---|---|---|
| FFh | -1 | 255 |
| + 01h | + 1 | + 1 |
| = 00h | = 0 (FEh) | = 256 → 00h |
| C bit = 1 | C bit = 1 | C bit = 1 |
| OV bit = 0 | OV bit = 0 | OV bit = 0 |
| DC bit = 1 | DC bit = 1 | DC bit = 1 |
| Z bit = 1 | Z bit = 1 | Z bit = 1 |

| Hex Value | Signed Values | Unsigned Values |
|---|---|---|
| 7Fh | 127 | 127 |
| + 01h | + 1 | + 1 |
| = 80h | = 128 → 00h | = 128 |
| C bit = 0 | C bit = 0 | C bit = 0 |
| OV bit = 1 | OV bit = 1 | OV bit = 1 |
| DC bit = 1 | DC bit = 1 | DC bit = 1 |
| Z bit = 0 | Z bit = 0 | Z bit = 0 |

© 2000 Microchip Technology Inc.

**FIGURE 3-1: PIC17C752/756A BLOCK DIAGRAM**





© 2000 Microchip Technology Inc.

**FIGURE 3-2: PIC17C762/766 BLOCK DIAGRAM**

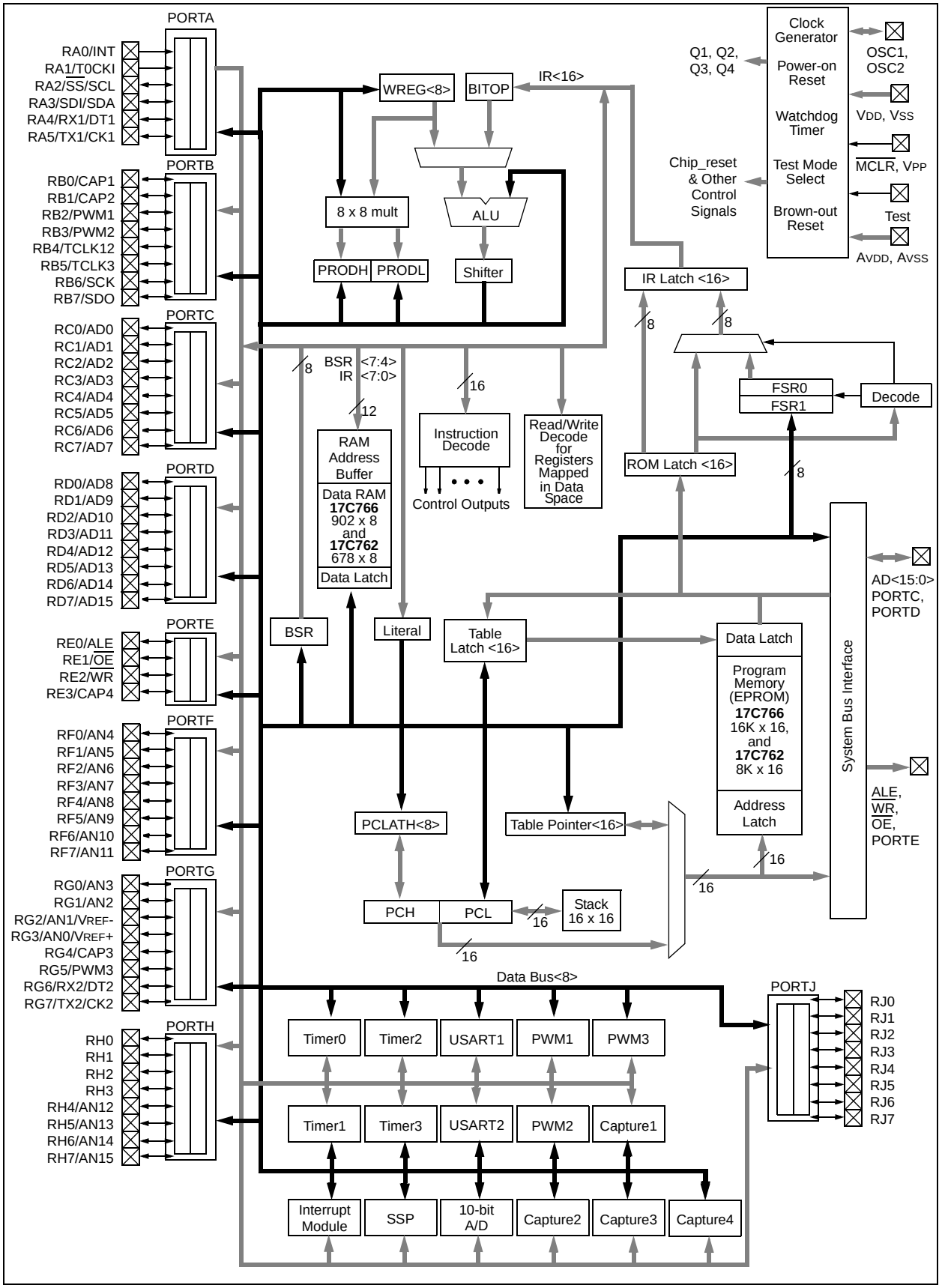


**TABLE 3-1: PINOUT DESCRIPTIONS**

| Name | PIC17C75X | | | PIC17C76X | | I/O/P Type | Buffer Type | Description |
|---|---|---|---|---|---|---|---|---|
| | DIP No. | PLCC No. | TQFP No. | PLCC No. | QFP No. | | | |
| OSC1/CLKIN | 47 | 50 | 39 | 62 | 49 | I | ST | Oscillator input in Crystal/Resonator or RC Oscillator mode. External clock input in External Clock mode. |
| OSC2/CLKOUT | 48 | 51 | 40 | 63 | 50 | O | — | Oscillator output. Connects to crystal or resonator in Crystal Oscillator mode. In RC Oscillator or External Clock modes, OSC2 pin outputs CLKOUT which has one fourth the frequency (FOSC/4) of OSC1 and denotes the instruction cycle rate. |
| $\overline{MCLR}$/VPP | 15 | 16 | 7 | 20 | 9 | I/P | ST | Master clear (RESET) input or Programming Voltage (VPP) input. This is the active low RESET input to the device. |
| | | | | | | | | PORTA pins have individual differentiations that are listed in the following descriptions: |
| RA0/INT | 56 | 60 | 48 | 72 | 58 | I | ST | RA0 can also be selected as an external interrupt input. Interrupt can be configured to be on positive or negative edge. Input only pin. |
| RA1/T0CKI | 41 | 44 | 33 | 56 | 43 | I | ST | RA1 can also be selected as an external interrupt input and the interrupt can be configured to be on positive or negative edge. RA1 can also be selected to be the clock input to the Timer0 timer/counter. Input only pin. |
| RA2/$\overline{SS}$/SCL | 42 | 45 | 34 | 57 | 44 | I/O(2) | ST | RA2 can also be used as the slave select input for the SPI or the clock input for the $I^2C$ bus. High voltage, high current, open drain port pin. |
| RA3/SDI/SDA | 43 | 46 | 35 | 58 | 45 | I/O(2) | ST | RA3 can also be used as the data input for the SPI or the data for the $I^2C$ bus. High voltage, high current, open drain port pin. |
| RA4/RX1/DT1 | 40 | 43 | 32 | 51 | 38 | I/O(1) | ST | RA4 can also be selected as the USART1 (SCI) Asynchronous Receive or USART1 (SCI) Synchronous Data. Output available from USART only. |
| RA5/TX1/CK1 | 39 | 42 | 31 | 50 | 37 | I/O(1) | ST | RA5 can also be selected as the USART1 (SCI) Asynchronous Transmit or USART1 (SCI) Synchronous Clock. Output available from USART only. |
| | | | | | | | | PORTB is a bi-directional I/O Port with software configurable weak pull-ups. |
| RB0/CAP1 | 55 | 59 | 47 | 71 | 57 | I/O | ST | RB0 can also be the Capture1 input pin. |
| RB1/CAP2 | 54 | 58 | 46 | 70 | 56 | I/O | ST | RB1 can also be the Capture2 input pin. |
| RB2/PWM1 | 50 | 54 | 42 | 66 | 52 | I/O | ST | RB2 can also be the PWM1 output pin. |
| RB3/PWM2 | 53 | 57 | 45 | 69 | 55 | I/O | ST | RB3 can also be the PWM2 output pin. |
| RB4/TCLK12 | 52 | 56 | 44 | 68 | 54 | I/O | ST | RB4 can also be the external clock input to Timer1 and Timer2. |
| RB5/TCLK3 | 51 | 55 | 43 | 67 | 53 | I/O | ST | RB5 can also be the external clock input to Timer3. |
| RB6/SCK | 44 | 47 | 36 | 59 | 46 | I/O | ST | RB6 can also be used as the master/slave clock for the SPI. |
| RB7/SDO | 45 | 48 | 37 | 60 | 47 | I/O | ST | RB7 can also be used as the data output for the SPI. |

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

**Note 1:** The output is only available by the peripheral operation.
**2:** Open drain input/output pin. Pin forced to input upon any device RESET.


© 2000 Microchip Technology Inc.

**TABLE 3-1: PINOUT DESCRIPTIONS (CONTINUED)**

| Name | PIC17C75X | | | PIC17C76X | | I/O/P Type | Buffer Type | Description |
|---|---|---|---|---|---|---|---|---|
| | DIP No. | PLCC No. | TQFP No. | PLCC No. | QFP No. | | | |
| | | | | | | | | PORTC is a bi-directional I/O Port. |
| RC0/AD0 | 2 | 3 | 58 | 3 | 72 | I/O | TTL | This is also the least significant byte (LSB) of the 16-bit wide system bus in Microprocessor mode or Extended Microcontroller mode. In multiplexed system bus configuration, these pins are address output as well as data input or output. |
| RC1/AD1 | 63 | 67 | 55 | 83 | 69 | I/O | TTL | |
| RC2/AD2 | 62 | 66 | 54 | 82 | 68 | I/O | TTL | |
| RC3/AD3 | 61 | 65 | 53 | 81 | 67 | I/O | TTL | |
| RC4/AD4 | 60 | 64 | 52 | 80 | 66 | I/O | TTL | |
| RC5/AD5 | 58 | 63 | 51 | 79 | 65 | I/O | TTL | |
| RC6/AD6 | 58 | 62 | 50 | 78 | 64 | I/O | TTL | |
| RC7/AD7 | 57 | 61 | 49 | 77 | 63 | I/O | TTL | |
| | | | | | | | | PORTD is a bi-directional I/O Port. |
| RD0/AD8 | 10 | 11 | 2 | 15 | 4 | I/O | TTL | This is also the most significant byte (MSB) of the 16-bit system bus in Microprocessor mode or Extended Microcontroller mode. In multiplexed system bus configuration, these pins are address output as well as data input or output. |
| RD1/AD9 | 9 | 10 | 1 | 14 | 3 | I/O | TTL | |
| RD2/AD10 | 8 | 9 | 64 | 9 | 78 | I/O | TTL | |
| RD3/AD11 | 7 | 8 | 63 | 8 | 77 | I/O | TTL | |
| RD4/AD12 | 6 | 7 | 62 | 7 | 76 | I/O | TTL | |
| RD5/AD13 | 5 | 6 | 61 | 6 | 75 | I/O | TTL | |
| RD6/AD14 | 4 | 5 | 60 | 5 | 74 | I/O | TTL | |
| RD7/AD15 | 3 | 4 | 59 | 4 | 73 | I/O | TTL | |
| | | | | | | | | PORTE is a bi-directional I/O Port. |
| RE0/ALE | 11 | 12 | 3 | 16 | 5 | I/O | TTL | In Microprocessor mode or Extended Microcontroller mode, RE0 is the Address Latch Enable (ALE) output. Address should be latched on the falling edge of ALE output. |
| RE1/$\overline{OE}$ | 12 | 13 | 4 | 17 | 6 | I/O | TTL | In Microprocessor or Extended Microcontroller mode, RE1 is the Output Enable ($\overline{OE}$) control output (active low). |
| RE2/$\overline{WR}$ | 13 | 14 | 5 | 18 | 7 | I/O | TTL | In Microprocessor or Extended Microcontroller mode, RE2 is the Write Enable ($\overline{WR}$) control output (active low). |
| RE3/CAP4 | 14 | 15 | 6 | 19 | 8 | I/O | ST | RE3 can also be the Capture4 input pin. |
| | | | | | | | | PORTF is a bi-directional I/O Port. |
| RF0/AN4 | 26 | 28 | 18 | 36 | 24 | I/O | ST | RF0 can also be analog input 4. |
| RF1/AN5 | 25 | 27 | 17 | 35 | 23 | I/O | ST | RF1 can also be analog input 5. |
| RF2/AN6 | 24 | 26 | 16 | 30 | 18 | I/O | ST | RF2 can also be analog input 6. |
| RF3/AN7 | 23 | 25 | 15 | 29 | 17 | I/O | ST | RF3 can also be analog input 7. |
| RF4/AN8 | 22 | 24 | 14 | 28 | 16 | I/O | ST | RF4 can also be analog input 8. |
| RF5/AN9 | 21 | 23 | 13 | 27 | 15 | I/O | ST | RF5 can also be analog input 9. |
| RF6/AN10 | 20 | 22 | 12 | 26 | 14 | I/O | ST | RF6 can also be analog input 10. |
| RF7/AN11 | 19 | 21 | 11 | 25 | 13 | I/O | ST | RF7 can also be analog input 11. |

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

**Note 1:** The output is only available by the peripheral operation.

**2:** Open drain input/output pin. Pin forced to input upon any device RESET.

**TABLE 3-1: PINOUT DESCRIPTIONS (CONTINUED)**

| Name | PIC17C75X | | | PIC17C76X | | I/O/P Type | Buffer Type | Description |
|---|---|---|---|---|---|---|---|---|
| | DIP No. | PLCC No. | TQFP No. | PLCC No. | QFP No. | | | |
| | | | | | | | | PORTG is a bi-directional I/O Port. |
| RG0/AN3 | 32 | 34 | 24 | 42 | 30 | I/O | ST | RG0 can also be analog input 3. |
| RG1/AN2 | 31 | 33 | 23 | 41 | 29 | I/O | ST | RG1 can also be analog input 2. |
| RG2/AN1/VREF- | 30 | 32 | 22 | 40 | 28 | I/O | ST | RG2 can also be analog input 1, or the ground reference voltage. |
| RG3/AN0/VREF+ | 29 | 31 | 21 | 39 | 27 | I/O | ST | RG3 can also be analog input 0, or the positive reference voltage. |
| RG4/CAP3 | 35 | 38 | 27 | 46 | 33 | I/O | ST | RG4 can also be the Capture3 input pin. |
| RG5/PWM3 | 36 | 39 | 28 | 47 | 34 | I/O | ST | RG5 can also be the PWM3 output pin. |
| RG6/RX2/DT2 | 38 | 41 | 30 | 49 | 36 | I/O | ST | RG6 can also be selected as the USART2 (SCI) Asynchronous Receive or USART2 (SCI) Synchronous Data. |
| RG7/TX2/CK2 | 37 | 40 | 29 | 48 | 35 | I/O | ST | RG7 can also be selected as the USART2 (SCI) Asynchronous Transmit or USART2 (SCI) Synchronous Clock. |
| | | | | | | | | PORTH is a bi-directional I/O Port. PORTH is only available on the PIC17C76X devices. |
| RH0 | — | — | — | 10 | 79 | I/O | ST | |
| RH1 | — | — | — | 11 | 80 | I/O | ST | |
| RH2 | — | — | — | 12 | 1 | I/O | ST | |
| RH3 | — | — | — | 13 | 2 | I/O | ST | |
| RH4/AN12 | — | — | — | 31 | 19 | I/O | ST | RH4 can also be analog input 12. |
| RH5/AN13 | — | — | — | 32 | 20 | I/O | ST | RH5 can also be analog input 13. |
| RH6/AN14 | — | — | — | 33 | 21 | I/O | ST | RH6 can also be analog input 14. |
| RH7/AN15 | — | — | — | 34 | 22 | I/O | ST | RH7 can also be analog input 15. |
| | | | | | | | | PORTJ is a bi-directional I/O Port. PORTJ is only available on the PIC17C76X devices. |
| RJ0 | — | — | — | 52 | 39 | I/O | ST | |
| RJ1 | — | — | — | 53 | 40 | I/O | ST | |
| RJ2 | — | — | — | 54 | 41 | I/O | ST | |
| RJ3 | — | — | — | 55 | 42 | I/O | ST | |
| RJ4 | — | — | — | 73 | 59 | I/O | ST | |
| RJ5 | — | — | — | 74 | 60 | I/O | ST | |
| RJ6 | — | — | — | 75 | 61 | I/O | ST | |
| RJ7 | — | — | — | 76 | 62 | I/O | ST | |
| TEST | 16 | 17 | 8 | 21 | 10 | I | ST | Test mode selection control input. Always tie to VSS for normal operation. |
| VSS | 17, 33, 49, 64 | 19, 36, 53, 68 | 9, 25, 41, 56 | 23, 44, 65, 84 | 11, 31, 51, 70 | P | | Ground reference for logic and I/O pins. |
| VDD | 1, 18, 34, 46 | 2, 20, 37, 49, | 10, 26, 38, 57 | 24, 45, 61, 2 | 12, 32, 48, 71 | P | | Positive supply for logic and I/O pins. |
| AVSS | 28 | 30 | 20 | 38 | 26 | P | | Ground reference for A/D converter. This pin **MUST** be at the same potential as VSS. |
| AVDD | 27 | 29 | 19 | 37 | 25 | P | | Positive supply for A/D converter. This pin **MUST** be at the same potential as VDD. |
| NC | — | 1, 18, 35, 52 | — | 1, 22, 43, 64 | — | | | No Connect. Leave these pins unconnected. |

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

**Note 1:** The output is only available by the peripheral operation.

**2:** Open drain input/output pin. Pin forced to input upon any device RESET.


© 2000 Microchip Technology Inc.

## 4.0 ON-CHIP OSCILLATOR CIRCUIT

The internal oscillator circuit is used to generate the device clock. Four device clock periods generate an internal instruction clock ($T_{CY}$).

There are four modes that the oscillator can operate in. They are selected by the device configuration bits during device programming. These modes are:

- LF Low Frequency ($F_{OSC} \leq 2$ MHz)
- XT Standard Crystal/Resonator Frequency (2 MHz ≤ $F_{OSC}$ ≤ 33 MHz)
- EC External Clock Input (Default oscillator configuration)
- RC External Resistor/Capacitor ($F_{OSC} \leq 4$ MHz)

There are two timers that offer necessary delays on power-up. One is the Oscillator Start-up Timer (OST), intended to keep the chip in RESET until the crystal oscillator is stable. The other is the Power-up Timer (PWRT), which provides a fixed delay of 96 ms (nominal) on POR and BOR. The PWRT is designed to keep the part in RESET while the power supply stabilizes. With these two timers on-chip, most applications need no external RESET circuitry.

SLEEP mode is designed to offer a very low current power-down mode. The user can wake from SLEEP through external RESET, Watchdog Timer Reset, or through an interrupt.

Several oscillator options are made available to allow the part to better fit the application. The RC oscillator option saves system cost while the LF crystal option saves power. Configuration bits are used to select various options.

### 4.1 Oscillator Configurations

#### 4.1.1 OSCILLATOR TYPES

The PIC17CXXX can be operated in four different oscillator modes. The user can program two configuration bits (FOSC1:FOSC0) to select one of these four modes:

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

The main difference between the LF and XT modes is the gain of the internal inverter of the oscillator circuit, which allows the different frequency ranges.

For more details on the device configuration bits, see Section 17.0.

#### 4.1.2 CRYSTAL OSCILLATOR/CERAMIC RESONATORS

In XT or LF modes, a crystal or ceramic resonator is connected to the OSC1/CLKIN and OSC2/CLKOUT pins to establish oscillation (Figure 4-2). The PIC17CXXX oscillator design requires the use of a parallel cut crystal. Use of a series cut crystal may give a frequency out of the crystal manufacturers specifications.

For frequencies above 24 MHz, it is common for the crystal to be an overtone mode crystal. Use of overtone mode crystals require a tank circuit to attenuate the gain at the fundamental frequency. Figure 4-3 shows an example circuit.

#### 4.1.3 OSCILLATOR/RESONATOR START-UP

As the device voltage increases from Vss, the oscillator will start its oscillations. The time required for the oscillator to start oscillating depends on many factors. These include:

- Crystal/resonator frequency
- Capacitor values used (C1 and C2)
- Device $V_{DD}$ rise time
- System temperature
- Series resistor value (and type) if used
- Oscillator mode selection of device (which selects the gain of the internal oscillator inverter)

Figure 4-1 shows an example of a typical oscillator/resonator start-up. The peak-to-peak voltage of the oscillator waveform can be quite low (less than 50% of device $V_{DD}$) when the waveform is centered at $V_{DD}/2$ (refer to parameter #D033 and parameter #D043 in the electrical specification section).

**FIGURE 4-1: OSCILLATOR/ RESONATOR START-UP CHARACTERISTICS**

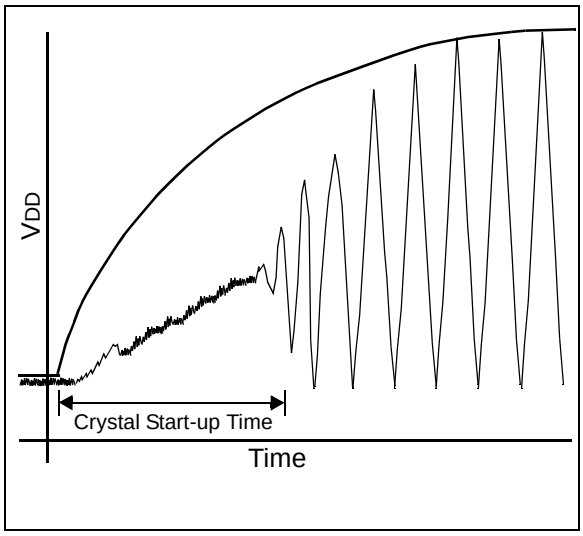
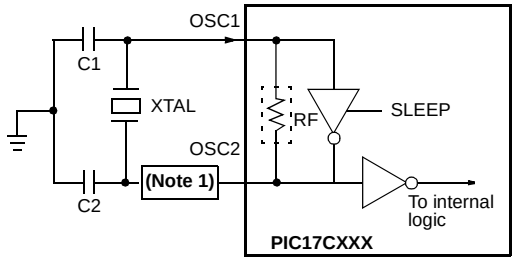


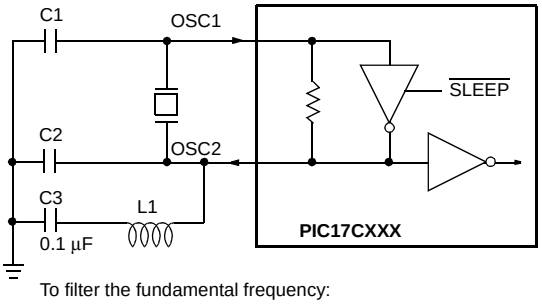
**FIGURE 4-2: CRYSTAL OR CERAMIC RESONATOR OPERATION (XT OR LF OSC CONFIGURATION)**




See Table 4-1 and Table 4-2 for recommended values of C1 and C2.

**Note 1:** A series resistor (Rs) may be required for AT strip cut crystals.

**FIGURE 4-3: CRYSTAL OPERATION, OVERTONE CRYSTALS (XT OSC CONFIGURATION)**




To filter the fundamental frequency:

$$\frac{1}{L1*C2} = (2\pi f)^2$$

Where f = tank circuit resonant frequency. This should be midway between the fundamental and the 3rd overtone frequencies of the crystal.
C3 blocks DC current to ground.

**TABLE 4-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS**

| Oscillator Type | Resonator Frequency | Capacitor Range C1 = C2(1) |
|---|---|---|
| LF | 455 kHz<br>2.0 MHz | 15 - 68 pF<br>10 - 33 pF |
| XT | 4.0 MHz<br>8.0 MHz<br>16.0 MHz | 22 - 68 pF<br>33 - 100 pF<br>33 - 100 pF |

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

**Note 1:** These values include all board capacitances on this pin. Actual capacitor value depends on board capacitance.

**Resonators Used:**

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

Resonators used did not have built-in capacitors.

**TABLE 4-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR**

| Osc Type | Freq | C1(2) | C2(2) |
|---|---|---|---|
| LF | 32 kHz<br>1 MHz<br>2 MHz | 100-150 pF<br>10-68 pF<br>10-68 pF | 100-150 pF<br>10-68 pF<br>10-68 pF |
| XT | 2 MHz<br>4 MHz<br>8 MHz<br>16 MHz<br>24 MHz(1)<br>32 MHz(1) | 47-100 pF<br>15-68 pF<br>15-47 pF<br>15-47 pF<br>15-47 pF<br>10-47 pF | 47-100 pF<br>15-68 pF<br>15-47 pF<br>15-47 pF<br>15-47 pF<br>10-47 pF |

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

**Note 1:** Overtone crystals are used at 24 MHz and higher. The circuit in Figure 4-3 should be used to select the desired harmonic frequency.

**2:** These values include all board capacitances on this pin. Actual capacitor value depends on board capacitance.

**Crystals Used:**

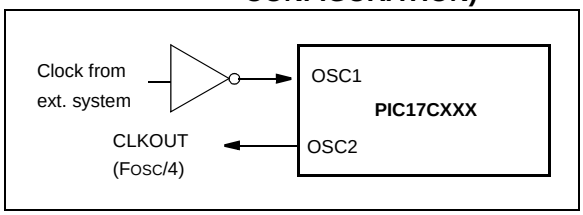
| | | |
|---|---|---|
| 32.768 kHz | Epson C-001R32.768K-A | ± 20 PPM |
| 1.0 MHz | ECS-10-13-1 | ± 50 PPM |
| 2.0 MHz | ECS-20-20-1 | ± 50 PPM |
| 4.0 MHz | ECS-40-20-1 | ± 50 PPM |
| 8.0 MHz | ECS ECS-80-S-4<br>ECS-80-18-1 | ± 50 PPM |
| 16.0 MHz | ECS-160-20-1 | ± 50 PPM |
| 25 MHz | CTS CTS25M |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


© 2000 Microchip Technology Inc.

### 4.1.4 EXTERNAL CLOCK OSCILLATOR

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

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




### 4.1.5 EXTERNAL CRYSTAL OSCILLATOR CIRCUIT

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

Figure 4-5 shows implementation of a parallel resonant oscillator circuit. The circuit is designed to use the fundamental frequency of the crystal. The 74AS04 inverter performs the 180-degree phase shift that a parallel oscillator requires. The 4.7 kΩ resistor provides the negative feedback for stability. The 10 kΩ potentiometer biases the 74AS04 in the linear region. This could be used for external oscillator designs.

**FIGURE 4-5: EXTERNAL PARALLEL RESONANT CRYSTAL OSCILLATOR CIRCUIT**

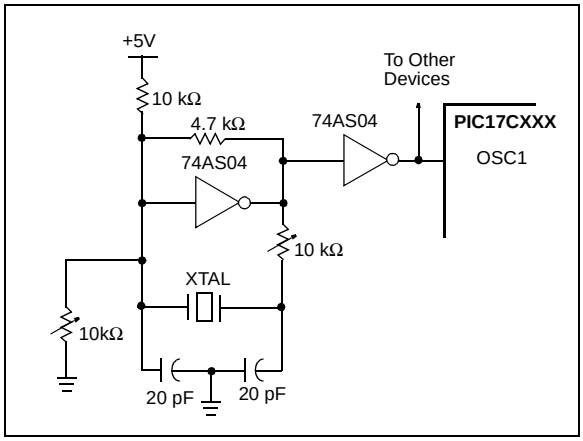
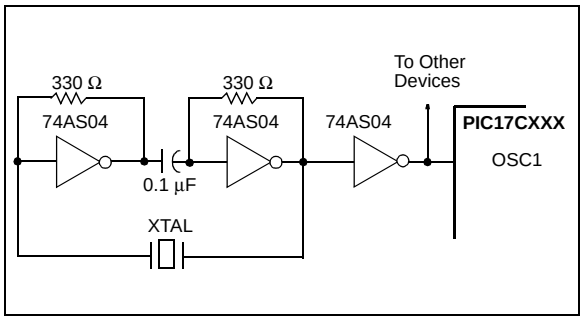



Figure 4-6 shows a series resonant oscillator circuit. This circuit is also designed to use the fundamental frequency of the crystal. The inverter performs a 180-degree phase shift in a series resonant oscillator circuit. The 330 Ω resistors provide the negative feedback to bias the inverters in their linear region.

**FIGURE 4-6: EXTERNAL SERIES RESONANT CRYSTAL OSCILLATOR CIRCUIT**



### 4.1.6 RC OSCILLATOR

For timing insensitive applications, the RC device option offers additional cost savings. RC oscillator frequency is a function of the supply voltage, the resistor (REXT) and capacitor (CEXT) values, and the operating temperature. In addition to this, oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect oscillation frequency, especially for low CEXT values. The user also needs to take into account variation due to tolerance of external R and C components used. Figure 4-7 shows how the R/C combination is connected to the PIC17CXXX. For REXT values below 2.2 kΩ, the oscillator operation may become unstable, or stop completely. For very high REXT values (e.g. 1 MΩ), the oscillator becomes sensitive to noise, humidity and leakage. Thus, we recommend to keep REXT between 3 kΩ and 100 kΩ.

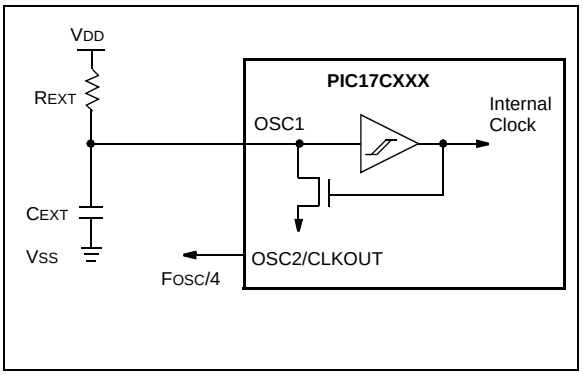
Although the oscillator will operate with no external capacitor (CEXT = 0 pF), we recommend using values above 20 pF for noise and stability reasons. With little or no external capacitance, oscillation frequency can vary dramatically due to changes in external capacitances, such as PCB trace capacitance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

See Section 21.0 for RC frequency variation from part to part due to normal process variation. The variation is larger for larger R (since leakage current variation will affect RC frequency more for large R) and for smaller C (since variation of input capacitance will affect RC frequency more).

See Section 21.0 for variation of oscillator frequency due to VDD for given REXT/CEXT values, as well as frequency variation due to operating temperature for given R, C, and VDD values.

The oscillator frequency, divided by 4, is available on the OSC2/CLKOUT pin and can be used for test purposes or to synchronize other logic (see Figure 4-8 for waveform).

**FIGURE 4-7: RC OSCILLATOR MODE**




#### 4.1.6.1 RC Start-up

As the device voltage increases, the RC will immediately start its oscillations once the pin voltage levels meet the input threshold specifications (parameter #D032 and parameter #D042 in the electrical specification section). The time required for the RC to start oscillating depends on many factors. These include:

- Resistor value used
- Capacitor value used
- Device VDD rise time
- System temperature


© 2000 Microchip Technology Inc.

## 4.2 Clocking Scheme/Instruction Cycle

The clock input (from OSC1) is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3 and Q4. Internally, the program counter (PC) is incremented every Q1 and the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 4-8.

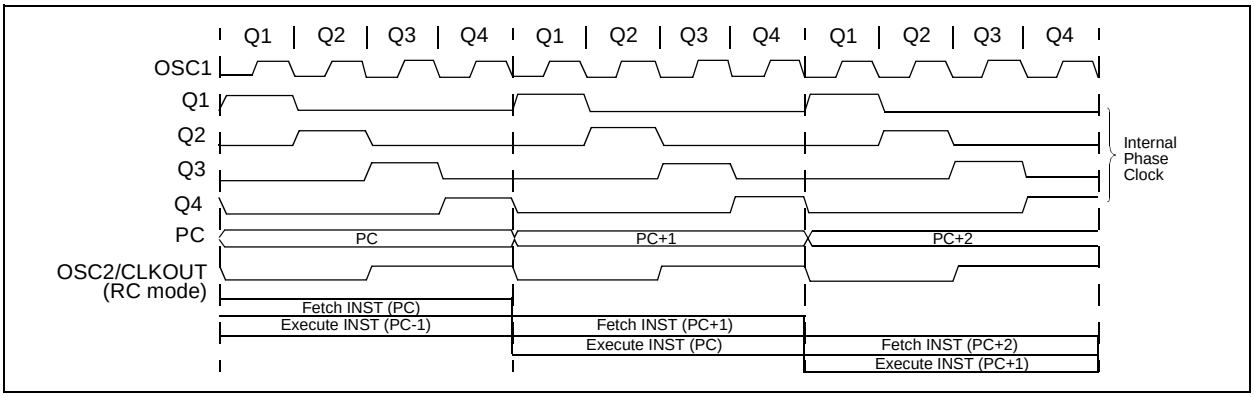
## 4.3 Instruction Flow/Pipelining

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 4-1).

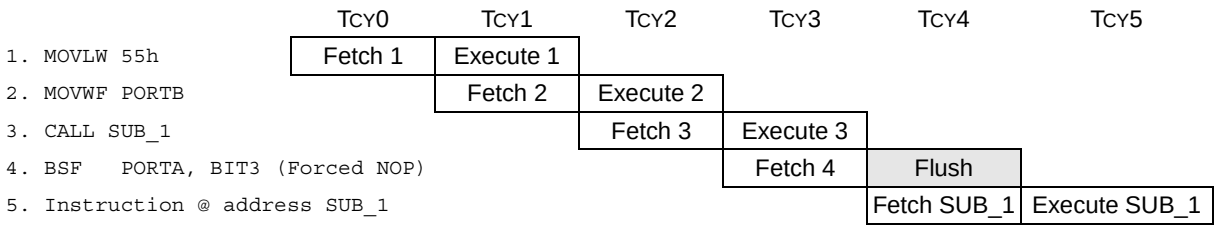
A fetch cycle begins with the program counter incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the “Instruction Register (IR)” in cycle Q1. This instruction is then decoded and executed during the Q2, Q3 and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

**FIGURE 4-8: CLOCK/INSTRUCTION CYCLE**




**EXAMPLE 4-1: INSTRUCTION PIPELINE FLOW**




All instructions are single cycle, except for any program branches. These take two cycles since the fetched instruction is “flushed” from the pipeline, while the new instruction is being fetched and then executed.

**NOTES:**


© 2000 Microchip Technology Inc.

## 5.0 RESET

The PIC17CXXX differentiates between various kinds of RESET:

- Power-on Reset (POR)
- Brown-out Reset
- $\overline{\text{MCLR}}$ Reset
- WDT Reset

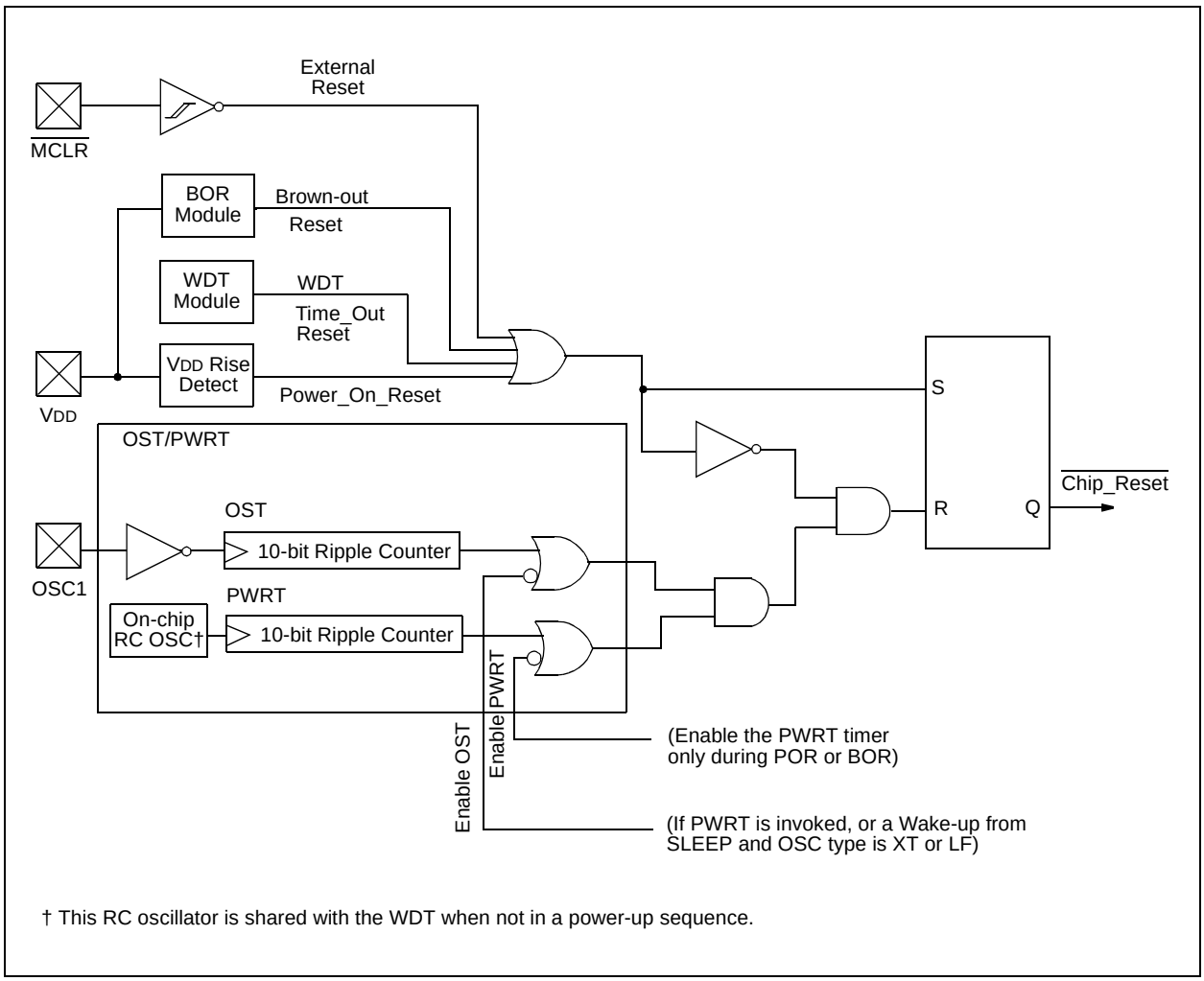
Some registers are not affected in any RESET condition, their status is unknown on POR and unchanged in any other RESET. Most other registers are forced to a "RESET state". The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits are set or cleared differently in different RESET situations, as indicated in Table 5-3. These bits, in conjunction with the $\overline{\text{POR}}$ and $\overline{\text{BOR}}$ bits, are used in software to determine the nature of the RESET. See Table 5-4 for a full description of the RESET states of all registers.

When the device enters the "RESET state", the Data Direction registers (DDR) are forced set, which will make the I/O hi-impedance inputs. The RESET state of some peripheral modules may force the I/O to other operations, such as analog inputs or the system bus.

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

A simplified block diagram of the On-Chip Reset Circuit is shown in Figure 5-1.

**FIGURE 5-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT**




† This RC oscillator is shared with the WDT when not in a power-up sequence.

## 5.1 Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST) and Brown-out Reset (BOR)

### 5.1.1 POWER-ON RESET (POR)

The Power-on Reset circuit holds the device in RESET until VDD is above the trip point (in the range of 1.4V - 2.3V). The devices produce an internal RESET for both rising and falling VDD. To take advantage of the POR, just tie the MCLR/VPP pin directly (or through a resistor) to VDD. This will eliminate external RC components usually needed to create Power-on Reset. A minimum rise time for VDD is required. See Electrical Specifications for details.

Figure 5-2 and Figure 5-3 show two possible POR circuits.

**FIGURE 5-2: USING ON-CHIP POR**

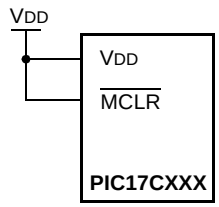
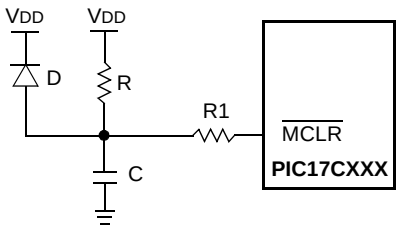



**FIGURE 5-3: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)**




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

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

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

### 5.1.2 POWER-UP TIMER (PWRT)

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

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

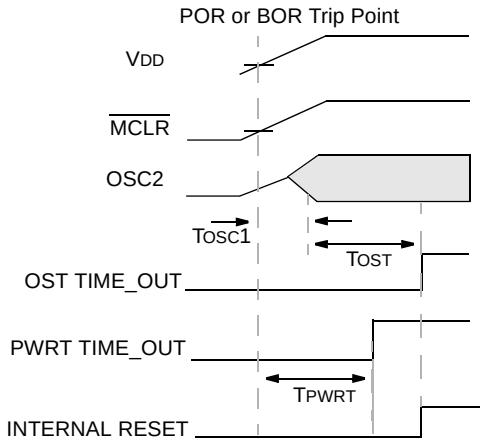
### 5.1.3 OSCILLATOR START-UP TIMER (OST)

The Oscillator Start-up Timer (OST) provides a 1024 oscillator cycle (1024TOSC) delay whenever the PWRT is invoked, or a wake-up from SLEEP event occurs in XT or LF mode. The PWRT and OST operate in parallel.

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

Figure 5-4 shows the operation of the OST circuit. In this figure, the oscillator is of such a low frequency that although enabled simultaneously, the OST does not time-out until after the Power-up Timer time-out.

**FIGURE 5-4: OSCILLATOR START-UP TIME (LOW FREQUENCY)**




This figure shows in greater detail the timings involved with the oscillator start-up timer. In this example, the low frequency crystal start-up time is larger than power-up time (TPWRT).

TOSC1 = time for the crystal oscillator to react to an oscillation level detectable by the Oscillator Start-up Timer (OST).

TOST = 1024TOSC.


© 2000 Microchip Technology Inc.

### 5.1.4 TIME-OUT SEQUENCE

On power-up, the time-out sequence is as follows: First, the internal POR signal goes high when the POR trip point is reached. If MCLR is high, then both the OST and PWRT timers start. In general, the PWRT time-out is longer, except with low frequency crystals/resonators. The total time-out also varies based on oscillator configuration. Table 5-1 shows the times that are associated with the oscillator configuration. Figure 5-5 and Figure 5-6 display these time-out sequences.

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

The time-out sequence begins from the first rising edge of MCLR.

Table 5-3 shows the RESET conditions for some special registers, while Table 5-4 shows the initialization conditions for all the registers.

**TABLE 5-1: TIME-OUT IN VARIOUS SITUATIONS**

| Oscillator Configuration | POR, BOR | Wake-up from SLEEP | MCLR Reset |
|---|---|---|---|
| XT, LF | Greater of: 96 ms or 1024TOSC | 1024TOSC | — |
| EC, RC | Greater of: 96 ms or 1024TOSC | — | — |

**TABLE 5-2: STATUS BITS AND THEIR SIGNIFICANCE**

| POR | BOR(1) | TO | PD | Event |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | Power-on Reset |
| 1 | 1 | 1 | 0 | MCLR Reset during SLEEP or interrupt wake-up from SLEEP |
| 1 | 1 | 0 | 1 | WDT Reset during normal operation |
| 1 | 1 | 0 | 0 | WDT Wake-up during SLEEP |
| 1 | 1 | 1 | 1 | MCLR Reset during normal operation |
| 1 | 0 | 1 | 1 | Brown-out Reset |
| 0 | 0 | 0 | x | Illegal, TO is set on POR |
| 0 | 0 | x | 0 | Illegal, PD is set on POR |
| x | x | 1 | 1 | CLRWDT instruction executed |

**Note 1:** When BODEN is enabled, else the BOR status bit is unknown.

**TABLE 5-3: RESET CONDITION FOR THE PROGRAM COUNTER AND THE CPUSTA REGISTER**

| Event | | PCH:PCL | CPUSTA(4) | OST Active |
|---|---|---|---|---|
| Power-on Reset | | 0000h | --11 1100 | Yes |
| Brown-out Reset | | 0000h | --11 1110 | Yes |
| MCLR Reset during normal operation | | 0000h | --11 1111 | No |
| MCLR Reset during SLEEP | | 0000h | --11 1011 | Yes(2) |
| WDT Reset during normal operation | | 0000h | --11 0111 | No |
| WDT Reset during SLEEP(3) | | 0000h | --11 0011 | Yes(2) |
| Interrupt Wake-up from SLEEP | GLINTD is set | PC + 1 | --11 1011 | Yes(2) |
| | GLINTD is clear | PC + 1(1) | --10 1011 | Yes(2) |

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

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

**2:** The OST is only active (on wake-up) when the oscillator is configured for XT or LF modes.

**3:** The Program Counter = 0; that is, the device branches to the RESET vector and places SFRs in WDT Reset states. This is different from the mid-range devices.

**4:** When BODEN is enabled, else the BOR status bit is unknown.

In Figure 5-5, Figure 5-6 and Figure 5-7, the TPWRT timer time-out is greater then the TOST timer time-out, as would be the case in higher frequency crystals. For lower frequency crystals (i.e., 32 kHz), TOST may be greater.

**FIGURE 5-5: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ TIED TO VDD)**

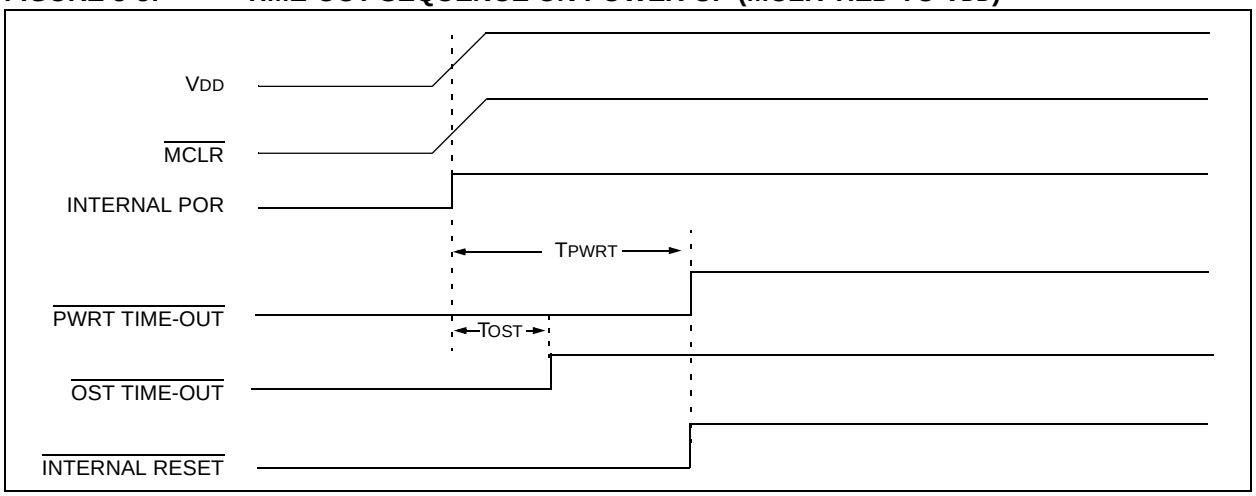



**FIGURE 5-6: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO VDD)**

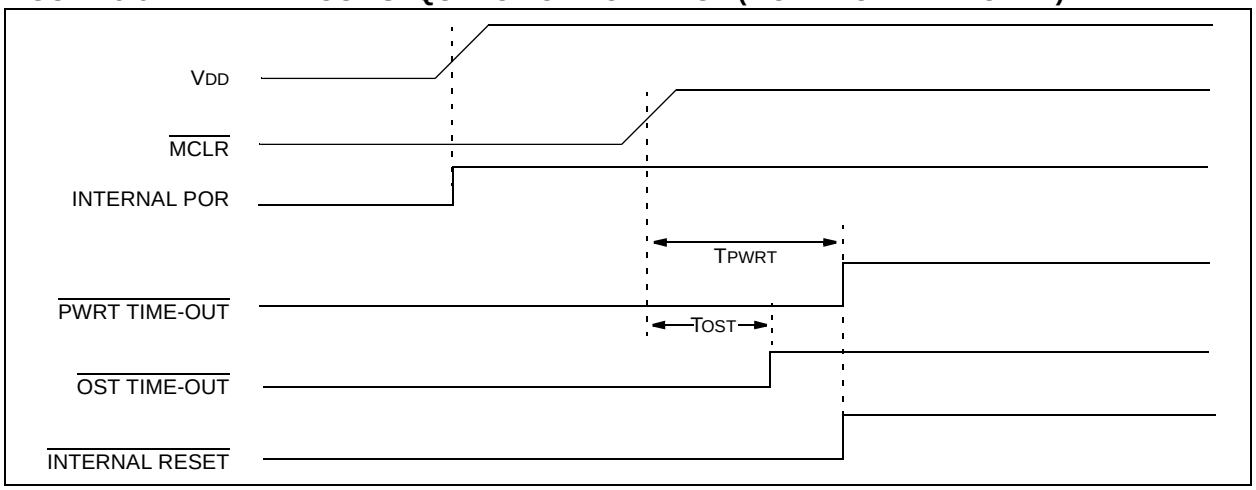
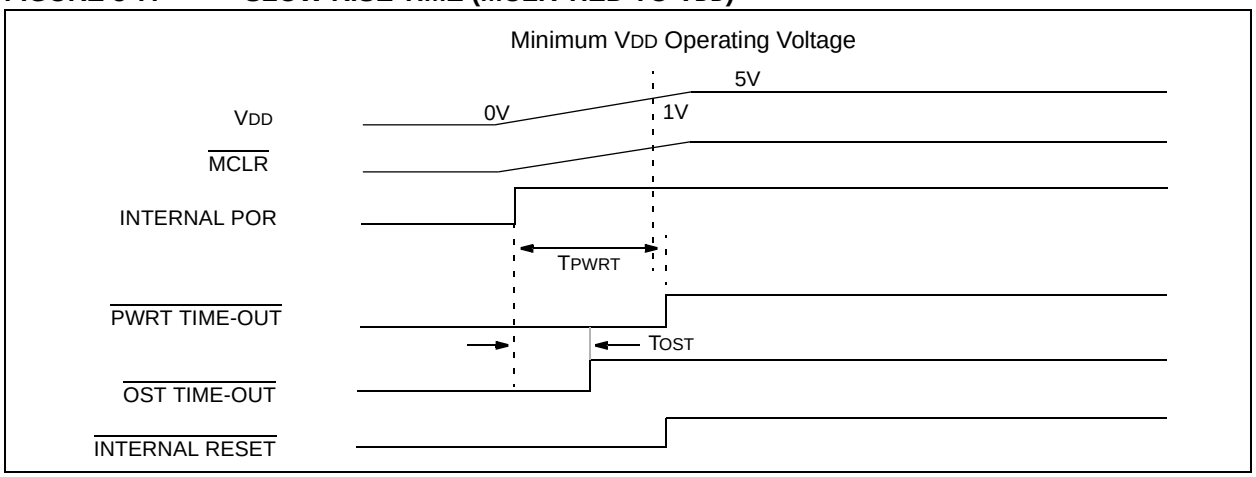



**FIGURE 5-7: SLOW RISE TIME ($\overline{\text{MCLR}}$ TIED TO VDD)**





© 2000 Microchip Technology Inc.

**TABLE 5-4: INITIALIZATION CONDITIONS FOR SPECIAL FUNCTION REGISTERS**

| Register | Address | Power-on Reset<br>Brown-out Reset | $\overline{\text{MCLR}}$ Reset<br>WDT Reset | Wake-up from SLEEP through Interrupt |
|---|---|---|---|---|
| **Unbanked** | | | | |
| INDF0 | 00h | N/A | N/A | N/A |
| FSR0 | 01h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PCL | 02h | 0000h | 0000h | PC + 1(2) |
| PCLATH | 03h | 0000 0000 | uuuu uuuu | uuuu uuuu |
| ALUSTA | 04h | 1111 xxxx | 1111 uuuu | 1111 uuuu |
| T0STA | 05h | 0000 000- | 0000 000- | 0000 000- |
| CPUSTA(3) | 06h | --11 11qq | --11 qquu | --uu qquu |
| INTSTA | 07h | 0000 0000 | 0000 0000 | uuuu uuuu(1) |
| INDF1 | 08h | N/A | N/A | N/A |
| FSR1 | 09h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| WREG | 0Ah | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR0L | 0Bh | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR0H | 0Ch | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TBLPTRL | 0Dh | 0000 0000 | 0000 0000 | uuuu uuuu |
| TBLPTRH | 0Eh | 0000 0000 | 0000 0000 | uuuu uuuu |
| BSR | 0Fh | 0000 0000 | 0000 0000 | uuuu uuuu |
| **Bank 0** | | | | |
| PORTA(4,6) | 10h | 0-xx 11xx | 0-uu 11uu | u-uu uuuu |
| DDRB | 11h | 1111 1111 | 1111 1111 | uuuu uuuu |
| PORTB(4) | 12h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RCSTA1 | 13h | 0000 -00x | 0000 -00u | uuuu -uuu |
| RCREG1 | 14h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXSTA1 | 15h | 0000 --1x | 0000 --1u | uuuu --uu |
| TXREG1 | 16h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| SPBRG1 | 17h | 0000 0000 | 0000 0000 | uuuu uuuu |

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

**Note 1:** One or more bits in INTSTA, PIR1, PIR2 will be affected (to cause wake-up).

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

**3:** See Table 5-3 for RESET value of specific condition.

**4:** This is the value that will be in the port output latch.

**5:** When the device is configured for Microprocessor or Extended Microcontroller mode, the operation of this port does not rely on these registers.

**6:** On any device RESET, these pins are configured as inputs.

**TABLE 5-4: INITIALIZATION CONDITIONS FOR SPECIAL FUNCTION REGISTERS (CONTINUED)**

| Register | Address | Power-on Reset Brown-out Reset | $\overline{\text{MCLR}}$ Reset WDT Reset | Wake-up from SLEEP through Interrupt |
|---|---|---|---|---|
| **Bank 1** | | | | |
| DDRC(5) | 10h | 1111 1111 | 1111 1111 | uuuu uuuu |
| PORTC(4,5) | 11h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| DDRD(5) | 12h | 1111 1111 | 1111 1111 | uuuu uuuu |
| PORTD(4,5) | 13h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| DDRE(5) | 14h | ---- 1111 | ---- 1111 | ---- uuuu |
| PORTE(4,5) | 15h | ---- xxxx | ---- uuuu | ---- uuuu |
| PIR1 | 16h | x000 0010 | u000 0010 | uuuu uuuu(1) |
| PIE1 | 17h | 0000 0000 | 0000 0000 | uuuu uuuu |
| **Bank 2** | | | | |
| TMR1 | 10h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR2 | 11h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR3L | 12h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR3H | 13h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PR1 | 14h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PR2 | 15h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PR3/CA1L | 16h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PR3/CA1H | 17h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| **Bank 3** | | | | |
| PW1DCL | 10h | xx-- ---- | uu-- ---- | uu-- ---- |
| PW2DCL | 11h | xx0- ---- | uu0- ---- | uuu- ---- |
| PW1DCH | 12h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PW2DCH | 13h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CA2L | 14h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CA2H | 15h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TCON1 | 16h | 0000 0000 | 0000 0000 | uuuu uuuu |
| TCON2 | 17h | 0000 0000 | 0000 0000 | uuuu uuuu |

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

**Note 1:** One or more bits in INTSTA, PIR1, PIR2 will be affected (to cause wake-up).

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

**3:** See Table 5-3 for RESET value of specific condition.

**4:** This is the value that will be in the port output latch.

**5:** When the device is configured for Microprocessor or Extended Microcontroller mode, the operation of this port does not rely on these registers.

**6:** On any device RESET, these pins are configured as inputs.


© 2000 Microchip Technology Inc.

**TABLE 5-4: INITIALIZATION CONDITIONS FOR SPECIAL FUNCTION REGISTERS (CONTINUED)**

| Register | Address | Power-on Reset<br>Brown-out Reset | $\overline{\text{MCLR}}$ Reset<br>WDT Reset | Wake-up from SLEEP through Interrupt |
|---|---|---|---|---|
| **Bank 4** | | | | |
| PIR2 | 10h | 000- 0010 | 000- 0010 | uuu- uuuu(1) |
| PIE2 | 11h | 000- 0000 | 000- 0000 | uuu- uuuu |
| Unimplemented | 12h | ---- ---- | ---- ---- | ---- ---- |
| RCSTA2 | 13h | 0000 -00x | 0000 -00u | uuuu -uuu |
| RCREG2 | 14h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXSTA2 | 15h | 0000 --1x | 0000 --1u | uuuu --uu |
| TXREG2 | 16h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| SPBRG2 | 17h | 0000 0000 | 0000 0000 | uuuu uuuu |
| **Bank 5** | | | | |
| DDRF | 10h | 1111 1111 | 1111 1111 | uuuu uuuu |
| PORTF(4) | 11h | 0000 0000 | 0000 0000 | uuuu uuuu |
| DDRG | 12h | 1111 1111 | 1111 1111 | uuuu uuuu |
| PORTG(4) | 13h | xxxx 0000 | uuuu 0000 | uuuu uuuu |
| ADCON0 | 14h | 0000 -0-0 | 0000 -0-0 | uuuu uuuu |
| ADCON1 | 15h | 000- 0000 | 000- 0000 | uuuu uuuu |
| ADRESL | 16h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ADRESH | 17h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| **Bank 6** | | | | |
| SSPADD | 10h | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPCON1 | 11h | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPCON2 | 12h | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPSTAT | 13h | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPBUF | 14h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| Unimplemented | 15h | ---- ---- | ---- ---- | ---- ---- |
| Unimplemented | 16h | ---- ---- | ---- ---- | ---- ---- |
| Unimplemented | 17h | ---- ---- | ---- ---- | ---- ---- |

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

**Note 1:** One or more bits in INTSTA, PIR1, PIR2 will be affected (to cause wake-up).

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

**3:** See Table 5-3 for RESET value of specific condition.

**4:** This is the value that will be in the port output latch.

**5:** When the device is configured for Microprocessor or Extended Microcontroller mode, the operation of this port does not rely on these registers.

**6:** On any device RESET, these pins are configured as inputs.

**TABLE 5-4: INITIALIZATION CONDITIONS FOR SPECIAL FUNCTION REGISTERS (CONTINUED)**

| Register | Address | Power-on Reset<br>Brown-out Reset | $\overline{\text{MCLR}}$ Reset<br>WDT Reset | Wake-up from SLEEP<br>through Interrupt |
|---|---|---|---|---|
| **Bank 7** | | | | |
| PW3DCL | 10h | xx0- ---- | uu0- ---- | uuu- ---- |
| PW3DCH | 11h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CA3L | 12h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CA3H | 13h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CA4L | 14h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CA4H | 15h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TCON3 | 16h | -000 0000 | -000 0000 | -uuu uuuu |
| Unimplemented | 17h | ---- ---- | ---- ---- | ---- ---- |
| **Bank 8** | | | | |
| DDRH | 10h | 1111 1111 | 1111 1111 | uuuu uuuu |
| PORTH(4) | 11h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| DDRJ | 12h | 1111 1111 | 1111 1111 | uuuu uuuu |
| PORTJ(4) | 13h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| **Unbanked** | | | | |
| PRODL | 18h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PRODH | 19h | xxxx xxxx | uuuu uuuu | uuuu uuuu |

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

**Note 1:** One or more bits in INTSTA, PIR1, PIR2 will be affected (to cause wake-up).

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

**3:** See Table 5-3 for RESET value of specific condition.

**4:** This is the value that will be in the port output latch.

**5:** When the device is configured for Microprocessor or Extended Microcontroller mode, the operation of this port does not rely on these registers.

**6:** On any device RESET, these pins are configured as inputs.


© 2000 Microchip Technology Inc.

### 5.1.5 BROWN-OUT RESET (BOR)

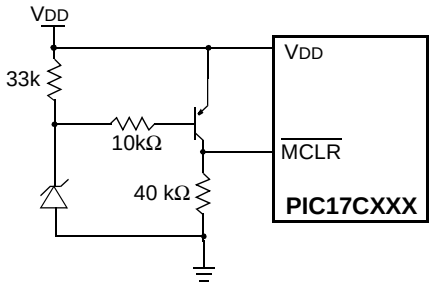
PIC17C7XX devices have on-chip Brown-out Reset circuitry. This circuitry places the device into a RESET when the device voltage falls below a trip point (BVDD). This ensures that the device does not continue program execution outside the valid operation range of the device. Brown-out Resets are typically used in AC line applications, or large battery applications, where large loads may be switched in (such as automotive).

| **Note:** | Before using the on-chip Brown-out for a voltage supervisory function, please review the electrical specifications to ensure that they meet your requirements. |
|---|---|

The BODEN configuration bit can disable (if clear/ programmed), or enable (if set) the Brown-out Reset circuitry. If VDD falls below BVDD (typically 4.0 V, paramter #D005 in electrical specification section), for greater than parameter #35, the Brown-out situation will reset the chip. A RESET is not guaranteed to occur if VDD falls below BVDD for less than paramter #35. The chip will remain in Brown-out Reset until VDD rises above BVDD. The Power-up Timer and Oscillator Start-up Timer will then be invoked. This will keep the chip in RESET the greater of 96 ms and 1024 TOSC. If VDD drops below BVDD while the Power-up Timer/Oscillator Start-up Timer is running, the chip will go back into a Brown-out Reset. The Power-up Timer/Oscillator Start-up Timer will be initialized. Once VDD rises above BVDD, the Power-up Timer/Oscillator Start-up Timer will start their time delays. Figure 5-10 shows typical Brown-out situations.

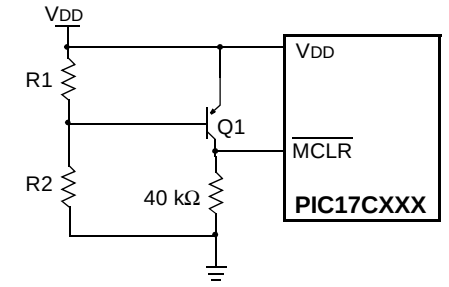
In some applications, the Brown-out Reset trip point of the device may not be at the desired level. Figure 5-8 and Figure 5-9 are two examples of external circuitry that may be implemented. Each needs to be evaluated to determine if they match the requirements of the application.

**FIGURE 5-8: EXTERNAL BROWN-OUT PROTECTION CIRCUIT 1**




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

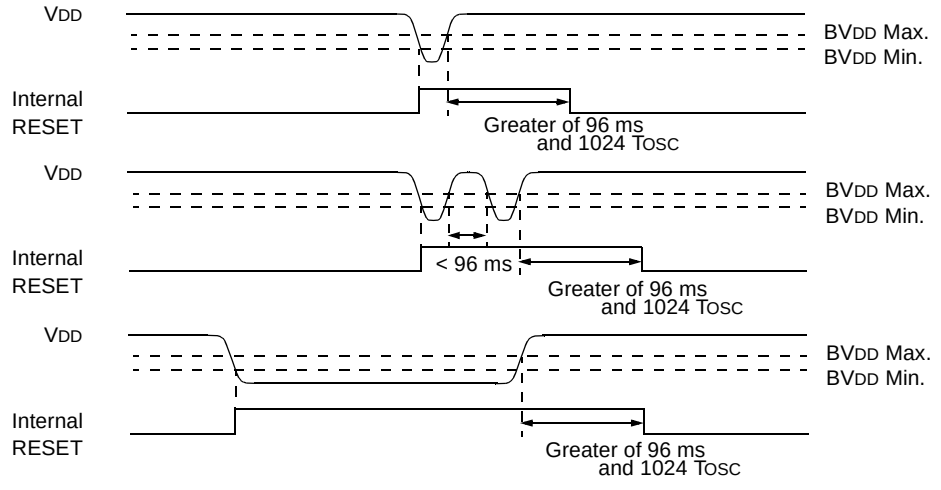
**FIGURE 5-9: EXTERNAL BROWN-OUT PROTECTION CIRCUIT 2**




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

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

**FIGURE 5-10: BROWN-OUT SITUATIONS**



**NOTES:**


© 2000 Microchip Technology Inc.

## 6.0 INTERRUPTS

PIC17C7XX devices have 18 sources of interrupt:

- External interrupt from the RA0/INT pin
- Change on RB7:RB0 pins
- TMR0 Overflow
- TMR1 Overflow
- TMR2 Overflow
- TMR3 Overflow
- USART1 Transmit buffer empty
- USART1 Receive buffer full
- USART2 Transmit buffer empty
- USART2 Receive buffer full
- SSP Interrupt
- SSP $I^2C$ bus collision interrupt
- A/D conversion complete
- Capture1
- Capture2
- Capture3
- Capture4
- T0CKI edge occurred

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

- CPUSTA
- INTSTA
- PIE1
- PIR1
- PIE2
- PIR2

The CPUSTA register contains the GLINTD bit. This is the Global Interrupt Disable bit. When this bit is set, all interrupts are disabled. This bit is part of the controller core functionality and is described in the Section 6.4.

When an interrupt is responded to, the GLINTD bit is automatically set to disable any further interrupts, the return address is pushed onto the stack and the PC is loaded with the interrupt vector address. There are four interrupt vectors. Each vector address is for a specific interrupt source (except the peripheral interrupts, which all vector to the same address). These sources are:

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

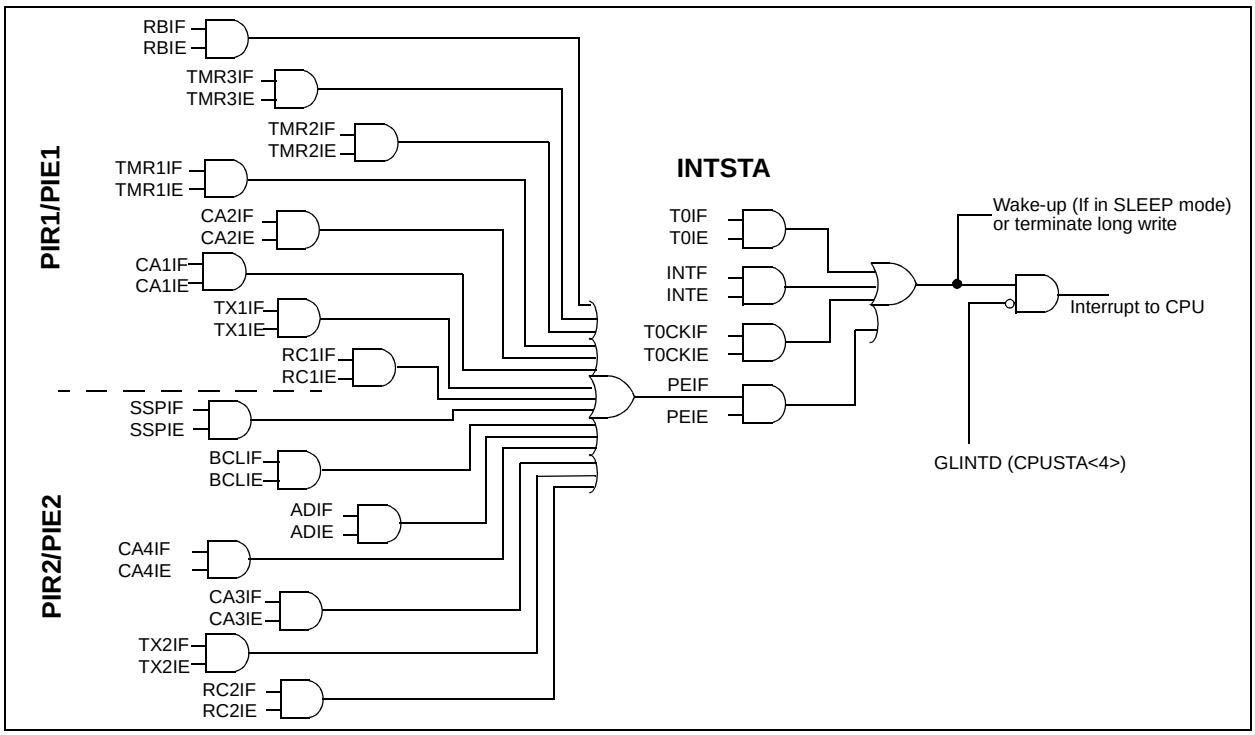
When program execution vectors to one of these interrupt vector addresses (except for the peripheral interrupts), the interrupt flag bit is automatically cleared. Vectoring to the peripheral interrupt vector address does not automatically clear the source of the interrupt. In the peripheral Interrupt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

When an interrupt condition is met, that individual interrupt flag bit will be set, regardless of the status of its corresponding mask bit or the GLINTD bit.

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

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

**FIGURE 6-1: INTERRUPT LOGIC**



## 6.1 Interrupt Status Register (INTSTA)

The Interrupt Status/Control register (INTSTA) contains the flag and enable bits for non-peripheral interrupts.

The PEIF bit is a read only, bit wise OR of all the peripheral flag bits in the PIR registers (Figure 6-4 and Figure 6-5).

| **Note:** | All interrupt flag bits get set by their specified condition, even if the corresponding interrupt enable bit is clear (interrupt disabled), or the GLINTD bit is set (all interrupts disabled). |
|---|---|

Care should be taken when clearing any of the INTSTA register enable bits when interrupts are enabled (GLINTD is clear). If any of the INTSTA flag bits (T0IF, INTF, T0CKIF, or PEIF) are set in the same instruction cycle as the corresponding interrupt enable bit is cleared, the device will vector to the RESET address (0x00).

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

**REGISTER 6-1: INTSTA REGISTER (ADDRESS: 07h, UNBANKED)**

| R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PEIF | T0CKIF | T0IF | INTF | PEIE | T0CKIE | T0IE | INTE |
| bit 7 | | | | | | | bit 0 |

bit 7 **PEIF**: Peripheral Interrupt Flag bit
This bit is the OR of all peripheral interrupt flag bits AND'ed with their corresponding enable bits. The interrupt logic forces program execution to address (20h) when a peripheral interrupt is pending.
1 = A peripheral interrupt is pending
0 = No peripheral interrupt is pending

bit 6 **T0CKIF**: External Interrupt on T0CKI Pin Flag bit
This bit is cleared by hardware, when the interrupt logic forces program execution to address (18h).
1 = The software specified edge occurred on the RA1/T0CKI pin
0 = The software specified edge did not occur on the RA1/T0CKI pin

bit 5 **T0IF**: TMR0 Overflow Interrupt Flag bit
This bit is cleared by hardware, when the interrupt logic forces program execution to address (10h).
1 = TMR0 overflowed
0 = TMR0 did not overflow

bit 4 **INTF**: External Interrupt on INT Pin Flag bit
This bit is cleared by hardware, when the interrupt logic forces program execution to address (08h).
1 = The software specified edge occurred on the RA0/INT pin
0 = The software specified edge did not occur on the RA0/INT pin

bit 3 **PEIE**: Peripheral Interrupt Enable bit
This bit acts as a global enable bit for the peripheral interrupts that have their corresponding enable bits set.
1 = Enable peripheral interrupts
0 = Disable peripheral interrupts

bit 2 **T0CKIE**: External Interrupt on T0CKI Pin Enable bit
1 = Enable software specified edge interrupt on the RA1/T0CKI pin
0 = Disable interrupt on the RA1/T0CKI pin

bit 1 **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enable TMR0 overflow interrupt
0 = Disable TMR0 overflow interrupt

bit 0 **INTE**: External Interrupt on RA0/INT Pin Enable bit
1 = Enable software specified edge interrupt on the RA0/INT pin
0 = Disable software specified edge interrupt on the RA0/INT pin

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |


© 2000 Microchip Technology Inc.

## 6.2 Peripheral Interrupt Enable Register1 (PIE1) and Register2 (PIE2)

These registers contains the individual enable bits for the peripheral interrupts.

**REGISTER 6-2: PIE1 REGISTER (ADDRESS: 17h, BANK 1)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| RBIE | TMR3IE | TMR2IE | TMR1IE | CA2IE | CA1IE | TX1IE | RC1IE |
| bit 7 | | | | | | | bit 0 |

bit 7 **RBIE**: PORTB Interrupt-on-Change Enable bit
1 = Enable PORTB interrupt-on-change
0 = Disable PORTB interrupt-on-change

bit 6 **TMR3IE**: TMR3 Interrupt Enable bit
1 = Enable TMR3 interrupt
0 = Disable TMR3 interrupt

bit 5 **TMR2IE**: TMR2 Interrupt Enable bit
1 = Enable TMR2 interrupt
0 = Disable TMR2 interrupt

bit 4 **TMR1IE**: TMR1 Interrupt Enable bit
1 = Enable TMR1 interrupt
0 = Disable TMR1 interrupt

bit 3 **CA2IE**: Capture2 Interrupt Enable bit
1 = Enable Capture2 interrupt
0 = Disable Capture2 interrupt

bit 2 **CA1IE**: Capture1 Interrupt Enable bit
1 = Enable Capture1 interrupt
0 = Disable Capture1 interrupt

bit 1 **TX1IE**: USART1 Transmit Interrupt Enable bit
1 = Enable USART1 Transmit buffer empty interrupt
0 = Disable USART1 Transmit buffer empty interrupt

bit 0 **RC1IE**: USART1 Receive Interrupt Enable bit
1 = Enable USART1 Receive buffer full interrupt
0 = Disable USART1 Receive buffer full interrupt

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ | |
| - n = Value at POR Reset | ’1’ = Bit is set | ’0’ = Bit is cleared | x = Bit is unknown |

**REGISTER 6-3: PIE2 REGISTER (ADDRESS: 11h, BANK 4)**

| R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| SSPIE | BCLIE | ADIE | — | CA4IE | CA3IE | TX2IE | RC2IE |
| bit 7 | | | | | | | bit 0 |

bit 7 **SSPIE**: Synchronous Serial Port Interrupt Enable bit
1 = Enable SSP interrupt
0 = Disable SSP interrupt

bit 6 **BCLIE**: Bus Collision Interrupt Enable bit
1 = Enable bus collision interrupt
0 = Disable bus collision interrupt

bit 5 **ADIE**: A/D Module Interrupt Enable bit
1 = Enable A/D module interrupt
0 = Disable A/D module interrupt

bit 4 **Unimplemented:** Read as '0'

bit 3 **CA4IE**: Capture4 Interrupt Enable bit
1 = Enable Capture4 interrupt
0 = Disable Capture4 interrupt

bit 2 **CA3IE**: Capture3 Interrupt Enable bit
1 = Enable Capture3 interrupt
0 = Disable Capture3 interrupt

bit 1 **TX2IE**: USART2 Transmit Interrupt Enable bit
1 = Enable USART2 Transmit buffer empty interrupt
0 = Disable USART2 Transmit buffer empty interrupt

bit 0 **RC2IE**: USART2 Receive Interrupt Enable bit
1 = Enable USART2 Receive buffer full interrupt
0 = Disable USART2 Receive buffer full interrupt

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |


© 2000 Microchip Technology Inc.

## 6.3 Peripheral Interrupt Request Register1 (PIR1) and Register2 (PIR2)

These registers contains the individual flag bits for the peripheral interrupts.

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

**REGISTER 6-4: PIR1 REGISTER (ADDRESS: 16h, BANK 1)**

| R/W-x | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-1 | R-0 |
|---|---|---|---|---|---|---|---|
| RBIF | TMR3IF | TMR2IF | TMR1IF | CA2IF | CA1IF | TX1IF | RC1IF |
| bit 7 | | | | | | | bit 0 |

bit 7 **RBIF**: PORTB Interrupt-on-Change Flag bit
1 = One of the PORTB inputs changed (software must end the mismatch condition)
0 = None of the PORTB inputs have changed

bit 6 **TMR3IF**: TMR3 Interrupt Flag bit

If Capture1 is enabled (CA1/$\overline{PR3}$ = 1):
1 = TMR3 overflowed
0 = TMR3 did not overflow

If Capture1 is disabled (CA1/$\overline{PR3}$ = 0):
1 = TMR3 value has rolled over to 0000h from equalling the period register (PR3H:PR3L) value
0 = TMR3 value has not rolled over to 0000h from equalling the period register (PR3H:PR3L) value

bit 5 **TMR2IF**: TMR2 Interrupt Flag bit
1 = TMR2 value has rolled over to 0000h from equalling the period register (PR2) value
0 = TMR2 value has not rolled over to 0000h from equalling the period register (PR2) value

bit 4 **TMR1IF**: TMR1 Interrupt Flag bit

If TMR1 is in 8-bit mode (T16 = 0):
1 = TMR1 value has rolled over to 0000h from equalling the period register (PR1) value
0 = TMR1 value has not rolled over to 0000h from equalling the period register (PR1) value

If Timer1 is in 16-bit mode (T16 = 1):
1 = TMR2:TMR1 value has rolled over to 0000h from equalling the period register (PR2:PR1) value
0 = TMR2:TMR1 value has not rolled over to 0000h from equalling the period register (PR2:PR1) value

bit 3 **CA2IF**: Capture2 Interrupt Flag bit
1 = Capture event occurred on RB1/CAP2 pin
0 = Capture event did not occur on RB1/CAP2 pin

bit 2 **CA1IF**: Capture1 Interrupt Flag bit
1 = Capture event occurred on RB0/CAP1 pin
0 = Capture event did not occur on RB0/CAP1 pin

bit 1 **TX1IF**: USART1 Transmit Interrupt Flag bit (state controlled by hardware)
1 = USART1 Transmit buffer is empty
0 = USART1 Transmit buffer is full

bit 0 **RC1IF**: USART1 Receive Interrupt Flag bit (state controlled by hardware)
1 = USART1 Receive buffer is full
0 = USART1 Receive buffer is empty

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

**REGISTER 6-5: PIR2 REGISTER (ADDRESS: 10h, BANK 4)**

| R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R-1 | R-0 |
|---|---|---|---|---|---|---|---|
| SSPIF | BCLIF | ADIF | — | CA4IF | CA3IF | TX2IF | RC2IF |
| bit 7 | | | | | | | bit 0 |

bit 7 **SSPIF**: Synchronous Serial Port (SSP) Interrupt Flag bit
1 = The SSP interrupt condition has occurred and must be cleared in software before returning from the Interrupt Service Routine. The conditions that will set this bit are:

SPI:
A transmission/reception has taken place.

$I^2C$ Slave/Master:
A transmission/reception has taken place.

$I^2C$ Master:
The initiated START condition was completed by the SSP module.
The initiated STOP condition was completed by the SSP module.
The initiated Restart condition was completed by the SSP module.
The initiated Acknowledge condition was completed by the SSP module.
A START condition occurred while the SSP module was idle (Multi-master system).
A STOP condition occurred while the SSP module was idle (Multi-master system).

0 = An SSP interrupt condition has NOT occurred

bit 6 **BCLIF**: Bus Collision Interrupt Flag bit
1 = A bus collision has occurred in the SSP, when configured for $I^2C$ Master mode
0 = No bus collision has occurred

bit 5 **ADIF**: A/D Module Interrupt Flag bit
1 = An A/D conversion is complete
0 = An A/D conversion is not complete

bit 4 **Unimplemented**: Read as '0'

bit 3 **CA4IF**: Capture4 Interrupt Flag bit
1 = Capture event occurred on RE3/CAP4 pin
0 = Capture event did not occur on RE3/CAP4 pin

bit 2 **CA3IF**: Capture3 Interrupt Flag bit
1 = Capture event occurred on RG4/CAP3 pin
0 = Capture event did not occur on RG4/CAP3 pin

bit 1 **TX2IF**:USART2 Transmit Interrupt Flag bit (state controlled by hardware)
1 = USART2 Transmit buffer is empty
0 = USART2 Transmit buffer is full

bit 0 **RC2IF**: USART2 Receive Interrupt Flag bit (state controlled by hardware)
1 = USART2 Receive buffer is full
0 = USART2 Receive buffer is empty

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |


© 2000 Microchip Technology Inc.

## 6.4 Interrupt Operation

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

The RETFIE instruction clears the GLINTD bit while forcing the Program Counter (PC) to the value loaded at the Top-of-Stack.

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

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

The PIC17C7XX devices have four interrupt vectors. These vectors and their hardware priority are shown in Table 6-1. If two enabled interrupts occur “at the same time”, the interrupt of the highest priority will be serviced first. This means that the vector address of that interrupt will be loaded into the program counter (PC).

**TABLE 6-1: INTERRUPT VECTORS/ PRIORITIES**

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

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

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

## 6.5 RA0/INT Interrupt

The external interrupt on the RA0/INT pin is edge triggered. Either the rising edge if the INTEDG bit (T0STA<7>) is set, or the falling edge if the INTEDG bit is clear. When a valid edge appears on the RA0/INT pin, the INTF bit (INTSTA<4>) is set. This interrupt can be disabled by clearing the INTE control bit (INTSTA<0>). The INT interrupt can wake the processor from SLEEP. See Section 17.4 for details on SLEEP operation.

## 6.6 T0CKI Interrupt

The external interrupt on the RA1/T0CKI pin is edge triggered. Either the rising edge if the T0SE bit (T0STA<6>) is set, or the falling edge if the T0SE bit is clear. When a valid edge appears on the RA1/T0CKI pin, the T0CKIF bit (INTSTA<6>) is set. This interrupt can be disabled by clearing the T0CKIE control bit (INTSTA<2>). The T0CKI interrupt can wake up the processor from SLEEP. See Section 17.4 for details on SLEEP operation.

## 6.7 Peripheral Interrupt

The peripheral interrupt flag indicates that at least one of the peripheral interrupts occurred (PEIF is set). The PEIF bit is a read only bit and is a bit wise OR of all the flag bits in the PIR registers AND’d with the corresponding enable bits in the PIE registers. Some of the peripheral interrupts can wake the processor from SLEEP. See Section 17.4 for details on SLEEP operation.

## 6.8 Context Saving During Interrupts

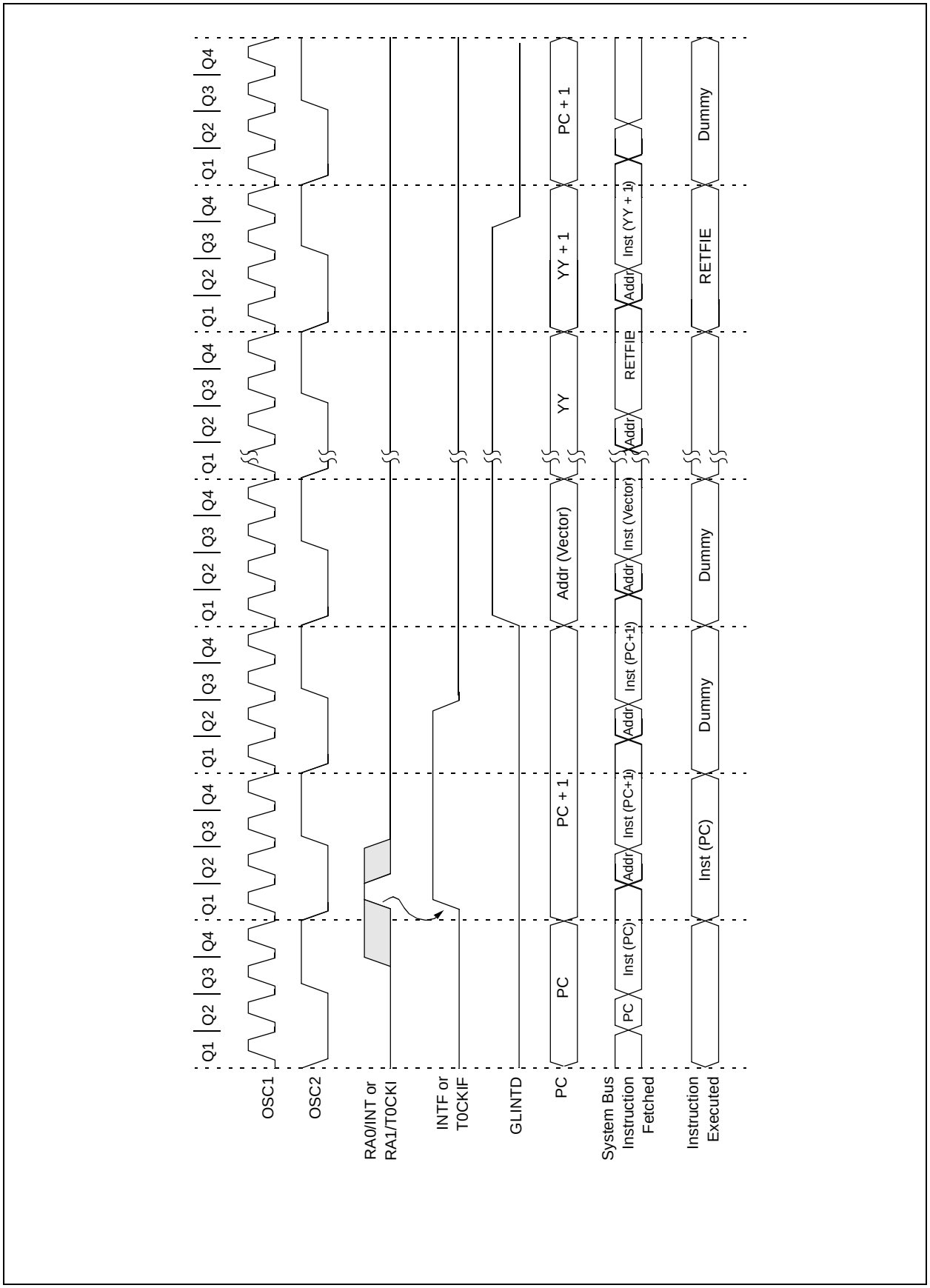
During an interrupt, only the returned PC value is saved on the stack. Typically, users may wish to save key registers during an interrupt; e.g. WREG, ALUSTA and the BSR registers. This requires implementation in software.

Example 6-2 shows the saving and restoring of information for an Interrupt Service Routine. This is for a simple interrupt scheme, where only one interrupt may occur at a time (no interrupt nesting). The SFRs are stored in the non-banked GPR area.

Example 6-2 shows the saving and restoring of information for a more complex Interrupt Service Routine. This is useful where nesting of interrupts is required. A maximum of 6 levels can be done by this example. The BSR is stored in the non-banked GPR area, while the other registers would be stored in a particular bank. Therefore, 6 saves may be done with this routine (since there are 6 non-banked GPR registers). These routines require a dedicated indirect addressing register, FSR0, to be selected for this.

The PUSH and POP code segments could either be in each Interrupt Service Routine, or could be subroutines that were called. Depending on the application, other registers may also need to be saved.

**FIGURE 6-2: INT PIN/T0CKI PIN INTERRUPT TIMING**





© 2000 Microchip Technology Inc.

**EXAMPLE 6-1: SAVING STATUS AND WREG IN RAM (SIMPLE)**

```
; The addresses that are used to store the CPUSTA and WREG values must be in the data memory
; address range of 1Ah - 1Fh. Up to 6 locations can be saved and restored using the MOVFP
; instruction. This instruction neither affects the status bits, nor corrupts the WREG register.
;
UNBANK1         EQU    0x01A       ; Address for 1st location to save
UNBANK2         EQU    0x01B       ; Address for 2nd location to save
UNBANK3         EQU    0x01C       ; Address for 3rd location to save
UNBANK4         EQU    0x01D       ; Address for 4th location to save
UNBANK5         EQU    0x01E       ; Address for 5th location to save
                                   ;    (Label Not used in program)
UNBANK6         EQU    0x01F       ; Address for 6th location to save
                                   ;    (Label Not used in program)
;
        :                          ; At Interrupt Vector Address
PUSH    MOVFP   ALUSTA, UNBANK1    ; Push ALUSTA value
        MOVFP   BSR, UNBANK2       ; Push BSR value
        MOVFP   WREG, UNBANK3      ; Push WREG value
        MOVFP   PCLATH, UNBANK4    ; Push PCLATH value
        ;
        :                          ; Interrupt Service Routine (ISR) code
        ;
POP     MOVFP   UNBANK4, PCLATH    ; Restore PCLATH value
        MOVFP   UNBANK3, WREG      ; Restore WREG value
        MOVFP   UNBANK2, BSR       ; Restore BSR value
        MOVFP   UNBANK1, ALUSTA    ; Restore ALUSTA value
;
        RETFIE                     ; Return from interrupt (enable interrupts)
```

### EXAMPLE 6-2: SAVING STATUS AND WREG IN RAM (NESTED)

```
; The addresses that are used to store the CPUSTA and WREG values must be in the data memory
; address range of 1Ah - 1Fh. Up to 6 locations can be saved and restored using the MOVFP
; instruction. This instruction neither affects the status bits, nor corrupts the WREG register.
; This routine uses the FRS0, so it controls the FS1 and FS0 bits in the ALUSTA register.
;
Nobank_FSR      EQU    0x40
Bank_FSR        EQU    0x41
ALU_Temp        EQU    0x42
WREG_TEMP       EQU    0x43
BSR_S1          EQU    0x01A        ; 1st location to save BSR
BSR_S2          EQU    0x01B        ; 2nd location to save BSR (Label Not used in program)
BSR_S3          EQU    0x01C        ; 3rd location to save BSR (Label Not used in program)
BSR_S4          EQU    0x01D        ; 4th location to save BSR (Label Not used in program)
BSR_S5          EQU    0x01E        ; 5th location to save BSR (Label Not used in program)
BSR_S6          EQU    0x01F        ; 6th location to save BSR (Label Not used in program)
;
INITIALIZATION                      ;
        CALL    CLEAR_RAM           ; Must Clear all Data RAM
;
INIT_POINTERS                       ; Must Initialize the pointers for POP and PUSH
        CLRF    BSR, F              ; Set All banks to 0
        CLRF    ALUSTA, F           ; FSR0 post increment
        BSF     ALUSTA, FS1
        CLRF    WREG, F             ; Clear WREG
        MOVLW   BSR_S1              ; Load FSR0 with 1st address to save BSR
        MOVWF   FSR0
        MOVWF   Nobank_FSR
        MOVLW   0x20
        MOVWF   Bank_FSR
        :
        :                           ; Your code
        :
        :                           ; At Interrupt Vector Address
PUSH    BSF     ALUSTA, FS0         ; FSR0 has auto-increment, does not affect status bits
        BCF     ALUSTA, FS1         ; does not affect status bits
        MOVFP   BSR, INDF0          ; No Status bits are affected
        CLRF    BSR, F              ; Peripheral and Data RAM Bank 0 No Status bits are affected
        MOVPF   ALUSTA, ALU_Temp    ;
        MOVPF   FSR0, Nobank_FSR    ; Save the FSR for BSR values
        MOVPF   WREG, WREG_TEMP     ;
        MOVFP   Bank_FSR, FSR0      ; Restore FSR value for other values
        MOVFP   ALU_Temp, INDF0     ; Push ALUSTA value
        MOVFP   WREG_TEMP, INDF0    ; Push WREG value
        MOVFP   PCLATH, INDF0       ; Push PCLATH value
        MOVPF   FSR0, Bank_FSR      ; Restore FSR value for other values
        MOVFP   Nobank_FSR, FSR0    ;
        ;
        :                           ; Interrupt Service Routine (ISR) code
        ;
POP     CLRF    ALUSTA, F           ; FSR0 has auto-decrement, does not affect status bits
        MOVFP   Bank_FSR, FSR0      ; Restore FSR value for other values
        DECF    FSR0, F             ;
        MOVFP   INDF0, PCLATH       ; Pop PCLATH value
        MOVFP   INDF0, WREG         ; Pop WREG value
        BSF     ALUSTA, FS1         ; FSR0 does not change
        MOVPF   INDF0, ALU_Temp     ; Pop ALUSTA value
        MOVPF   FSR0, Bank_FSR      ; Restore FSR value for other values
        DECF    Nobank_FSR, F       ;
        MOVFP   Nobank_FSR, FSR0    ; Save the FSR for BSR values
        MOVFP   ALU_Temp, ALUSTA    ;
        MOVFP   INDF0, BSR          ; No Status bits are affected
;
        RETFIE                      ; Return from interrupt (enable interrupts)
```


© 2000 Microchip Technology Inc.

## 7.0 MEMORY ORGANIZATION

There are two memory blocks in the PIC17C7XX; program memory and data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle.

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

### 7.1 Program Memory Organization

PIC17C7XX devices have a 16-bit program counter capable of addressing a 64K x 16 program memory space. The RESET vector is at 0000h and the interrupt vectors are at 0008h, 0010h, 0018h, and 0020h (Figure 7-1).

#### 7.1.1 PROGRAM MEMORY OPERATION

The PIC17C7XX can operate in one of four possible program memory configurations. The configuration is selected by configuration bits. The possible modes are:

- Microprocessor
- Microcontroller
- Extended Microcontroller
- Protected Microcontroller

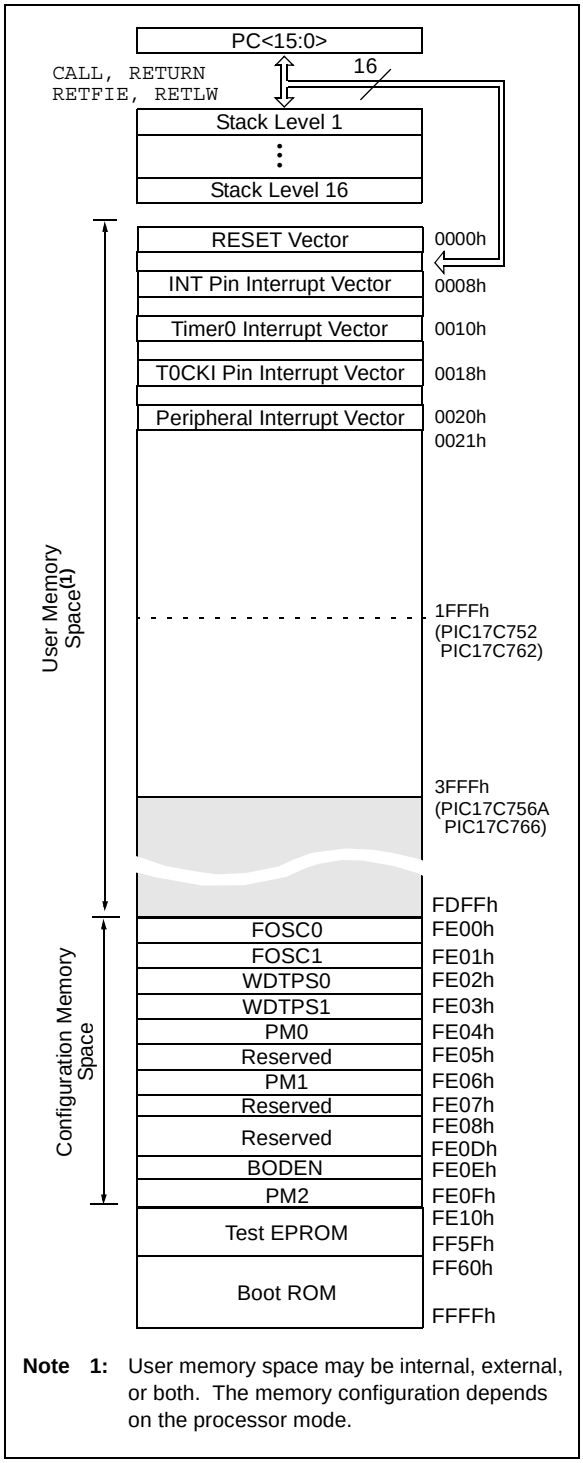
The **Microcontroller** and **Protected Microcontroller** modes only allow internal execution. Any access beyond the program memory reads unknown data. The Protected Microcontroller mode also enables the code protection feature.

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

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

The different modes allow different access to the configuration bits, test memory and boot ROM. Table 7-1 lists which modes can access which areas in memory. Test Memory and Boot Memory are not required for normal operation of the device. Care should be taken to ensure that no unintended branches occur to these areas.

**FIGURE 7-1: PROGRAM MEMORY MAP AND STACK**




**Note 1:** User memory space may be internal, external, or both. The memory configuration depends on the processor mode.

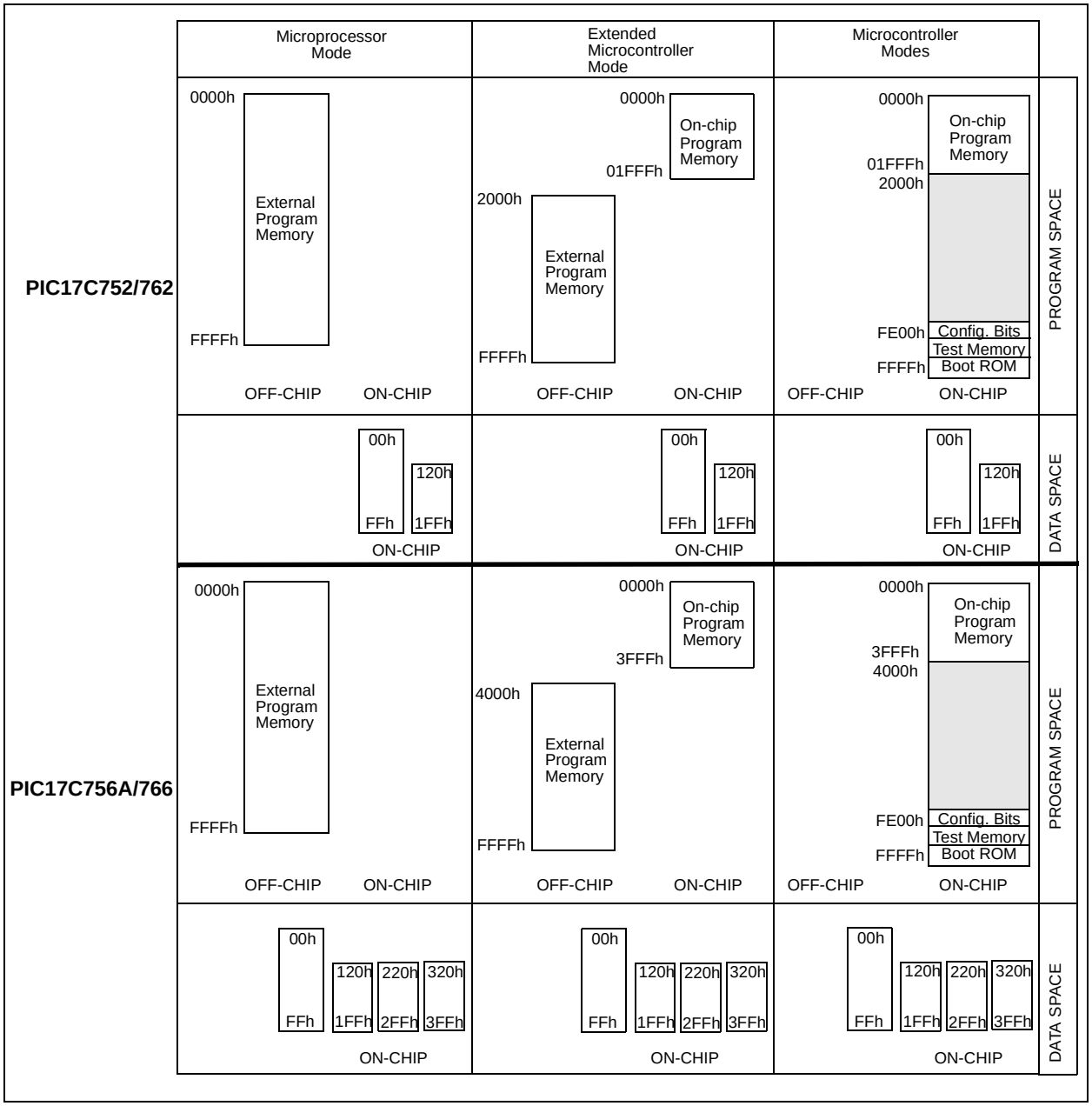
**TABLE 7-1: MODE MEMORY ACCESS**

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

The PIC17C7XX can operate in modes where the program memory is off-chip. They are the Microprocessor and Extended Microcontroller modes. The Microprocessor mode is the default for an unprogrammed device.

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

**FIGURE 7-2: MEMORY MAP IN DIFFERENT MODES**



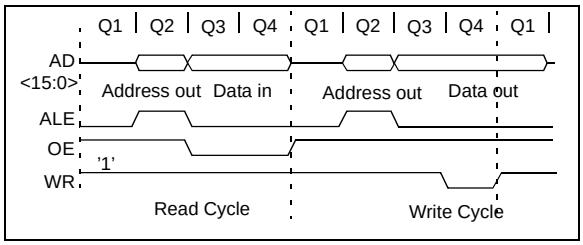


© 2000 Microchip Technology Inc.

### 7.1.2 EXTERNAL MEMORY INTERFACE

When either Microprocessor or Extended Microcontroller mode is selected, PORTC, PORTD and PORTE are configured as the system bus. PORTC and PORTD are the multiplexed address/data bus and PORTE<2:0> is for the control signals. External components are needed to demultiplex the address and data. This can be done as shown in Figure 7-4. The waveforms of address and data are shown in Figure 7-3. For complete timings, please refer to the electrical specification section.

**FIGURE 7-3: EXTERNAL PROGRAM MEMORY ACCESS WAVEFORMS**




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

As the speed of the processor increases, external EPROM memory with faster access time must be used. Table 7-2 lists external memory speed requirements for a given PIC17C7XX device frequency.

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

The following selection is for use with Microchip EPROMs. For interfacing to other manufacturers memory, please refer to the electrical specifications of the desired PIC17C7XX device, as well as the desired memory device to ensure compatibility.

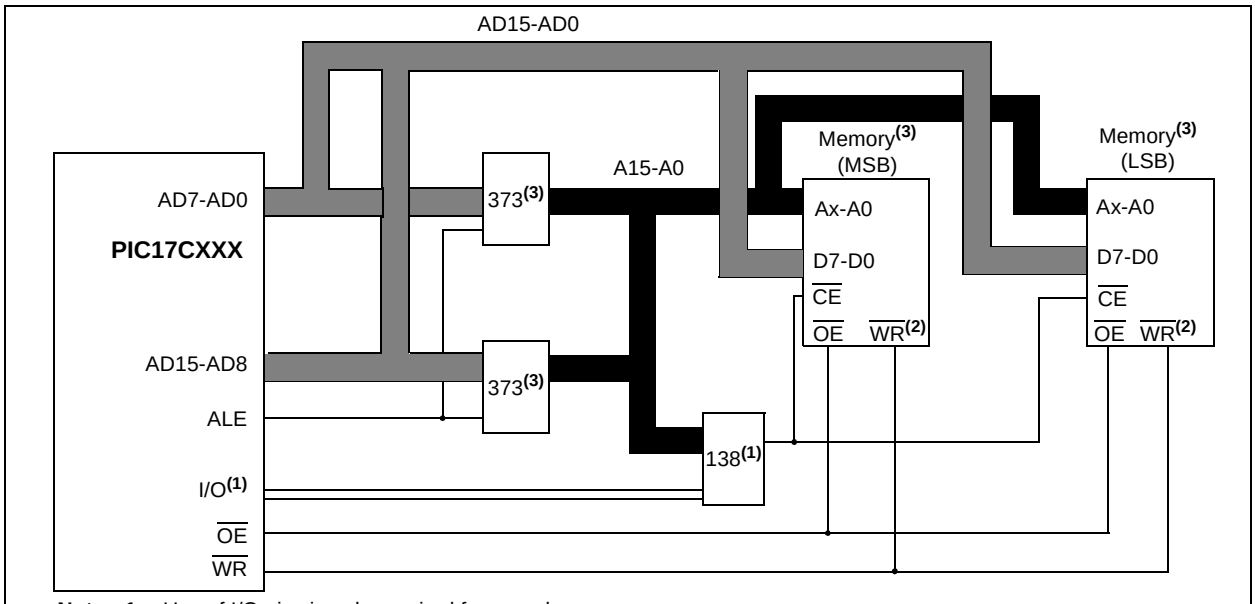
**TABLE 7-2: EPROM MEMORY ACCESS TIME ORDERING SUFFIX**

| PIC17C7XX Oscillator Frequency | Instruction Cycle Time ($T_{CY}$) | EPROM Suffix |
|---|---|---|
| 8 MHz | 500 ns | -25 |
| 16 MHz | 250 ns | -15 |
| 20 MHz | 200 ns | -10 |
| 25 MHz | 160 ns | -70 |

**Note:** The access times for this requires the use of fast SRAMs.

The electrical specifications now include timing specifications for the memory interface with PIC17**LC**XXX devices. These specifications reflect the capability of the device by characterization. Please validate your design with these timings.

**FIGURE 7-4: TYPICAL EXTERNAL PROGRAM MEMORY CONNECTION DIAGRAM**




**Note 1:** Use of I/O pins is only required for paged memory.
**2:** This signal is unused for ROM and EPROM devices.
**3:** 16-bit wide devices are now common and could be used instead of 8-bit wide devices.

## 7.2 Data Memory Organization

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

Portions of data memory are banked, this occurs in both areas. The GPR area is banked to allow greater than 232 bytes of general purpose RAM.

Banking requires the use of control bits for bank selection. These control bits are located in the Bank Select Register (BSR). If an access is made to the unbanked region, the BSR bits are ignored. Figure 7-5 shows the data memory map organization.

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

The entire data memory can be accessed either directly, or indirectly (through file select registers FSR0 and FSR1) (see Section 7.4). Indirect addressing uses the appropriate control bits of the BSR for access into the banked areas of data memory. The BSR is explained in greater detail in Section 7.8.

### 7.2.1 GENERAL PURPOSE REGISTER (GPR)

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

All the PIC17C7XX devices have banked memory in the GPR area. To facilitate switching between these banks, the `MOVLR bank` instruction has been added to the instruction set. GPRs are not initialized by a Power-on Reset and are unchanged on all other RESETS.

### 7.2.2 SPECIAL FUNCTION REGISTERS (SFR)

The SFRs are used by the CPU and peripheral functions to control the operation of the device (Figure 7-5). These registers are static RAM.

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

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


© 2000 Microchip Technology Inc.

**FIGURE 7-5: PIC17C7XX REGISTER FILE MAP**

| Addr | Unbanked |
|---|---|
| 00h | INDF0 |
| 01h | FSR0 |
| 02h | PCL |
| 03h | PCLATH |
| 04h | ALUSTA |
| 05h | T0STA |
| 06h | CPUSTA |
| 07h | INTSTA |
| 08h | INDF1 |
| 09h | FSR1 |
| 0Ah | WREG |
| 0Bh | TMR0L |
| 0Ch | TMR0H |
| 0Dh | TBLPTRL |
| 0Eh | TBLPTRH |
| 0Fh | BSR |

| | Bank 0 | Bank 1(1) | Bank 2(1) | Bank 3(1) | Bank 4(1) | Bank 5(1) | Bank 6(1) | Bank 7(1) | Bank 8(1,4) |
|---|---|---|---|---|---|---|---|---|---|
| 10h | PORTA | DDRC | TMR1 | PW1DCL | PIR2 | DDRF | SSPADD | PW3DCL | DDRH |
| 11h | DDRB | PORTC | TMR2 | PW2DCL | PIE2 | PORTF | SSPCON1 | PW3DCH | PORTH |
| 12h | PORTB | DDRD | TMR3L | PW1DCH | — | DDRG | SSPCON2 | CA3L | DDRJ |
| 13h | RCSTA1 | PORTD | TMR3H | PW2DCH | RCSTA2 | PORTG | SSPSTAT | CA3H | PORTJ |
| 14h | RCREG1 | DDRE | PR1 | CA2L | RCREG2 | ADCON0 | SSPBUF | CA4L | — |
| 15h | TXSTA1 | PORTE | PR2 | CA2H | TXSTA2 | ADCON1 | — | CA4H | — |
| 16h | TXREG1 | PIR1 | PR3L/CA1L | TCON1 | TXREG2 | ADRESL | — | TCON3 | — |
| 17h | SPBRG1 | PIE1 | PR3H/CA1H | TCON2 | SPBRG2 | ADRESH | — | — | — |

| | Unbanked |
|---|---|
| 18h | PRODL |
| 19h | PRODH |
| 1Ah<br><br>1Fh | General Purpose RAM |

| | Bank 0(2) | Bank 1(2) | Bank 2(2) | Bank 3(2,3) |
|---|---|---|---|---|
| 20h<br><br><br><br><br><br>FFh | General Purpose RAM | General Purpose RAM | General Purpose RAM | General Purpose RAM |

**Note 1:** SFR file locations 10h - 17h are banked. The lower nibble of the BSR specifies the bank. All unbanked SFRs ignore the Bank Select Register (BSR) bits.

**2:** General Purpose Registers (GPR) locations 20h - FFh, 120h - 1FFh, 220h - 2FFh, and 320h - 3FFh are banked. The upper nibble of the BSR specifies this bank. All other GPRs ignore the Bank Select Register (BSR) bits.

**3:** RAM bank 3 is not implemented on the PIC17C752 and the PIC17C762. Reading any unimplemented register reads '0's.

**4:** Bank 8 is only implemented on the PIC17C76X devices.

**TABLE 7-3: SPECIAL FUNCTION REGISTERS**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Unbanked** | | | | | | | | | | | |
| 00h | INDF0 | Uses contents of FSR0 to address Data Memory (not a physical register) | | | | | | | | ---- ---- | ---- ---- |
| 01h | FSR0 | Indirect Data Memory Address Pointer 0 | | | | | | | | xxxx xxxx | uuuu uuuu |
| 02h | PCL | Low order 8-bits of PC | | | | | | | | 0000 0000 | 0000 0000 |
| 03h(1) | PCLATH | Holding Register for upper 8-bits of PC | | | | | | | | 0000 0000 | uuuu uuuu |
| 04h | ALUSTA | FS3 | FS2 | FS1 | FS0 | OV | Z | DC | C | 1111 xxxx | 1111 uuuu |
| 05h | T0STA | INTEDG | T0SE | T0CS | T0PS3 | T0PS2 | T0PS1 | T0PS0 | — | 0000 000- | 0000 000- |
| 06h(2) | CPUSTA | — | — | STKAV | GLINTD | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | --11 11qq | --11 qquu |
| 07h | INTSTA | PEIF | T0CKIF | T0IF | INTF | PEIE | T0CKIE | T0IE | INTE | 0000 0000 | 0000 0000 |
| 08h | INDF1 | Uses contents of FSR1 to address Data Memory (not a physical register) | | | | | | | | ---- ---- | ---- ---- |
| 09h | FSR1 | Indirect Data Memory Address Pointer 1 | | | | | | | | xxxx xxxx | uuuu uuuu |
| 0Ah | WREG | Working Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 0Bh | TMR0L | TMR0 Register; Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 0Ch | TMR0H | TMR0 Register; High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 0Dh | TBLPTRL | Low Byte of Program Memory Table Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 0Eh | TBLPTRH | High Byte of Program Memory Table Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 0Fh | BSR | Bank Select Register | | | | | | | | 0000 0000 | 0000 0000 |
| **Bank 0** | | | | | | | | | | | |
| 10h | PORTA(4,6) | RBPU | — | RA5/TX1/ CK1 | RA4/RX1/ DT1 | RA3/SDI/ SDA | RA2/$\overline{\text{SS}}$/ SCL | RA1/T0CKI | RA0/INT | 0-xx 11xx | 0-uu 11uu |
| 11h | DDRB | Data Direction Register for PORTB | | | | | | | | 1111 1111 | 1111 1111 |
| 12h | PORTB(4) | RB7/ SDO | RB6/ SCK | RB5/ TCLK3 | RB4/ TCLK12 | RB3/ PWM2 | RB2/ PWM1 | RB1/ CAP2 | RB0/ CAP1 | xxxx xxxx | uuuu uuuu |
| 13h | RCSTA1 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 14h | RCREG1 | Serial Port Receive Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 15h | TXSTA1 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 16h | TXREG1 | Serial Port Transmit Register (for USART1) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 17h | SPBRG1 | Baud Rate Generator Register (for USART1) | | | | | | | | 0000 0000 | 0000 0000 |
| **Bank 1** | | | | | | | | | | | |
| 10h | DDRC(5) | Data Direction Register for PORTC | | | | | | | | 1111 1111 | 1111 1111 |
| 11h | PORTC(4,5) | RC7/AD7 | RC6/AD6 | RC5/AD5 | RC4/AD4 | RC3/AD3 | RC2/AD2 | RC1/AD1 | RC0/AD0 | xxxx xxxx | uuuu uuuu |
| 12h | DDRD(5) | Data Direction Register for PORTD | | | | | | | | 1111 1111 | 1111 1111 |
| 13h | PORTD(4,5) | RD7/ AD15 | RD6/ AD14 | RD5/ AD13 | RD4/ AD12 | RD3/ AD11 | RD2/ AD10 | RD1/AD9 | RD0/AD8 | xxxx xxxx | uuuu uuuu |
| 14h | DDRE(5) | Data Direction Register for PORTE | | | | | | | | ---- 1111 | ---- 1111 |
| 15h | PORTE(4,5) | — | — | — | — | RE3/ CAP4 | RE2/$\overline{\text{WR}}$ | RE1/$\overline{\text{OE}}$ | RE0/ALE | ---- xxxx | ---- uuuu |
| 16h | PIR1 | RBIF | TMR3IF | TMR2IF | TMR1IF | CA2IF | CA1IF | TX1IF | RC1IF | x000 0010 | u000 0010 |
| 17h | PIE1 | RBIE | TMR3IE | TMR2IE | TMR1IE | CA2IE | CA1IE | TX1IE | RC1IE | 0000 0000 | 0000 0000 |

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

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

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

**3:** Bank 8 and associated registers are only implemented on the PIC17C76X devices.

**4:** This is the value that will be in the port output latch.

**5:** When the device is configured for Microprocessor or Extended Microcontroller mode, the operation of this port does not rely on these registers.

**6:** On any device RESET, these pins are configured as inputs.


© 2000 Microchip Technology Inc.

**TABLE 7-3: SPECIAL FUNCTION REGISTERS (CONTINUED)**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bank 2** | | | | | | | | | | | |
| 10h | TMR1 | Timer1's Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 11h | TMR2 | Timer2's Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 12h | TMR3L | Timer3's Register; Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 13h | TMR3H | Timer3's Register; High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 14h | PR1 | Timer1's Period Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 15h | PR2 | Timer2's Period Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 16h | PR3L/CA1L | Timer3's Period Register - Low Byte/Capture1 Register; Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 17h | PR3H/CA1H | Timer3's Period Register - High Byte/Capture1 Register; High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| **Bank 3** | | | | | | | | | | | |
| 10h | PW1DCL | DC1 | DC0 | — | — | — | — | — | — | xx-- ---- | uu-- ---- |
| 11h | PW2DCL | DC1 | DC0 | TM2PW2 | — | — | — | — | — | xx0- ---- | uu0- ---- |
| 12h | PW1DCH | DC9 | DC8 | DC7 | DC6 | DC5 | DC4 | DC3 | DC2 | xxxx xxxx | uuuu uuuu |
| 13h | PW2DCH | DC9 | DC8 | DC7 | DC6 | DC5 | DC4 | DC3 | DC2 | xxxx xxxx | uuuu uuuu |
| 14h | CA2L | Capture2 Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 15h | CA2H | Capture2 High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 16h | TCON1 | CA2ED1 | CA2ED0 | CA1ED1 | CA1ED0 | T16 | TMR3CS | TMR2CS | TMR1CS | 0000 0000 | 0000 0000 |
| 17h | TCON2 | CA2OVF | CA1OVF | PWM2ON | PWM1ON | CA1/$\overline{\text{PR3}}$ | TMR3ON | TMR2ON | TMR1ON | 0000 0000 | 0000 0000 |
| **Bank 4** | | | | | | | | | | | |
| 10h | PIR2 | SSPIF | BCLIF | ADIF | — | CA4IF | CA3IF | TX2IF | RC2IF | 000- 0010 | 000- 0010 |
| 11h | PIE2 | SSPIE | BCLIE | ADIE | — | CA4IE | CA3IE | TX2IE | RC2IE | 000- 0000 | 000- 0000 |
| 12h | Unimplemented | — | — | — | — | — | — | — | — | ---- ---- | ---- ---- |
| 13h | RCSTA2 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 14h | RCREG2 | Serial Port Receive Register for USART2 | | | | | | | | xxxx xxxx | uuuu uuuu |
| 15h | TXSTA2 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 16h | TXREG2 | Serial Port Transmit Register for USART2 | | | | | | | | xxxx xxxx | uuuu uuuu |
| 17h | SPBRG2 | Baud Rate Generator for USART2 | | | | | | | | 0000 0000 | 0000 0000 |
| **Bank 5:** | | | | | | | | | | | |
| 10h | DDRF | Data Direction Register for PORTF | | | | | | | | 1111 1111 | 1111 1111 |
| 11h | PORTF(4) | RF7/ AN11 | RF6/ AN10 | RF5/ AN9 | RF4/ AN8 | RF3/ AN7 | RF2/ AN6 | RF1/ AN5 | RF0/ AN4 | 0000 0000 | 0000 0000 |
| 12h | DDRG | Data Direction Register for PORTG | | | | | | | | 1111 1111 | 1111 1111 |
| 13h | PORTG(4) | RG7/ TX2/CK2 | RG6/ RX2/DT2 | RG5/ PWM3 | RG4/ CAP3 | RG3/ AN0 | RG2/ AN1 | RG1/ AN2 | RG0/ AN3 | xxxx 0000 | uuuu 0000 |
| 14h | ADCON0 | CHS3 | CHS2 | CHS1 | CHS0 | — | GO/$\overline{\text{DONE}}$ | — | ADON | 0000 -0-0 | 0000 -0-0 |
| 15h | ADCON1 | ADCS1 | ADCS0 | ADFM | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 000- 0000 | 000- 0000 |
| 16h | ADRESL | A/D Result Register Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 17h | ADRESH | A/D Result Register High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |

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

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

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

**3:** Bank 8 and associated registers are only implemented on the PIC17C76X devices.

**4:** This is the value that will be in the port output latch.

**5:** When the device is configured for Microprocessor or Extended Microcontroller mode, the operation of this port does not rely on these registers.

**6:** On any device RESET, these pins are configured as inputs.

**TABLE 7-3: SPECIAL FUNCTION REGISTERS (CONTINUED)**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{MCLR}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bank 6** | | | | | | | | | | | |
| 10h | SSPADD | SSP Address Register in $I^2C$ Slave mode. SSP Baud Rate Reload Register in $I^2C$ Master mode | | | | | | | | 0000 0000 | 0000 0000 |
| 11h | SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 0000 0000 | 0000 0000 |
| 12h | SSPCON2 | GCEN | AKSTAT | AKDT | AKEN | RCEN | PEN | RSEN | SEN | 0000 0000 | 0000 0000 |
| 13h | SSPSTAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 0000 0000 | 0000 0000 |
| 14h | SSPBUF | Synchronous Serial Port Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 15h | Unimplemented | — | — | — | — | — | — | — | — | ---- ---- | ---- ---- |
| 16h | Unimplemented | — | — | — | — | — | — | — | — | ---- ---- | ---- ---- |
| 17h | Unimplemented | — | — | — | — | — | — | — | — | ---- ---- | ---- ---- |
| **Bank 7** | | | | | | | | | | | |
| 10h | PW3DCL | DC1 | DC0 | TM2PW3 | — | — | — | — | — | xx0- ---- | uu0- ---- |
| 11h | PW3DCH | DC9 | DC8 | DC7 | DC6 | DC5 | DC4 | DC3 | DC2 | xxxx xxxx | uuuu uuuu |
| 12h | CA3L | Capture3 Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 13h | CA3H | Capture3 High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 14h | CA4L | Capture4 Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 15h | CA4H | Capture4 High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 16h | TCON3 | — | CA4OVF | CA3OVF | CA4ED1 | CA4ED0 | CA3ED1 | CA3ED0 | PWM3ON | -000 0000 | -000 0000 |
| 17h | Unimplemented | — | — | — | — | — | — | — | — | ---- ---- | ---- ---- |
| **Bank 8(3)** | | | | | | | | | | | |
| 10h(3) | DDRH | Data Direction Register for PORTH | | | | | | | | 1111 1111 | 1111 1111 |
| 11h(3) | PORTH(4) | RH7/ AN15 | RH6/ AN14 | RH5/ AN13 | RH4/ AN12 | RH3 | RH2 | RH1 | RH0 | xxxx xxxx | uuuu uuuu |
| 12h(3) | DDRJ | Data Direction Register for PORTJ | | | | | | | | 1111 1111 | 1111 1111 |
| 13h(3) | PORTJ(4) | RJ7 | RJ6 | RJ5 | RJ4 | RJ3 | RJ2 | RJ1 | RJ0 | xxxx xxxx | uuuu uuuu |
| 14h(3) | Unimplemented | — | — | — | — | — | — | — | — | ---- ---- | ---- ---- |
| 15h(3) | Unimplemented | — | — | — | — | — | — | — | — | ---- ---- | ---- ---- |
| 16h(3) | Unimplemented | — | — | — | — | — | — | — | — | ---- ---- | ---- ---- |
| 17h(3) | Unimplemented | — | — | — | — | — | — | — | — | ---- ---- | ---- ---- |
| **Unbanked** | | | | | | | | | | | |
| 18h | PRODL | Low Byte of 16-bit Product (8 x 8 Hardware Multiply) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 19h | PRODH | High Byte of 16-bit Product (8 x 8 Hardware Multiply) | | | | | | | | xxxx xxxx | uuuu uuuu |

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

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

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

**3:** Bank 8 and associated registers are only implemented on the PIC17C76X devices.

**4:** This is the value that will be in the port output latch.

**5:** When the device is configured for Microprocessor or Extended Microcontroller mode, the operation of this port does not rely on these registers.

**6:** On any device RESET, these pins are configured as inputs.


© 2000 Microchip Technology Inc.

### 7.2.2.1 ALU Status Register (ALUSTA)

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

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

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

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

**Note 1:** The C and DC bits operate as a $\overline{\text{borrow}}$ and $\overline{\text{digit borrow}}$ bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

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

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

**REGISTER 7-1: ALUSTA REGISTER (ADDRESS: 04h, UNBANKED)**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| FS3 | FS2 | FS1 | FS0 | OV | Z | DC | C |
| bit 7 | | | | | | | bit 0 |

bit 7-6 **FS3:FS2**: FSR1 Mode Select bits
`00` = Post auto-decrement FSR1 value
`01` = Post auto-increment FSR1 value
`1x` = FSR1 value does not change

bit 5-4 **FS1:FS0**: FSR0 Mode Select bits
`00` = Post auto-decrement FSR0 value
`01` = Post auto-increment FSR0 value
`1x` = FSR0 value does not change

bit 3 **OV**: Overflow bit
This bit is used for signed arithmetic (2's complement). It indicates an overflow of the 7-bit magnitude, which causes the sign bit (bit7) to change state.
`1` = Overflow occurred for signed arithmetic (in this arithmetic operation)
`0` = No overflow occurred

bit 2 **Z**: Zero bit
`1` = The result of an arithmetic or logic operation is zero
`0` = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{\text{borrow}}$ bit
For `ADDWF` and `ADDLW` instructions.
`1` = A carry-out from the 4th low order bit of the result occurred
`0` = No carry-out from the 4th low order bit of the result

**Note:** For borrow, the polarity is reversed.

bit 0 **C**: Carry/$\overline{\text{borrow}}$ bit

For `ADDWF` and `ADDLW` instructions. Note that a subtraction is executed by adding the two's complement of the second operand.
For rotate (`RRCF`, `RLCF`) instructions, this bit is loaded with either the high or low order bit of the source register.
`1` = A carry-out from the Most Significant bit of the result occurred
`0` = No carry-out from the Most Significant bit of the result

**Note:** For borrow, the polarity is reversed.

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

#### 7.2.2.2 CPU Status Register (CPUSTA)

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

The $\overline{POR}$ bit allows the differentiation between a Power-on Reset, external $\overline{MCLR}$ Reset, or a WDT Reset. The $\overline{BOR}$ bit indicates if a Brown-out Reset occurred.

| | |
|---|---|
| **Note 1:** | The $\overline{BOR}$ status bit is a don't care and is not necessarily predictable if the Brown-out circuit is disabled (when the BODEN bit in the Configuration word is programmed). |

**REGISTER 7-2: CPUSTA REGISTER (ADDRESS: 06h, UNBANKED)**

| U-0 | U-0 | R-1 | R/W-1 | R-1 | R-1 | R/W-0 | R/W-1 |
|---|---|---|---|---|---|---|---|
| — | — | STKAV | GLINTD | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| bit 7 | | | | | | | bit 0 |

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

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

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

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

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

bit 1 **$\overline{POR}$**: Power-on Reset Status bit
1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set by software)

bit 0 **$\overline{BOR}$**: Brown-out Reset Status bit

When BODEN Configuration bit is set (enabled):
1 = No Brown-out Reset occurred
0 = A Brown-out Reset occurred (must be set by software)

When BODEN Configuration bit is clear (disabled):
Don't care

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |


© 2000 Microchip Technology Inc.

#### 7.2.2.3 TMR0 Status/Control Register (T0STA)

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

**REGISTER 7-3: T0STA REGISTER (ADDRESS: 05h, UNBANKED)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 |
|---|---|---|---|---|---|---|---|
| INTEDG | T0SE | T0CS | T0PS3 | T0PS2 | T0PS1 | T0PS0 | — |
| bit 7 | | | | | | | bit 0 |

bit 7 **INTEDG**: RA0/INT Pin Interrupt Edge Select bit
This bit selects the edge upon which the interrupt is detected.
1 = Rising edge of RA0/INT pin generates interrupt
0 = Falling edge of RA0/INT pin generates interrupt

bit 6 **T0SE**: Timer0 External Clock Input Edge Select bit
This bit selects the edge upon which TMR0 will increment.

When T0CS = 0 (External Clock):
1 = Rising edge of RA1/T0CKI pin increments TMR0 and/or sets the T0CKIF bit
0 = Falling edge of RA1/T0CKI pin increments TMR0 and/or sets a T0CKIF bit

When T0CS = 1 (Internal Clock):
Don't care

bit 5 **T0CS**: Timer0 Clock Source Select bit
This bit selects the clock source for Timer0.
1 = Internal instruction clock cycle ($T_{CY}$)
0 = External clock input on the T0CKI pin

bit 4-1 **T0PS3:T0PS0**: Timer0 Prescale Selection bits
These bits select the prescale value for Timer0.

| T0PS3:T0PS0 | Prescale Value |
|---|---|
| 0000 | 1:1 |
| 0001 | 1:2 |
| 0010 | 1:4 |
| 0011 | 1:8 |
| 0100 | 1:16 |
| 0101 | 1:32 |
| 0110 | 1:64 |
| 0111 | 1:128 |
| 1xxx | 1:256 |

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

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

## 7.3 Stack Operation

PIC17C7XX devices have a 16 x 16-bit hardware stack (Figure 7-1). The stack is not part of either the program or data memory space, and the stack pointer is neither readable nor writable. The PC (Program Counter) is "PUSH'd" onto the stack when a CALL or LCALL instruction is executed, or an interrupt is acknowledged. The stack is "POP'd" in the event of a RETURN, RETLW, or a RETFIE instruction execution. PCLATH is not affected by a "PUSH" or a "POP" operation.

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

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

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

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

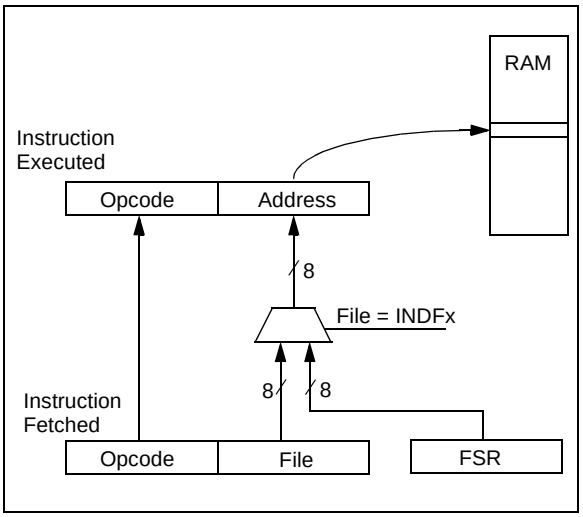
After the device is "PUSH'd" sixteen times (without a "POP"), the seventeenth push overwrites the value from the first push. The eighteenth push overwrites the second push (and so on).

## 7.4 Indirect Addressing

Indirect addressing is a mode of addressing data memory where the data memory address in the instruction is not fixed. That is, the register that is to be read or written can be modified by the program. This can be useful for data tables in the data memory. Figure 7-6 shows the operation of indirect addressing. This depicts the moving of the value to the data memory address specified by the value of the FSR register.

Example 7-1 shows the use of indirect addressing to clear RAM in a minimum number of instructions. A similar concept could be used to move a defined number of bytes (block) of data to the USART transmit register (TXREG). The starting address of the block of data to be transmitted could easily be modified by the program.

**FIGURE 7-6: INDIRECT ADDRESSING**




### 7.4.1 INDIRECT ADDRESSING REGISTERS

The PIC17C7XX has four registers for indirect addressing. These registers are:

- INDF0 and FSR0
- INDF1 and FSR1

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

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


© 2000 Microchip Technology Inc.

### 7.4.2 INDIRECT ADDRESSING OPERATION

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

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

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

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

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

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

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

A simple program to clear RAM from 20h - FFh is shown in Example 7-1.

**EXAMPLE 7-1: INDIRECT ADDRESSING**

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

## 7.5 Table Pointer (TBLPTRL and TBLPTRH)

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

The TABLRD and the TABLWT instructions allow transfer of data between program and data space. The table pointer serves as the 16-bit address of the data word within the program memory. For a more complete description of these registers and the operation of Table Reads and Table Writes, see Section 8.0.

## 7.6 Table Latch (TBLATH, TBLATL)

The table latch (TBLAT) is a 16-bit register, with TBLATH and TBLATL referring to the high and low bytes of the register. It is not mapped into data or program memory. The table latch is used as a temporary holding latch during data transfer between program and data memory (see TABLRD, TABLWT, TLRD and TLWT instruction descriptions). For a more complete description of these registers and the operation of Table Reads and Table Writes, see Section 8.0.

## 7.7 Program Counter Module

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

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

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

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

Figure 7-7 and Figure 7-8 show the operation of the program counter for various situations.

**FIGURE 7-7: PROGRAM COUNTER OPERATION**

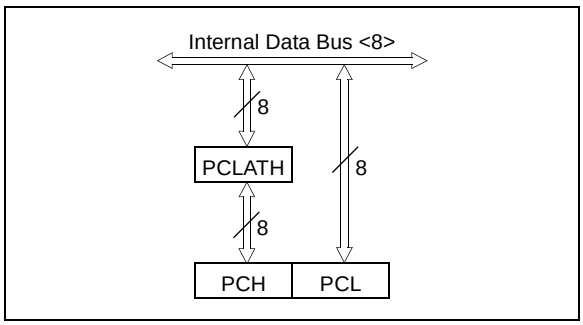
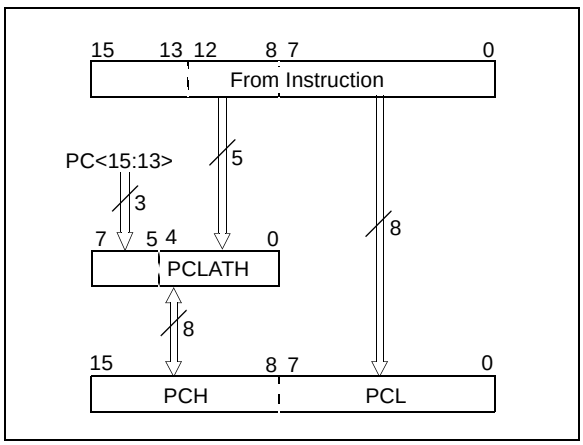



**FIGURE 7-8: PROGRAM COUNTER USING THE CALL AND GOTO INSTRUCTIONS**




Using Figure 7-7, the operations of the PC and PCLATH for different instructions are as follows:

a) LCALL instructions:
   An 8-bit destination address is provided in the instruction (opcode). PCLATH is unchanged.
   PCLATH → PCH
   Opcode<7:0> → PCL

b) Read instructions on PCL:
   Any instruction that reads PCL.
   PCL → data bus → ALU or destination
   PCH → PCLATH

c) Write instructions on PCL:
   Any instruction that writes to PCL.
   8-bit data → data bus → PCL
   PCLATH → PCH

d) Read-Modify-Write instructions on PCL:
   Any instruction that does a read-write-modify operation on PCL, such as ADDWF PCL.
   Read: PCL → data bus → ALU
   Write: 8-bit result → data bus → PCL
   PCLATH → PCH

e) RETURN instruction:
   Stack<MRU> → PC<15:0>

Using Figure 7-8, the operation of the PC and PCLATH for GOTO and CALL instructions is as follows:

CALL, GOTO instructions:

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

Opcode<12:0> → PC<12:0>

PC<15:13> → PCLATH<7:5>

Opcode<12:8> → PCLATH<4:0>

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

The following PC related operations do not change PCLATH:

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


© 2000 Microchip Technology Inc.

## 7.8 Bank Select Register (BSR)

The BSR is used to switch between banks in the data memory area (Figure 7-9). In the PIC17C7XX devices, the entire byte is implemented. The lower nibble is used to select the peripheral register bank. The upper nibble is used to select the general purpose memory bank.

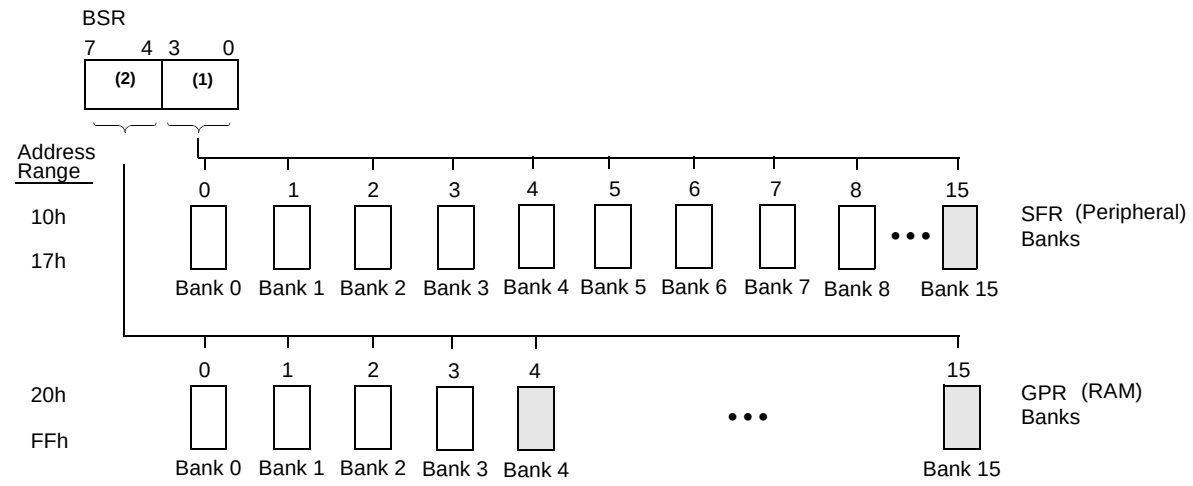
All the Special Function Registers (SFRs) are mapped into the data memory space. In order to accommodate the large number of registers, a banking scheme has been used. A segment of the SFRs, from address 10h to address 17h, is banked. The lower nibble of the bank select register (BSR) selects the currently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

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

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

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

**FIGURE 7-9: BSR OPERATION**




**Note 1:** For the SFRs only Banks 0 through 8 are implemented. Selection of an unimplemented bank is not recommended. **Bank 15 is reserved for Microchip use, reading of registers in this bank may cause random values to be read.**

**2:** For the GPRs, Bank 3 is unimplemented on the PIC17C752 and the PIC17C762. Selection of an unimplemented bank is not recommended.

**3:** SFR Bank 8 is only implemented on the PIC17C76X.

**NOTES:**


© 2000 Microchip Technology Inc.

## 8.0 TABLE READS AND TABLE WRITES

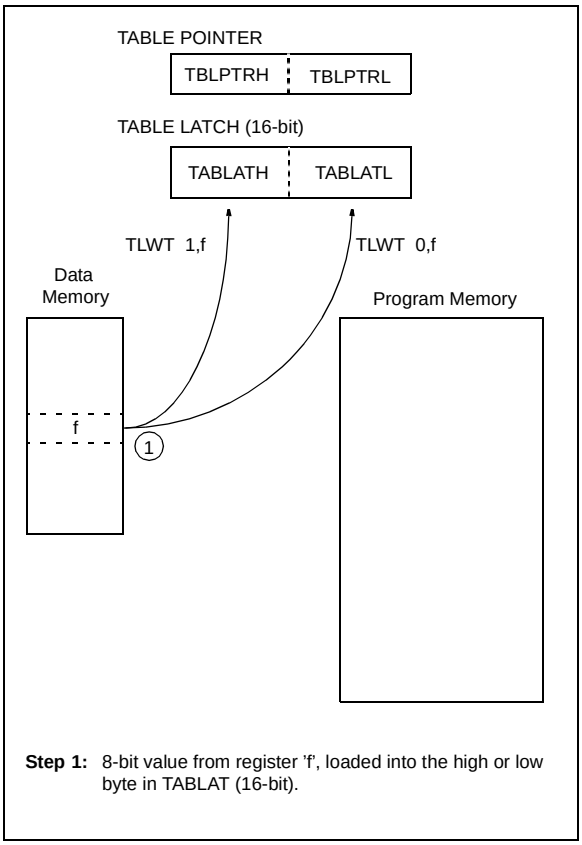
The PIC17C7XX has four instructions that allow the processor to move data from the data memory space to the program memory space, and vice versa. Since the program memory space is 16-bits wide and the data memory space is 8-bits wide, two operations are required to move 16-bit values to/from the data memory.

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

The program memory can be internal or external. For the program memory access to be external, the device needs to be operating in Microprocessor or Extended Microcontroller mode.

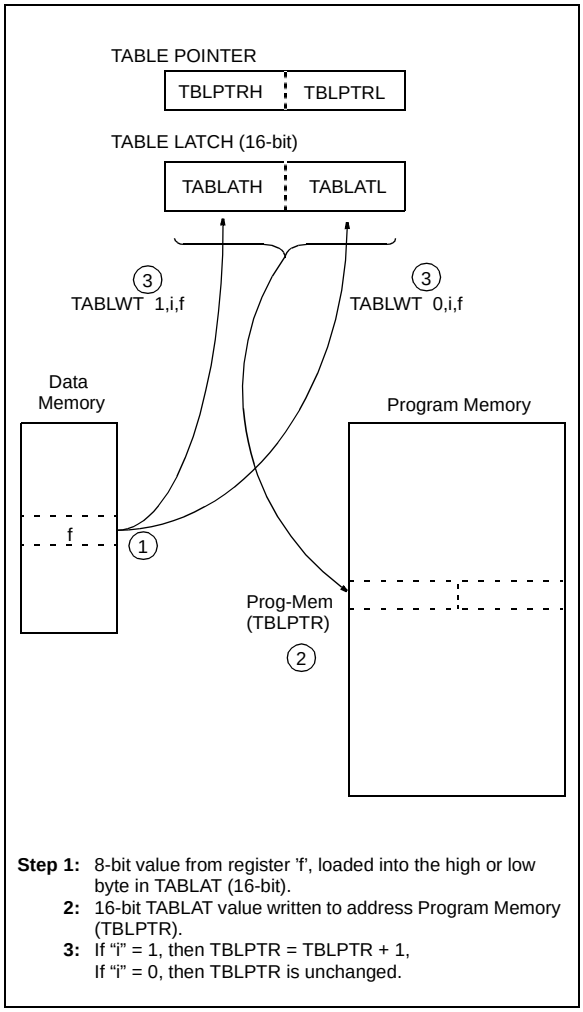
Figure 8-1 through Figure 8-4 show the operation of these four instructions. The steps show the sequence of operation.

**FIGURE 8-1: `TLWT` INSTRUCTION OPERATION**



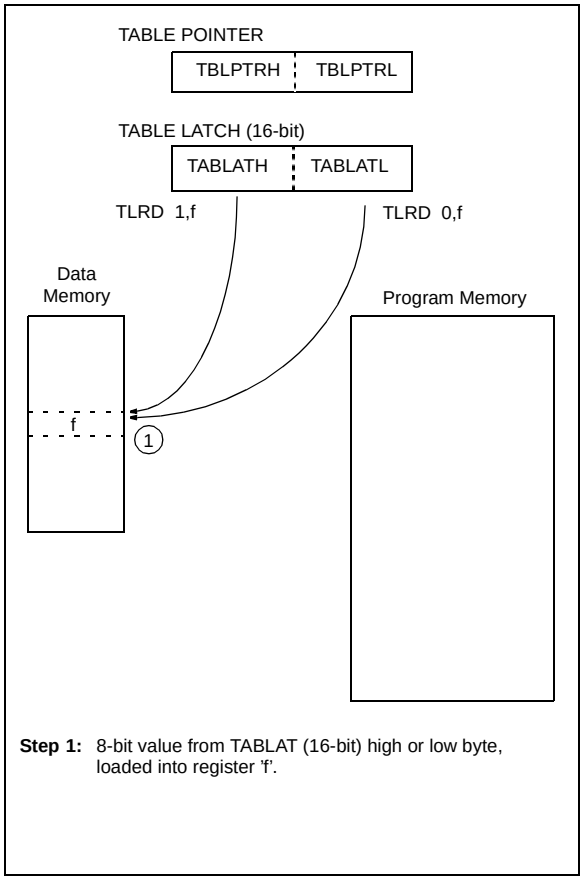

**Step 1:** 8-bit value from register 'f', loaded into the high or low byte in TABLAT (16-bit).

**FIGURE 8-2: `TABLWT` INSTRUCTION OPERATION**



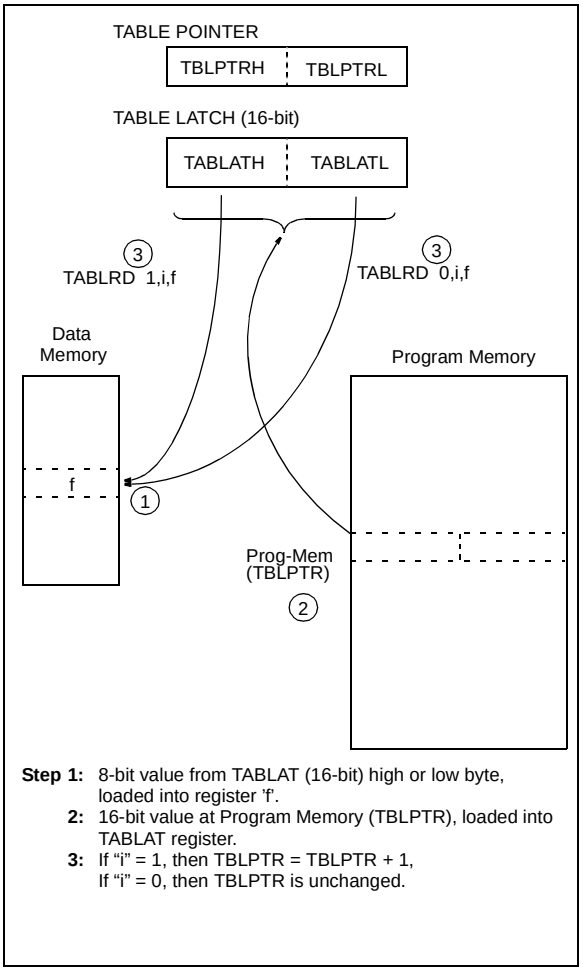

**Step 1:** 8-bit value from register 'f', loaded into the high or low byte in TABLAT (16-bit).
**2:** 16-bit TABLAT value written to address Program Memory (TBLPTR).
**3:** If "i" = 1, then TBLPTR = TBLPTR + 1,
If "i" = 0, then TBLPTR is unchanged.

**FIGURE 8-3: TLRD INSTRUCTION OPERATION**




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

**FIGURE 8-4: TABLRD INSTRUCTION OPERATION**




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


© 2000 Microchip Technology Inc.

## 8.1 Table Writes to Internal Memory

A table write operation to internal memory causes a long write operation. The long write is necessary for programming the internal EPROM. Instruction execution is halted while in a long write cycle. The long write will be terminated by any enabled interrupt. To ensure that the EPROM location has been well programmed, a minimum programming time is required (see specification #D114). Having only one interrupt enabled to terminate the long write ensures that no unintentional interrupts will prematurely terminate the long write.

The sequence of events for programming an internal program memory location should be:

1. Disable all interrupt sources, except the source to terminate EPROM program write.
2. Raise $\overline{MCLR}/V_{PP}$ pin to the programming voltage.
3. Clear the WDT.
4. Do the table write. The interrupt will terminate the long write.
5. Verify the memory location (table read).

**Note 1:** Programming requirements must be met. See timing specification in electrical specifications for the desired device. Violating these specifications (including temperature) may result in EPROM locations that are not fully programmed and may lose their state over time.

**2:** If the $V_{PP}$ requirement is not met, the table write is a 2-cycle write and the program memory is unchanged.

### 8.1.1 TERMINATING LONG WRITES

An interrupt source or RESET are the only events that terminate a long write operation. Terminating the long write from an interrupt source requires that the interrupt enable and flag bits are set. The GLINTD bit only enables the vectoring to the interrupt address.

If the T0CKI, RA0/INT, or TMR0 interrupt source is used to terminate the long write, the interrupt flag of the highest priority enabled interrupt, will terminate the long write and automatically be cleared.

**Note 1:** If an interrupt is pending, the `TABLWT` is aborted (a `NOP` is executed). The highest priority pending interrupt, from the T0CKI, RA0/INT, or TMR0 sources that is enabled, has its flag cleared.

**2:** If the interrupt is not being used for the program write timing, the interrupt should be disabled. This will ensure that the interrupt is not lost, nor will it terminate the long write prematurely.

If a peripheral interrupt source is used to terminate the long write, the interrupt enable and flag bits must be set. The interrupt flag will not be automatically cleared upon the vectoring to the interrupt vector address.

The GLINTD bit determines whether the program will branch to the interrupt vector when the long write is terminated. If GLINTD is clear, the program will vector, if GLINTD is set, the program will not vector to the interrupt address.

**TABLE 8-1: INTERRUPT - TABLE WRITE INTERACTION**

| Interrupt Source | GLINTD | Enable Bit | Flag Bit | Action |
|---|---|---|---|---|
| RA0/INT, TMR0, T0CKI | 0 | 1 | 1 | Terminate long table write (to internal program memory), branch to interrupt vector (branch clears flag bit). |
| | 0 | 1 | 0 | None. |
| | 1 | 0 | x | None. |
| | 1 | 1 | 1 | Terminate long table write, do not branch to interrupt vector (flag is automatically cleared). |
| Peripheral | 0 | 1 | 1 | Terminate long table write, branch to interrupt vector. |
| | 0 | 1 | 0 | None. |
| | 1 | 0 | x | None. |
| | 1 | 1 | 1 | Terminate long table write, do not branch to interrupt vector (flag remains set). |

## 8.2 Table Writes to External Memory

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

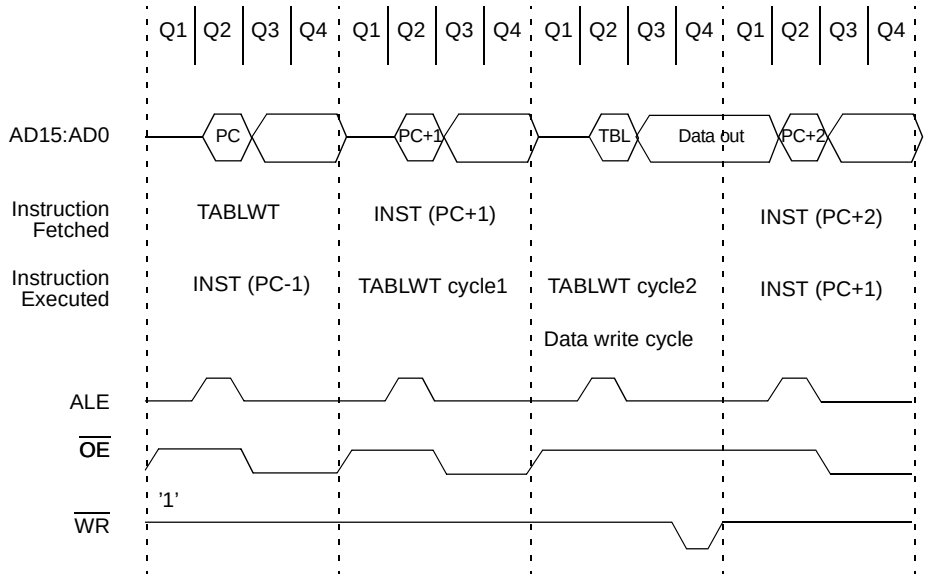
### 8.2.1 TABLE WRITE CODE

The "i" operand of the TABLWT instruction can specify that the value in the 16-bit TBLPTR register is automatically incremented (for the next write). In Example 8-1, the TBLPTR register is not automatically incremented.

**EXAMPLE 8-1: TABLE WRITE**

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

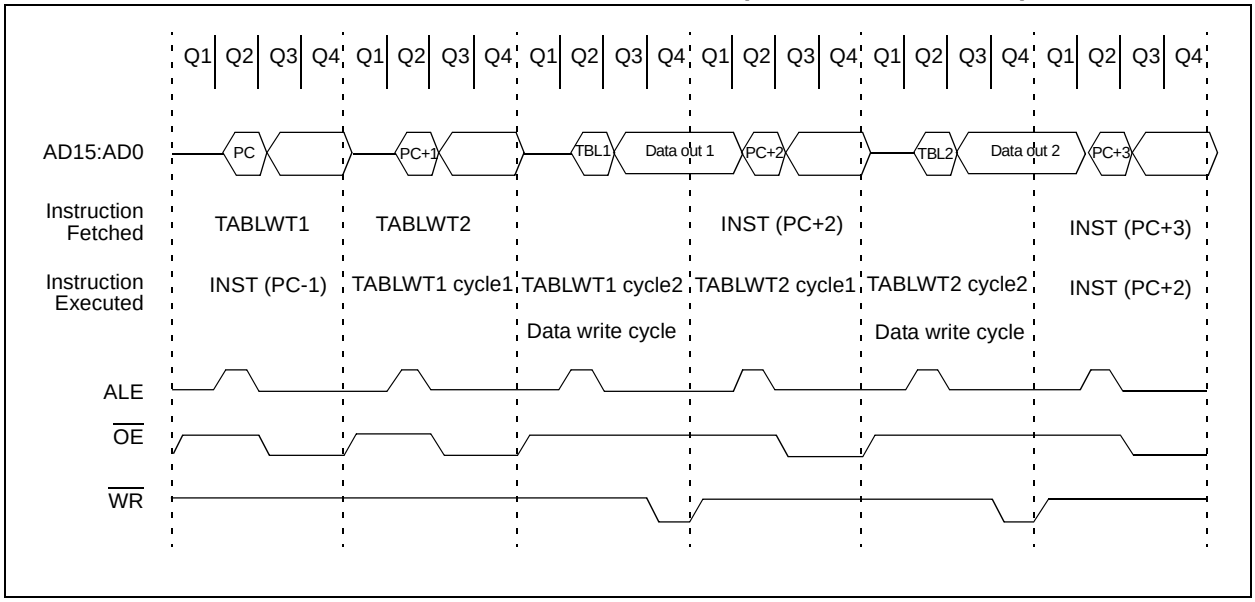
**FIGURE 8-5: TABLWT WRITE TIMING (EXTERNAL MEMORY)**




**Note:** If external write and GLINTD = '1' and Enable bit = '1', then when '1' → Flag bit, do table write. The highest pending interrupt is cleared.


© 2000 Microchip Technology Inc.

**FIGURE 8-6: CONSECUTIVE `TABLWT` WRITE TIMING (EXTERNAL MEMORY)**



## 8.3 Table Reads

The table read allows the program memory to be read. This allows constants to be stored in the program memory space and retrieved into data memory when needed. Example 8-2 reads the 16-bit value at program memory address TBLPTR. After the dummy byte has been read from the TABLATH, the TABLATH is loaded with the 16-bit data from program memory address TBLPTR and then increments the TBLPTR value. The first read loads the data into the latch and can be considered a dummy read (unknown data loaded into 'f'). INDF0 should be configured for either auto-increment or auto-decrement.

**EXAMPLE 8-2: TABLE READ**

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

**FIGURE 8-7: `TABLRD` TIMING**

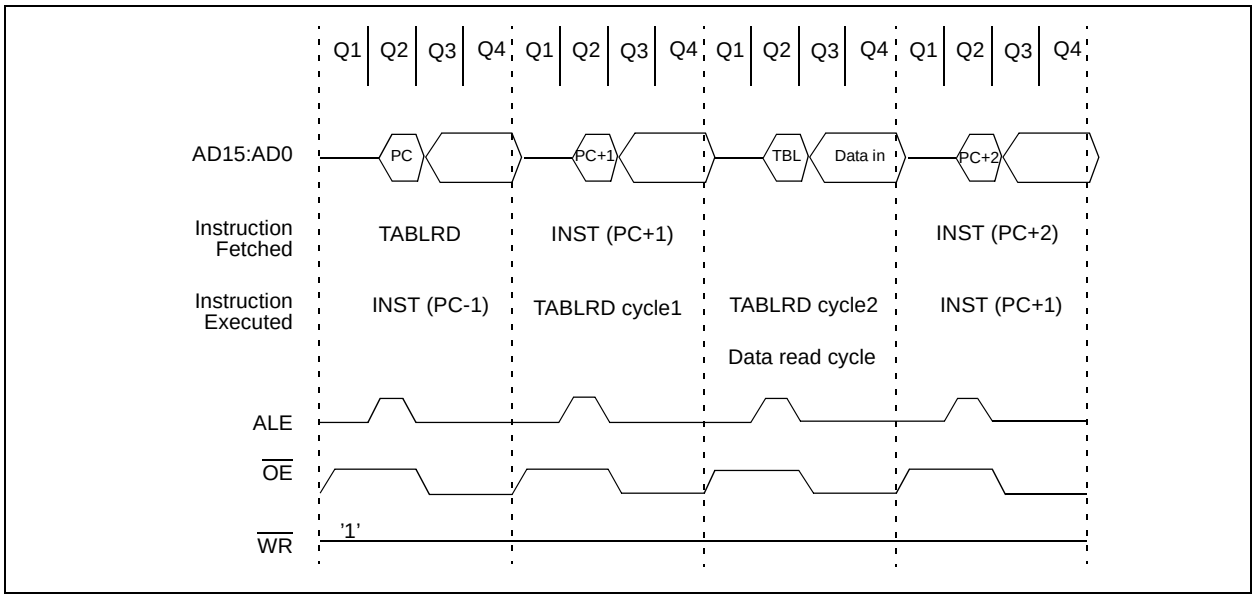
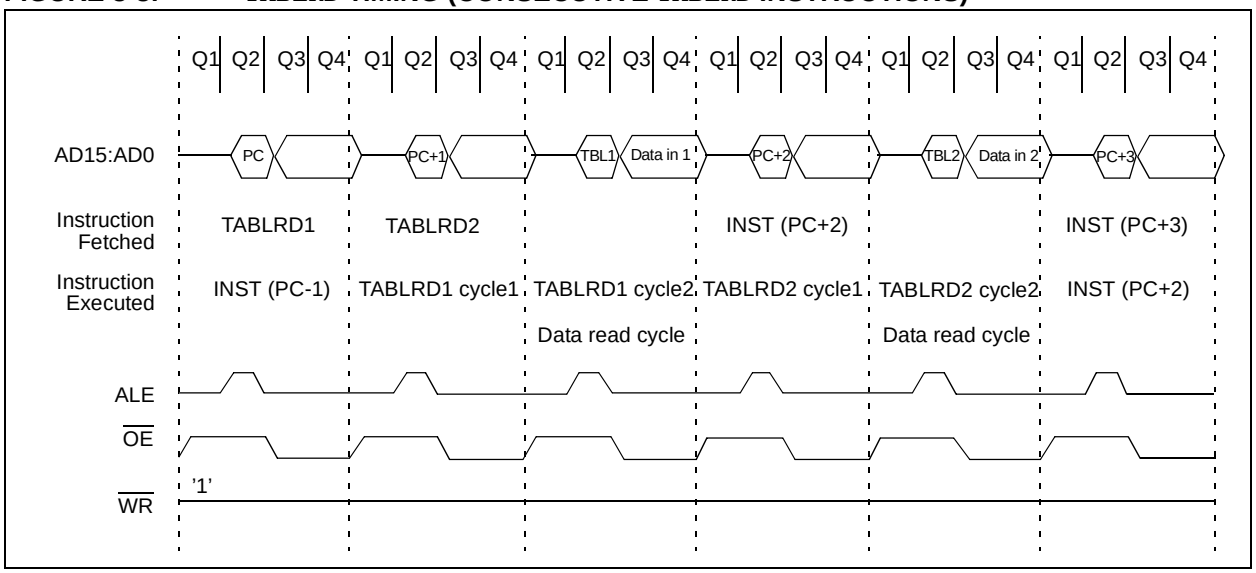



**FIGURE 8-8: `TABLRD` TIMING (CONSECUTIVE `TABLRD` INSTRUCTIONS)**





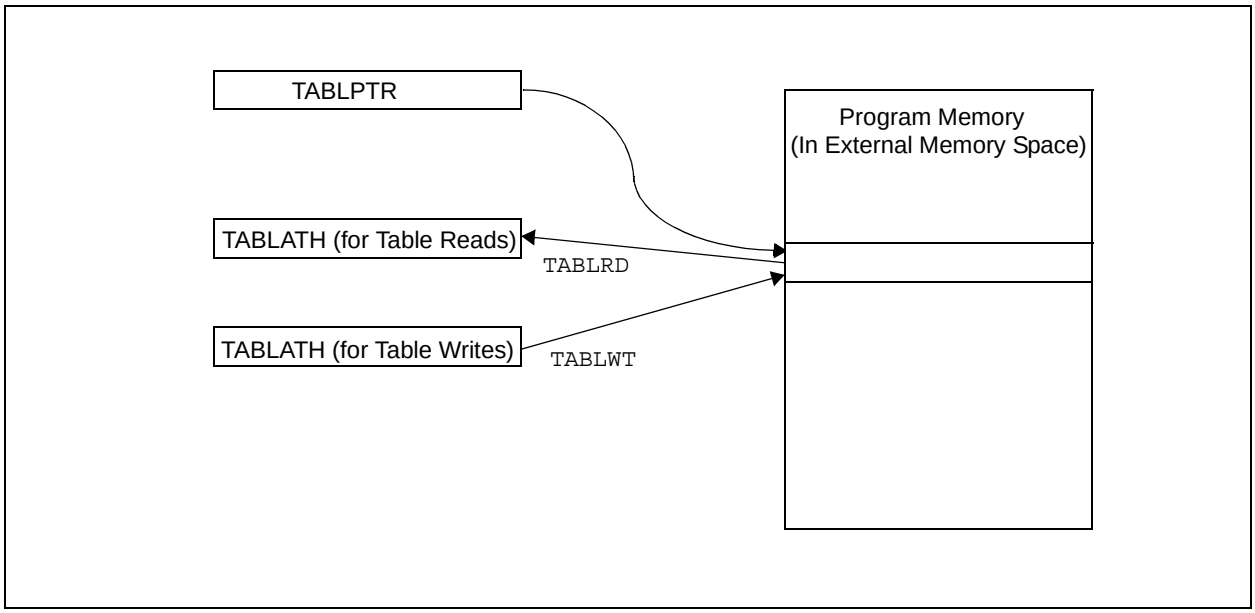
© 2000 Microchip Technology Inc.

## 8.4 Operation with External Memory Interface

When the table reads/writes are accessing external memory (via the external system interface bus), the table latch for the table reads is different from the table latch for the table writes (see Figure 8-9).

This means that you cannot do a TABLRD instruction, and use the values that were loaded into the table latches for a TABLWT instruction. Any table write sequence should use both the TLWT and then the TABLWT instructions.

**FIGURE 8-9: ACCESSING EXTERNAL MEMORY WITH TABLRD AND TABLWT INSTRUCTIONS**



**NOTES:**


© 2000 Microchip Technology Inc.

## 9.0 HARDWARE MULTIPLIER

All PIC17C7XX devices have an 8 x 8 hardware multiplier included in the ALU of the device. By making the multiply a hardware operation, it completes in a single instruction cycle. This is an unsigned multiply that gives a 16-bit result. The result is stored into the 16-bit Product register (PRODH:PRODL). The multiplier does not affect any flags in the ALUSTA register.

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

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

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

Table 9-1 shows a performance comparison between PIC17CXXX devices using the single cycle hardware multiply and performing the same function without the hardware multiply.

Example 9-1 shows the sequence to do an 8 x 8 unsigned multiply. Only one instruction is required when one argument of the multiply is already loaded in the WREG register.

Example 9-2 shows the sequence to do an 8 x 8 signed multiply. To account for the sign bits of the arguments, each argument's most significant bit (MSb) is tested and the appropriate subtractions are done.

**EXAMPLE 9-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE**

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

**EXAMPLE 9-2: 8 x 8 SIGNED MULTIPLY ROUTINE**

```
MOVFP    ARG1, WREG
MULWF    ARG2        ; ARG1 * ARG2 ->
                     ;   PRODH:PRODL
BTFSC    ARG2, SB    ; Test Sign Bit
SUBWF    PRODH, F    ; PRODH = PRODH
                     ;         - ARG1
MOVFP    ARG2, WREG
BTFSC    ARG1, SB    ; Test Sign Bit
SUBWF    PRODH, F    ; PRODH = PRODH
                     ;         - ARG2
```

**TABLE 9-1: PERFORMANCE COMPARISON**

| Routine | Multiply Method | Program Memory (Words) | Cycles (Max) | Time | | |
|---|---|---|---|---|---|---|
| | | | | @ 33 MHz | @ 16 MHz | @ 8 MHz |
| 8 x 8 unsigned | Without hardware multiply | 13 | 69 | 8.364 µs | 17.25 µs | 34.50 µs |
| | Hardware multiply | 1 | 1 | 0.121 µs | 0.25 µs | 0.50 µs |
| 8 x 8 signed | Without hardware multiply | — | — | — | — | — |
| | Hardware multiply | 6 | 6 | 0.727 µs | 1.50 µs | 3.0 µs |
| 16 x 16 unsigned | Without hardware multiply | 21 | 242 | 29.333 µs | 60.50 µs | 121.0 µs |
| | Hardware multiply | 24 | 24 | 2.91 µs | 6.0 µs | 12.0 µs |
| 16 x 16 signed | Without hardware multiply | 52 | 254 | 30.788 µs | 63.50 µs | 127.0 µs |
| | Hardware multiply | 36 | 36 | 4.36 µs | 9.0 µs | 18.0 µs |

© 2000 Microchip Technology Inc.

Example 9-3 shows the sequence to do a 16 x 16 unsigned multiply. Equation 9-1 shows the algorithm that is used. The 32-bit result is stored in 4 registers, RES3:RES0.

**EQUATION 9-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM**

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

**EXAMPLE 9-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE**

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


© 2000 Microchip Technology Inc.

Example 9-4 shows the sequence to do a 16 x 16 signed multiply. Equation 9-2 shows the algorithm used. The 32-bit result is stored in four registers, RES3:RES0. To account for the sign bits of the arguments, each argument pairs most significant bit (MSb) is tested and the appropriate subtractions are done.

**EQUATION 9-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM**

$$
\begin{aligned}
&RES3:RES0 \\
&\quad = ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&\quad = (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad\ \ (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad\ \ (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad\ \ (ARG1L \bullet ARG2L) + \\
&\quad\ \ (-1 \bullet ARG2H\langle 7\rangle \bullet ARG1H:ARG1L \bullet 2^{16}) + \\
&\quad\ \ (-1 \bullet ARG1H\langle 7\rangle \bullet ARG2H:ARG2L \bullet 2^{16})
\end{aligned}
$$

**EXAMPLE 9-4: 16 x 16 SIGNED MULTIPLY ROUTINE**

```
   MOVFP    ARG1L, WREG
   MULWF    ARG2L        ; ARG1L * ARG2L ->
                         ;   PRODH:PRODL
   MOVPF    PRODH, RES1 ;
   MOVPF    PRODL, RES0 ;
;
   MOVFP    ARG1H, WREG
   MULWF    ARG2H        ; ARG1H * ARG2H ->
                         ;   PRODH:PRODL
   MOVPF    PRODH, RES3 ;
   MOVPF    PRODL, RES2 ;
;
   MOVFP    ARG1L, WREG
   MULWF    ARG2H        ; ARG1L * ARG2H ->
                         ;   PRODH:PRODL
   MOVPF    PRODL, WREG ;
   ADDWF    RES1, F      ; Add cross
   MOVPF    PRODH, WREG ;    products
   ADDWFC   RES2, F      ;
   CLRF     WREG, F      ;
   ADDWFC   RES3, F      ;
;
   MOVFP    ARG1H, WREG ;
   MULWF    ARG2L        ; ARG1H * ARG2L ->
                         ;   PRODH:PRODL
   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F      ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F      ;
   CLRF     WREG, F      ;
   ADDWFC   RES3, F      ;
;
   BTFSS    ARG2H, 7     ; ARG2H:ARG2L neg?
   GOTO     SIGN_ARG1    ; no, check ARG1
   MOVFP    ARG1L, WREG ;
   SUBWF    RES2         ;
   MOVFP    ARG1H, WREG ;
   SUBWFB   RES3
;
SIGN_ARG1
   BTFSS    ARG1H, 7     ; ARG1H:ARG1L neg?
   GOTO     CONT_CODE    ; no, done
   MOVFP    ARG2L, WREG ;
   SUBWF    RES2         ;
   MOVFP    ARG2H, WREG ;
   SUBWFB   RES3
;
CONT_CODE
     :
```

**NOTES:**


© 2000 Microchip Technology Inc.

## 10.0 I/O PORTS

PIC17C75X devices have seven I/O ports, PORTA through PORTG. PIC17C76X devices have nine I/O ports, PORTA through PORTJ. PORTB through PORTJ have a corresponding Data Direction Register (DDR), which is used to configure the port pins as inputs or outputs. Some of these ports pins are multiplexed with alternate functions.

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

PORTA, PORTB, PORTE<3>, PORTF, PORTG and the upper four bits of PORTH are multiplexed with the peripheral features of the device. These peripheral features are:

- Timer Modules
- Capture Modules
- PWM Modules
- USART/SCI Modules
- SSP Module
- A/D Module
- External Interrupt pin

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

- PWM Module
- SSP Module
- USART/SCI Module

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

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

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

When the device enters the "RESET state", the Data Direction registers (DDR) are forced set, which will make the I/O hi-impedance inputs. The RESET state of some peripheral modules may force the I/O to other operations, such as analog inputs or the system bus.

## 10.1 PORTA Register

PORTA is a 6-bit wide latch. PORTA does not have a corresponding Data Direction Register (DDR). Upon a device RESET, the PORTA pins are forced to be hi-impedance inputs. For the RA4 and RA5 pins, the peripheral module controls the output. When a device RESET occurs, the peripheral module is disabled, so these pins are forced to be hi-impedance inputs.

Reading PORTA reads the status of the pins.

The RA0 pin is multiplexed with the external interrupt, INT. The RA1 pin is multiplexed with TMR0 clock input, RA2 and RA3 are multiplexed with the SSP functions, and RA4 and RA5 are multiplexed with the USART1 functions. The control of RA2, RA3, RA4 and RA5 as outputs, is automatically configured by their multiplexed peripheral module when the module is enabled.

### 10.1.1 USING RA2, RA3 AS OUTPUTS

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

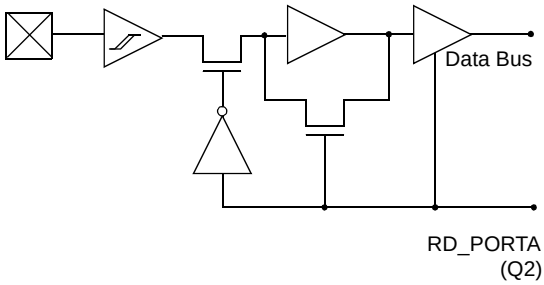
| | |
|---|---|
| **Note:** | When using the RA2 or RA3 pin(s) as output(s), read-modify-write instructions (such as BCF, BSF, BTG) on PORTA are not recommended.<br><br>Such operations read the port pins, do the desired operation, and then write this value to the data latch. This may inadvertently cause the RA2 or RA3 pins to switch from input to output (or vice-versa).<br><br>To avoid this possibility, use a shadow register for PORTA. Do the bit operations on this shadow register and then move it to PORTA. |

Example 10-1 shows an instruction sequence to initialize PORTA. The Bank Select Register (BSR) must be selected to Bank 0 for the port to be initialized. The following example uses the MOVLB instruction to load the BSR register for bank selection.

**EXAMPLE 10-1: INITIALIZING PORTA**

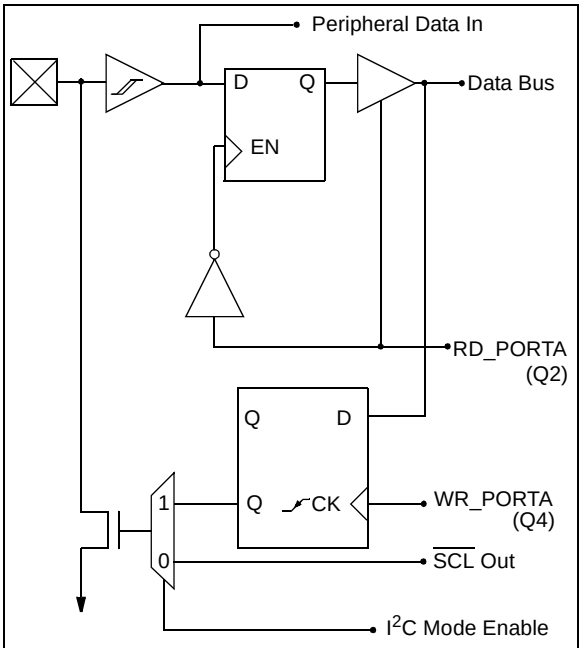
```
    MOVLB   0       ; Select Bank 0
    MOVLW   0xF3    ;
    MOVWF   PORTA   ; Initialize PORTA
                    ;   RA<3:2> are output low
                    ;   RA<5:4> and RA<1:0>
                    ;   are inputs
                    ;   (outputs floating)
```

**FIGURE 10-1: RA0 AND RA1 BLOCK DIAGRAM**




**Note:** Input pins have protection diodes to VDD and VSS.

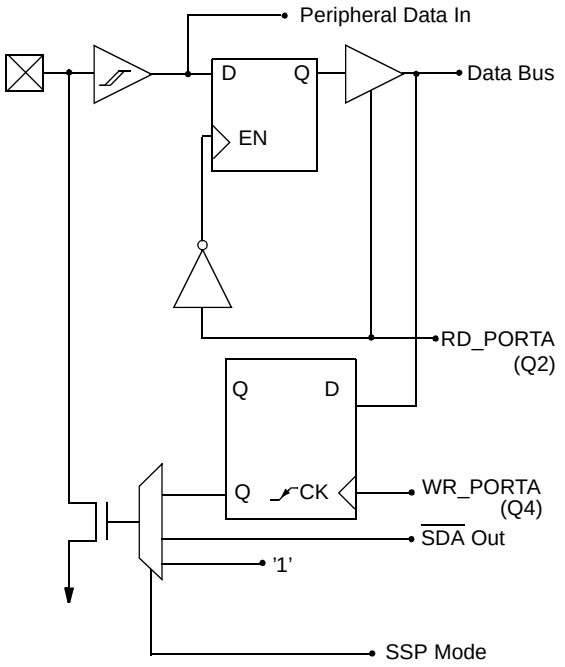
**FIGURE 10-2: RA2 BLOCK DIAGRAM**




**Note:** I/O pin has protection diodes to VSS.

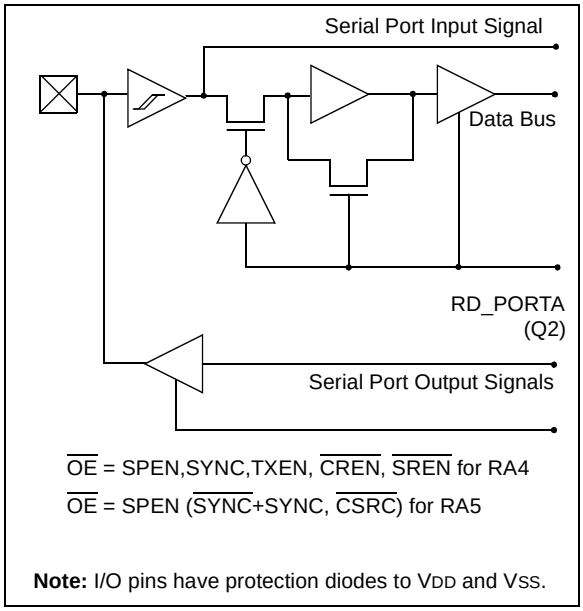

© 2000 Microchip Technology Inc.

**FIGURE 10-3: RA3 BLOCK DIAGRAM**




**Note:** I/O pin has protection diodes to VSS.

**FIGURE 10-4: RA4 AND RA5 BLOCK DIAGRAM**




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

**TABLE 10-1: PORTA FUNCTIONS**

| Name | Bit0 | Buffer Type | Function |
|---|---|---|---|
| RA0/INT | bit0 | ST | Input or external interrupt input. |
| RA1/T0CKI | bit1 | ST | Input or clock input to the TMR0 timer/counter and/or an external interrupt input. |
| RA2/SS/SCL | bit2 | ST | Input/output or slave select input for the SPI, or clock input for the $I^2C$ bus. Output is open drain type. |
| RA3/SDI/SDA | bit3 | ST | Input/output or data input for the SPI, or data for the $I^2C$ bus. Output is open drain type. |
| RA4/RX1/DT1 | bit4 | ST | Input or USART1 Asynchronous Receive input, or USART1 Synchronous Data input/output. |
| RA5/TX1/CK1 | bit5 | ST | Input or USART1 Asynchronous Transmit output, or USART1 Synchronous Clock input/output. |
| RBPU | bit7 | — | Control bit for PORTB weak pull-ups. |

Legend: ST = Schmitt Trigger input

**TABLE 10-2: REGISTERS/BITS ASSOCIATED WITH PORTA**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | MCLR, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 10h, Bank 0 | PORTA(1) | RBPU | — | RA5/ TX1/CK1 | RA4/ RX1/DT1 | RA3/ SDI/SDA | RA2/ SS/SCL | RA1/T0CKI | RA0/INT | 0-xx 11xx | 0-uu 11uu |
| 05h, Unbanked | T0STA | INTEDG | T0SE | T0CS | T0PS3 | T0PS2 | T0PS1 | T0PS0 | — | 0000 000- | 0000 000- |
| 13h, Bank 0 | RCSTA1 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 15h, Bank 0 | TXSTA1 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |

Legend: x = unknown, u = unchanged, - = unimplemented, reads as '0'. Shaded cells are not used by PORTA.
**Note 1:** On any device RESET, these pins are configured as inputs.

## 10.2 PORTB and DDRB Registers

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

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

PORTB also has an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB0 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB0) are compared with the value in the PORTB data latch. The "mismatch" outputs of RB7:RB0 are OR'd together to set the PORTB Interrupt Flag bit, RBIF (PIR1<7>).

This interrupt can wake the device from SLEEP. The user, in the Interrupt Service Routine, can clear the interrupt by:

a) Read-Write PORTB (such as: `MOVPF PORTB, PORTB`). This will end the mismatch condition.

b) Then, clear the RBIF bit.

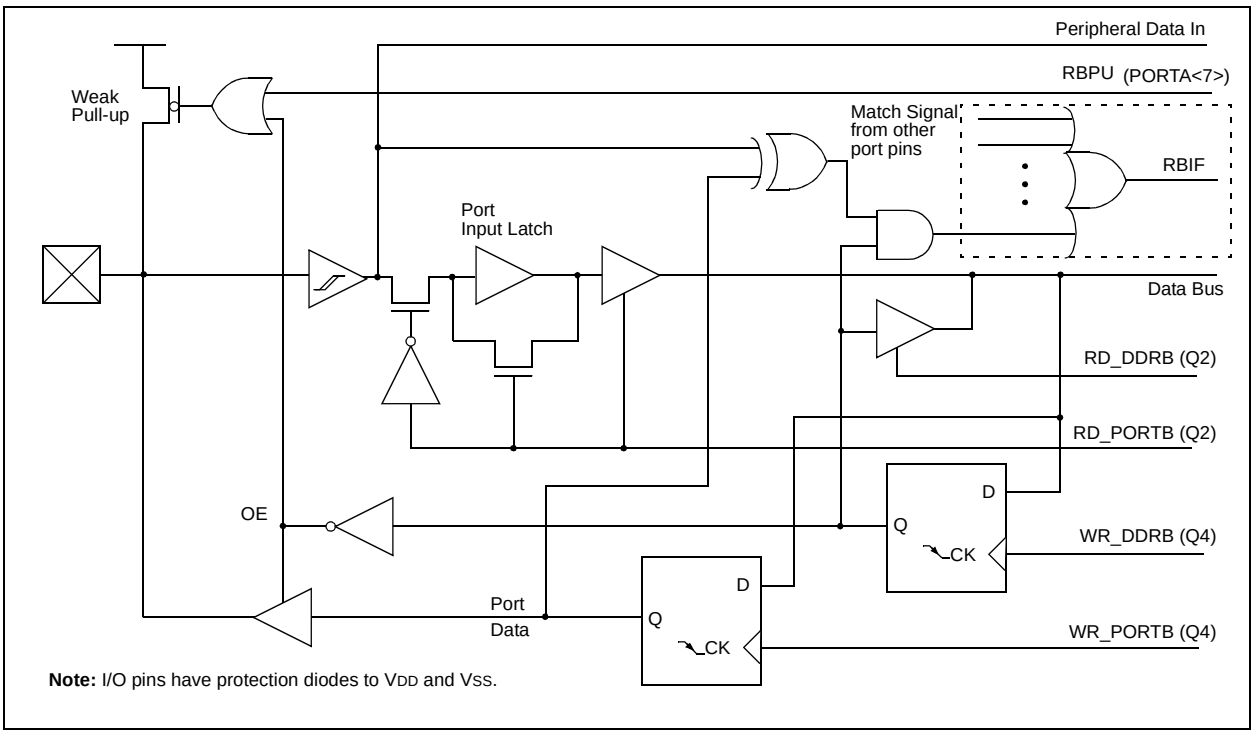
A mismatch condition will continue to set the RBIF bit. Reading, then writing PORTB, will end the mismatch condition and allow the RBIF bit to be cleared.

This interrupt-on-mismatch feature, together with software configurable pull-ups on this port, allows easy interface to a keypad and makes it possible for wake-up on key depression. For an example, refer to Application Note AN552, "Implementing Wake-up on Keystroke."

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

| **Note:** | On a device RESET, the RBIF bit is indeterminate, since the value in the latch may be different than the pin. |
|---|---|

**FIGURE 10-5: BLOCK DIAGRAM OF RB5:RB4 AND RB1:RB0 PORT PINS**




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

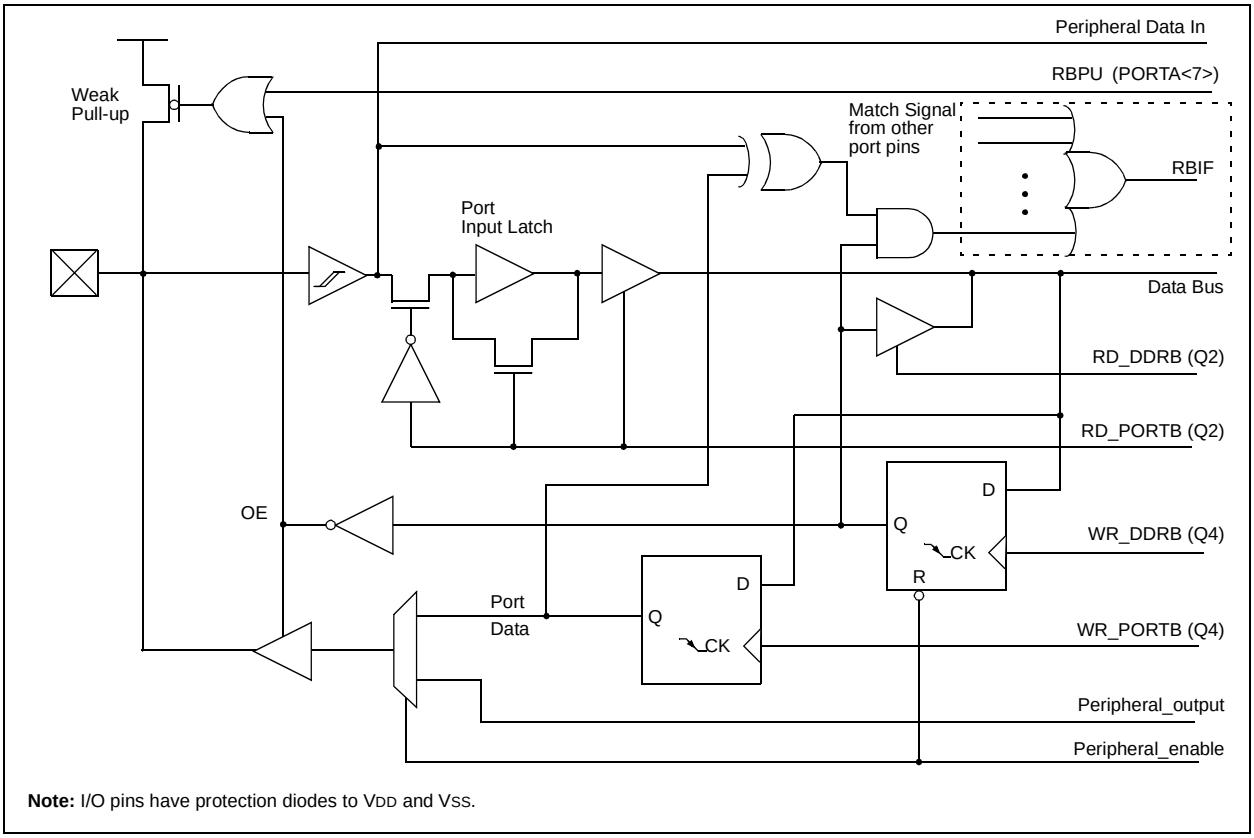

© 2000 Microchip Technology Inc.

Example 10-2 shows an instruction sequence to initialize PORTB. The Bank Select Register (BSR) must be selected to Bank 0 for the port to be initialized. The following example uses the MOVLB instruction to load the BSR register for bank selection.

**EXAMPLE 10-2: INITIALIZING PORTB**

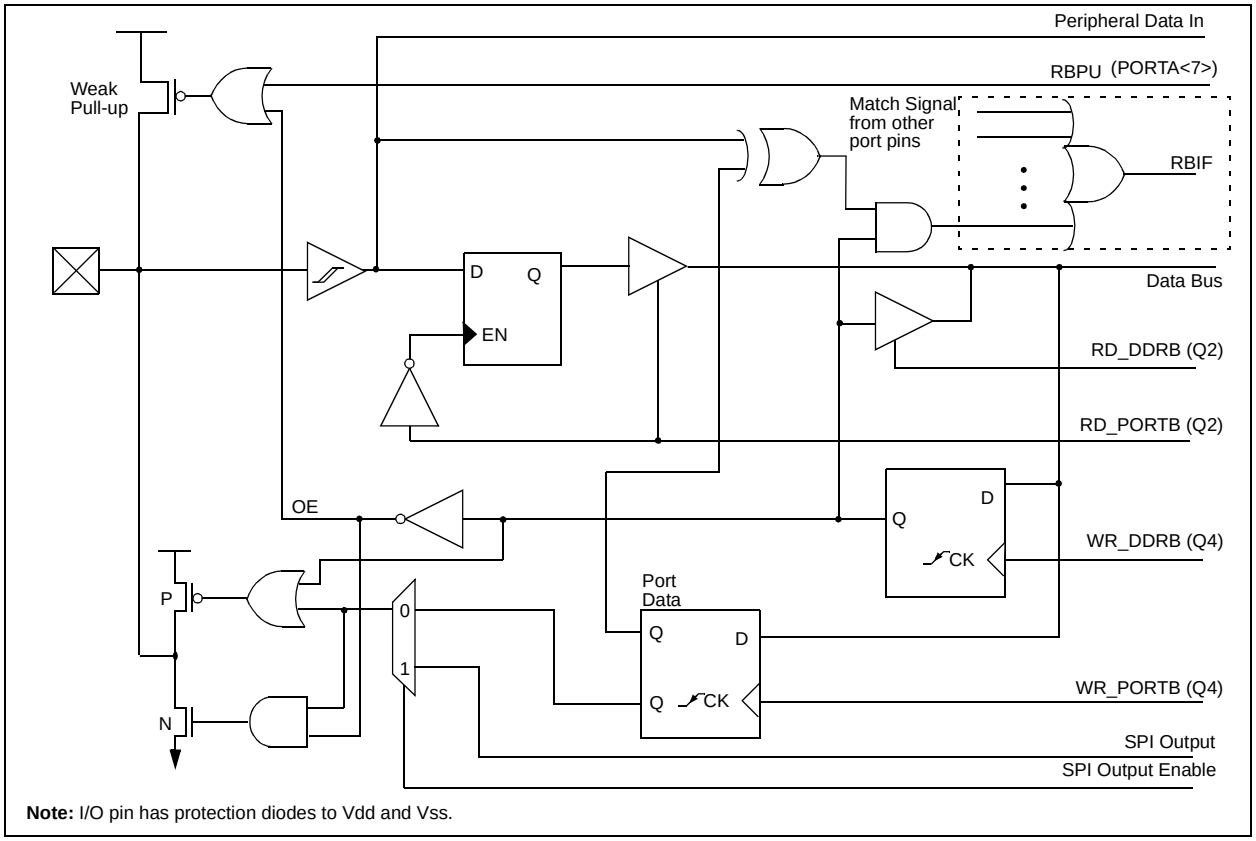
```
MOVLB   0          ; Select Bank 0
CLRF    PORTB, F   ; Init PORTB by clearing
                   ;  output data latches
MOVLW   0xCF       ; Value used to initialize
                   ;  data direction
MOVWF   DDRB       ; Set RB<3:0> as inputs
                   ;   RB<5:4> as outputs
       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

**FIGURE 10-6: BLOCK DIAGRAM OF RB3:RB2 PORT PINS**



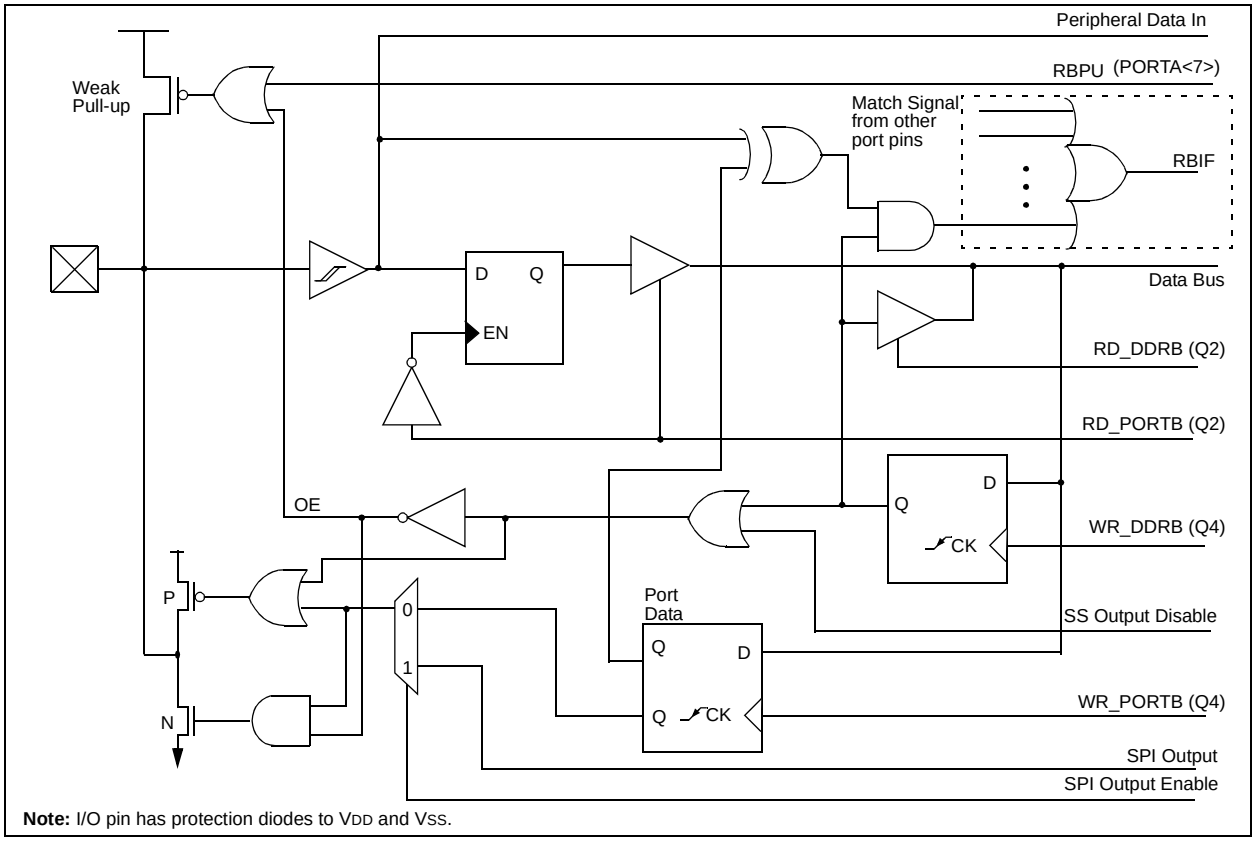

**Note:** I/O pins have protection diodes to VDD and VSS.

**FIGURE 10-7: BLOCK DIAGRAM OF RB6 PORT PIN**




**Note:** I/O pin has protection diodes to Vdd and Vss.

**FIGURE 10-8: BLOCK DIAGRAM OF RB7 PORT PIN**




**Note:** I/O pin has protection diodes to VDD and VSS.


© 2000 Microchip Technology Inc.

**TABLE 10-3: PORTB FUNCTIONS**

| Name | Bit | Buffer Type | Function |
|---|---|---|---|
| RB0/CAP1 | bit0 | ST | Input/output or the Capture1 input pin. Software programmable weak pull-up and interrupt-on-change features. |
| RB1/CAP2 | bit1 | ST | Input/output or the Capture2 input pin. Software programmable weak pull-up and interrupt-on-change features. |
| RB2/PWM1 | bit2 | ST | Input/output or the PWM1 output pin. Software programmable weak pull-up and interrupt-on-change features. |
| RB3/PWM2 | bit3 | ST | Input/output or the PWM2 output pin. Software programmable weak pull-up and interrupt-on-change features. |
| RB4/TCLK12 | bit4 | ST | Input/output or the external clock input to Timer1 and Timer2. Software programmable weak pull-up and interrupt-on-change features. |
| RB5/TCLK3 | bit5 | ST | Input/output or the external clock input to Timer3. Software programmable weak pull-up and interrupt-on-change features. |
| RB6/SCK | bit6 | ST | Input/output or the Master/Slave clock for the SPI. Software programmable weak pull-up and interrupt-on-change features. |
| RB7/SDO | bit7 | ST | Input/output or data output for the SPI. Software programmable weak pull-up and interrupt-on-change features. |

Legend: ST = Schmitt Trigger input

**TABLE 10-4: REGISTERS/BITS ASSOCIATED WITH PORTB**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 12h, Bank 0 | PORTB | RB7/ SDO | RB6/ SCK | RB5/ TCLK3 | RB4/ TCLK12 | RB3/ PWM2 | RB2/ PWM1 | RB1/ CAP2 | RB0/ CAP1 | xxxx xxxx | uuuu uuuu |
| 11h, Bank 0 | DDRB | Data Direction Register for PORTB | | | | | | | | 1111 1111 | 1111 1111 |
| 10h, Bank 0 | PORTA | $\overline{\text{RBPU}}$ | — | RA5/ TX1/CK1 | RA4/ RX1/DT1 | RA3/ SDI/SDA | $\overline{\text{SS}}$/SCL | RA1/T0CKI | RA0/INT | 0-xx 11xx | 0-uu 11uu |
| 06h, Unbanked | CPUSTA | — | — | STKAV | GLINTD | TO | PD | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | --11 11qq | --11 qquu |
| 07h, Unbanked | INTSTA | PEIF | T0CKIF | T0IF | INTF | PEIE | T0CKIE | T0IE | INTE | 0000 0000 | 0000 0000 |
| 16h, Bank 1 | PIR1 | RBIF | TMR3IF | TMR2IF | TMR1IF | CA2IF | CA1IF | TX1IF | RC1IF | x000 0010 | u000 0010 |
| 17h, Bank 1 | PIE1 | RBIE | TMR3IE | TMR2IE | TMR1IE | CA2IE | CA1IE | TX1IE | RC1IE | 0000 0000 | 0000 0000 |
| 16h, Bank 3 | TCON1 | CA2ED1 | CA2ED0 | CA1ED1 | CA1ED0 | T16 | TMR3CS | TMR2CS | TMR1CS | 0000 0000 | 0000 0000 |
| 17h, Bank 3 | TCON2 | CA2OVF | CA1OVF | PWM2ON | PWM1ON | CA1/$\overline{\text{PR3}}$ | TMR3ON | TMR2ON | TMR1ON | 0000 0000 | 0000 0000 |

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

## 10.3 PORTC and DDRC Registers

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

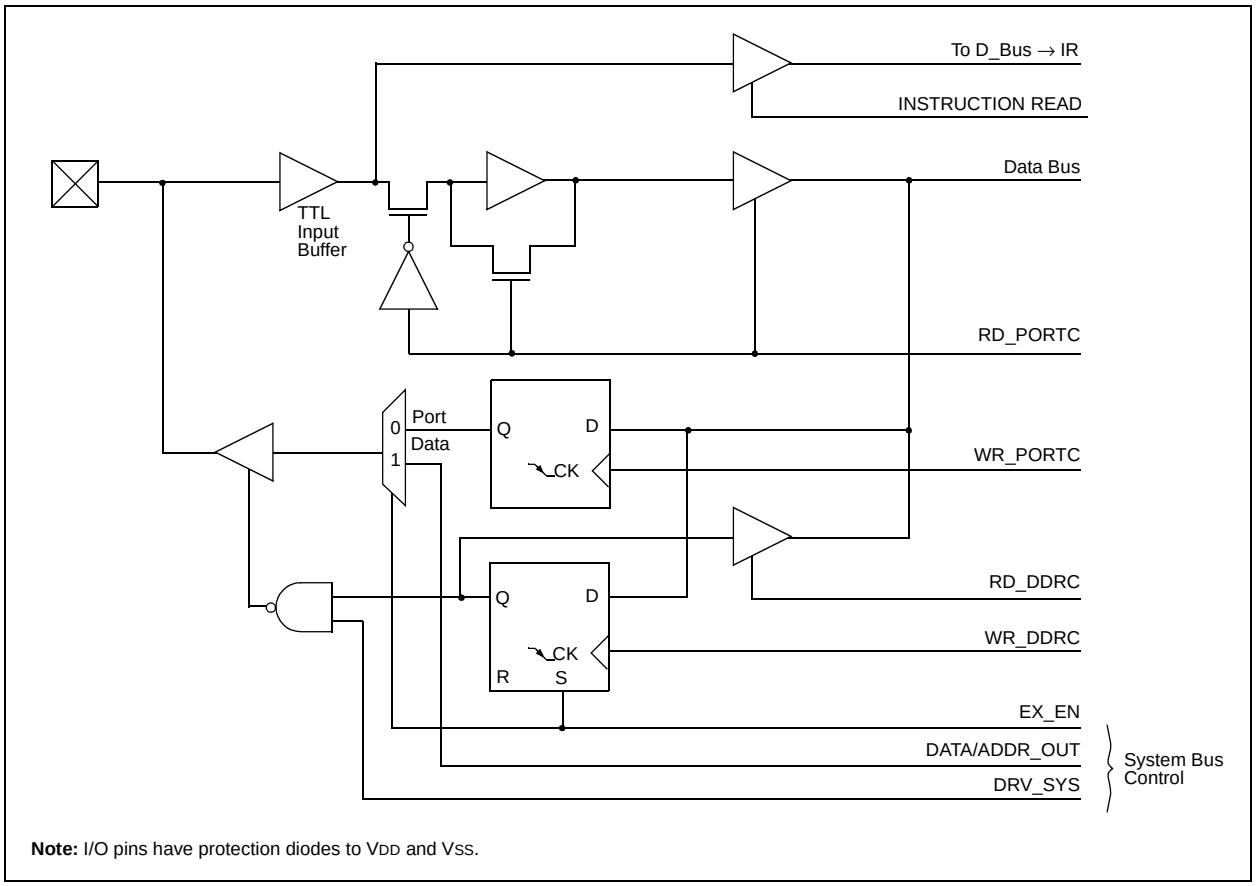
| **Note:** | This port is configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, this port is a general purpose I/O. |
|---|---|

Example 10-3 shows an instruction sequence to initialize PORTC. The Bank Select Register (BSR) must be selected to Bank 1 for the port to be initialized. The following example uses the MOVLB instruction to load the BSR register for bank selection.

**EXAMPLE 10-3: INITIALIZING PORTC**

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

**FIGURE 10-9: BLOCK DIAGRAM OF RC7:RC0 PORT PINS**




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


© 2000 Microchip Technology Inc.

**TABLE 10-5: PORTC FUNCTIONS**

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

Legend: TTL = TTL input

**TABLE 10-6: REGISTERS/BITS ASSOCIATED WITH PORTC**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 11h, Bank 1 | PORTC | RC7/ AD7 | RC6/ AD6 | RC5/ AD5 | RC4/ AD4 | RC3/ AD3 | RC2/ AD2 | RC1/ AD1 | RC0/ AD0 | xxxx xxxx | uuuu uuuu |
| 10h, Bank 1 | DDRC | Data Direction Register for PORTC | | | | | | | | 1111 1111 | 1111 1111 |

Legend: x = unknown, u = unchanged

## 10.4 PORTD and DDRD Registers

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

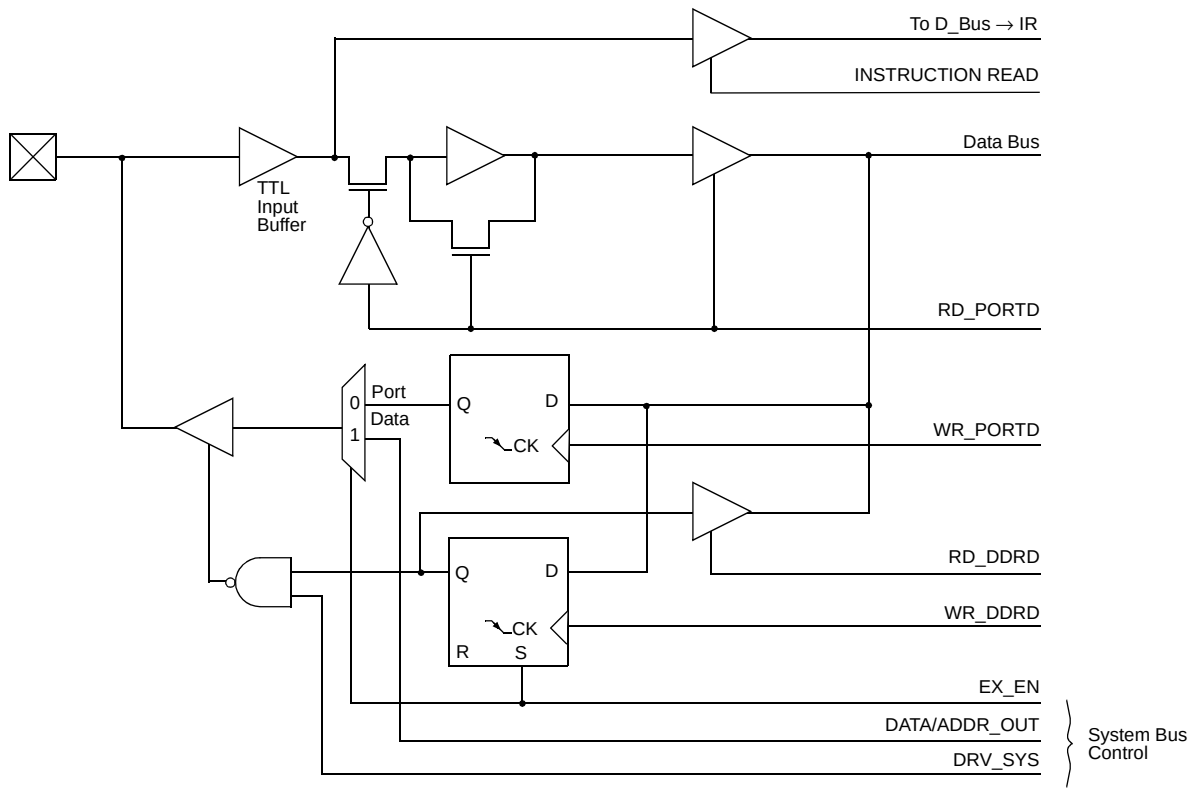
**Note:** This port is configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, this port is a general purpose I/O.

Example 10-4 shows an instruction sequence to initialize PORTD. The Bank Select Register (BSR) must be selected to Bank 1 for the port to be initialized. The following example uses the MOVLB instruction to load the BSR register for bank selection.

**EXAMPLE 10-4: INITIALIZING PORTD**

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

**FIGURE 10-10: BLOCK DIAGRAM OF RD7:RD0 PORT PINS (IN I/O PORT MODE)**




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


© 2000 Microchip Technology Inc.

**TABLE 10-7: PORTD FUNCTIONS**

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

Legend: TTL = TTL input

**TABLE 10-8: REGISTERS/BITS ASSOCIATED WITH PORTD**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 13h, Bank 1 | PORTD | RD7/ AD15 | RD6/ AD14 | RD5/ AD13 | RD4/ AD12 | RD3/ AD11 | RD2/ AD10 | RD1/ AD9 | RD0/ AD8 | xxxx xxxx | uuuu uuuu |
| 12h, Bank 1 | DDRD | Data Direction Register for PORTD | | | | | | | | 1111 1111 | 1111 1111 |

Legend: x = unknown, u = unchanged

## 10.5 PORTE and DDRE Register

PORTE is a 4-bit bi-directional port. The corresponding data direction register is DDRE. A '1' in DDRE configures the corresponding port pin as an input. A '0' in the DDRE register configures the corresponding port pin as an output. Reading PORTE reads the status of the pins, whereas writing to PORTE will write to the port latch. PORTE is multiplexed with the system bus. When operating as the system bus, PORTE contains the control signals for the address/data bus (AD15:AD0). These control signals are Address Latch Enable (ALE), Output Enable ($\overline{OE}$) and Write ($\overline{WR}$). The control signals $\overline{OE}$ and $\overline{WR}$ are active low signals. The timing for the system bus is shown in the Electrical Specifications section.

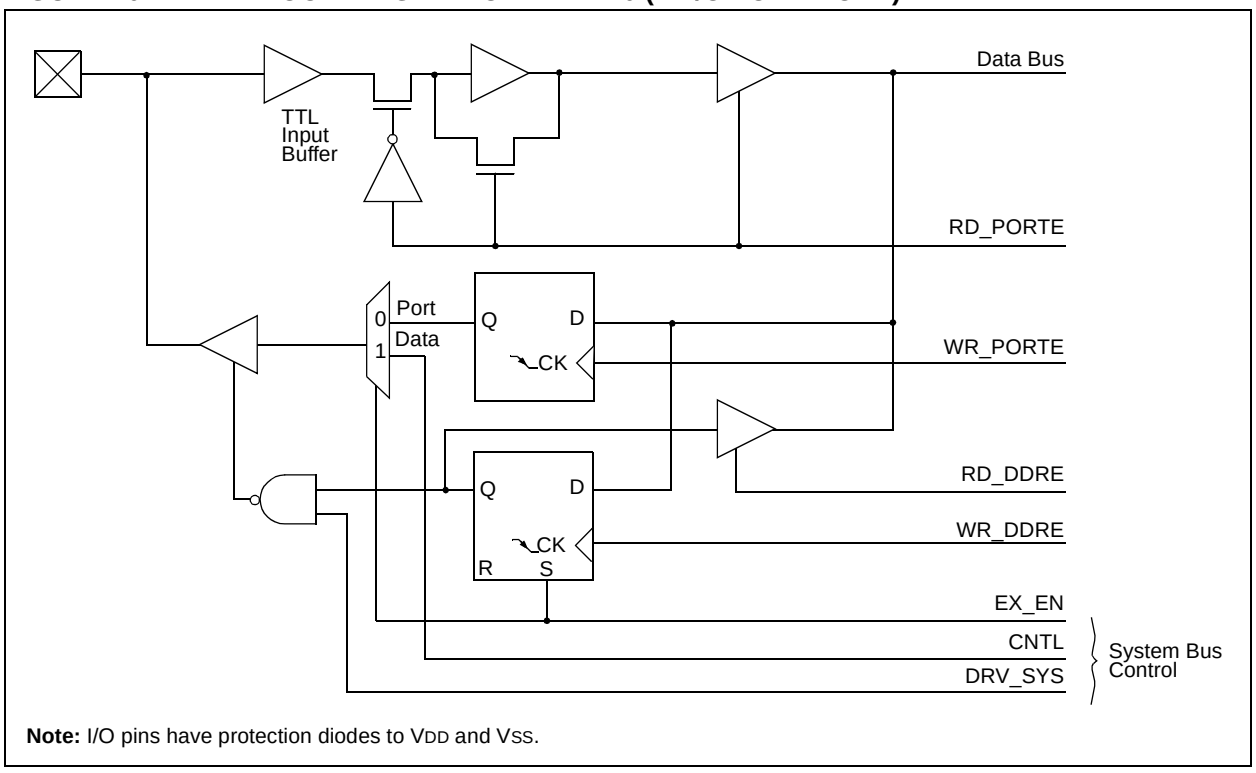
| **Note:** | Three pins of this port are configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. The other pin is a general purpose I/O or Capture4 pin. In the two other microcontroller modes, RE2:RE0 are general purpose I/O pins. |
|---|---|

Example 10-5 shows an instruction sequence to initialize PORTE. The Bank Select Register (BSR) must be selected to Bank 1 for the port to be initialized. The following example uses the MOVLB instruction to load the BSR register for bank selection.

**EXAMPLE 10-5: INITIALIZING PORTE**

```
MOVLB   1           ; Select Bank 1
CLRF    PORTE, F    ; Initialize PORTE data
                    ; latches before setting
                    ; the data direction
                    ; register
MOVLW    0x03       ; Value used to initialize
                    ; data direction
MOVWF    DDRE       ; Set RE<1:0> as inputs
                    ; RE<3:2> as outputs
                    ; RE<7:4> are always
                    ; read as '0'
```

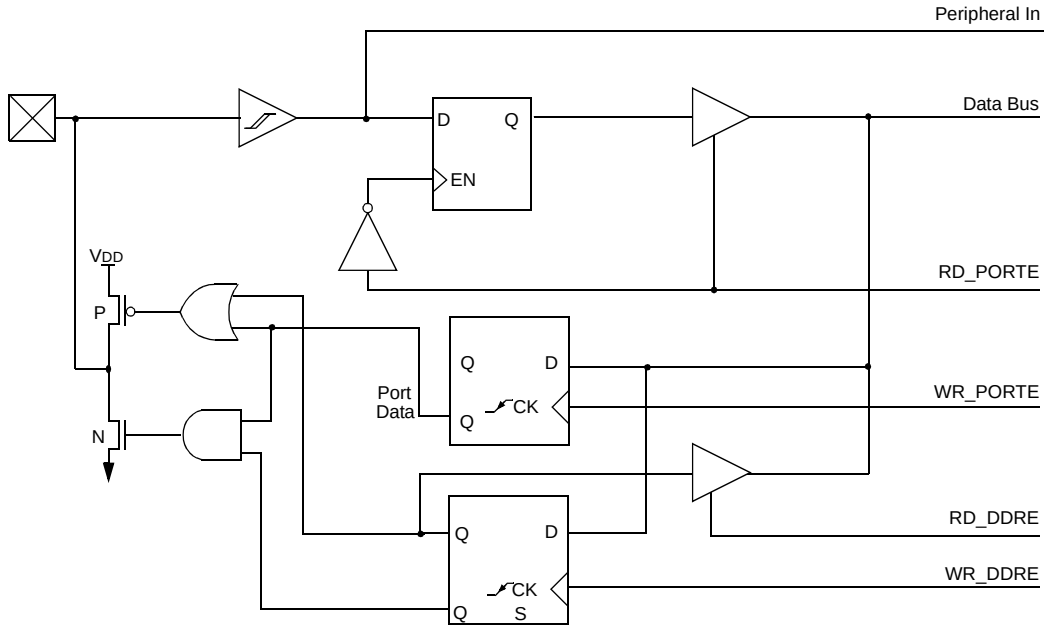
**FIGURE 10-11: BLOCK DIAGRAM OF RE2:RE0 (IN I/O PORT MODE)**




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


© 2000 Microchip Technology Inc.

**FIGURE 10-12: BLOCK DIAGRAM OF RE3/CAP4 PORT PIN**




**Note:** I/O pin has protection diodes to VDD and VSS.

**TABLE 10-9: PORTE FUNCTIONS**

| Name | Bit | Buffer Type | Function |
|---|---|---|---|
| RE0/$\overline{ALE}$ | bit0 | TTL | Input/output or system bus Address Latch Enable ($\overline{ALE}$) control pin. |
| RE1/$\overline{OE}$ | bit1 | TTL | Input/output or system bus Output Enable ($\overline{OE}$) control pin. |
| RE2/$\overline{WR}$ | bit2 | TTL | Input/output or system bus Write ($\overline{WR}$) control pin. |
| RE3/CAP4 | bit3 | ST | Input/output or Capture4 input pin. |

Legend: TTL = TTL input, ST = Schmitt Trigger input

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

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{MCLR}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 15h, Bank 1 | PORTE | — | — | — | — | RE3/CAP4 | RE2/WR | RE1/OE | RE0/ALE | ---- xxxx | ---- uuuu |
| 14h, Bank 1 | DDRE | Data Direction Register for PORTE | | | | | | | | ---- 1111 | ---- 1111 |
| 14h, Bank 7 | CA4L | Capture4 Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 15h, Bank 7 | CA4H | Capture4 High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 16h, Bank 7 | TCON3 | — | CA4OVF | CA3OVF | CA4ED1 | CA4ED0 | CA3ED1 | CA3ED0 | PWM3ON | -000 0000 | -000 0000 |

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

## 10.6 PORTF and DDRF Registers

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

All eight bits of PORTF are multiplexed with 8 channels of the 10-bit A/D converter.

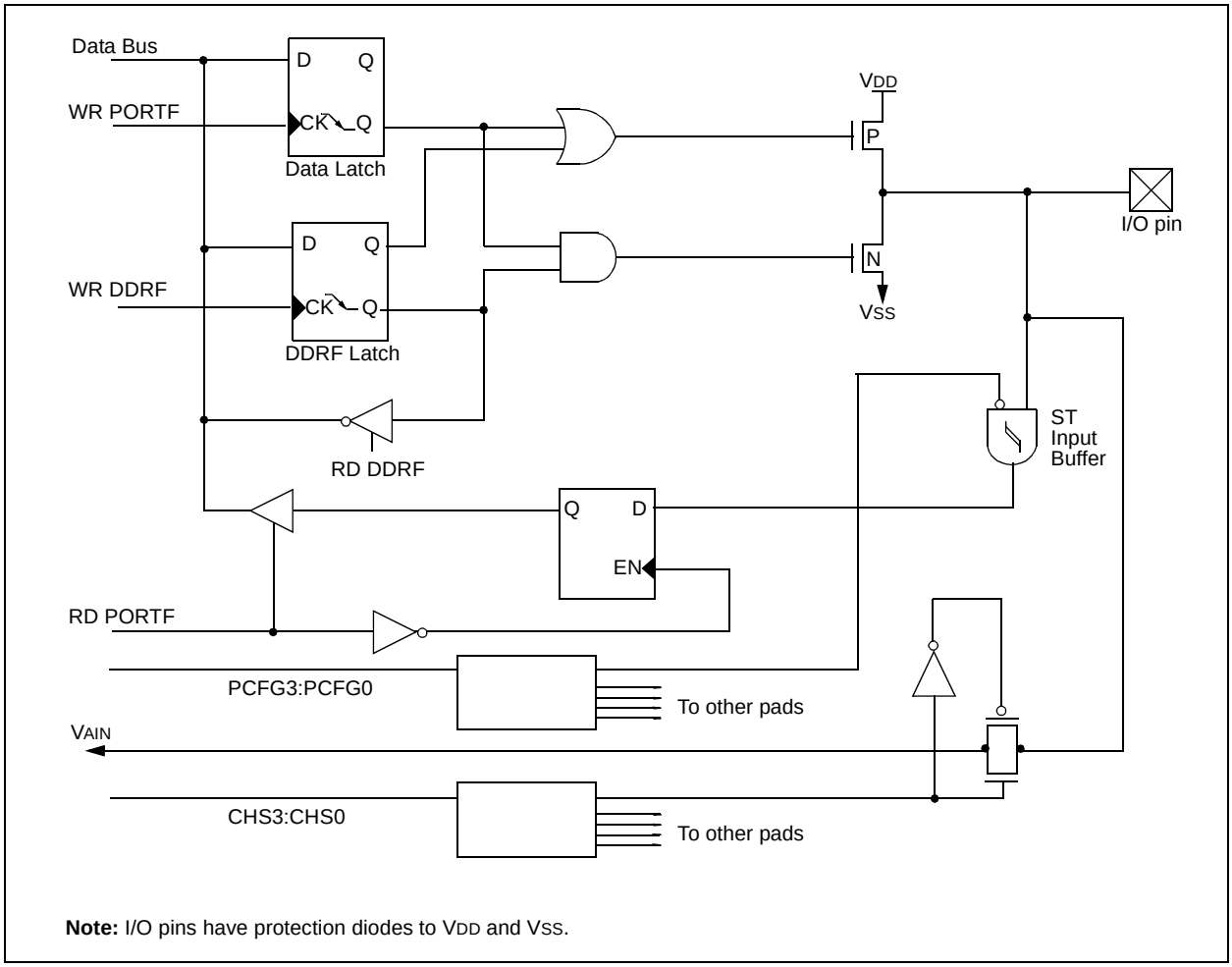
Upon RESET, the entire Port is automatically configured as analog inputs and must be configured in software to be a digital I/O.

Example 10-6 shows an instruction sequence to initialize PORTF. The Bank Select Register (BSR) must be selected to Bank 5 for the port to be initialized. The following example uses the MOVLB instruction to load the BSR register for bank selection.

**EXAMPLE 10-6: INITIALIZING PORTF**

```
MOVLB  5          ; Select Bank 5
MOVWF  0x0E       ; Configure PORTF as
MOVWF  ADCON1     ; Digital
CLRF   PORTF, F   ; Initialize PORTF data
                  ;   latches before
                  ;   the data direction
                  ;   register
MOVLW   0x03      ; Value used to init
                  ;   data direction
MOVWF   DDRF      ; Set RF<1:0> as inputs
                  ;   RF<7:2> as outputs
```

**FIGURE 10-13: BLOCK DIAGRAM OF RF7:RF0**




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


© 2000 Microchip Technology Inc.

**TABLE 10-11: PORTF FUNCTIONS**

| Name | Bit | Buffer Type | Function |
|---|---|---|---|
| RF0/AN4 | bit0 | ST | Input/output or analog input 4. |
| RF1/AN5 | bit1 | ST | Input/output or analog input 5. |
| RF2/AN6 | bit2 | ST | Input/output or analog input 6. |
| RF3/AN7 | bit3 | ST | Input/output or analog input 7. |
| RF4/AN8 | bit4 | ST | Input/output or analog input 8. |
| RF5/AN9 | bit5 | ST | Input/output or analog input 9. |
| RF6/AN10 | bit6 | ST | Input/output or analog input 10. |
| RF7/AN11 | bit7 | ST | Input/output or analog input 11. |

Legend: ST = Schmitt Trigger input

**TABLE 10-12: REGISTERS/BITS ASSOCIATED WITH PORTF**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 10h, Bank 5 | DDRF | Data Direction Register for PORTF | | | | | | | | 1111 1111 | 1111 1111 |
| 11h, Bank 5 | PORTF | RF7/ AN11 | RF6/ AN10 | RF5/ AN9 | RF4/ AN8 | RF3/ AN7 | RF2/ AN6 | RF1/ AN5 | RF0/ AN4 | 0000 0000 | 0000 0000 |
| 15h, Bank 5 | ADCON1 | ADCS1 | ADCS0 | ADFM | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 000- 0000 | 000- 0000 |

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

## 10.7 PORTG and DDRG Registers

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

The lower four bits of PORTG are multiplexed with four channels of the 10-bit A/D converter.

The remaining bits of PORTG are multiplexed with peripheral output and inputs. RG4 is multiplexed with the CAP3 input, RG5 is multiplexed with the PWM3 output, RG6 and RG7 are multiplexed with the USART2 functions.

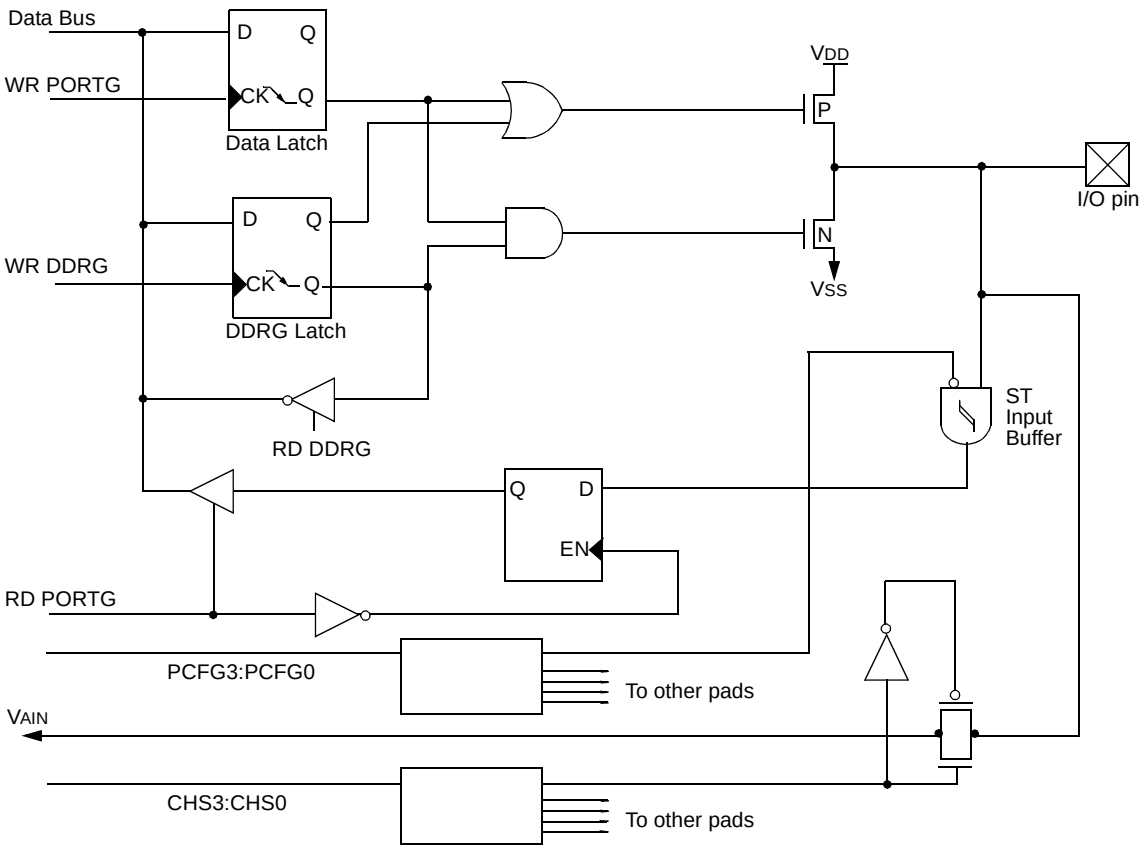
Upon RESET, RG3:RG0 is automatically configured as analog inputs and must be configured in software to be a digital I/O.

Example 10-7 shows the instruction sequence to initialize PORTG. The Bank Select Register (BSR) must be selected to Bank 5 for the port to be initialized. The following example uses the MOVLB instruction to load the BSR register for bank selection.

**EXAMPLE 10-7: INITIALIZING PORTG**

```
MOVLB  5              ; Select Bank 5
MOVLW  0x0E           ; Configure PORTG as
MOVPF  WREG, ADCON1   ; digital
CLRF   PORTG, F      ; Initialize PORTG data
                      ;   latches before
                      ;   the data direction
                      ;   register
MOVLW   0x03          ; Value used to init
                      ;   data direction
MOVWF   DDRG         ; Set RG<1:0> as inputs
                      ;   RG<7:2> as outputs
```

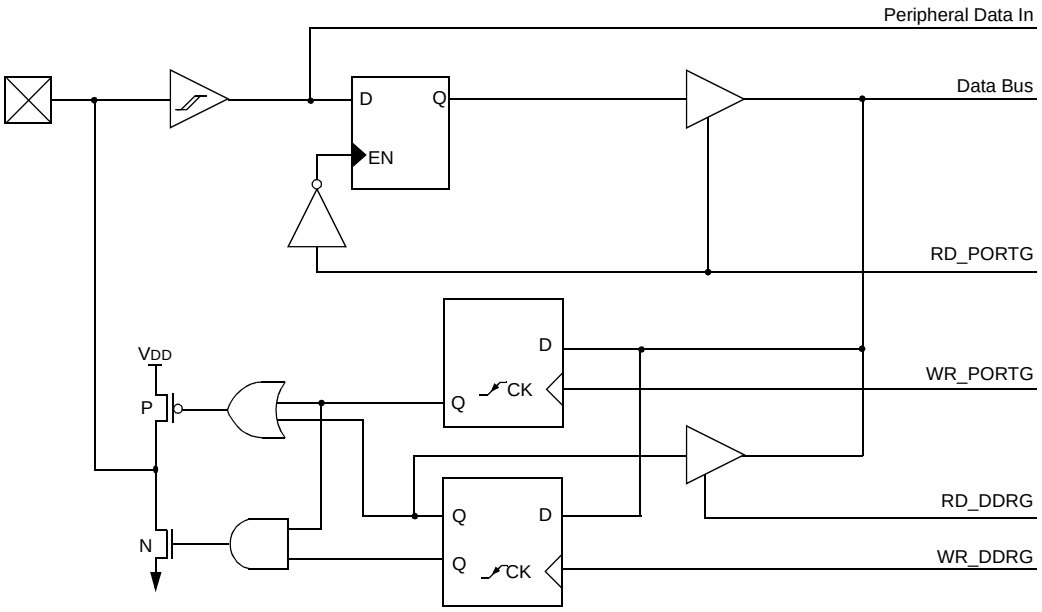
**FIGURE 10-14: BLOCK DIAGRAM OF RG3:RG0**




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

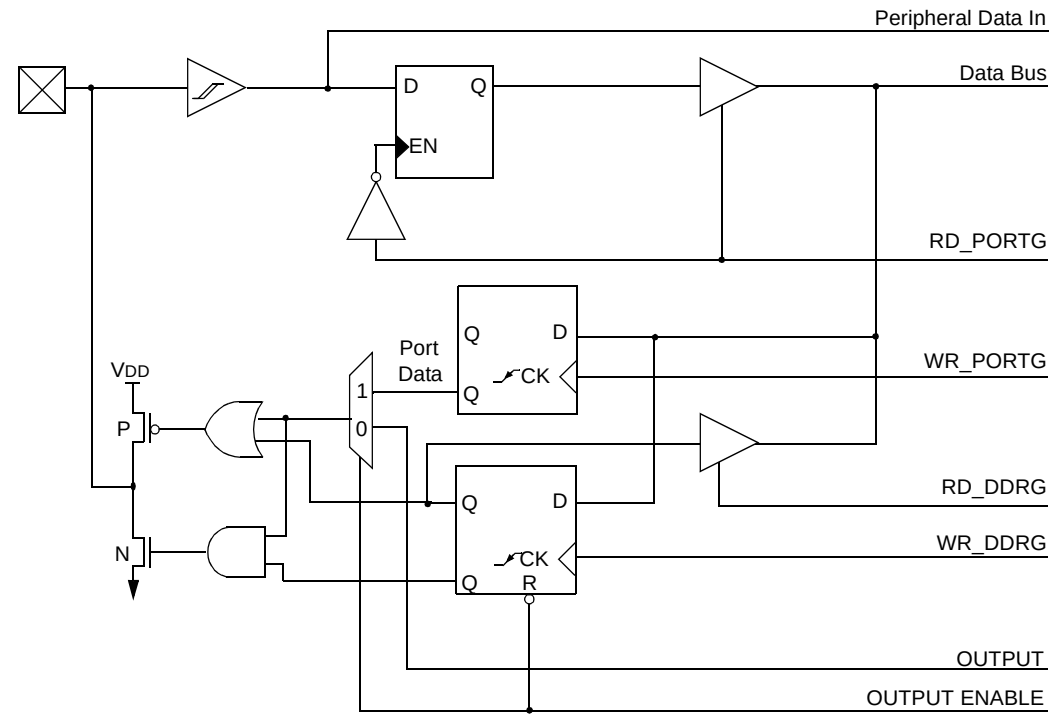

© 2000 Microchip Technology Inc.

**FIGURE 10-15: RG4 BLOCK DIAGRAM**




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

**FIGURE 10-16: RG7:RG5 BLOCK DIAGRAM**




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

**TABLE 10-13: PORTG FUNCTIONS**

| Name | Bit | Buffer Type | Function |
|---|---|---|---|
| RG0/AN3 | bit0 | ST | Input/output or analog input 3. |
| RG1/AN2 | bit1 | ST | Input/output or analog input 2. |
| RG2/AN1/VREF- | bit2 | ST | Input/output or analog input 1 or the ground reference voltage. |
| RG3/AN0/VREF+ | bit3 | ST | Input/output or analog input 0 or the positive reference voltage. |
| RG4/CAP3 | bit4 | ST | Input/output or the Capture3 input pin. |
| RG5/PWM3 | bit5 | ST | Input/output or the PWM3 output pin. |
| RG6/RX2/DT2 | bit6 | ST | Input/output or the USART2 (SCI) Asynchronous Receive or USART2 (SCI) Synchronous Data. |
| RG7/TX2/CK2 | bit7 | ST | Input/output or the USART2 (SCI) Asynchronous Transmit or USART2 (SCI) Synchronous Clock. |

Legend: ST = Schmitt Trigger input

**TABLE 10-14: REGISTERS/BITS ASSOCIATED WITH PORTG**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 12h, Bank 5 | DDRG | Data Direction Register for PORTG | | | | | | | | 1111 1111 | 1111 1111 |
| 13h, Bank 5 | PORTG | RG7/ TX2/CK2 | RG6/ RX2/DT2 | RG5/ PWM3 | RG4/ CAP3 | RG3/ AN0 | RG2/ AN1 | RG1/ AN2 | RG0/ AN3 | xxxx 0000 | uuuu 0000 |
| 15h, Bank 5 | ADCON1 | ADCS1 | ADCS0 | ADFM | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 000- 0000 | 000- 0000 |

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


© 2000 Microchip Technology Inc.

## 10.8 PORTH and DDRH Registers (PIC17C76X only)

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

The upper four bits of PORTH are multiplexed with 4 channels of the 10-bit A/D converter.

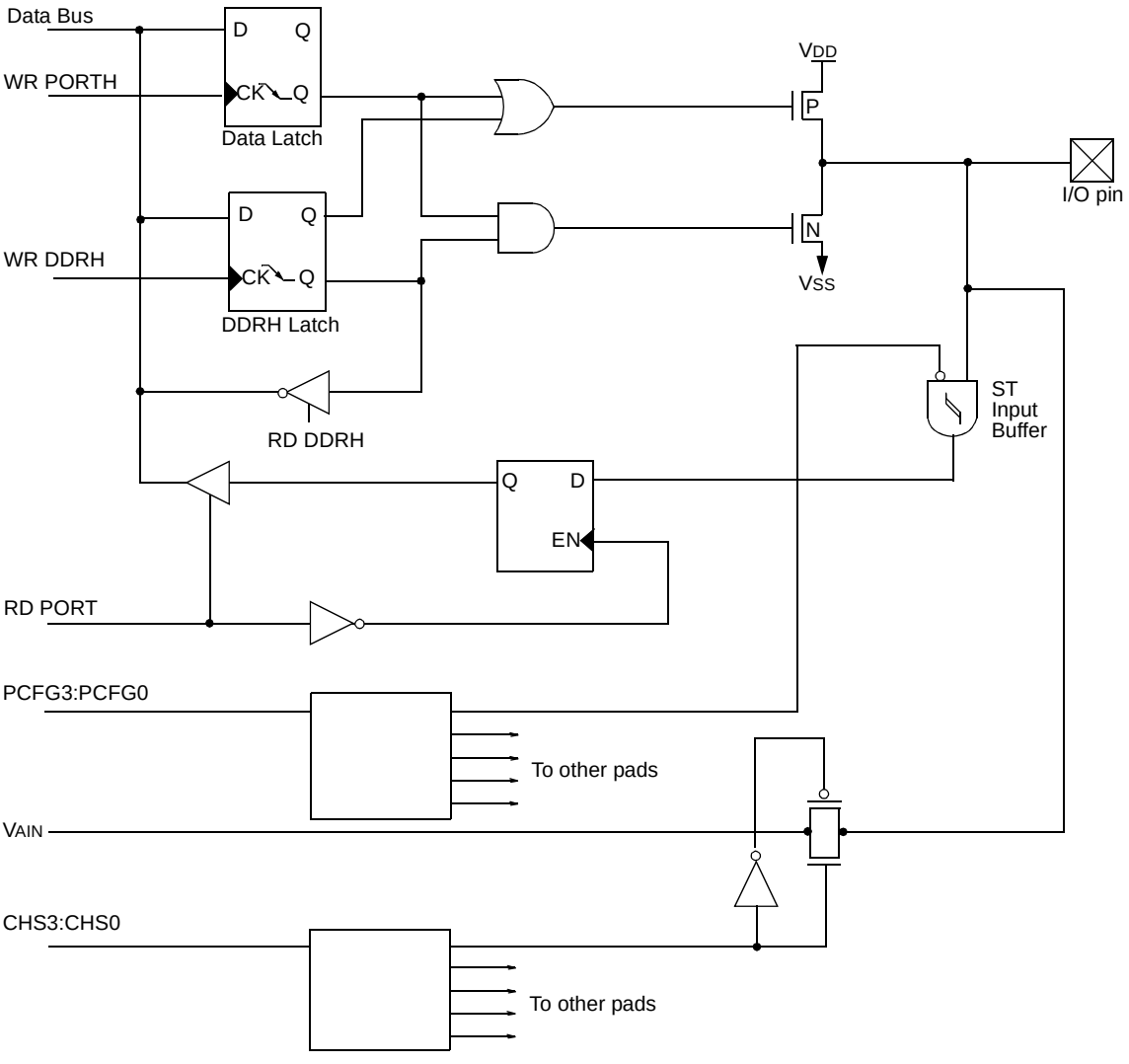
The remaining bits of PORTH are general purpose I/O.

Upon RESET, RH7:RH4 are automatically configured as analog inputs and must be configured in software to be a digital I/O.

**EXAMPLE 10-8: INITIALIZING PORTH**

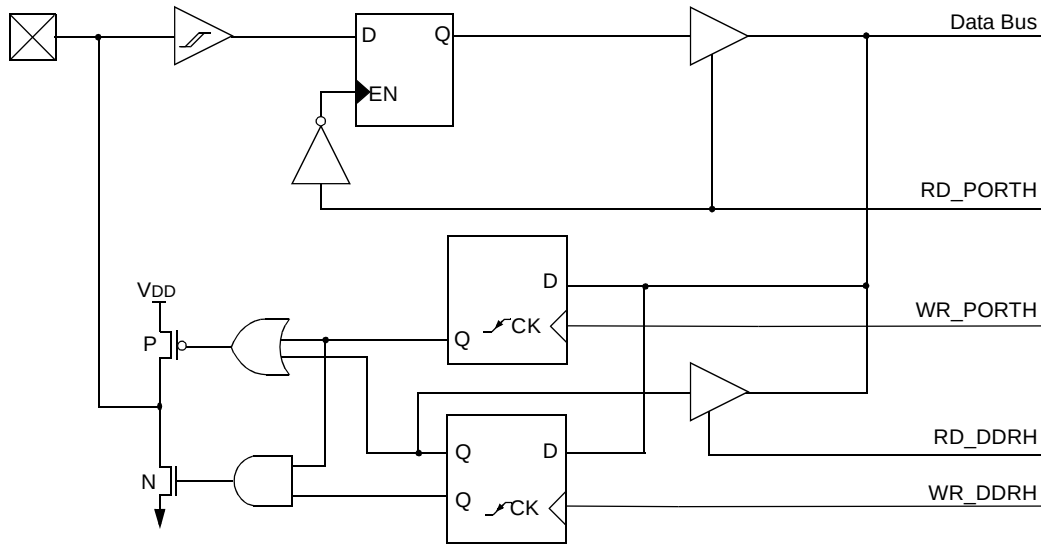
```
MOVLB   8           ; Select Bank 8
MOVLW   0x0E        ; Configure PORTH as
MOVPF   ADCON1      ; digital
CLRF    PORTH, F    ; Initialize PORTH data
                    ;   latches before
                    ;   the data direction
                    ;   register
MOVLW    0x03       ; Value used to init
                    ;   data direction
MOVWF    DDRH       ; Set RH<1:0> as inputs
                    ;   RH<7:2> as outputs
```

**FIGURE 10-17: BLOCK DIAGRAM OF RH7:RH4**




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

**FIGURE 10-18: RH3:RH0 BLOCK DIAGRAM**




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

**TABLE 10-15: PORTH FUNCTIONS**

| Name | Bit | Buffer Type | Function |
|---|---|---|---|
| RH0 | bit0 | ST | Input/output. |
| RH1 | bit1 | ST | Input/output. |
| RH2 | bit2 | ST | Input/output. |
| RH3 | bit3 | ST | Input/output. |
| RH4/AN12 | bit4 | ST | Input/output or analog input 12. |
| RH5/AN13 | bit5 | ST | Input/output or analog input 13. |
| RH6/AN14 | bit6 | ST | Input/output or analog input 14. |
| RH7/AN15 | bit7 | ST | Input/output or analog input 15. |

Legend: ST = Schmitt Trigger input

**TABLE 10-16: REGISTERS/BITS ASSOCIATED WITH PORTH**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 10h, Bank 8 | DDRH | Data Direction Register for PORTH | | | | | | | | 1111 1111 | 1111 1111 |
| 11h, Bank 8 | PORTH | RH7/AN15 | RH6/AN14 | RH5/AN13 | RH4/AN12 | RH3 | RH2 | RH1 | RH0 | 0000 xxxx | 0000 uuuu |
| 15h, Bank 5 | ADCON1 | ADCS1 | ADCS0 | ADFM | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 000- 0000 | 000- 0000 |

Legend: x = unknown, u = unchanged


© 2000 Microchip Technology Inc.

## 10.9 PORTJ and DDRJ Registers (PIC17C76X only)

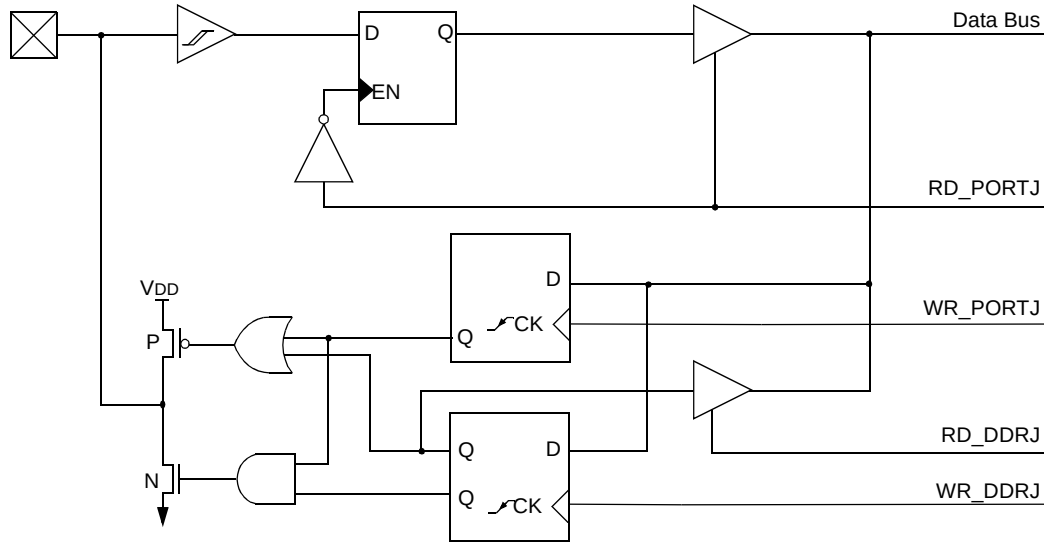
PORTJ is an 8-bit wide, bi-directional port. The corresponding data direction register is DDRJ. A '1' in DDRJ configures the corresponding port pin as an input. A '0' in the DDRJ register configures the corresponding port pin as an output. Reading PORTJ reads the status of the pins, whereas writing to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

PORTJ is a general purpose I/O port.

**EXAMPLE 10-9: INITIALIZING PORTJ**

```
MOVLB   8           ; Select Bank 8
CLRF    PORTJ, F    ; Initialize PORTJ data
                    ; latches before setting
                    ; the data direction
                    ; register
MOVLW   0xCF        ; Value used to initialize
                    ; data direction
MOVWF   DDRJ        ; Set RJ<3:0> as inputs
                    ; RJ<5:4> as outputs
                    ; RJ<7:6> as inputs
```

**FIGURE 10-19: PORTJ BLOCK DIAGRAM**




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

**TABLE 10-17: PORTJ FUNCTIONS**

| Name | Bit | Buffer Type | Function |
|---|---|---|---|
| RJ0 | bit0 | ST | Input/output |
| RJ1 | bit1 | ST | Input/output |
| RJ2 | bit2 | ST | Input/output |
| RJ3 | bit3 | ST | Input/output |
| RJ4 | bit4 | ST | Input/output |
| RJ5 | bit5 | ST | Input/output |
| RJ6 | bit6 | ST | Input/output |
| RJ7 | bit7 | ST | Input/output |

Legend: ST = Schmitt Trigger input

**TABLE 10-18: REGISTERS/BITS ASSOCIATED WITH PORTJ**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on, POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 12h, Bank 8 | DDRJ | Data Direction Register for PORTJ | | | | | | | | 1111 1111 | 1111 1111 |
| 13h, Bank 8 | PORTJ | RJ7 | RJ6 | RJ5 | RJ4 | RJ3 | RJ2 | RJ1 | RJ0 | xxxx xxxx | uuuu uuuu |

Legend: x = unknown, u = unchanged


© 2000 Microchip Technology Inc.

## 10.10 I/O Programming Considerations

### 10.10.1 BI-DIRECTIONAL I/O PORTS

Any instruction which writes, operates internally as a read, followed by a write operation. For example, the BCF and BSF instructions read the register into the CPU, execute the bit operation and write the result back to the register. Caution must be used when these instructions are applied to a port with both inputs and outputs defined. For example, a BSF operation on bit5 of PORTB, will cause all eight bits of PORTB to be read into the CPU. Then the BSF operation takes place on bit5 and PORTB is written to the output latches. If another bit of PORTB is used as a bi-directional I/O pin (e.g. bit0) and it is defined as an input at this time, the input signal present on the pin itself would be read into the CPU and 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.

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

Example 10-10 shows the possible effect of two sequential read-modify-write instructions on an I/O port.

**EXAMPLE 10-10: READ-MODIFY-WRITE INSTRUCTIONS ON AN I/O PORT**

```
; Initial PORT settings: PORTB<7:4> Inputs
;                        PORTB<3:0> Outputs
; PORTB<7:6> have pull-ups and are
; not connected to other circuitry
;
;                      PORT latch  PORT pins
;                      ----------  ---------
;
   BCF   PORTB, 7    ; 01pp pppp   11pp pppp
   BCF   PORTB, 6    ; 10pp pppp   11pp pppp

   BCF   DDRB, 7     ; 10pp pppp   11pp pppp
   BCF   DDRB, 6     ; 10pp pppp   10pp pppp
;
; Note that the user may have expected the
; pin values to be 00pp pppp. The 2nd BCF
; caused RB7 to be latched as the pin value
; (High).
```

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

### 10.10.2 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 10-20). Therefore, care must be exercised if a write followed by a read operation is carried out on the same I/O port. The sequence of instructions should be such to allow the pin voltage to stabilize (load dependent) before executing the instruction that reads the values on that I/O port. Otherwise, the previous state of that pin may be read into the CPU, rather than the "new" state. When in doubt, it is better to separate these instructions with a `NOP`, or another instruction not accessing this I/O port.

Figure 10-21 shows the I/O model which causes this situation. As the effective capacitance (C) becomes larger, the rise/fall time of the I/O pin increases. As the device frequency increases, or the effective capacitance increases, the possibility of this subsequent PORTx read-modify-write instruction issue increases. This effective capacitance includes the effects of the board traces.

The best way to address this is to add a series resistor at the I/O pin. This resistor allows the I/O pin to get to the desired level before the next instruction.

The use of `NOP` instructions between the subsequent PORTx read-modify-write instructions, is a lower cost solution, but has the issue that the number of `NOP` instructions is dependent on the effective capacitance C and the frequency of the device.

**FIGURE 10-20: SUCCESSIVE I/O OPERATION**

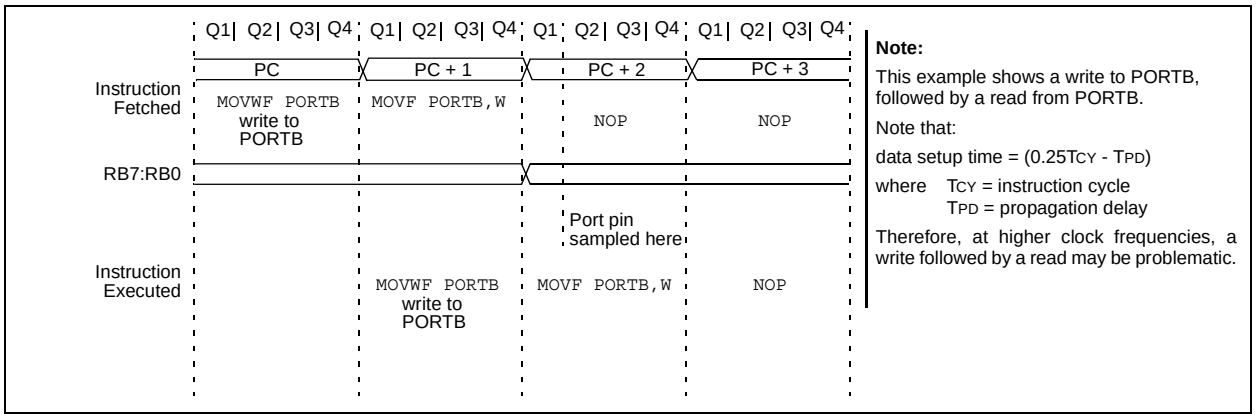
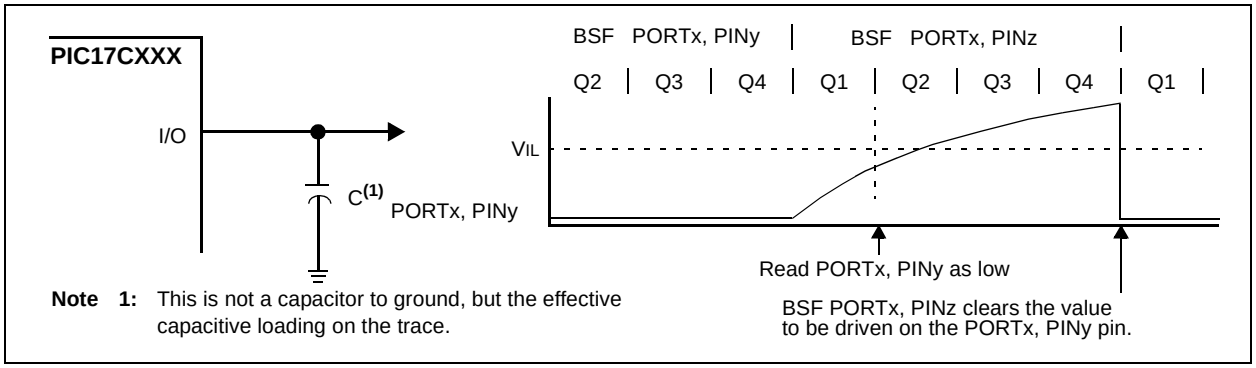



**FIGURE 10-21: I/O CONNECTION ISSUES**




**Note 1:** This is not a capacitor to ground, but the effective capacitive loading on the trace.


© 2000 Microchip Technology Inc.

## 11.0 OVERVIEW OF TIMER RESOURCES

The PIC17C7XX has four timer modules. Each module can generate an interrupt to indicate that an event has occurred. These timers are called:

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

For enhanced time base functionality, four input Captures and three Pulse Width Modulation (PWM) outputs are possible. The PWMs use the Timer1 and Timer2 resources and the input Captures use the Timer3 resource.

### 11.1 Timer0 Overview

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

When Timer0 uses an external clock source, it has the flexibility to allow user selection of the incrementing edge, rising or falling.

The Timer0 module also has a programmable prescaler. The T0PS3:T0PS0 bits (T0STA<4:1>) determine the prescale value. TMR0 can increment at the following rates: 1:1, 1:2, 1:4, 1:8, 1:16, 1:32, 1:64, 1:128, 1:256.

Synchronization of the external clock occurs after the prescaler. When the prescaler is used, the external clock frequency may be higher than the device's frequency. The maximum external frequency on the T0CKI pin is 50 MHz, given the high and low time requirements of the clock.

### 11.2 Timer1 Overview

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

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

### 11.3 Timer2 Overview

The Timer2 module is an 8-bit timer/counter with an 8-bit period register (PR2). When the TMR2 value rolls over from the period match value to 0h, the TMR2IF flag is set and an interrupt will be generated, if enabled. In Counter mode, the clock comes from the RB4/TCLK12 pin, which can also provide the clock for the Timer1 module.

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

### 11.4 Timer3 Overview

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

When operating in the four Capture modes, the period registers become the second (of four) 16-bit capture registers.

### 11.5 Role of the Timer/Counters

The timer modules are general purpose, but have dedicated resources associated with them. TImer1 and Timer2 are the time bases for the three Pulse Width Modulation (PWM) outputs, while Timer3 is the time base for the four input captures.

**NOTES:**


© 2000 Microchip Technology Inc.

## 12.0 TIMER0

The Timer0 module consists of a 16-bit timer/counter, TMR0. The high byte is register TMR0H and the low byte is register TMR0L. A software programmable 8-bit prescaler makes Timer0 an effective 24-bit overflow timer. The clock source is software programmable as either the internal instruction clock, or an external clock on the RA1/T0CKI pin. The control bits for this module are in register T0STA (Figure 12-1).

**REGISTER 12-1: T0STA REGISTER (ADDRESS: 05h, UNBANKED)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 |
|---|---|---|---|---|---|---|---|
| INTEDG | T0SE | T0CS | T0PS3 | T0PS2 | T0PS1 | T0PS0 | — |
| bit 7 | | | | | | | bit 0 |

bit 7 **INTEDG**: RA0/INT Pin Interrupt Edge Select bit
This bit selects the edge upon which the interrupt is detected.
1 = Rising edge of RA0/INT pin generates interrupt
0 = Falling edge of RA0/INT pin generates interrupt

bit 6 **T0SE**: Timer0 Clock Input Edge Select bit
This bit selects the edge upon which TMR0 will increment.

When T0CS = 0 (External Clock):
1 = Rising edge of RA1/T0CKI pin increments TMR0 and/or sets the T0CKIF bit
0 = Falling edge of RA1/T0CKI pin increments TMR0 and/or sets the T0CKIF bit

When T0CS = 1 (Internal Clock):
Don't care

bit 5 **T0CS**: Timer0 Clock Source Select bit
This bit selects the clock source for TMR0.
1 = Internal instruction clock cycle ($T_{CY}$)
0 = External clock input on the T0CKI pin

bit 4-1 **T0PS3:T0PS0**: Timer0 Prescale Selection bits
These bits select the prescale value for TMR0.

| T0PS3:T0PS0 | Prescale Value |
|---|---|
| 0000 | 1:1 |
| 0001 | 1:2 |
| 0010 | 1:4 |
| 0011 | 1:8 |
| 0100 | 1:16 |
| 0101 | 1:32 |
| 0110 | 1:64 |
| 0111 | 1:128 |
| 1xxx | 1:256 |

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

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

## 12.1 Timer0 Operation

When the T0CS (T0STA<5>) bit is set, TMR0 increments on the internal clock. When T0CS is clear, TMR0 increments on the external clock (RA1/T0CKI pin). The external clock edge can be selected in software. When the T0SE (T0STA<6>) bit is set, the timer will increment on the rising edge of the RA1/T0CKI pin. When T0SE is clear, the timer will increment on the falling edge of the RA1/T0CKI pin. The prescaler can be programmed to introduce a prescale of 1:1 to 1:256. The timer increments from 0000h to FFFFh and rolls over to 0000h. On overflow, the TMR0 Interrupt Flag bit (T0IF) is set. The TMR0 interrupt can be masked by clearing the corresponding TMR0 Interrupt Enable bit (T0IE). The TMR0 Interrupt Flag bit (T0IF) is automatically cleared when vectoring to the TMR0 interrupt vector.

## 12.2 Using Timer0 with External Clock

When an external clock input is used for Timer0, it is synchronized with the internal phase clocks. Figure 12-2 shows the synchronization of the external clock. This synchronization is done after the prescaler. The output of the prescaler (PSOUT) is sampled twice in every instruction cycle to detect a rising or a falling edge. The timing requirements for the external clock are detailed in the electrical specification section.

### 12.2.1 DELAY FROM EXTERNAL CLOCK EDGE

Since the prescaler output is synchronized with the internal clocks, there is a small delay from the time the external clock edge occurs to the time TMR0 is actually incremented. Figure 12-2 shows that this delay is between 3TOSC and 7TOSC. Thus, for example, measuring the interval between two edges (e.g. period) will be accurate within ±4TOSC (±121 ns @ 33 MHz).

**FIGURE 12-1: TIMER0 MODULE BLOCK DIAGRAM**

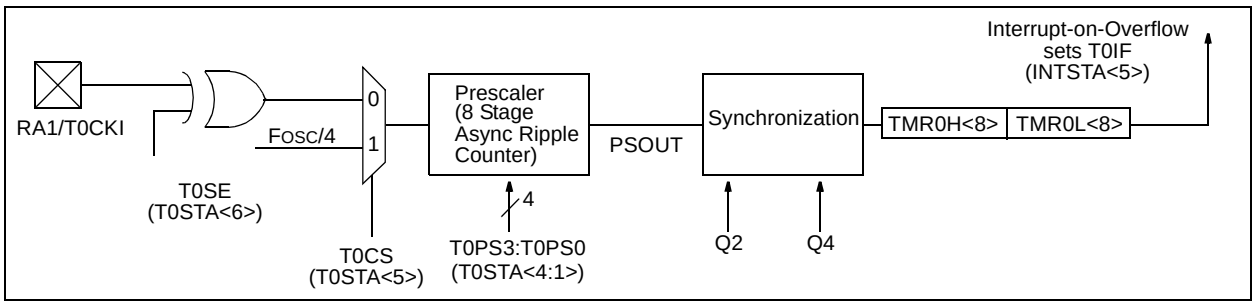
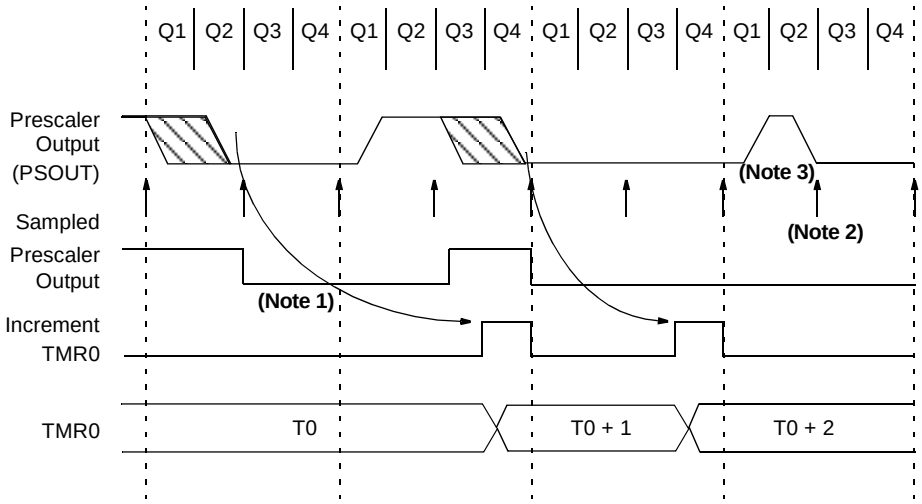



**FIGURE 12-2: TMR0 TIMING WITH EXTERNAL CLOCK (INCREMENT ON FALLING EDGE)**




**Note 1:** The delay from the T0CKI edge to the TMR0 increment is 3Tosc to 7Tosc.
**2:** ↑ = PSOUT is sampled here.
**3:** The PSOUT high time is too short and is missed by the sampling circuit.


© 2000 Microchip Technology Inc.

## 12.3 Read/Write Consideration for TMR0

Although TMR0 is a 16-bit timer/counter, only 8-bits at a time can be read or written during a single instruction cycle. Care must be taken during any read or write.

### 12.3.1 READING 16-BIT VALUE

The problem in reading the entire 16-bit value is that after reading the low (or high) byte, its value may change from FFh to 00h.

Example 12-1 shows a 16-bit read. To ensure a proper read, interrupts must be disabled during this routine.

**EXAMPLE 12-1: 16-BIT READ**

```
MOVPF   TMR0L, TMPLO    ;read low tmr0
MOVPF   TMR0H, TMPHI    ;read high tmr0
MOVFP   TMPLO, WREG     ;tmplo -> wreg
CPFSLT  TMR0L           ;tmr0l < wreg?
RETURN                  ;no then return
MOVPF   TMR0L, TMPLO    ;read low tmr0
MOVPF   TMR0H, TMPHI    ;read high tmr0
RETURN                  ;return
```

### 12.3.2 WRITING A 16-BIT VALUE TO TMR0

Since writing to either TMR0L or TMR0H will effectively inhibit increment of that half of the TMR0 in the next cycle (following write), but not inhibit increment of the other half, the user must write to TMR0L first and TMR0H second, in two consecutive instructions, as shown in Example 12-2. The interrupt must be disabled. Any write to either TMR0L or TMR0H clears the prescaler.

**EXAMPLE 12-2: 16-BIT WRITE**

```
BSF     CPUSTA, GLINTD ; Disable interrupts
MOVFP   RAM_L, TMR0L   ;
MOVFP   RAM_H, TMR0H   ;
BCF     CPUSTA, GLINTD ; Done, enable
                       ;    interrupts
```

## 12.4 Prescaler Assignments

Timer0 has an 8-bit prescaler. The prescaler selection is fully under software control; i.e., it can be changed “on the fly” during program execution. Clearing the prescaler is recommended before changing its setting. The value of the prescaler is “unknown” and assigning a value that is less than the present value, makes it difficult to take this unknown time into account.

**FIGURE 12-3: TMR0 TIMING: WRITE HIGH OR LOW BYTE**

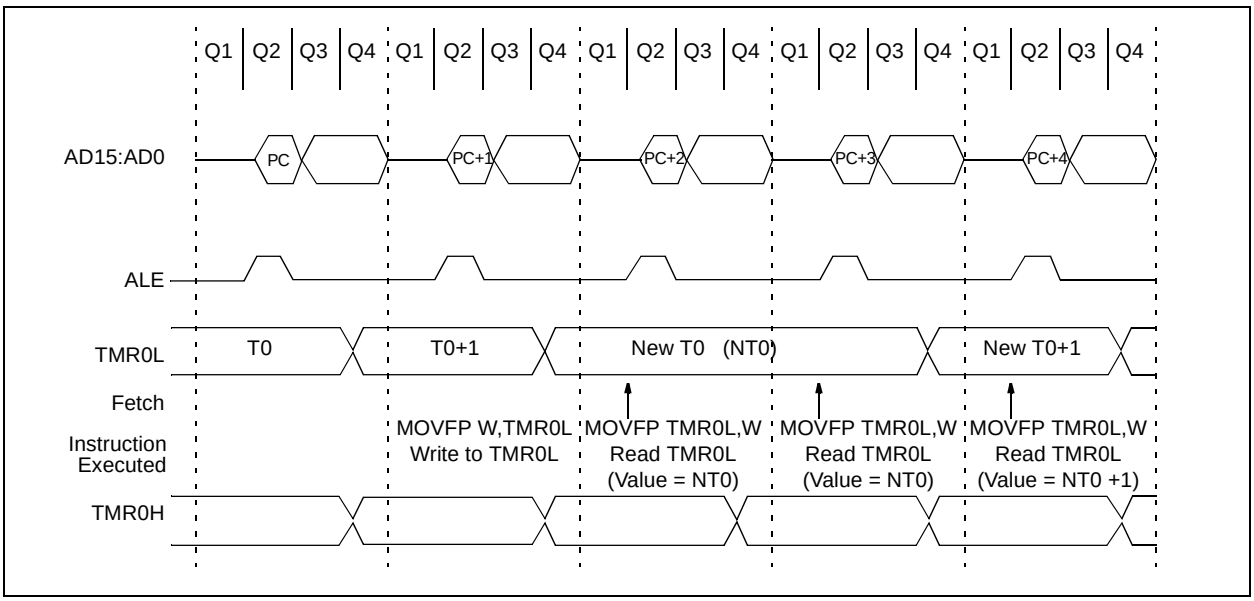
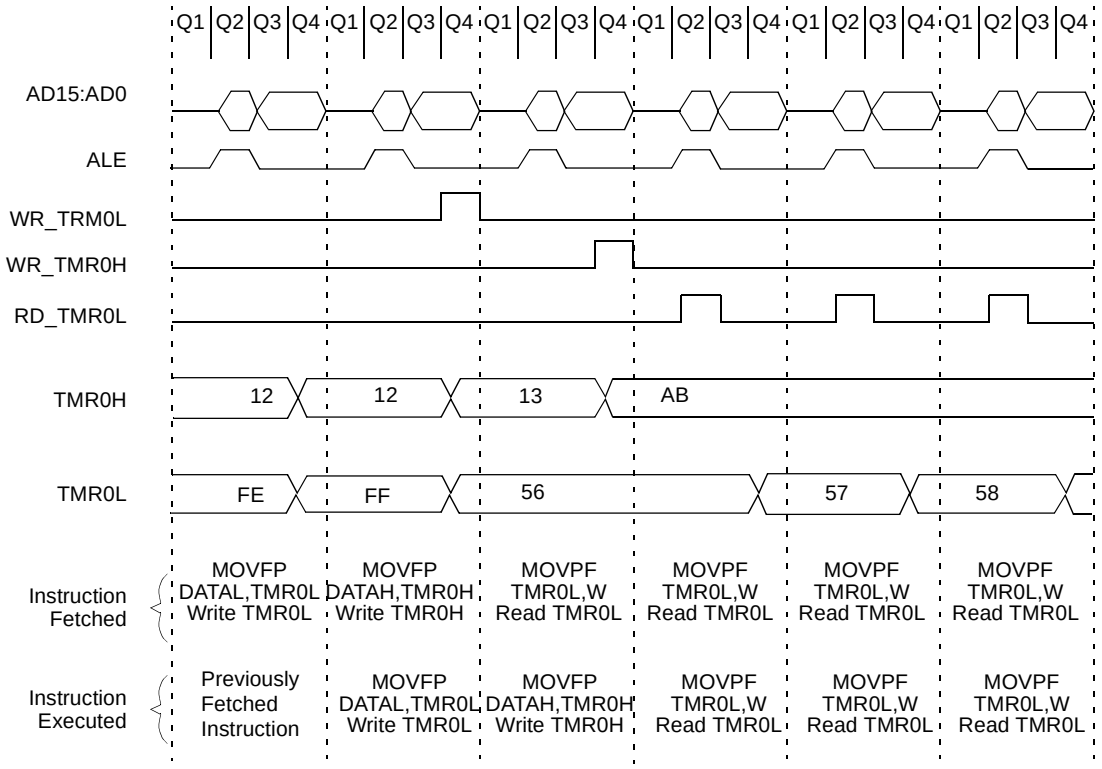


**FIGURE 12-4: TMR0 READ/WRITE IN TIMER MODE**




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

**TABLE 12-1: REGISTERS/BITS ASSOCIATED WITH TIMER0**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 05h, Unbanked | T0STA | INTEDG | T0SE | T0CS | T0PS3 | T0PS2 | T0PS1 | T0PS0 | — | 0000 000- | 0000 000- |
| 06h, Unbanked | CPUSTA | — | — | STKAV | GLINTD | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | --11 11qq | --11 qquu |
| 07h, Unbanked | INTSTA | PEIF | T0CKIF | T0IF | INTF | PEIE | T0CKIE | T0IE | INTE | 0000 0000 | 0000 0000 |
| 0Bh, Unbanked | TMR0L | TMR0 Register; Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 0Ch, Unbanked | TMR0H | TMR0 Register; High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |

Legend: x = unknown, u = unchanged, - = unimplemented, read as a '0', q = value depends on condition. Shaded cells are not used by Timer0.


© 2000 Microchip Technology Inc.

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

The PIC17C7XX has a wealth of timers and time based functions to ease the implementation of control applications. These time base functions include three PWM outputs and four Capture inputs.

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

Timer3 is a 16-bit timer/counter which uses the TMR3H and TMR3L registers. Timer3 also has two additional registers (PR3H/CA1H:PR3L/CA1L) that are configurable as a 16-bit period register or a 16-bit capture register. TMR3 can be software configured to increment from the internal system clock (FOSC/4), or from an external signal on the RB5/TCLK3 pin. Timer3 is the time base for all of the 16-bit captures.

Six other registers comprise the Capture2, Capture3, and Capture4 registers (CA2H:CA2L, CA3H:CA3L, and CA4H:CA4L).

Figure 13-1, Figure 13-2 and Figure 13-3 are the control registers for the operation of Timer1, Timer2 and Timer3, as well as PWM1, PWM2, PWM3, Capture1, Capture2, Capture3 and Capture4.

Table 13-1 shows the Timer resource requirements for these time base functions. Each timer is an open resource so that multiple functions may operate with it.

**TABLE 13-1: TIME-BASE FUNCTION/ RESOURCE REQUIREMENTS**

| Time Base Function | Timer Resource |
|---|---|
| PWM1 | Timer1 |
| PWM2 | Timer1 or Timer2 |
| PWM3 | Timer1 or Timer2 |
| Capture1 | Timer3 |
| Capture2 | Timer3 |
| Capture3 | Timer3 |
| Capture4 | Timer3 |

**REGISTER 13-1: TCON1 REGISTER (ADDRESS: 16h, BANK 3)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CA2ED1 | CA2ED0 | CA1ED1 | CA1ED0 | T16 | TMR3CS | TMR2CS | TMR1CS |
| bit 7 | | | | | | | bit 0 |

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

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

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

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

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

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

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

**REGISTER 13-2: TCON2 REGISTER (ADDRESS: 17h, BANK 3)**

| R-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CA2OVF | CA1OVF | PWM2ON | PWM1ON | CA1/$\overline{PR3}$ | TMR3ON | TMR2ON | TMR1ON |
| bit 7 | | | | | | | bit 0 |

bit 7 **CA2OVF**: Capture2 Overflow Status bit
This bit indicates that the capture value had not been read from the capture register pair (CA2H:CA2L) before the next capture event occurred. The capture register retains the oldest unread capture value (last capture before overflow). Subsequent capture events will not update the capture register with the TMR3 value until the capture register has been read (both bytes).
1 = Overflow occurred on Capture2 register
0 = No overflow occurred on Capture2 register

bit 6 **CA1OVF**: Capture1 Overflow Status bit
This bit indicates that the capture value had not been read from the capture register pair (PR3H/CA1H:PR3L/CA1L), before the next capture event occurred. The capture register retains the oldest unread capture value (last capture before overflow). Subsequent capture events will not update the capture register with the TMR3 value until the capture register has been read (both bytes).
1 = Overflow occurred on Capture1 register
0 = No overflow occurred on Capture1 register

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

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

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

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

bit 1 **TMR2ON**: Timer2 On bit
This bit controls the incrementing of the TMR2 register. When TMR2:TMR1 form the 16-bit timer (T16 is set), TMR2ON must be set. This allows the MSB of the timer to increment.
1 = Starts Timer2 (must be enabled if the T16 bit (TCON1<3>) is set)
0 = Stops Timer2

bit 0 **TMR1ON**: Timer1 On bit

When T16 is set (in 16-bit Timer mode):
1 = Starts 16-bit TMR2:TMR1
0 = Stops 16-bit TMR2:TMR1

When T16 is clear (in 8-bit Timer mode:
1 = Starts 8-bit Timer1
0 = Stops 8-bit Timer1

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |


© 2000 Microchip Technology Inc.

**REGISTER 13-3: TCON3 REGISTER (ADDRESS: 16h, BANK 7)**

| U-0 | R-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | CA4OVF | CA3OVF | CA4ED1 | CA4ED0 | CA3ED1 | CA3ED0 | PWM3ON |
| bit 7 | | | | | | | bit 0 |

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

bit 6 **CA4OVF**: Capture4 Overflow Status bit
This bit indicates that the capture value had not been read from the capture register pair (CA4H:CA4L) before the next capture event occurred. The capture register retains the oldest unread capture value (last capture before overflow). Subsequent capture events will not update the capture register with the TMR3 value until the capture register has been read (both bytes).
1 = Overflow occurred on Capture4 registers
0 = No overflow occurred on Capture4 registers

bit 5 **CA3OVF**: Capture3 Overflow Status bit
This bit indicates that the capture value had not been read from the capture register pair (CA3H:CA3L) before the next capture event occurred. The capture register retains the oldest unread capture value (last capture before overflow). Subsequent capture events will not update the capture register with the TMR3 value until the capture register has been read (both bytes).
1 = Overflow occurred on Capture3 registers
0 = No overflow occurred on Capture3 registers

bit 4-3 **CA4ED1:CA4ED0**: Capture4 Mode Select bits
00 = Capture on every falling edge
01 = Capture on every rising edge
10 = Capture on every 4th rising edge
11 = Capture on every 16th rising edge

bit 2-1 **CA3ED1:CA3ED0**: Capture3 Mode Select bits
00 = Capture on every falling edge
01 = Capture on every rising edge
10 = Capture on every 4th rising edge
11 = Capture on every 16th rising edge

bit 0 **PWM3ON**: PWM3 On bit
1 = PWM3 is enabled (the RG5/PWM3 pin ignores the state of the DDRG<5> bit)
0 = PWM3 is disabled (the RG5/PWM3 pin uses the state of the DDRG<5> bit for data direction)

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

## 13.1 Timer1 and Timer2

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

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

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

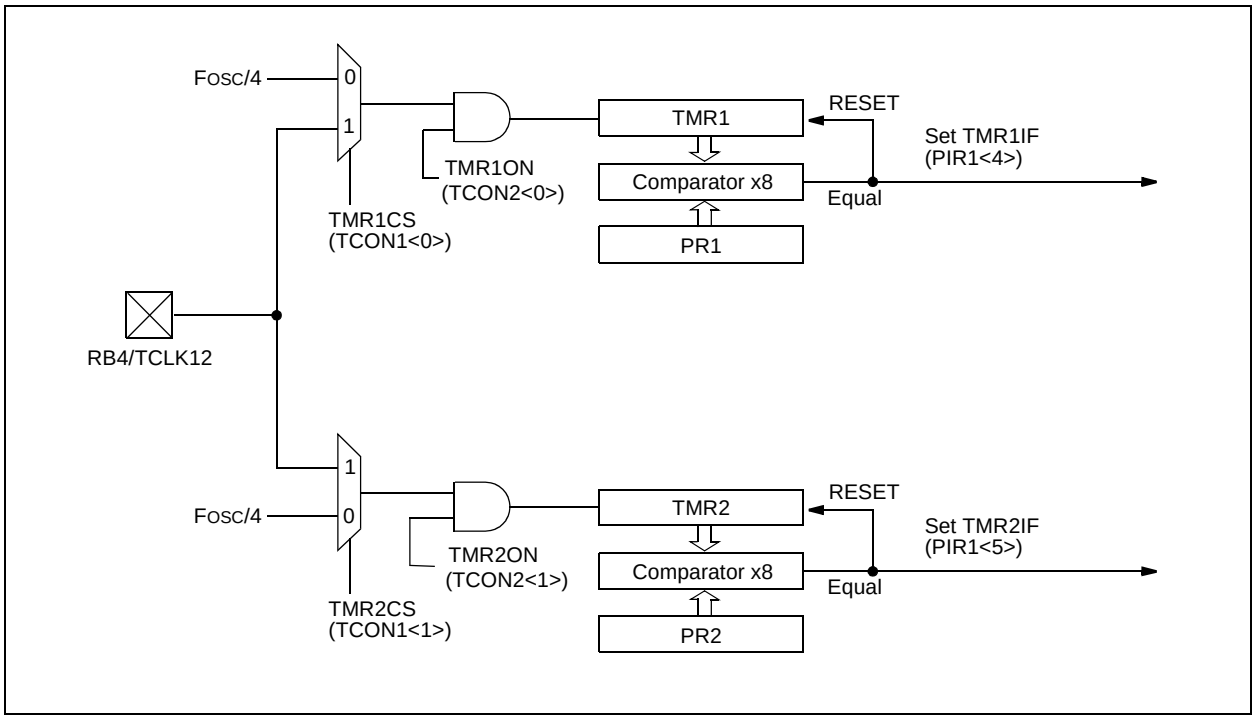
Each timer also has a corresponding interrupt enable bit (TMRxIE). The timer interrupt can be enabled/disabled by setting/clearing this bit. For peripheral interrupts to be enabled, the Peripheral Interrupt Enable bit must be set (PEIE = '1') and global interrupt must be enabled (GLINTD = '0').

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

#### 13.1.1.1 External Clock Input for Timer1 and Timer2

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

**FIGURE 13-1: TIMER1 AND TIMER2 IN TWO 8-BIT TIMER/COUNTER MODE**





© 2000 Microchip Technology Inc.

### 13.1.2 TIMER1 AND TIMER2 IN 16-BIT MODE

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

When selecting the clock source for the 16-bit timer, the TMR1CS bit controls the entire 16-bit timer and TMR2CS is a “don’t care”, however, ensure that TMR2ON is set (allows TMR2 to increment). When TMR1CS is clear, the timer increments once every instruction cycle (FOSC/4). When TMR1CS is set, the timer increments on every falling edge of the RB4/TCLK12 pin. For the 16-bit timer to increment, both TMR1ON and TMR2ON bits must be set (Table 13-2).

#### 13.1.2.1 External Clock Input for TMR2:TMR1

When TMR1CS is set, the 16-bit TMR2:TMR1 increments on the falling edge of clock input TCLK12. The input on the RB4/TCLK12 pin is sampled and synchronized by the internal phase clocks twice every instruction cycle. This causes a delay from the time a falling edge appears on RB4/TCLK12 to the time TMR2:TMR1 is actually incremented. For the external clock input timing requirements, see the Electrical Specification section.

**TABLE 13-2: TURNING ON 16-BIT TIMER**

| T16 | TMR2ON | TMR1ON | Result |
|---|---|---|---|
| 1 | 1 | 1 | 16-bit timer (TMR2:TMR1) ON |
| 1 | 0 | 1 | Only TMR1 increments |
| 1 | x | 0 | 16-bit timer OFF |
| 0 | 1 | 1 | Timers in 8-bit mode |

**FIGURE 13-2: TMR2 AND TMR1 IN 16-BIT TIMER/COUNTER MODE**

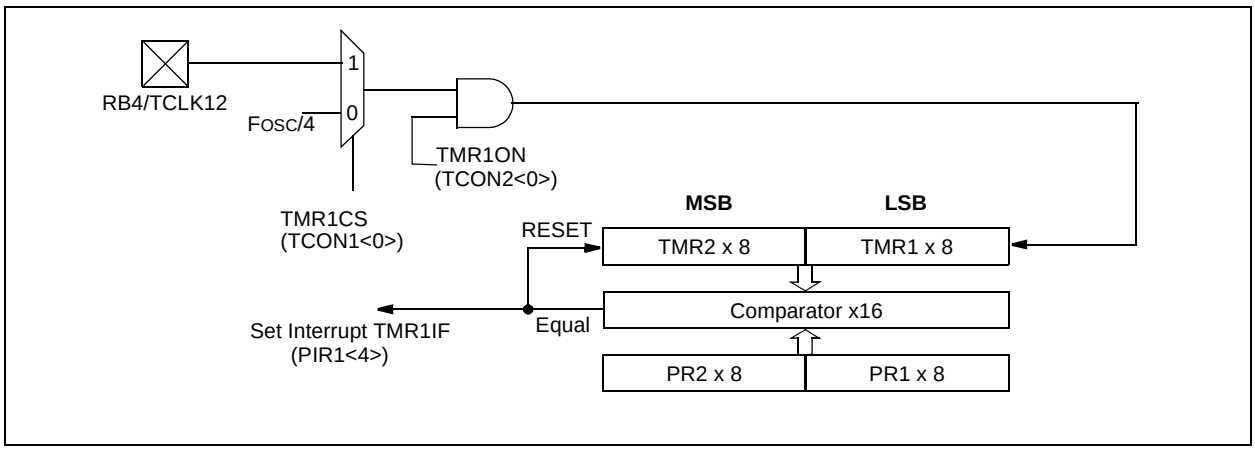


**TABLE 13-3: SUMMARY OF TIMER1, TIMER2 AND TIMER3 REGISTERS**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 16h, Bank 3 | TCON1 | CA2ED1 | CA2ED0 | CA1ED1 | CA1ED0 | T16 | TMR3CS | TMR2CS | TMR1CS | 0000 0000 | 0000 0000 |
| 17h, Bank 3 | TCON2 | CA2OVF | CA1OVF | PWM2ON | PWM1ON | CA1/$\overline{\text{PR3}}$ | TMR3ON | TMR2ON | TMR1ON | 0000 0000 | 0000 0000 |
| 16h, Bank 7 | TCON3 | — | CA4OVF | CA3OVF | CA4ED1 | CA4ED0 | CA3ED1 | CA3ED0 | PWM3ON | -000 0000 | -000 0000 |
| 10h, Bank 2 | TMR1 | Timer1's Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 11h, Bank 2 | TMR2 | Timer2's Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 16h, Bank 1 | PIR1 | RBIF | TMR3IF | TMR2IF | TMR1IF | CA2IF | CA1IF | TX1IF | RC1IF | x000 0010 | u000 0010 |
| 17h, Bank 1 | PIE1 | RBIE | TMR3IE | TMR2IE | TMR1IE | CA2IE | CA1IE | TX1IE | RC1IE | 0000 0000 | 0000 0000 |
| 07h, Unbanked | INTSTA | PEIF | T0CKIF | T0IF | INTF | PEIE | T0CKIE | T0IE | INTE | 0000 0000 | 0000 0000 |
| 06h, Unbanked | CPUSTA | — | — | STKAV | GLINTD | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | --11 11qq | --11 qquu |
| 14h, Bank 2 | PR1 | Timer1 Period Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 15h, Bank 2 | PR2 | Timer2 Period Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 10h, Bank 3 | PW1DCL | DC1 | DC0 | — | — | — | — | — | — | xx-- ---- | uu-- ---- |
| 11h, Bank 3 | PW2DCL | DC1 | DC0 | TM2PW2 | — | — | — | — | — | xx0- ---- | uu0- ---- |
| 10h, Bank 7 | PW3DCL | DC1 | DC0 | TM2PW3 | — | — | — | — | — | xx0- ---- | uu0- ---- |
| 12h, Bank 3 | PW1DCH | DC9 | DC8 | DC7 | DC6 | DC5 | DC4 | DC3 | DC2 | xxxx xxxx | uuuu uuuu |
| 13h, Bank 3 | PW2DCH | DC9 | DC8 | DC7 | DC6 | DC5 | DC4 | DC3 | DC2 | xxxx xxxx | uuuu uuuu |
| 11h, Bank 7 | PW3DCH | DC9 | DC8 | DC7 | DC6 | DC5 | DC4 | DC3 | DC2 | xxxx xxxx | uuuu uuuu |

Legend: x = unknown, u = unchanged, - = unimplemented, read as a '0', q = value depends on condition.
Shaded cells are not used by Timer1 or Timer2.


© 2000 Microchip Technology Inc.

### 13.1.3 USING PULSE WIDTH MODULATION (PWM) OUTPUTS WITH TIMER1 AND TIMER2

Three high speed pulse width modulation (PWM) outputs are provided. The PWM1 output uses Timer1 as its time base, while PWM2 and PWM3 may independently be software configured to use either Timer1 or Timer2 as the time base. The PWM outputs are on the RB2/PWM1, RB3/PWM2 and RG5/PWM3 pins.

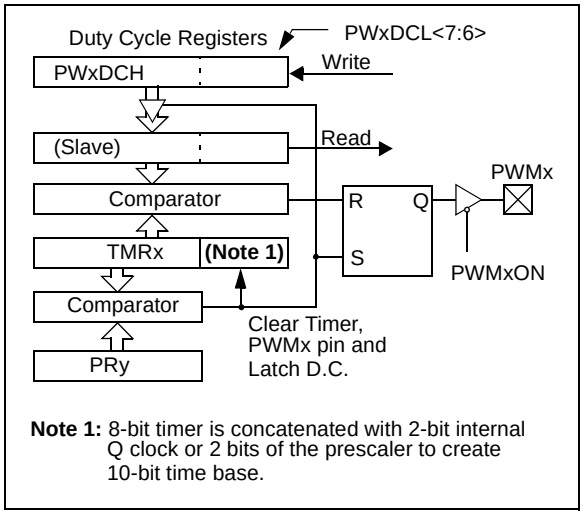
Each PWM output has a maximum resolution of 10-bits. At 10-bit resolution, the PWM output frequency is 32.2 kHz (@ 32 MHz clock) and at 8-bit resolution the PWM output frequency is 128.9 kHz. The duty cycle of the output can vary from 0% to 100%.

Figure 13-3 shows a simplified block diagram of a PWM module.

The duty cycle registers are double buffered for glitch free operation. Figure 13-4 shows how a glitch could occur if the duty cycle registers were not double buffered.

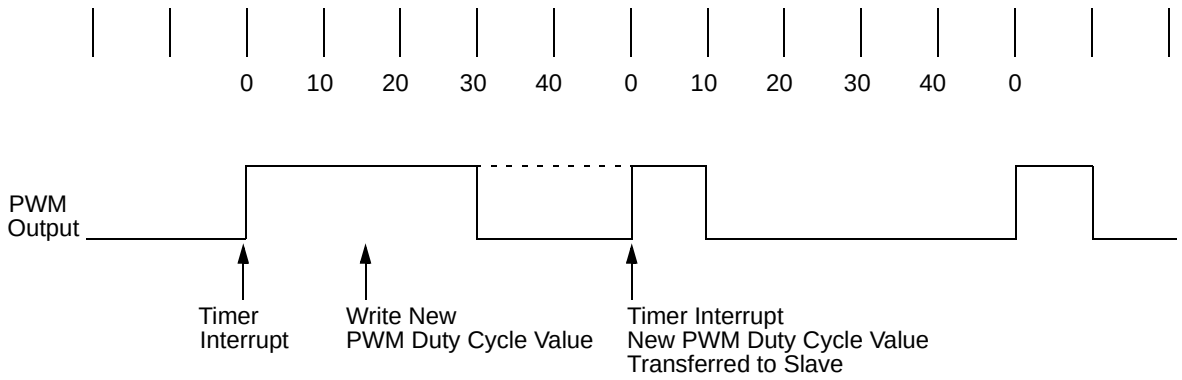
The user needs to set the PWM1ON bit (TCON2<4>) to enable the PWM1 output. When the PWM1ON bit is set, the RB2/PWM1 pin is configured as PWM1 output and forced as an output, irrespective of the data direction bit (DDRB<2>). When the PWM1ON bit is clear, the pin behaves as a port pin and its direction is controlled by its data direction bit (DDRB<2>). Similarly, the PWM2ON (TCON2<5>) bit controls the configuration of the RB3/PWM2 pin and the PWM3ON (TCON3<0>) bit controls the configuration of the RG5/PWM3 pin.

**FIGURE 13-3: SIMPLIFIED PWM BLOCK DIAGRAM**




**Note 1:** 8-bit timer is concatenated with 2-bit internal Q clock or 2 bits of the prescaler to create 10-bit time base.

**FIGURE 13-4: PWM OUTPUT (NOT BUFFERED)**




**Note:** The dotted line shows PWM output if duty cycle registers were not double buffered. If the new duty cycle is written after the timer has passed that value, then the PWM does not reset at all during the current cycle, causing a "glitch".

In this example, PWM period = 50. Old duty cycle is 30. New duty cycle value is 10.

#### 13.1.3.1 PWM Periods

The period of the PWM1 output is determined by Timer1 and its period register (PR1). The period of the PWM2 and PWM3 outputs can be individually software configured to use either Timer1 or Timer2 as the time-base. For PWM2, when TM2PW2 bit (PW2DCL<5>) is clear, the time base is determined by TMR1 and PR1 and when TM2PW2 is set, the time base is determined by Timer2 and PR2. For PWM3, when TM2PW3 bit (PW3DCL<5>) is clear, the time base is determined by TMR1 and PR1, and when TM2PW3 is set, the time base is determined by Timer2 and PR2.

Running two different PWM outputs on two different timers allows different PWM periods. Running all PWMs from Timer1 allows the best use of resources by freeing Timer2 to operate as an 8-bit timer. Timer1 and Timer2 cannot be used as a 16-bit timer if any PWM is being used.

The PWM periods can be calculated as follows:

$$\text{period of PWM1} = [(PR1) + 1] \times 4T_{OSC}$$

$$\text{period of PWM2} = [(PR1) + 1] \times 4T_{OSC} \text{ or } [(PR2) + 1] \times 4T_{OSC}$$

$$\text{period of PWM3} = [(PR1) + 1] \times 4T_{OSC} \text{ or } [(PR2) + 1] \times 4T_{OSC}$$

The duty cycle of PWMx is determined by the 10-bit value DCx<9:0>. The upper 8-bits are from register PWxDCH and the lower 2-bits are from PWxDCL<7:6> (PWxDCH:PWxDCL<7:6>). Table 13-4 shows the maximum PWM frequency ($F_{PWM}$), given the value in the period register.

The number of bits of resolution that the PWM can achieve depends on the operation frequency of the device as well as the PWM frequency ($F_{PWM}$).

Maximum PWM resolution (bits) for a given PWM frequency:

$$= \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

where: $F_{PWM}$ = 1 / period of PWM

The PWMx duty cycle is as follows:

$$\text{PWMx Duty Cycle} = (DCx) \times T_{OSC}$$

where DCx represents the 10-bit value from PWxDCH:PWxDCL.

If DCx = 0, then the duty cycle is zero. If PRx = PWxDCH, then the PWM output will be low for one to four Q-clocks (depending on the state of the PWxDCL<7:6> bits). For a duty cycle to be 100%, the PWxDCH value must be greater then the PRx value.

The duty cycle registers for both PWM outputs are double buffered. When the user writes to these registers, they are stored in master latches. When TMR1 (or TMR2) overflows and a new PWM period begins, the master latch values are transferred to the slave latches and the PWMx pin is forced high.

| **Note:** | For PW1DCH, PW1DCL, PW2DCH, PW2DCL, PW3DCH and PW3DCL registers, a write operation writes to the "master latches", while a read operation reads the "slave latches". As a result, the user may not read back what was just written to the duty cycle registers (until transferred to slave latch). |
|---|---|

The user should also avoid any "read-modify-write" operations on the duty cycle registers, such as: `ADDWF PW1DCH`. This may cause duty cycle outputs that are unpredictable.

**TABLE 13-4: PWM FREQUENCY vs. RESOLUTION AT 33 MHz**

| PWM Frequency | Frequency (kHz) | | | | |
|---|---|---|---|---|---|
| | **32.2** | **64.5** | **90.66** | **128.9** | **515.6** |
| PRx Value | 0xFF | 0x7F | 0x5A | 0x3F | 0x0F |
| High Resolution | 10-bit | 9-bit | 8.5-bit | 8-bit | 6-bit |
| Standard Resolution | 8-bit | 7-bit | 6.5-bit | 6-bit | 4-bit |

#### 13.1.3.2 PWM INTERRUPTS

The PWM modules make use of the TMR1 and/or TMR2 interrupts. A timer interrupt is generated when TMR1 or TMR2 equals its period register and on the following increment is cleared to zero. This interrupt also marks the beginning of a PWM cycle. The user can write new duty cycle values before the timer rollover. The TMR1 interrupt is latched into the TMR1IF bit and the TMR2 interrupt is latched into the TMR2IF bit. These flags must be cleared in software.


© 2000 Microchip Technology Inc.

#### 13.1.3.3 External Clock Source

The PWMs will operate, regardless of the clock source of the timer. The use of an external clock has ramifications that must be understood. Because the external TCLK12 input is synchronized internally (sampled once per instruction cycle), the time TCLK12 changes to the time the timer increments, will vary by as much as 1TCY (one instruction cycle). This will cause jitter in the duty cycle as well as the period of the PWM output.

This jitter will be ±1TCY, unless the external clock is synchronized with the processor clock. Use of one of the PWM outputs as the clock source to the TCLK12 input, will supply a synchronized clock.

In general, when using an external clock source for PWM, its frequency should be much less than the device frequency (FOSC).

#### 13.1.3.4 Maximum Resolution/Frequency for External Clock Input

The use of an external clock for the PWM time base (Timer1 or Timer2) limits the PWM output to a maximum resolution of 8-bits. The PWxDCL<7:6> bits must be kept cleared. Use of any other value will distort the PWM output. All resolutions are supported when internal clock mode is selected. The maximum attainable frequency is also lower. This is a result of the timing requirements of an external clock input for a timer (see the Electrical Specification section). The maximum PWM frequency, when the timers clock source is the RB4/TCLK12 pin, is shown in Table 13-4 (Standard Resolution mode).

**TABLE 13-5: REGISTERS/BITS ASSOCIATED WITH PWM**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 16h, Bank 3 | TCON1 | CA2ED1 | CA2ED0 | CA1ED1 | CA1ED0 | T16 | TMR3CS | TMR2CS | TMR1CS | 0000 0000 | 0000 0000 |
| 17h, Bank 3 | TCON2 | CA2OVF | CA1OVF | PWM2ON | PWM1ON | CA1/$\overline{\text{PR3}}$ | TMR3ON | TMR2ON | TMR1ON | 0000 0000 | 0000 0000 |
| 16h, Bank 7 | TCON3 | — | CA4OVF | CA3OVF | CA4ED1 | CA4ED0 | CA3ED1 | CA3ED0 | PWM3ON | -000 0000 | -000 0000 |
| 10h, Bank 2 | TMR1 | Timer1's Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 11h, Bank 2 | TMR2 | Timer2's Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 16h, Bank 1 | PIR1 | RBIF | TMR3IF | TMR2IF | TMR1IF | CA2IF | CA1IF | TX1IF | RC1IF | x000 0010 | u000 0010 |
| 17h, Bank 1 | PIE1 | RBIE | TMR3IE | TMR2IE | TMR1IE | CA2IE | CA1IE | TX1IE | RC1IE | 0000 0000 | 0000 0000 |
| 07h, Unbanked | INTSTA | PEIF | T0CKIF | T0IF | INTF | PEIE | T0CKIE | T0IE | INTE | 0000 0000 | 0000 0000 |
| 06h, Unbanked | CPUSTA | — | — | STKAV | GLINTD | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | POR | BOR | --11 11qq | --11 qquu |
| 14h, Bank 2 | PR1 | Timer1 Period Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 15h, Bank 2 | PR2 | Timer2 Period Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 10h, Bank 3 | PW1DCL | DC1 | DC0 | — | — | — | — | — | — | xx-- ---- | uu-- ---- |
| 11h, Bank 3 | PW2DCL | DC1 | DC0 | TM2PW2 | — | — | — | — | — | xx0- ---- | uu0- ---- |
| 10h, Bank 7 | PW3DCL | DC1 | DC0 | TM2PW3 | — | — | — | — | — | xx0- ---- | uu0- ---- |
| 12h, Bank 3 | PW1DCH | DC9 | DC8 | DC7 | DC6 | DC5 | DC4 | DC3 | DC2 | xxxx xxxx | uuuu uuuu |
| 13h, Bank 3 | PW2DCH | DC9 | DC8 | DC7 | DC6 | DC5 | DC4 | DC3 | DC2 | xxxx xxxx | uuuu uuuu |
| 11h, Bank 7 | PW3DCH | DC9 | DC8 | DC7 | DC6 | DC5 | DC4 | DC3 | DC2 | xxxx xxxx | uuuu uuuu |

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

## 13.2 Timer3

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

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

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

- Three capture and one period register mode
- Four capture register mode

The PIC17C7XX has up to four 16-bit capture registers that capture the 16-bit value of TMR3 when events are detected on capture pins. There are four capture pins (RB0/CAP1, RB1/CAP2, RG4/CAP3, and RE3/CAP4), one for each capture register pair. The capture pins are multiplexed with the I/O pins. An event can be:

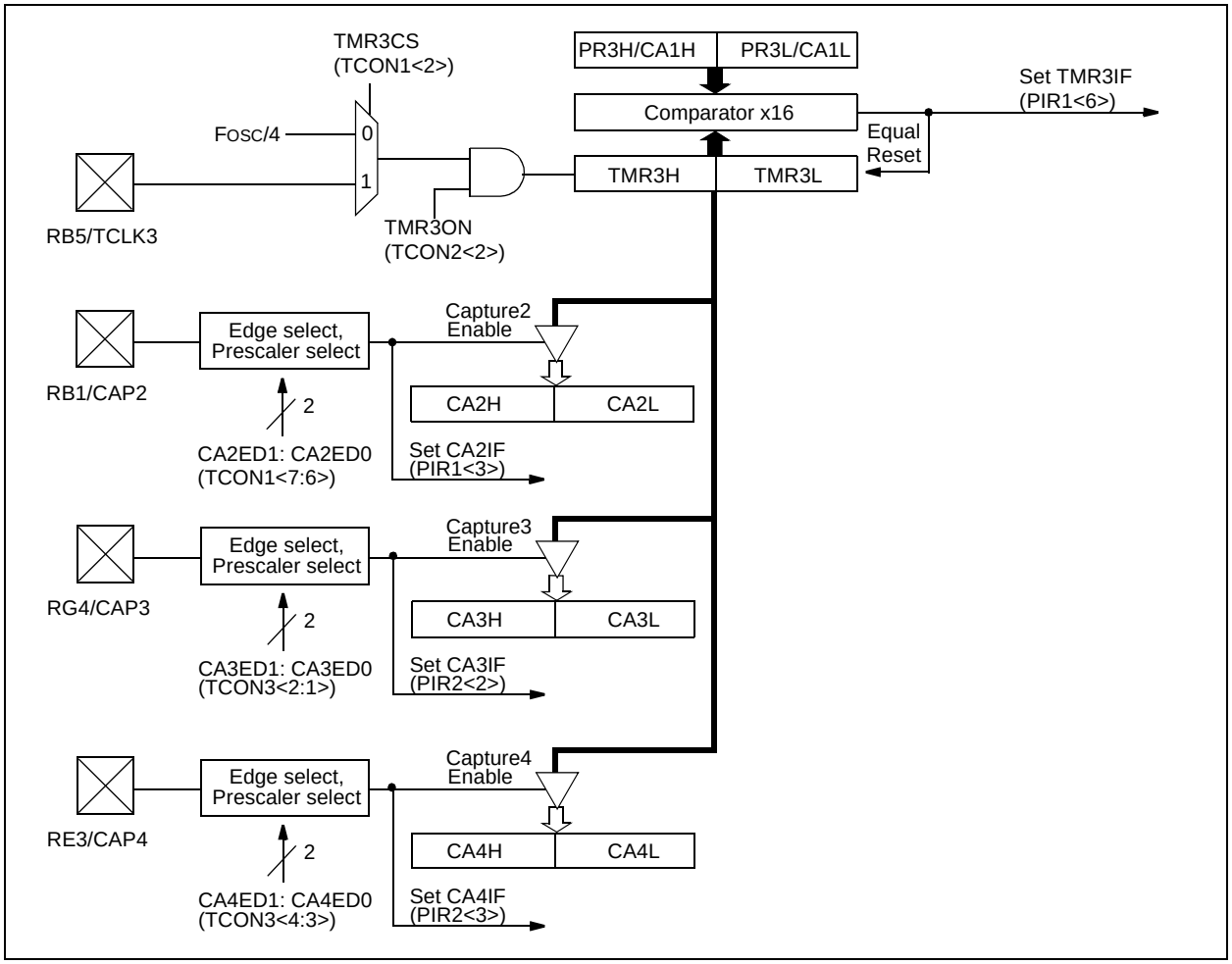
- A rising edge
- A falling edge
- Every 4th rising edge
- Every 16th rising edge

Each 16-bit capture register has an interrupt flag associated with it. The flag is set when a capture is made. The capture modules are truly part of the Timer3 block. Figure 13-5 and Figure 13-6 show the block diagrams for the two modes of operation.

### 13.2.1 THREE CAPTURE AND ONE PERIOD REGISTER MODE

In this mode, registers PR3H/CA1H and PR3L/CA1L constitute a 16-bit period register. A block diagram is shown in Figure 13-5. The timer increments until it equals the period register and then resets to 0000h on the next timer clock. TMR3 Interrupt Flag bit (TMR3IF) is set at this point. This interrupt can be disabled by clearing the TMR3 Interrupt Enable bit (TMR3IE). TMR3IF must be cleared in software.

**FIGURE 13-5: TIMER3 WITH THREE CAPTURE AND ONE PERIOD REGISTER BLOCK DIAGRAM**





© 2000 Microchip Technology Inc.

This mode (3 Capture, 1 Period) is selected if control bit CA1/$\overline{PR3}$ is clear. In this mode, the Capture1 register, consisting of high byte (PR3H/CA1H) and low byte (PR3L/CA1L), is configured as the period control register for TMR3. Capture1 is disabled in this mode and the corresponding interrupt bit, CA1IF, is never set. TMR3 increments until it equals the value in the period register and then resets to 0000h on the next timer clock.

All other Captures are active in this mode.

#### 13.2.1.1 Capture Operation

The CAxED1 and CAxED0 bits determine the event on which capture will occur. The possible events are:

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

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

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

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

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

The capture overflow status flag bit is double buffered. The master bit is set if one captured word is already residing in the Capture register (CAxH:CAxL) and another “event” has occurred on the CAPx pin. The new event will not transfer the TMR3 value to the capture register, protecting the previous unread capture value. When the user reads both the high and the low bytes (in any order) of the Capture register, the master overflow bit is transferred to the slave overflow bit (CAxOVF) and then the master bit is reset. The user can then read TCONx to determine the value of CAxOVF.

The recommended sequence to read capture registers and capture overflow flag bits is shown in Example 13-1.

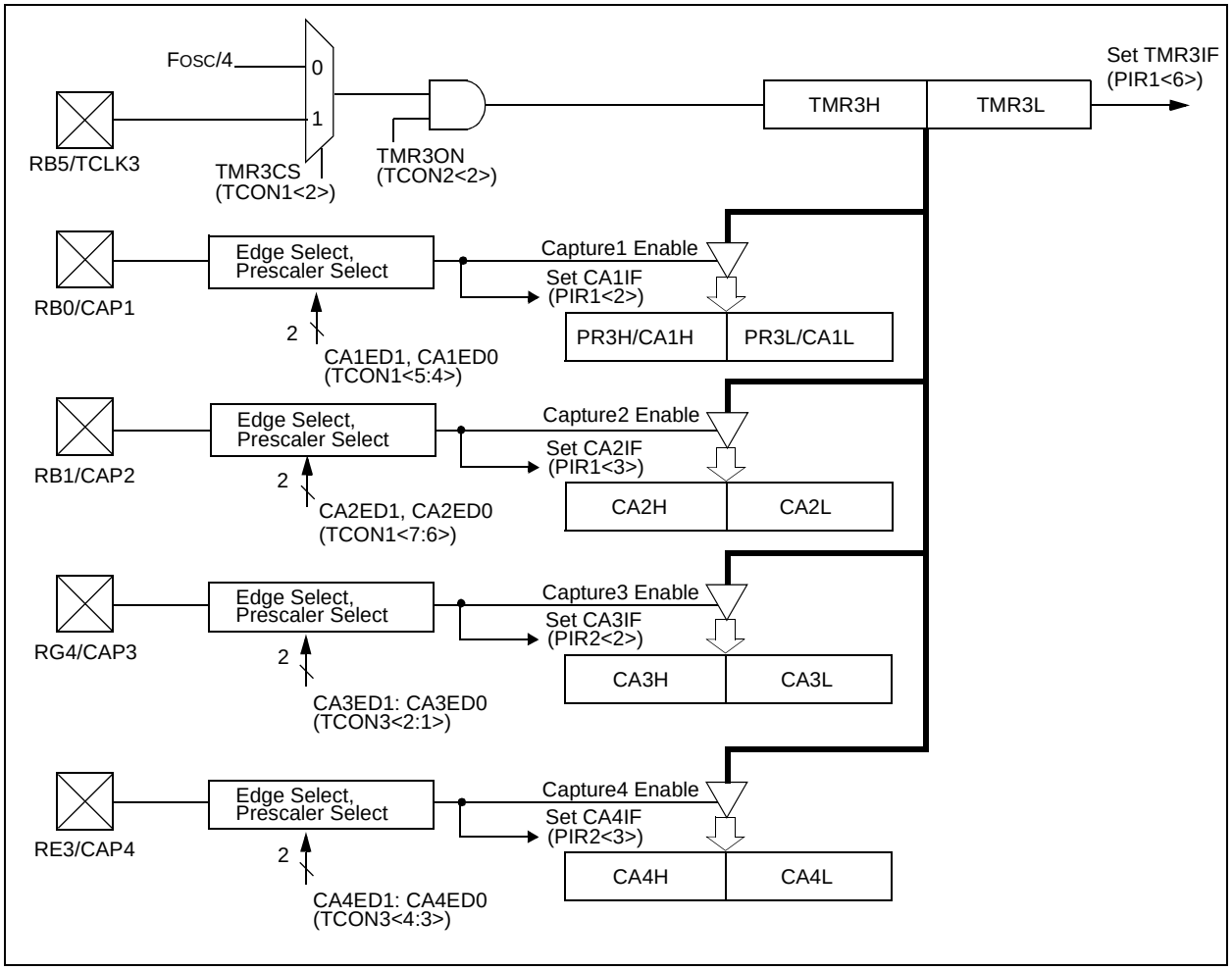
### 13.2.2 FOUR CAPTURE MODE

This mode is selected by setting bit CA1/$\overline{PR3}$. A block diagram is shown in Figure 13-6. In this mode, TMR3 runs without a period register and increments from 0000h to FFFFh and rolls over to 0000h. The TMR3 interrupt Flag (TMR3IF) is set on this rollover. The TMR3IF bit must be cleared in software.

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

All the captures operate in the same manner. Refer to Section 13.2.1 for the operation of capture.

**FIGURE 13-6: TIMER3 WITH FOUR CAPTURES BLOCK DIAGRAM**





© 2000 Microchip Technology Inc.

### 13.2.3 READING THE CAPTURE REGISTERS

The Capture overflow status flag bits are double buffered. The master bit is set if one captured word is already residing in the Capture register and another "event" has occurred on the CAPx pin. The new event will not transfer the TMR3 value to the capture register, protecting the previous unread capture value. When the user reads both the high and the low bytes (in any order) of the Capture register, the master overflow bit is transferred to the slave overflow bit (CAxOVF) and then the master bit is reset. The user can then read TCONx to determine the value of CAxOVF.

An example of an instruction sequence to read capture registers and capture overflow flag bits is shown in Example 13-1. Depending on the capture source, different registers will need to be read.

**EXAMPLE 13-1: SEQUENCE TO READ CAPTURE REGISTERS**

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

**TABLE 13-6: REGISTERS ASSOCIATED WITH CAPTURE**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 16h, Bank 3 | TCON1 | CA2ED1 | CA2ED0 | CA1ED1 | CA1ED0 | T16 | TMR3CS | TMR2CS | TMR1CS | 0000 0000 | 0000 0000 |
| 17h, Bank 3 | TCON2 | CA2OVF | CA1OVF | PWM2ON | PWM1ON | CA1/$\overline{\text{PR3}}$ | TMR3ON | TMR2ON | TMR1ON | 0000 0000 | 0000 0000 |
| 16h, Bank 7 | TCON3 | — | CA4OVF | CA3OVF | CA4ED1 | CA4ED0 | CA3ED1 | CA3ED0 | PWM3ON | -000 0000 | -000 0000 |
| 12h, Bank 2 | TMR3L | Holding Register for the Low Byte of the 16-bit TMR3 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 13h, Bank 2 | TMR3H | Holding Register for the High Byte of the 16-bit TMR3 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 16h, Bank 1 | PIR1 | RBIF | TMR3IF | TMR2IF | TMR1IF | CA2IF | CA1IF | TX1IF | RC1IF | x000 0010 | u000 0010 |
| 17h, Bank 1 | PIE1 | RBIE | TMR3IE | TMR2IE | TMR1IE | CA2IE | CA1IE | TX1IE | RC1IE | 0000 0000 | 0000 0000 |
| 10h, Bank 4 | PIR2 | SSPIF | BCLIF | ADIF | — | CA4IF | CA3IF | TX2IF | RC2IF | 000- 0010 | 000- 0010 |
| 11h, Bank 4 | PIE2 | SSPIE | BCLIE | ADIE | — | CA4IE | CA3IE | TX2IE | RC2IE | 000- 0000 | 000- 0000 |
| 07h, Unbanked | INTSTA | PEIF | T0CKIF | T0IF | INTF | PEIE | T0CKIE | T0IE | INTE | 0000 0000 | 0000 0000 |
| 06h, Unbanked | CPUSTA | — | — | STKAV | GLINTD | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | --11 11qq | --11 qquu |
| 16h, Bank 2 | PR3L/CA1L | Timer3 Period Register, Low Byte/Capture1 Register, Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 17h, Bank 2 | PR3H/CA1H | Timer3 Period Register, High Byte/Capture1 Register, High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 14h, Bank 3 | CA2L | Capture2 Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 15h, Bank 3 | CA2H | Capture2 High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 12h, Bank 7 | CA3L | Capture3 Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 13h, Bank 7 | CA3H | Capture3 High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 14h, Bank 7 | CA4L | Capture4 Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 15h, Bank 7 | CA4H | Capture4 High Byte | | | | | | | | xxxx xxxx | uuuu uuuu |

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

### 13.2.4 EXTERNAL CLOCK INPUT FOR TIMER3

When TMR3CS is set, the 16-bit TMR3 increments on the falling edge of clock input TCLK3. The input on the RB5/TCLK3 pin is sampled and synchronized by the internal phase clocks, twice every instruction cycle. This causes a delay from the time a falling edge appears on TCLK3 to the time TMR3 is actually incremented. For the external clock input timing requirements, see the Electrical Specification section. Figure 13-7 shows the timing diagram when operating from an external clock.

### 13.2.5 READING/WRITING TIMER3

Since Timer3 is a 16-bit timer and only 8-bits at a time can be read or written, care should be taken when reading or writing while the timer is running. The best method is to stop the timer, perform any read or write operation and then restart Timer3 (using the TMR3ON bit). However, if it is necessary to keep Timer3 free-running, care must be taken. For writing to the 16-bit TMR3, Example 13-2 may be used. For reading the 16-bit TMR3, Example 13-3 may be used. Interrupts must be disabled during this routine.

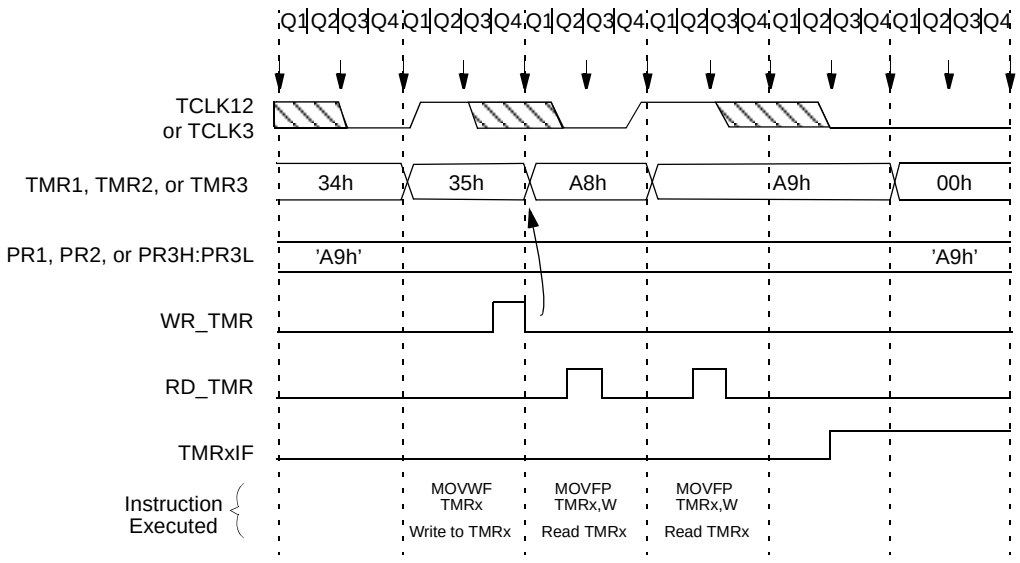
**EXAMPLE 13-2: WRITING TO TMR3**

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

**EXAMPLE 13-3: READING FROM TMR3**

```
    MOVPF   TMR3L, TMPLO        ; read low TMR3
    MOVPF   TMR3H, TMPHI        ; read high TMR3
    MOVFP   TMPLO, WREG         ; tmplo –> wreg
    CPFSLT  TMR3L               ; TMR3L < wreg?
    RETURN                      ; no then return
    MOVPF   TMR3L, TMPLO        ; read low TMR3
    MOVPF   TMR3H, TMPHI        ; read high TMR3
    RETURN                      ; return
```

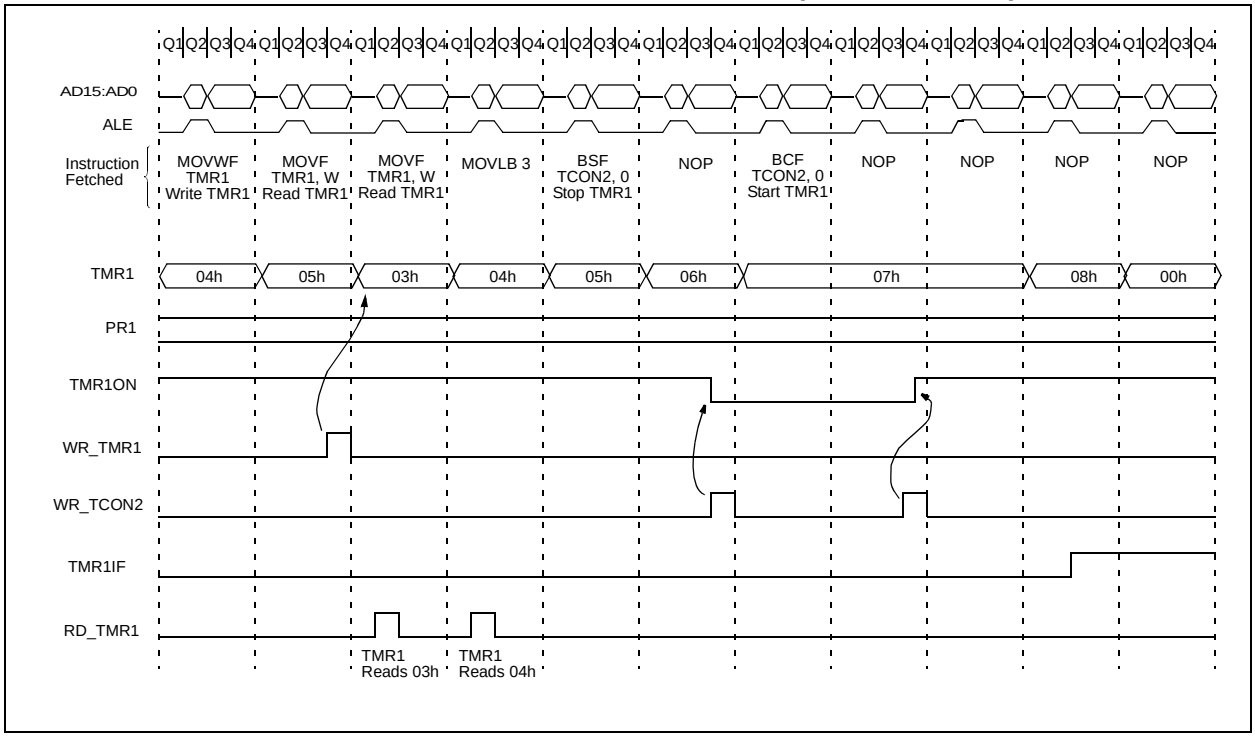
**FIGURE 13-7: TIMER1, TIMER2 AND TIMER3 OPERATION (IN COUNTER MODE)**




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


© 2000 Microchip Technology Inc.

**FIGURE 13-8: TIMER1, TIMER2 AND TIMER3 OPERATION (IN TIMER MODE)**



**NOTES:**


© 2000 Microchip Technology Inc.

## 14.0 UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART) MODULES

Each USART module is a serial I/O module. There are two USART modules that are available on the PIC17C7XX. They are specified as USART1 and USART2. The description of the operation of these modules is generic in regard to the register names and pin names used. Table 14-1 shows the generic names that are used in the description of operation and the actual names for both USART1 and USART2. Since the control bits in each register have the same function, their names are the same (there is no need to differentiate).

The Transmit Status and Control Register (TXSTA) is shown in Figure 14-1, while the Receive Status and Control Register (RCSTA) is shown in Figure 14-2.

**TABLE 14-1: USART MODULE GENERIC NAMES**

| Generic Name | USART1 Name | USART2 Name |
|---|---|---|
| *Registers* | | |
| RCSTA | RCSTA1 | RCSTA2 |
| TXSTA | TXSTA1 | TXSTA2 |
| SPBRG | SPBRG1 | SPBRG2 |
| RCREG | RCREG1 | RCREG2 |
| TXREG | TXREG1 | TXREG2 |
| *Interrupt Control Bits* | | |
| RCIE | RC1IE | RC2IE |
| RCIF | RC1IF | RC2IF |
| TXIE | TX1IE | TX2IE |
| TXIF | TX1IF | TX2IF |
| *Pins* | | |
| RX/DT | RA4/RX1/DT1 | RG6/RX2/DT2 |
| TX/CK | RA5/TX1/CK1 | RG7/TX2/CK2 |

**REGISTER 14-1: TXSTA1 REGISTER (ADDRESS: 15h, BANK 0)**
**TXSTA2 REGISTER (ADDRESS: 15h, BANK 4)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | U-0 | R-1 | R/W-x |
|---|---|---|---|---|---|---|---|
| CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D |
| bit 7 | | | | | | | bit 0 |

bit 7 **CSRC**: Clock Source Select bit

Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

Asynchronous mode:
Don't care

bit 6 **TX9**: 9-bit Transmit Select bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled
SREN/CREN overrides TXEN in SYNC mode

bit 4 **SYNC**: USART Mode Select bit
(Synchronous/Asynchronous)
1 = Synchronous mode
0 = Asynchronous mode

bit 3-2 **Unimplemented**: Read as '0'

bit 1 **TRMT**: Transmit Shift Register (TSR) Empty bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D**: 9th bit of Transmit Data (can be used to calculate the parity in software)

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

© 2000 Microchip Technology Inc.

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

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

The SPEN (RCSTA<7>) bit has to be set in order to configure the I/O pins as the Serial Communication Interface (USART).

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

- SPEN
- TXEN
- SREN
- CREN
- CSRC

**REGISTER 14-2: RCSTA1 REGISTER (ADDRESS: 13h, BANK 0)**
**RCSTA2 REGISTER (ADDRESS: 13h, BANK 4)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R-0 | R-0 | R-x |
|---|---|---|---|---|---|---|---|
| SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D |
| bit 7 | | | | | | | bit 0 |

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

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

bit 5 **SREN**: Single Receive Enable bit
This bit enables the reception of a single byte. After receiving the byte, this bit is automatically cleared.

Synchronous mode:
1 = Enable reception
0 = Disable reception

**Note:** This bit is ignored in synchronous slave reception.

Asynchronous mode:
Don't care

bit 4 **CREN**: Continuous Receive Enable bit
This bit enables the continuous reception of serial data.

Asynchronous mode:
1 = Enable continuous reception
0 = Disables continuous reception

Synchronous mode:
1 = Enables continuous reception until CREN is cleared (CREN overrides SREN)
0 = Disables continuous reception

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

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

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

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

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |


© 2000 Microchip Technology Inc.

**FIGURE 14-1: USART TRANSMIT**

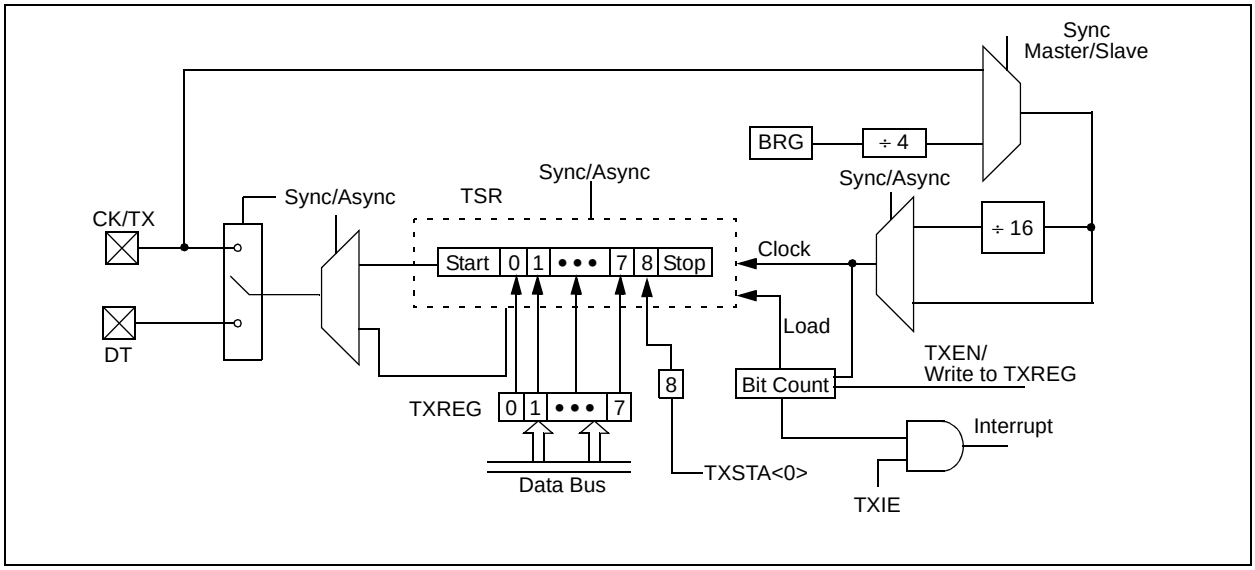
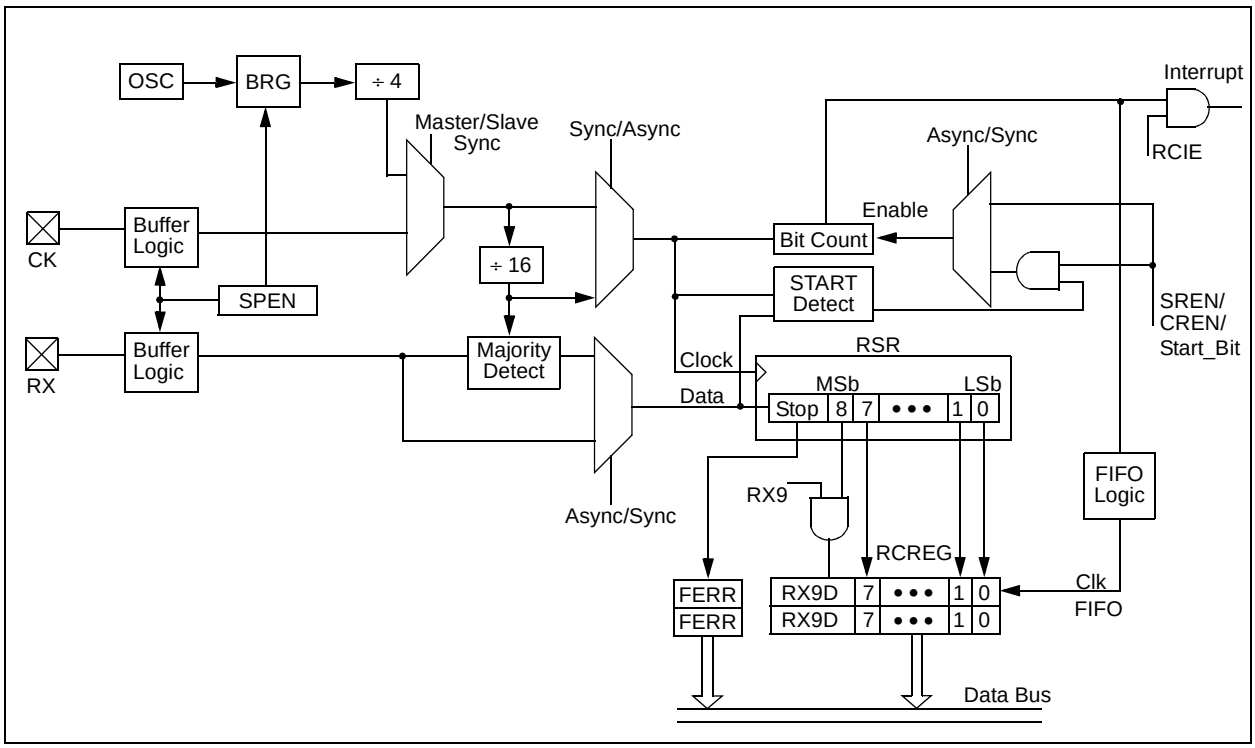



**FIGURE 14-2: USART RECEIVE**



## 14.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit baud rate generator. The SPBRG register controls the period of a free running 8-bit timer. Table 14-2 shows the formula for computation of the baud rate for different USART modes. These only apply when the USART is in Synchronous Master mode (internal clock) and Asynchronous mode.

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

**TABLE 14-2: BAUD RATE FORMULA**

| SYNC | Mode | Baud Rate |
|---|---|---|
| 0 | Asynchronous | FOSC/(64(X+1)) |
| 1 | Synchronous | FOSC/(4(X+1)) |

X = value in SPBRG (0 to 255)

Example 14-1 shows the calculation of the baud rate error for the following conditions:

FOSC = 16 MHz

Desired Baud Rate = 9600

SYNC = 0

**EXAMPLE 14-1: CALCULATING BAUD RATE ERROR**

Desired Baud Rate = FOSC / (64 (X + 1))

9600 = 16000000 /(64 (X + 1))

X = 25.042 → 25

Calculated Baud Rate = 16000000 / (64 (25 + 1))

= 9615

$$\text{Error} = \frac{\text{(Calculated Baud Rate - Desired Baud Rate)}}{\text{Desired Baud Rate}}$$

= (9615 - 9600) / 9600

= 0.16%

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

Effects of Reset

After any device RESET, the SPBRG register is cleared. The SPBRG register will need to be loaded with the desired value after each RESET.

**TABLE 14-3: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR**

| | Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| USART1 | 13h, Bank 0 | RCSTA1 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| | 15h, Bank 0 | TXSTA1 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| | 17h, Bank 0 | SPBRG1 | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |
| USART2 | 13h, Bank 4 | RCSTA2 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| | 15h, Bank 4 | TXSTA2 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| | 17h, Bank 4 | SPBRG2 | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |

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


© 2000 Microchip Technology Inc.

**TABLE 14-4: BAUD RATES FOR SYNCHRONOUS MODE**

| BAUD RATE (K) | FOSC = 33 MHz | | | FOSC = 25 MHz | | | FOSC = 20 MHz | | | FOSC = 16 MHz | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) |
| 0.3 | NA | — | — | NA | — | — | NA | — | — | NA | — | — |
| 1.2 | NA | — | — | NA | — | — | NA | — | — | NA | — | — |
| 2.4 | NA | — | — | NA | — | — | NA | — | — | NA | — | — |
| 9.6 | NA | — | — | NA | — | — | NA | — | — | NA | — | — |
| 19.2 | NA | — | — | NA | — | — | 19.53 | +1.73 | 255 | 19.23 | +0.16 | 207 |
| 76.8 | 77.10 | +0.39 | 106 | 77.16 | +0.47 | 80 | 76.92 | +0.16 | 64 | 76.92 | +0.16 | 51 |
| 96 | 95.93 | -0.07 | 85 | 96.15 | +0.16 | 64 | 96.15 | +0.16 | 51 | 95.24 | -0.79 | 41 |
| 300 | 294.64 | -1.79 | 27 | 297.62 | -0.79 | 20 | 294.1 | -1.96 | 16 | 307.69 | +2.56 | 12 |
| 500 | 485.29 | -2.94 | 16 | 480.77 | -3.85 | 12 | 500 | 0 | 9 | 500 | 0 | 7 |
| HIGH | 8250 | — | 0 | 6250 | — | 0 | 5000 | — | 0 | 4000 | — | 0 |
| LOW | 32.22 | — | 255 | 24.41 | — | 255 | 19.53 | — | 255 | 15.625 | — | 255 |

| BAUD RATE (K) | FOSC = 10 MHz | | | FOSC = 7.159 MHz | | | FOSC = 5.068 MHz | | |
|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) |
| 0.3 | NA | — | — | NA | — | — | NA | — | — |
| 1.2 | NA | — | — | NA | — | — | NA | — | — |
| 2.4 | NA | — | — | NA | — | — | NA | — | — |
| 9.6 | 9.766 | +1.73 | 255 | 9.622 | +0.23 | 185 | 9.6 | 0 | 131 |
| 19.2 | 19.23 | +0.16 | 129 | 19.24 | +0.23 | 92 | 19.2 | 0 | 65 |
| 76.8 | 75.76 | -1.36 | 32 | 77.82 | +1.32 | 22 | 79.2 | +3.13 | 15 |
| 96 | 96.15 | +0.16 | 25 | 94.20 | -1.88 | 18 | 97.48 | +1.54 | 12 |
| 300 | 312.5 | +4.17 | 7 | 298.3 | -0.57 | 5 | 316.8 | +5.60 | 3 |
| 500 | 500 | 0 | 4 | NA | — | — | NA | — | — |
| HIGH | 2500 | — | 0 | 1789.8 | — | 0 | 1267 | — | 0 |
| LOW | 9.766 | — | 255 | 6.991 | — | 255 | 4.950 | — | 255 |

| BAUD RATE (K) | FOSC = 3.579 MHz | | | FOSC = 1 MHz | | | FOSC = 32.768 kHz | | |
|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) |
| 0.3 | NA | — | — | NA | — | — | 0.303 | +1.14 | 26 |
| 1.2 | NA | — | — | 1.202 | +0.16 | 207 | 1.170 | -2.48 | 6 |
| 2.4 | NA | — | — | 2.404 | +0.16 | 103 | NA | — | — |
| 9.6 | 9.622 | +0.23 | 92 | 9.615 | +0.16 | 25 | NA | — | — |
| 19.2 | 19.04 | -0.83 | 46 | 19.24 | +0.16 | 12 | NA | — | — |
| 76.8 | 74.57 | -2.90 | 11 | 83.34 | +8.51 | 2 | NA | — | — |
| 96 | 99.43 | _3.57 | 8 | NA | — | — | NA | — | — |
| 300 | 298.3 | -0.57 | 2 | NA | — | — | NA | — | — |
| 500 | NA | — | — | NA | — | — | NA | — | — |
| HIGH | 894.9 | — | 0 | 250 | — | 0 | 8.192 | — | 0 |
| LOW | 3.496 | — | 255 | 0.976 | — | 255 | 0.032 | — | 255 |

© 2000 Microchip Technology Inc.

**TABLE 14-5: BAUD RATES FOR ASYNCHRONOUS MODE**

| BAUD RATE (K) | FOSC = 33 MHz | | | FOSC = 25 MHz | | | FOSC = 20 MHz | | | FOSC = 16 MHz | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) |
| 0.3 | NA | — | — | NA | — | — | NA | — | — | NA | — | — |
| 1.2 | NA | — | — | NA | — | — | 1.221 | +1.73 | 255 | 1.202 | +0.16 | 207 |
| 2.4 | 2.398 | -0.07 | 214 | 2.396 | 0.14 | 162 | 2.404 | +0.16 | 129 | 2.404 | +0.16 | 103 |
| 9.6 | 9.548 | -0.54 | 53 | 9.53 | -0.76 | 40 | 9.469 | -1.36 | 32 | 9.615 | +0.16 | 25 |
| 19.2 | 19.09 | -0.54 | 26 | 19.53 | +1.73 | 19 | 19.53 | +1.73 | 15 | 19.23 | +0.16 | 12 |
| 76.8 | 73.66 | -4.09 | 6 | 78.13 | +1.73 | 4 | 78.13 | +1.73 | 3 | 83.33 | +8.51 | 2 |
| 96 | 103.12 | +7.42 | 4 | 97.65 | +1.73 | 3 | 104.2 | +8.51 | 2 | NA | — | — |
| 300 | 257.81 | -14.06 | 1 | 390.63 | +30.21 | 0 | 312.5 | +4.17 | 0 | NA | — | — |
| 500 | 515.62 | +3.13 | 0 | NA | — | — | NA | — | — | NA | — | — |
| HIGH | 515.62 | — | 0 | — | — | 0 | 312.5 | — | 0 | 250 | — | 0 |
| LOW | 2.014 | — | 255 | 1.53 | — | 255 | 1.221 | — | 255 | 0.977 | — | 255 |

| BAUD RATE (K) | FOSC = 10 MHz | | | FOSC = 7.159 MHz | | | FOSC = 5.068 MHz | | |
|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) |
| 0.3 | NA | — | — | NA | — | — | 0.31 | +3.13 | 255 |
| 1.2 | 1.202 | +0.16 | 129 | 1.203 | _0.23 | 92 | 1.2 | 0 | 65 |
| 2.4 | 2.404 | +0.16 | 64 | 2.380 | -0.83 | 46 | 2.4 | 0 | 32 |
| 9.6 | 9.766 | +1.73 | 15 | 9.322 | -2.90 | 11 | 9.9 | -3.13 | 7 |
| 19.2 | 19.53 | +1.73 | 7 | 18.64 | -2.90 | 5 | 19.8 | +3.13 | 3 |
| 76.8 | 78.13 | +1.73 | 1 | NA | — | — | 79.2 | +3.13 | 0 |
| 96 | NA | — | — | NA | — | — | NA | — | — |
| 300 | NA | — | — | NA | — | — | NA | — | — |
| 500 | NA | — | — | NA | — | — | NA | — | — |
| HIGH | 156.3 | — | 0 | 111.9 | — | 0 | 79.2 | — | 0 |
| LOW | 0.610 | — | 255 | 0.437 | — | 255 | 0.309 | — | 255 |

| BAUD RATE (K) | FOSC = 3.579 MHz | | | FOSC = 1 MHz | | | FOSC = 32.768 kHz | | |
|---|---|---|---|---|---|---|---|---|---|
| | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) | KBAUD | %ERROR | SPBRG VALUE (DECIMAL) |
| 0.3 | 0.301 | +0.23 | 185 | 0.300 | +0.16 | 51 | 0.256 | -14.67 | 1 |
| 1.2 | 1.190 | -0.83 | 46 | 1.202 | +0.16 | 12 | NA | — | — |
| 2.4 | 2.432 | +1.32 | 22 | 2.232 | -6.99 | 6 | NA | — | — |
| 9.6 | 9.322 | -2.90 | 5 | NA | — | — | NA | — | — |
| 19.2 | 18.64 | -2.90 | 2 | NA | — | — | NA | — | — |
| 76.8 | NA | — | — | NA | — | — | NA | — | — |
| 96 | NA | — | — | NA | — | — | NA | — | — |
| 300 | NA | — | — | NA | — | — | NA | — | — |
| 500 | NA | — | — | NA | — | — | NA | — | — |
| HIGH | 55.93 | — | 0 | 15.63 | — | 0 | 0.512 | — | 0 |
| LOW | 0.218 | — | 255 | 0.061 | — | 255 | 0.002 | — | 255 |


© 2000 Microchip Technology Inc.

## 14.2 USART Asynchronous Mode

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

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

The USART Asynchronous module consists of the following components:

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

### 14.2.1 USART ASYNCHRONOUS TRANSMITTER

The USART transmitter block diagram is shown in Figure 14-1. The heart of the transmitter is the transmit shift register (TSR). The shift register obtains its data from the read/write transmit buffer (TXREG). TXREG is loaded with data in software. The TSR is not loaded until the STOP bit has been transmitted from the previous load. As soon as the STOP bit is transmitted, the TSR is loaded with new data from the TXREG (if available). Once TXREG transfers the data to the TSR (occurs in one TCY at the end of the current BRG cycle), the TXREG is empty and an interrupt bit, TXIF, is set. This interrupt can be enabled/disabled by setting/clearing the TXIE bit. TXIF will be set, regardless of TXIE and cannot be reset in software. It will reset only when new data is loaded into TXREG. While TXIF indicates the status of the TXREG, the TRMT (TXSTA<1>) bit shows the status of the TSR.

TRMT is a read only bit which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR is empty.

| **Note:** | The TSR is not mapped in data memory, so it is not available to the user. |
|---|---|

Transmission is enabled by setting the TXEN (TXSTA<5>) bit. The actual transmission will not occur until TXREG has been loaded with data and the baud rate generator (BRG) has produced a shift clock (Figure 14-3). The transmission can also be started by first loading TXREG and then setting TXEN. Normally, when transmission is first started, the TSR is empty, so a transfer to TXREG will result in an immediate transfer to TSR, resulting in an empty TXREG. A back-to-back transfer is thus possible (Figure 14-4). Clearing TXEN during a transmission will cause the transmission to be aborted. This will reset the transmitter and the TX/CK pin will revert to hi-impedance.

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

Steps to follow when setting up an Asynchronous Transmission:

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

**FIGURE 14-3: ASYNCHRONOUS MASTER TRANSMISSION**

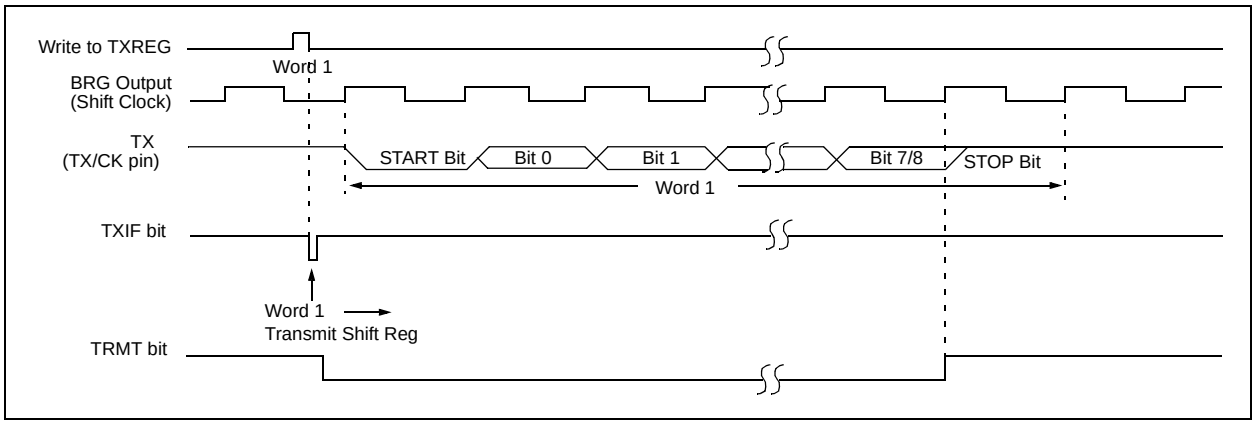
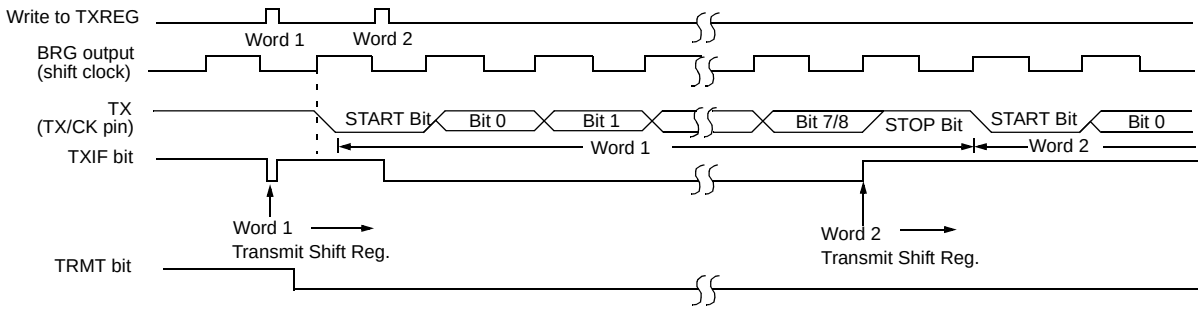


**FIGURE 14-4: ASYNCHRONOUS MASTER TRANSMISSION (BACK TO BACK)**




**Note:** This timing diagram shows two consecutive transmissions.

**TABLE 14-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 16h, Bank 1 | PIR1 | RBIF | TMR3IF | TMR2IF | TMR1IF | CA2IF | CA1IF | TX1IF | RC1IF | x000 0010 | u000 0010 |
| 17h, Bank 1 | PIE1 | RBIE | TMR3IE | TMR2IE | TMR1IE | CA2IE | CA1IE | TX1IE | RC1IE | 0000 0000 | 0000 0000 |
| 13h, Bank 0 | RCSTA1 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 16h, Bank 0 | TXREG1 | Serial Port Transmit Register (USART1) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 15h, Bank 0 | TXSTA1 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 17h, Bank 0 | SPBRG1 | Baud Rate Generator Register (USART1) | | | | | | | | 0000 0000 | 0000 0000 |
| 10h, Bank 4 | PIR2 | SSPIF | BCLIF | ADIF | — | CA4IF | CA3IF | TX2IF | RC2IF | 000- 0010 | 000- 0010 |
| 11h, Bank 4 | PIE2 | SSPIE | BCLIE | ADIE | — | CA4IE | CA3IE | TX2IE | RC2IE | 000- 0000 | 000- 0000 |
| 13h, Bank 4 | RCSTA2 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 16h, Bank 4 | TXREG2 | Serial Port Transmit Register (USART2) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 15h, Bank 4 | TXSTA2 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 17h, Bank 4 | SPBRG2 | Baud Rate Generator Register (USART2) | | | | | | | | 0000 0000 | 0000 0000 |

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


© 2000 Microchip Technology Inc.

### 14.2.2 USART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 14-2. The data comes in the RX/DT pin and drives the data recovery block. The data recovery block is actually a high speed shifter operating at 16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC.

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

The heart of the receiver is the receive (serial) shift register (RSR). After sampling the STOP bit, the received data in the RSR is transferred to the RCREG (if it is empty). If the transfer is complete, the interrupt bit, RCIF, is set. The actual interrupt can be enabled/disabled by setting/clearing the RCIE bit. RCIF is a read only bit which is cleared by the hardware. It is cleared when RCREG has been read and is empty. RCREG is a double buffered register (i.e., it is a two-deep FIFO). It is possible for two bytes of data to be received and transferred to the RCREG FIFO and a third byte begin shifting to the RSR. On detection of the STOP bit of the third byte, if the RCREG is still full, then the overrun error bit, OERR (RCSTA<1>) will be set. The word in the RSR will be lost. RCREG can be read twice to retrieve the two bytes in the FIFO. The OERR bit has to be cleared in software which is done by resetting the receive logic (CREN is set). If the OERR bit is set, transfers from the RSR to RCREG are inhibited, so it is essential to clear the OERR bit if it is set. The framing error bit FERR (RCSTA<2>) is set if a STOP bit is not detected.

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

### 14.2.3 SAMPLING

The data on the RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX/DT pin. The sampling is done on the seventh, eighth and ninth falling edges of a x16 clock (Figure 14-5).

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

**FIGURE 14-5: RX PIN SAMPLING SCHEME**

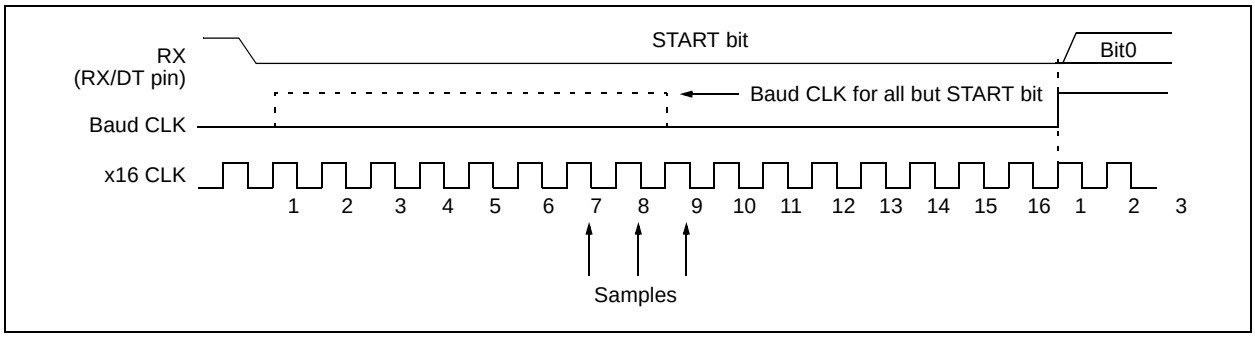
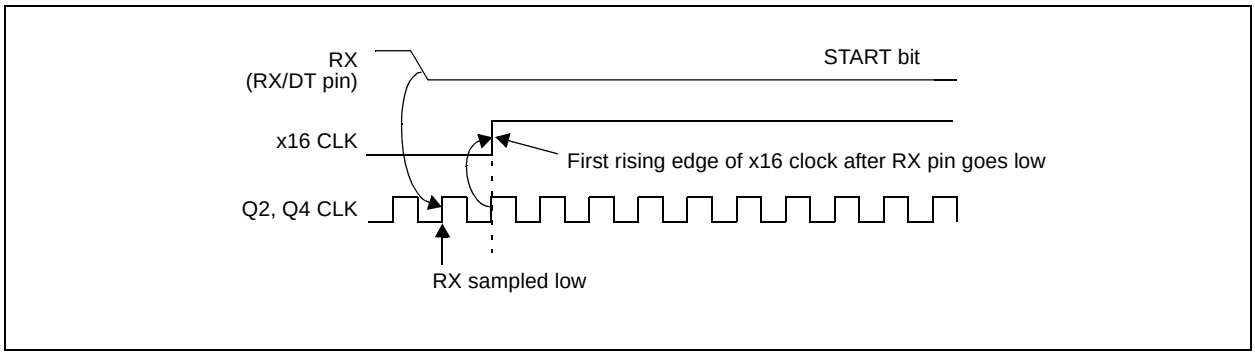



**FIGURE 14-6: START BIT DETECT**

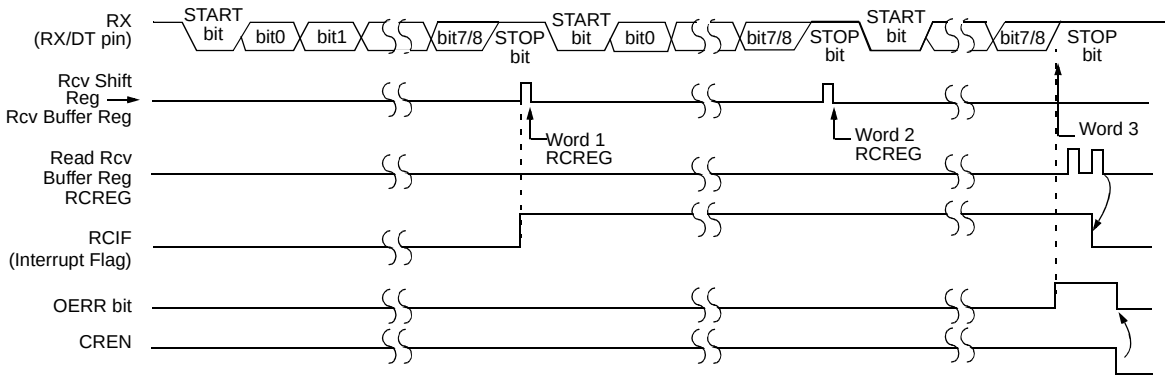


Steps to follow when setting up an Asynchronous Reception:

1. Initialize the SPBRG register for the appropriate baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are desired, then set the RCIE bit.
4. If 9-bit reception is desired, then set the RX9 bit.
5. Enable the reception by setting the CREN bit.
6. The RCIF bit will be set when reception completes and an interrupt will be generated if the RCIE bit was set.
7. Read RCSTA to get the ninth bit (if enabled) and FERR bit to determine if any error occurred during reception.
8. Read RCREG for the 8-bit received data.
9. If an overrun error occurred, clear the error by clearing the OERR bit.

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

**FIGURE 14-7: ASYNCHRONOUS RECEPTION**




**Note:** This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

**TABLE 14-7: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 16h, Bank 1 | PIR1 | RBIF | TMR3IF | TMR2IF | TMR1IF | CA2IF | CA1IF | TX1IF | RC1IF | x000 0010 | u000 0010 |
| 17h, Bank 1 | PIE1 | RBIE | TMR3IE | TMR2IE | TMR1IE | CA2IE | CA1IE | TX1IE | RC1IE | 0000 0000 | 0000 0000 |
| 13h, Bank 0 | RCSTA1 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 14h, Bank 0 | RCREG1 | RX7 | RX6 | RX5 | RX4 | RX3 | RX2 | RX1 | RX0 | xxxx xxxx | uuuu uuuu |
| 15h, Bank 0 | TXSTA1 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 17h, Bank 0 | SPBRG1 | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |
| 10h, Bank 4 | PIR2 | SSPIF | BCLIF | ADIF | — | CA4IF | CA3IF | TX2IF | RC2IF | 000- 0010 | 000- 0010 |
| 11h, Bank 4 | PIE2 | SSPIE | BCLIE | ADIE | — | CA4IE | CA3IE | TX2IE | RC2IE | 000- 0000 | 000- 0000 |
| 13h, Bank 4 | RCSTA2 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 14h, Bank 4 | RCREG2 | RX7 | RX6 | RX5 | RX4 | RX3 | RX2 | RX1 | RX0 | xxxx xxxx | uuuu uuuu |
| 15h, Bank 4 | TXSTA2 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 17h, Bank 4 | SPBRG2 | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |

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


© 2000 Microchip Technology Inc.

## 14.3 USART Synchronous Master Mode

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

### 14.3.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 14-1. The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer TXREG. TXREG is loaded with data in software. The TSR is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from TXREG (if available). Once TXREG transfers the data to the TSR (occurs in one $T_{CY}$ at the end of the current BRG cycle), TXREG is empty and the TXIF bit is set. This interrupt can be enabled/disabled by setting/clearing the TXIE bit. TXIF will be set regardless of the state of bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into TXREG. While TXIF indicates the status of TXREG, TRMT (TXSTA<1>) shows the status of the TSR. TRMT is a read only bit which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR is empty. The TSR is not mapped in data memory, so it is not available to the user.

Transmission is enabled by setting the TXEN (TXSTA<5>) bit. The actual transmission will not occur until TXREG has been loaded with data. The first data bit will be shifted out on the next available rising edge of the clock on the TX/CK pin. Data out is stable around the falling edge of the synchronous clock (Figure 14-9). The transmission can also be started by first loading TXREG and then setting TXEN. This is advantageous when slow baud rates are selected, since BRG is kept in RESET when the TXEN, CREN, and SREN bits are clear. Setting the TXEN bit will start the BRG, creating a shift clock immediately. Normally when transmission is first started, the TSR is empty, so a transfer to TXREG will result in an immediate transfer to the TSR, resulting in an empty TXREG. Back-to-back transfers are possible.

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

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

Steps to follow when setting up a Synchronous Master Transmission:

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

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

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

**TABLE 14-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 16h, Bank 1 | PIR1 | RBIF | TMR3IF | TMR2IF | TMR1IF | CA2IF | CA1IF | TX1IF | RC1IF | x000 0010 | u000 0010 |
| 17h, Bank 1 | PIE1 | RBIE | TMR3IE | TMR2IE | TMR1IE | CA2IE | CA1IE | TX1IE | RC1IE | 0000 0000 | 0000 0000 |
| 13h, Bank 0 | RCSTA1 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 16h, Bank 0 | TXREG1 | TX7 | TX6 | TX5 | TX4 | TX3 | TX2 | TX1 | TX0 | xxxx xxxx | uuuu uuuu |
| 15h, Bank 0 | TXSTA1 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 17h, Bank 0 | SPBRG1 | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |
| 10h, Bank 4 | PIR2 | SSPIF | BCLIF | ADIF | — | CA4IF | CA3IF | TX2IF | RC2IF | 000- 0010 | 000- 0010 |
| 11h, Bank 4 | PIE2 | SSPIE | BCLIE | ADIE | — | CA4IE | CA3IE | TX2IE | RC2IE | 000- 0000 | 000- 0000 |
| 13h, Bank 4 | RCSTA2 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 16h, Bank 4 | TXREG2 | TX7 | TX6 | TX5 | TX4 | TX3 | TX2 | TX1 | TX0 | xxxx xxxx | uuuu uuuu |
| 15h, Bank 4 | TXSTA2 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 17h, Bank 4 | SPBRG2 | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |

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

**FIGURE 14-8: SYNCHRONOUS TRANSMISSION**

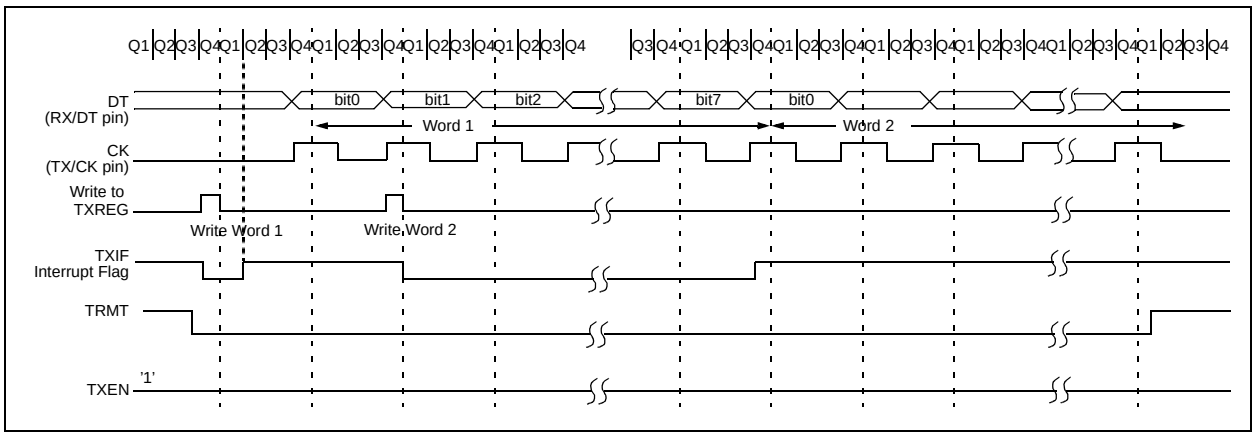
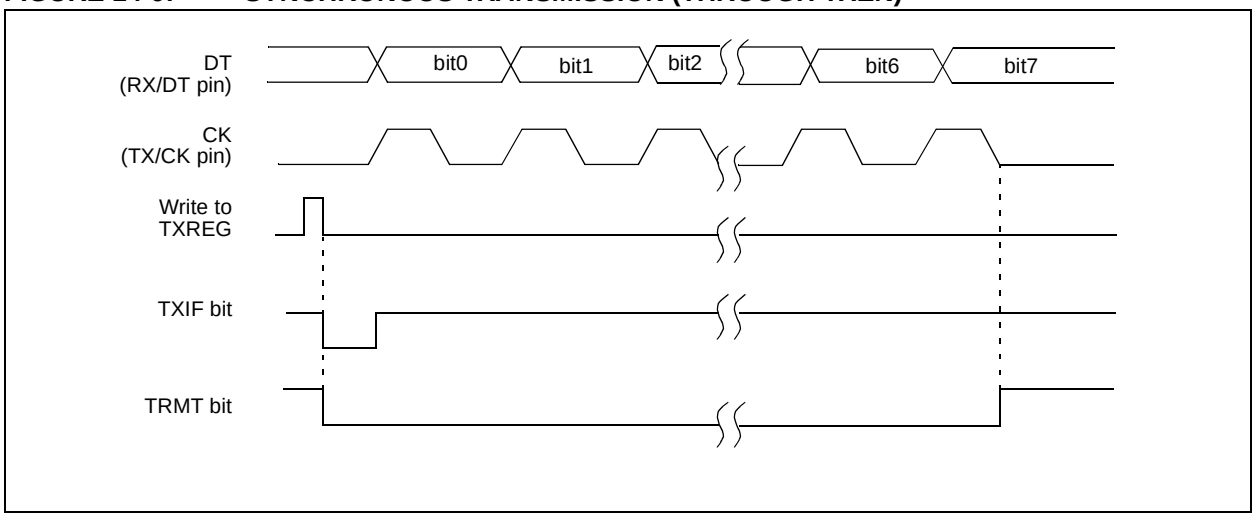



**FIGURE 14-9: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**





© 2000 Microchip Technology Inc.

### 14.3.2 USART SYNCHRONOUS MASTER RECEPTION

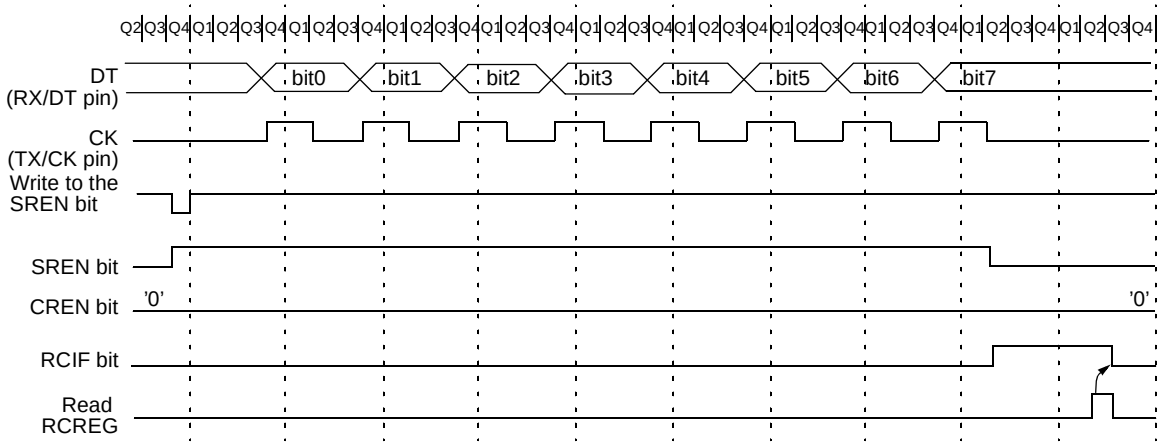
Once Synchronous mode is selected, reception is enabled by setting either the SREN (RCSTA<5>) bit or the CREN (RCSTA<4>) bit. Data is sampled on the RX/DT pin on the falling edge of the clock. If SREN is set, then only a single word is received. If CREN is set, the reception is continuous until CREN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

Steps to follow when setting up a Synchronous Master Reception:

1. Initialize the SPBRG register for the appropriate baud rate. See Section 14.1 for details.
2. Enable the synchronous master serial port by setting bits SYNC, SPEN, and CSRC.
3. If interrupts are desired, then set the RCIE bit.
4. If 9-bit reception is desired, then set the RX9 bit.
5. If a single reception is required, set bit SREN. For continuous reception set bit CREN.
6. The RCIF bit will be set when reception is complete and an interrupt will be generated if the RCIE bit was set.
7. Read RCSTA to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading RCREG.
9. If any error occurred, clear the error by clearing CREN.

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

**FIGURE 14-10: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)**




**Note:** Timing diagram demonstrates SYNC Master mode with SREN = 1.

**TABLE 14-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 16h, Bank 1 | PIR1 | RBIF | TMR3IF | TMR2IF | TMR1IF | CA2IF | CA1IF | TX1IF | RC1IF | x000 0010 | u000 0010 |
| 17h, Bank 1 | PIE1 | RBIE | TMR3IE | TMR2IE | TMR1IE | CA2IE | CA1IE | TX1IE | RC1IE | 0000 0000 | 0000 0000 |
| 13h, Bank 0 | RCSTA1 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 14h, Bank 0 | RCREG1 | RX7 | RX6 | RX5 | RX4 | RX3 | RX2 | RX1 | RX0 | xxxx xxxx | uuuu uuuu |
| 15h, Bank 0 | TXSTA1 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 17h, Bank 0 | SPBRG1 | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |
| 10h, Bank 4 | PIR2 | SSPIF | BCLIF | ADIF | — | CA4IF | CA3IF | TX2IF | RC2IF | 000- 0010 | 000- 0010 |
| 11h, Bank 4 | PIE2 | SSPIE | BCLIE | ADIE | — | CA4IE | CA3IE | TX2IE | RC2IE | 000- 0000 | 000- 0000 |
| 13h, Bank 4 | RCSTA2 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 14h, Bank 4 | RCREG2 | RX7 | RX6 | RX5 | RX4 | RX3 | RX2 | RX1 | RX0 | xxxx xxxx | uuuu uuuu |
| 15h, Bank 4 | TXSTA2 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 17h, Bank 4 | SPBRG2 | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |

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


© 2000 Microchip Technology Inc.

## 14.4 USART Synchronous Slave Mode

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

### 14.4.1 USART SYNCHRONOUS SLAVE TRANSMIT

The operation of the SYNC Master and Slave modes are identical except in the case of the SLEEP mode.

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

Steps to follow when setting up a Synchronous Slave Transmission:

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

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

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

### 14.4.2 USART SYNCHRONOUS SLAVE RECEPTION

Operation of the Synchronous Master and Slave modes are identical except in the case of the SLEEP mode. Also, SREN is a “don't care” in Slave mode.

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

Steps to follow when setting up a Synchronous Slave Reception:

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

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

### TABLE 14-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 16h, Bank 1 | PIR1 | RBIF | TMR3IF | TMR2IF | TMR1IF | CA2IF | CA1IF | TX1IF | RC1IF | x000 0010 | u000 0010 |
| 17h, Bank 1 | PIE1 | RBIE | TMR3IE | TMR2IE | TMR1IE | CA2IE | CA1IE | TX1IE | RC1IE | 0000 0000 | 0000 0000 |
| 13h, Bank 0 | RCSTA1 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 15h, Bank 0 | TXSTA1 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 16h, Bank 0 | TXREG1 | TX7 | TX6 | TX5 | TX4 | TX3 | TX2 | TX1 | TX0 | xxxx xxxx | uuuu uuuu |
| 17h, Bank 0 | SPBRG1 | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |
| 10h, Bank 4 | PIR2 | SSPIF | BCLIF | ADIF | — | CA4IF | CA3IF | TX2IF | RC2IF | 000- 0010 | 000- 0010 |
| 11h, Bank 4 | PIE2 | SSPIE | BCLIE | ADIE | — | CA4IE | CA3IE | TX2IE | RC2IE | 000- 0000 | 000- 0000 |
| 13h, Bank 4 | RCSTA2 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 16h, Bank 4 | TXREG2 | TX7 | TX6 | TX5 | TX4 | TX3 | TX2 | TX1 | TX0 | xxxx xxxx | uuuu uuuu |
| 15h, Bank 4 | TXSTA2 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 17h, Bank 4 | SPBRG2 | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |

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

### TABLE 14-11: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 16h, Bank1 | PIR1 | RBIF | TMR3IF | TMR2IF | TMR1IF | CA2IF | CA1IF | TX1IF | RC1IF | x000 0010 | u000 0010 |
| 17h, Bank1 | PIE1 | RBIE | TMR3IE | TMR2IE | TMR1IE | CA2IE | CA1IE | TX1IE | RC1IE | 0000 0000 | 0000 0000 |
| 13h, Bank0 | RCSTA1 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 14h, Bank0 | RCREG1 | RX7 | RX6 | RX5 | RX4 | RX3 | RX2 | RX1 | RX0 | xxxx xxxx | uuuu uuuu |
| 15h, Bank 0 | TXSTA1 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 17h, Bank 0 | SPBRG1 | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |
| 10h, Bank 4 | PIR2 | SSPIF | BCLIF | ADIF | — | CA4IF | CA3IF | TX2IF | RC2IF | 000- 0010 | 000- 0010 |
| 11h, Bank 4 | PIE2 | SSPIE | BCLIE | ADIE | — | CA4IE | CA3IE | TX2IE | RC2IE | 000- 0000 | 000- 0000 |
| 13h, Bank 4 | RCSTA2 | SPEN | RX9 | SREN | CREN | — | FERR | OERR | RX9D | 0000 -00x | 0000 -00u |
| 14h, Bank 4 | RCREG2 | RX7 | RX6 | RX5 | RX4 | RX3 | RX2 | RX1 | RX0 | xxxx xxxx | uuuu uuuu |
| 15h, Bank 4 | TXSTA2 | CSRC | TX9 | TXEN | SYNC | — | — | TRMT | TX9D | 0000 --1x | 0000 --1u |
| 17h, Bank 4 | SPBRG2 | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |

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


© 2000 Microchip Technology Inc.

## 15.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

The Master Synchronous Serial Port (MSSP) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D converters, etc. The MSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit™ ($I^2C$)

Figure 15-1 shows a block diagram for the SPI mode, while Figure 15-2 and Figure 15-3 show the block diagrams for the two different $I^2C$ modes of operation.

**FIGURE 15-1: SPI MODE BLOCK DIAGRAM**

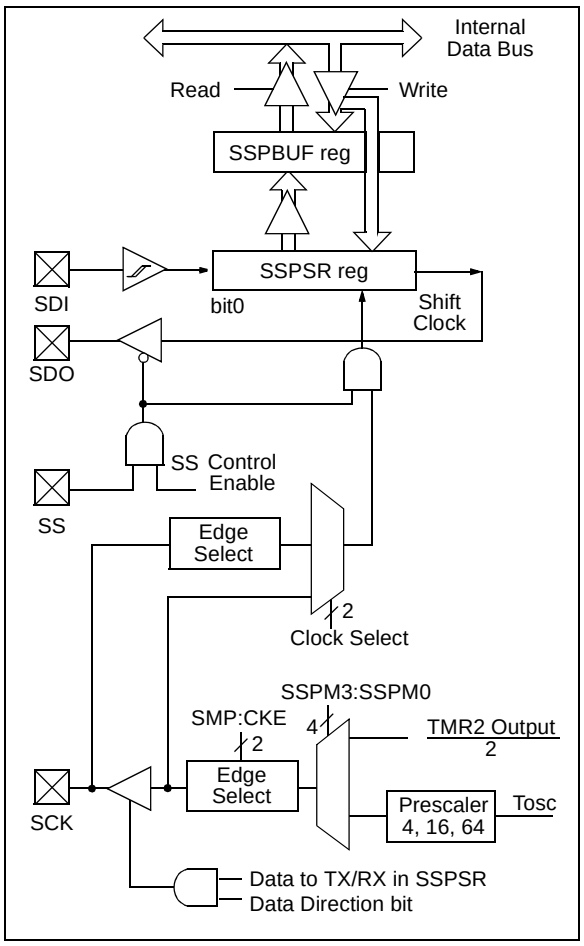



**FIGURE 15-2: $I^2C$ SLAVE MODE BLOCK DIAGRAM**

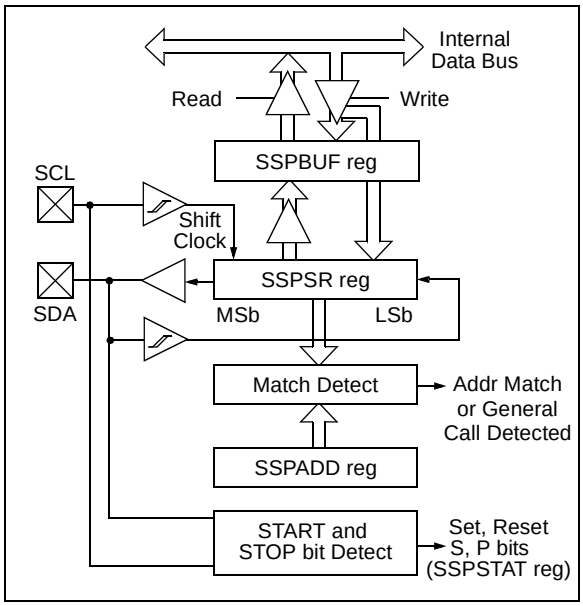
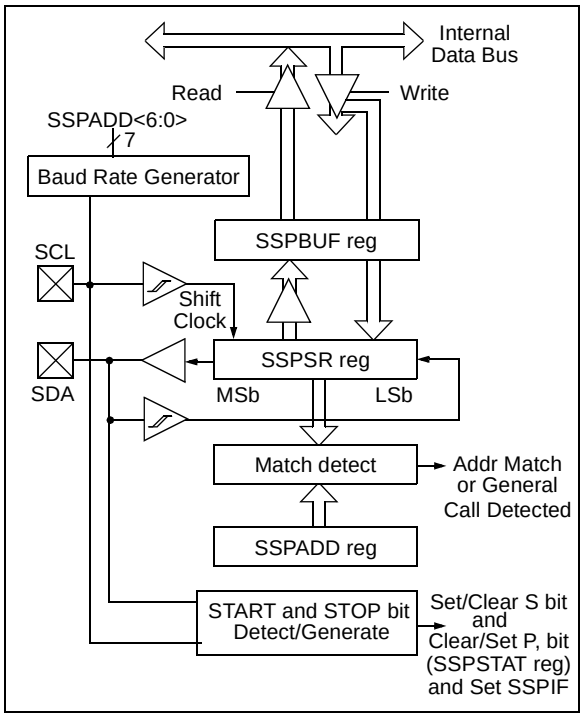



**FIGURE 15-3: $I^2C$ MASTER MODE BLOCK DIAGRAM**



**REGISTER 15-1: SSPSTAT: SYNC SERIAL PORT STATUS REGISTER (ADDRESS: 13h, BANK 6)**

| R/W-0 | R/W-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF |
| bit 7 | | | | | | | bit 0 |

bit 7 **SMP:** Sample bit

SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time

SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode

In $I^2C$ Master or Slave mode:
1 = Slew rate control disabled for Standard Speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for High Speed mode (400 kHz)

bit 6 **CKE**: SPI Clock Edge Select (Figure 15-6, Figure 15-8 and Figure 15-9)

CKP = 0:
1 = Data transmitted on rising edge of SCK
0 = Data transmitted on falling edge of SCK

CKP = 1:
1 = Data transmitted on falling edge of SCK
0 = Data transmitted on rising edge of SCK

bit 5 **D/$\overline{A}$**: Data/$\overline{Address}$ bit ($I^2C$ mode only)
1 = Indicates that the last byte received or transmitted was data
0 = Indicates that the last byte received or transmitted was address

bit 4 **P**: STOP bit
($I^2C$ mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)
1 = Indicates that a STOP bit has been detected last (this bit is '0' on RESET)
0 = STOP bit was not detected last

bit 3 **S**: START bit
($I^2C$ mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)
1 = Indicates that a START bit has been detected last (this bit is '0' on RESET)
0 = START bit was not detected last

bit 2 **R/$\overline{W}$**: Read/Write bit Information ($I^2C$ mode only)
This bit holds the R/W bit information following the last address match. This bit is only valid from the address match to the next START bit, STOP bit, or not ACK bit.

In $I^2C$ Slave mode:
1 = Read
0 = Write

In $I^2C$ Master mode:
1 = Transmit is in progress
0 = Transmit is not in progress
Or'ing this bit with SEN, RSEN, PEN, RCEN, or ACKEN will indicate if the MSSP is in IDLE mode.

bit 1 **UA**: Update Address (10-bit $I^2C$ mode only)
1 = Indicates that the user needs to update the address in the SSPADD register
0 = Address does not need to be updated

bit 0 **BF**: Buffer Full Status bit

Receive (SPI and $I^2C$ modes)
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty

Transmit ($I^2C$ mode only)
1 = Data transmit in progress (does not include the $\overline{ACK}$ and STOP bits), SSPBUF is full
0 = Data transmit complete (does not include the $\overline{ACK}$ and STOP bits), SSPBUF is empty

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |


© 2000 Microchip Technology Inc.

**REGISTER 15-2: SSPCON1: SYNC SERIAL PORT CONTROL REGISTER1 (ADDRESS 11h, BANK 6)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | $\overline{\text{SSPM1}}$ | $\overline{\text{SSPM0}}$ |
| bit 7 | | | | | | | bit 0 |

bit 7 **WCOL**: Write Collision Detect bit

Master mode:
1 = A write to the SSPBUF register was attempted while the $I^2C$ conditions were not valid for a transmission to be started
0 = No collision

Slave mode:
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6 **SSPOV**: Receive Overflow Indicator bit

In SPI mode:
1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. In Slave mode, the user must read the SSPBUF, even if only transmitting data, to avoid setting overflow. In Master mode, the overflow bit is not set, since each new reception (and transmission) is initiated by writing to the SSPBUF register. (Must be cleared in software.)
0 = No overflow

In $I^2C$ mode:
1 = A byte is received while the SSPBUF register is still holding the previous byte. SSPOV is a "don't care" in Transmit mode. (Must be cleared in software.)
0 = No overflow

bit 5 **SSPEN**: Synchronous Serial Port Enable bit
In both modes, when enabled, these pins must be properly configured as input or output.

In SPI mode:
1 = Enables serial port and configures SCK, SDO, SDI and $\overline{\text{SS}}$ as the source of the serial port pins
0 = Disables serial port and configures these pins as I/O port pins

In $I^2C$ mode:
1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins
0 = Disables serial port and configures these pins as I/O port pins

**Note:** In SPI mode, these pins must be properly configured as input or output.

bit 4 **CKP**: Clock Polarity Select bit

In SPI mode:
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level

In $I^2C$ Slave mode:
SCK release control
1 = Enable clock
0 = Holds clock low (clock stretch). (Used to ensure data setup time.)

In $I^2C$ Master mode:
Unused in this mode

bit 3-0 **SSPM3:SSPM0**: Synchronous Serial Port Mode Select bits
0000 = SPI Master mode, clock = FOSC/4
0001 = SPI Master mode, clock = FOSC/16
0010 = SPI Master mode, clock = FOSC/64
0011 = SPI Master mode, clock = TMR2 output/2
0100 = SPI Slave mode, clock = SCK pin, $\overline{\text{SS}}$ pin control enabled
0101 = SPI Slave mode, clock = SCK pin, $\overline{\text{SS}}$ pin control disabled, $\overline{\text{SS}}$ can be used as I/O pin
0110 = $I^2C$ Slave mode, 7-bit address
0111 = $I^2C$ Slave mode, 10-bit address
1000 = $I^2C$ Master mode, clock = FOSC / (4 * (SSPADD+1) )
1xx1 = Reserved
1x1x = Reserved

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

**REGISTER 15-3: SSPCON2: SYNC SERIAL PORT CONTROL REGISTER2 (ADDRESS 12h, BANK 6)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN |
| bit 7 | | | | | | | bit 0 |

bit 7 **GCEN**: General Call Enable bit (in $I^2C$ Slave mode only)
1 = Enable interrupt when a general call address (0000h) is received in the SSPSR
0 = General call address disabled

bit 6 **ACKSTAT**: Acknowledge Status bit (in $I^2C$ Master mode only)

In Master Transmit mode:
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT**: Acknowledge Data bit (in $I^2C$ Master mode only)

In Master Receive mode:
Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.
1 = Not Acknowledge
0 = Acknowledge

bit 4 **ACKEN**: Acknowledge Sequence Enable bit (in $I^2C$ Master mode only)

In Master Receive mode:
1 = Initiate Acknowledge sequence on SDA and SCL pins and transmit AKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence idle

**Note:** If the $I^2C$ module is not in the IDLE mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

bit 3 **RCEN**: Receive Enable bit (in $I^2C$ Master mode only)
1 = Enables Receive mode for $I^2C$
0 = Receive idle

**Note:** If the $I^2C$ module is not in the IDLE mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

bit 2 **PEN**: STOP Condition Enable bit (in $I^2C$ Master mode only)

SCK Release Control:
1 = Initiate STOP condition on SDA and SCL pins. Automatically cleared by hardware.
0 = STOP condition idle

**Note:** If the $I^2C$ module is not in the IDLE mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

bit 1 **RSEN**: Repeated Start Condition Enabled bit (in $I^2C$ Master mode only)
1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition idle

**Note:** If the $I^2C$ module is not in the IDLE mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

bit 0 **SEN**: START Condition Enabled bit (In $I^2C$ Master mode only)
1 = Initiate START condition on SDA and SCL pins. Automatically cleared by hardware.
0 = START condition idle.

**Note:** If the $I^2C$ module is not in the IDLE mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |


© 2000 Microchip Technology Inc.

## 15.1 SPI Mode

The SPI mode allows 8-bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

- Serial Data Out (SDO)
- Serial Data In (SDI)
- Serial Clock (SCK)

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select ($\overline{SS}$)

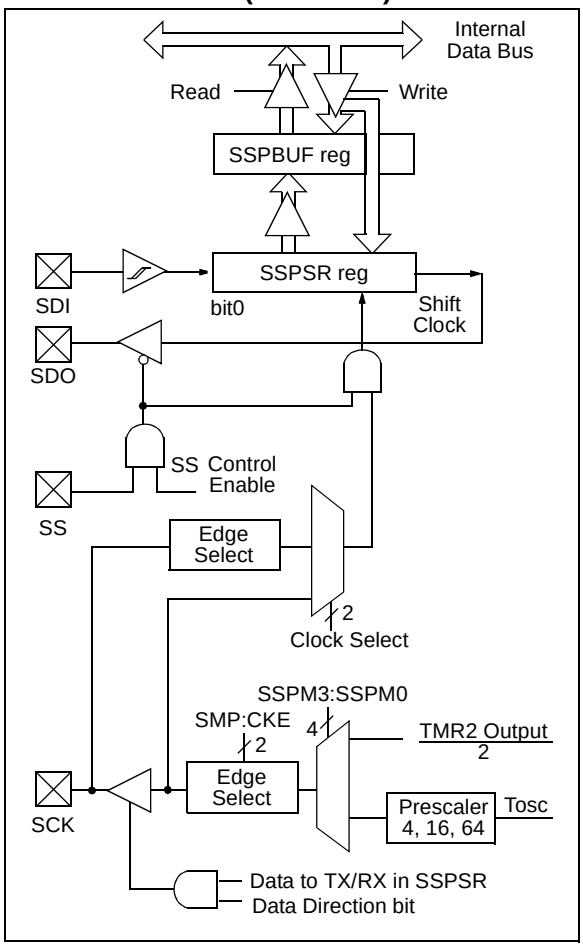
### 15.1.1 OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits in the SSPCON1 register (SSPCON1<5:0>) and SSPSTAT<7:6>. These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

Figure 15-4 shows the block diagram of the MSSP module when in SPI mode.

**FIGURE 15-4: MSSP BLOCK DIAGRAM (SPI MODE)**




The MSSP consists of a transmit/receive Shift Register (SSPSR) and a Buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR, until the received data is ready. Once the 8-bits of data have been received, that byte is moved to the SSPBUF register. Then the buffer full detect bit BF (SSPSTAT<0>) and the interrupt flag bit SSPIF (PIR2<7>) are set. This double buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored, and the write collision detect bit WCOL (SSPCON1<7>) will be set. User software must clear the WCOL bit so that it can be determined if the following write(s) to the SSPBUF register completed successfully.

When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. Buffer full bit, BF (SSPSTAT<0>), indicates when SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF is read, bit BF is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally the MSSP interrupt is used to determine when the transmission/reception has completed. The SSPBUF must be read and/or written. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur. Example 15-1 shows the loading of the SSPBUF (SSPSR) for data transmission.

**EXAMPLE 15-1: LOADING THE SSPBUF (SSPSR) REGISTER**

```
     MOVLB 6                ; Bank 6
LOOP BTFSS SSPSTAT, BF      ; Has data been
                            ;   received
                            ;   (transmit
                            ;   complete)?
     GOTO  LOOP             ; No
     MOVPF SSPBUF, RXDATA   ; Save in user RAM
     MOVFP TXDATA, SSPBUF   ; New data to xmit
```

The SSPSR is not directly readable, or writable and can only be accessed by addressing the SSPBUF register. Additionally, the MSSP status register (SSPSTAT) indicates the various status conditions.

### 15.1.2 ENABLING SPI I/O

To enable the serial port, MSSP Enable bit, SSPEN (SSPCON1<5>), must be set. To reset or reconfigure SPI mode, clear bit SSPEN, re-initialize the SSPCON registers and then set bit SSPEN. This configures the SDI, SDO, SCK and $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the DDR register) appropriately programmed. That is:

- SDI is automatically controlled by the SPI module
- SDO must have DDRB<7> cleared
- SCK (Master mode) must have DDRB<6> cleared
- SCK (Slave mode) must have DDRB<6> set
- $\overline{SS}$ must have PORTA<2> set

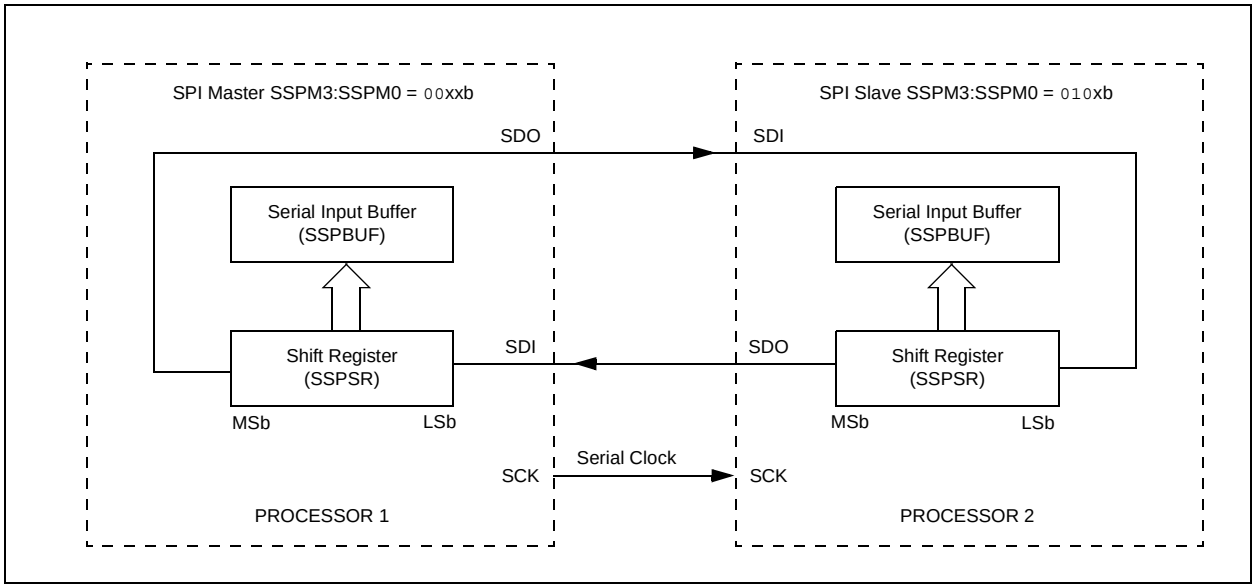
Any serial port function that is not desired may be overridden by programming the corresponding data direction (DDR) register to the opposite value.

### 15.1.3 TYPICAL CONNECTION

Figure 15-5 shows a typical connection between two microcontrollers. The master controller (Processor 1) initiates the data transfer by sending the SCK signal. Data is shifted out of both shift registers on their programmed clock edge and latched on the opposite edge of the clock. Both processors should be programmed to same Clock Polarity (CKP), then both controllers would send and receive data at the same time. Whether the data is meaningful (or dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends data — Slave sends dummy data
- Master sends data — Slave sends data
- Master sends dummy data — Slave sends data

**FIGURE 15-5: SPI MASTER/SLAVE CONNECTION**





© 2000 Microchip Technology Inc.

### 15.1.4 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave (Processor 2, Figure 15-5) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a "Line Activity Monitor" mode.

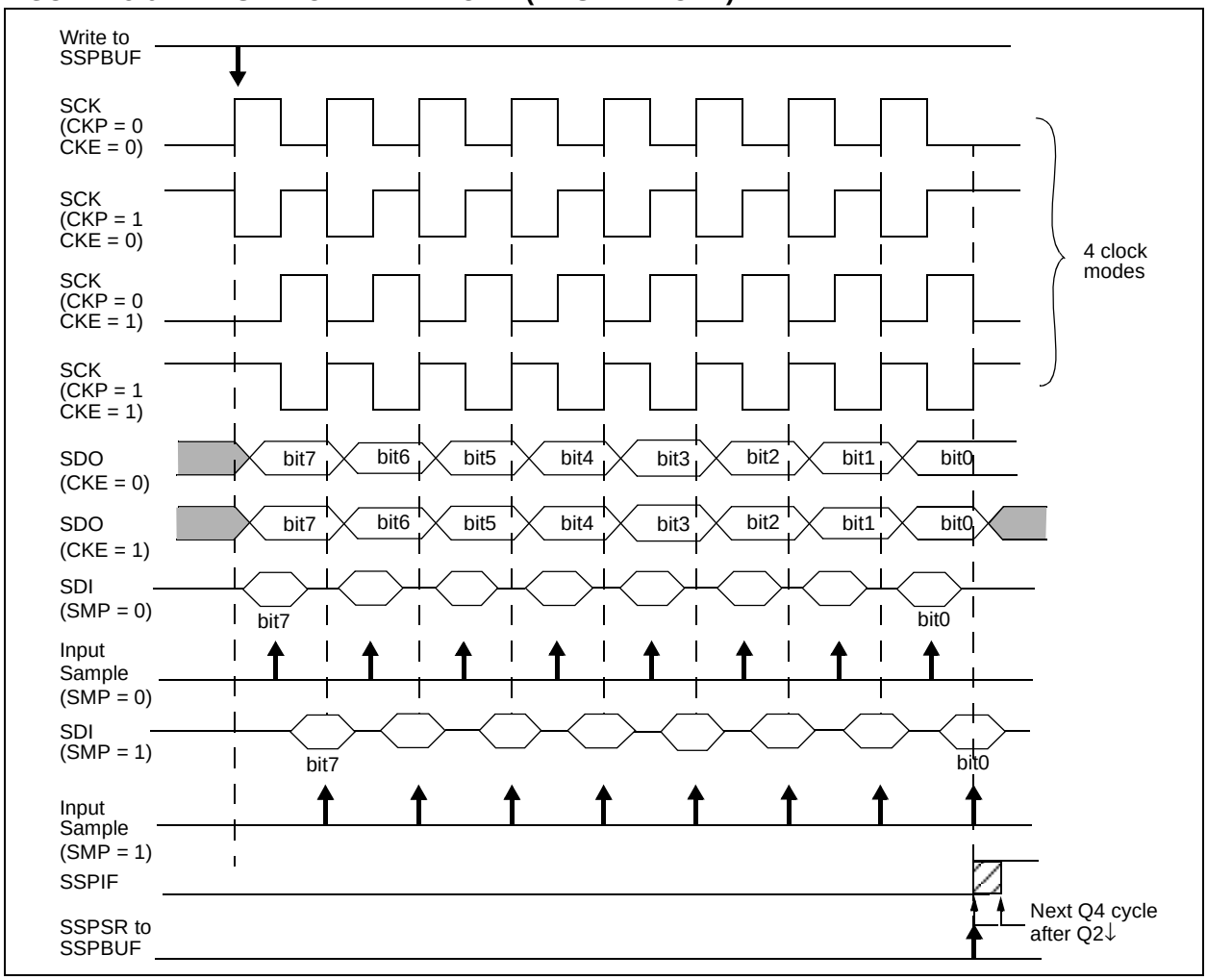
The clock polarity is selected by appropriately programming bit CKP (SSPCON1<4>). This then, would give waveforms for SPI communication as shown in Figure 15-6, Figure 15-8 and Figure 15-9, where the MSb is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- FOSC/4 (or TCY)
- FOSC/16 (or 4 • TCY)
- FOSC/64 (or 16 • TCY)
- Timer2 output/2

This allows a maximum bit clock frequency (at 33 MHz) of 8.25 MHz.

Figure 15-6 shows the waveforms for Master mode. When CKE = 1, the SDO data is valid before there is a clock edge on SCK. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPBUF is loaded with the received data is shown.

**FIGURE 15-6: SPI MODE WAVEFORM (MASTER MODE)**



### 15.1.5 SLAVE MODE

In Slave mode, the data is transmitted and received as the external clock pulses appear on SCK. When the last bit is latched, the interrupt flag bit SSPIF (PIR2<7>) is set.

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in SLEEP mode, the slave can transmit/receive data. When a byte is received, the device will wake-up from SLEEP.

### 15.1.6 SLAVE SELECT SYNCHRONIZATION

The $\overline{SS}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with $\overline{SS}$ pin control enabled (SSPCON1<3:0> = 04h). The pin must not be driven low for the $\overline{SS}$ pin to function as an input. The RA2 Data Latch must be high. When the $\overline{SS}$ pin is low, transmission and reception are enabled and the SDO pin is driven. When the $\overline{SS}$ pin goes high, the SDO pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable, depending on the application.

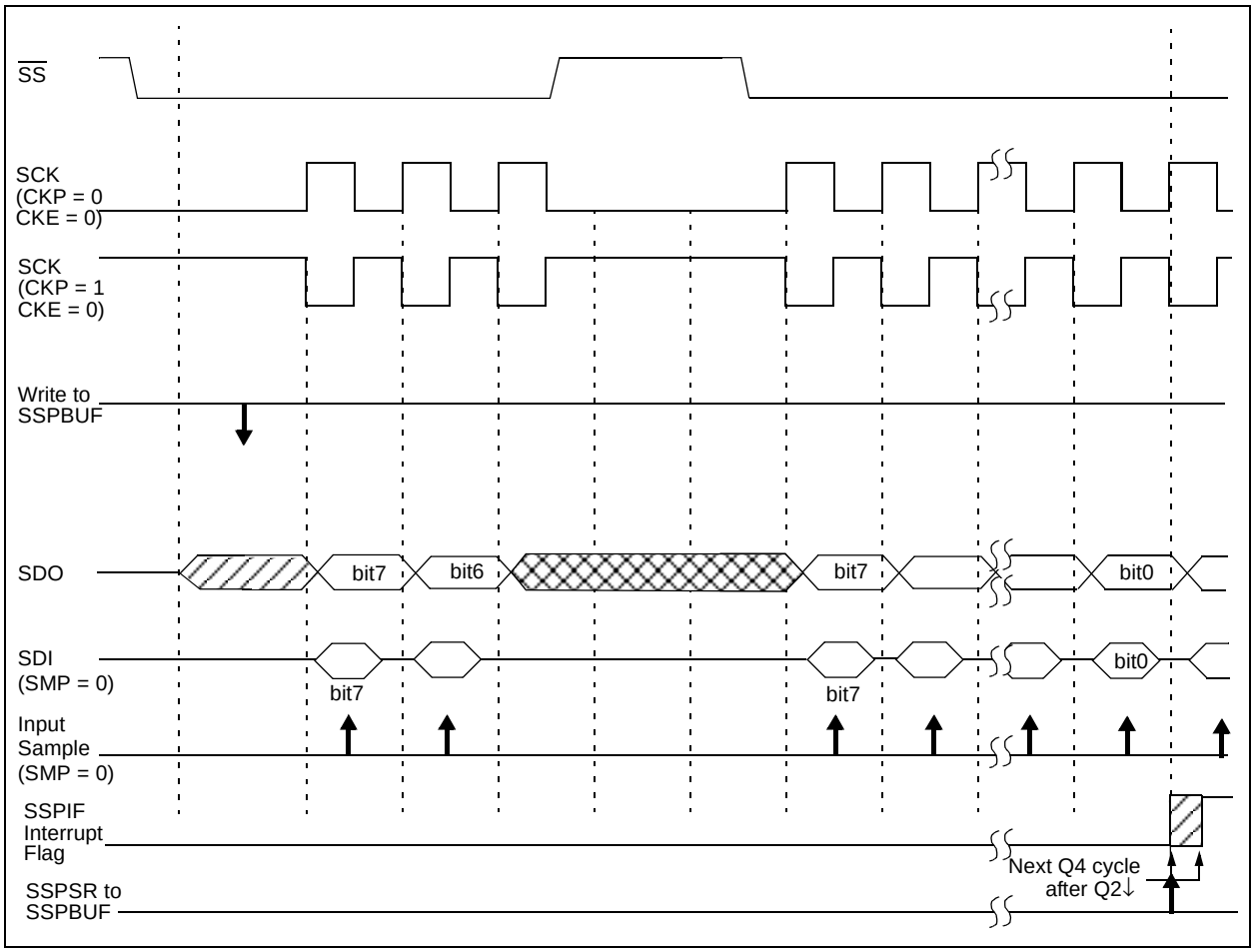
**Note 1:** When the SPI is in Slave mode with $\overline{SS}$ pin control enabled (SSPCON<3:0> = 0100), the SPI module will reset if the $\overline{SS}$ pin is set to $V_{DD}$.

**2:** If the SPI is used in Slave mode with CKE = '1', then the $\overline{SS}$ pin control must be enabled.

When the SPI module resets, the bit counter is forced to 0. This can be done by either forcing the $\overline{SS}$ pin to a high level, or clearing the SSPEN bit.

To emulate two-wire communication, the SDO pin can be connected to the SDI pin. When the SPI needs to operate as a receiver, the SDO pin can be configured as an input. This disables transmissions from the SDO. The SDI can always be left as an input (SDI function), since it cannot create a bus conflict.

**FIGURE 15-7: SLAVE SYNCHRONIZATION WAVEFORM**





© 2000 Microchip Technology Inc.

**FIGURE 15-8: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)**

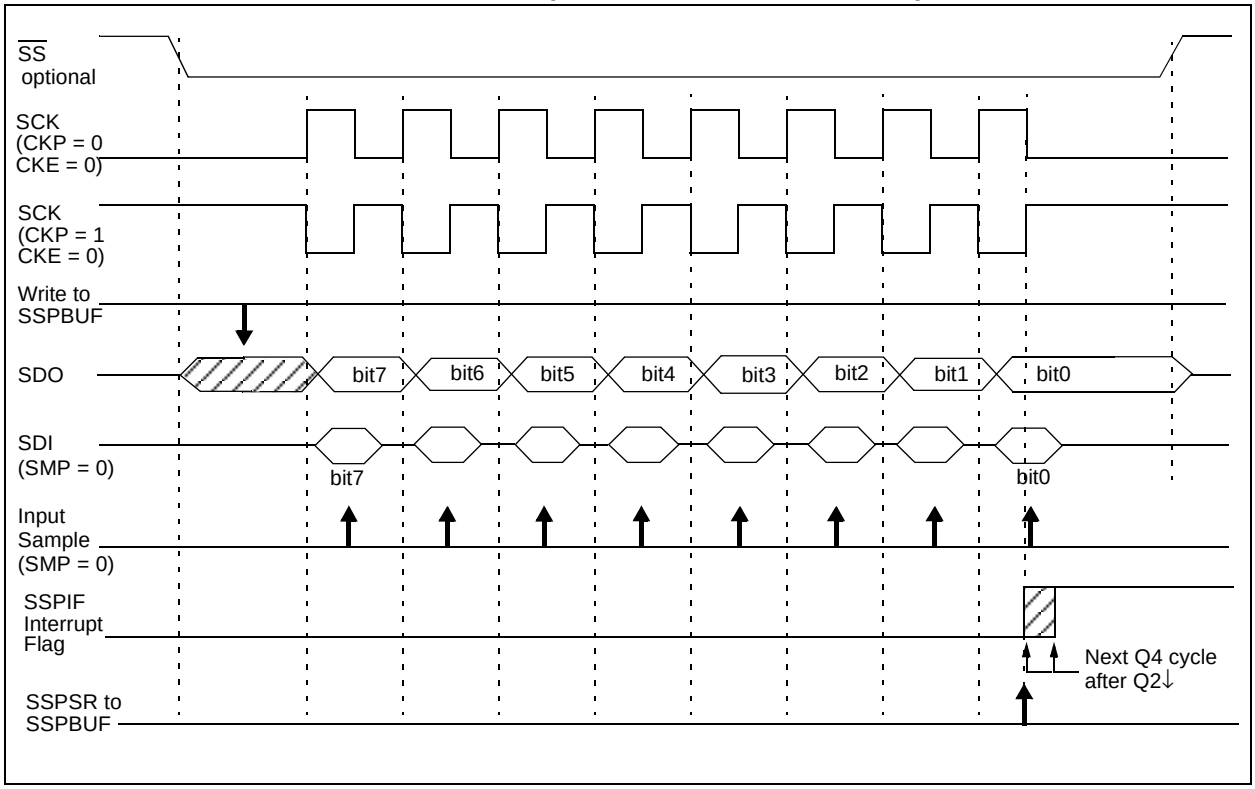
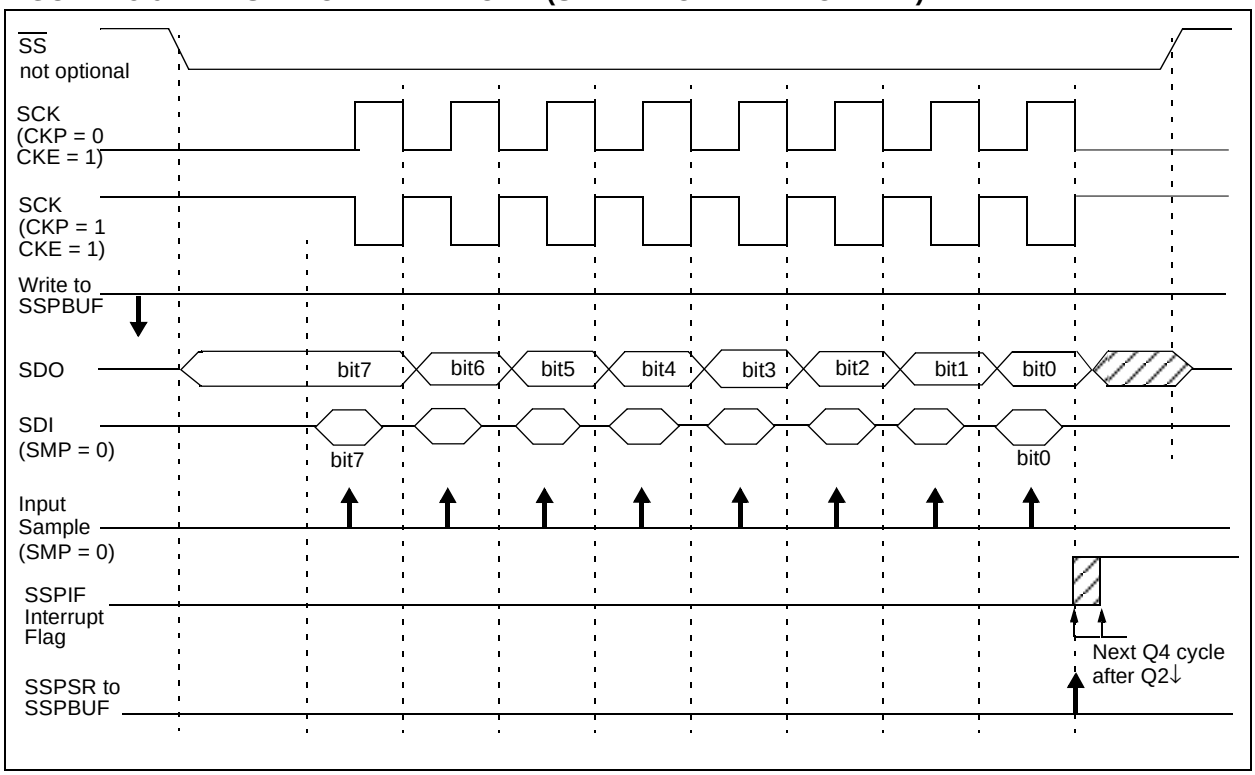



**FIGURE 15-9: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)**



### 15.1.7 SLEEP OPERATION

In Master mode, all module clocks are halted, and the transmission/reception will remain in that state until the device wakes from SLEEP. After the device returns to normal mode, the module will continue to transmit/receive data.

In Slave mode, the SPI transmit/receive shift register operates asynchronously to the device. This allows the device to be placed in SLEEP mode and data to be shifted into the SPI transmit/receive shift register. When all 8-bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device from SLEEP.

### 15.1.8 EFFECTS OF A RESET

A RESET disables the MSSP module and terminates the current transfer.

**TABLE 15-1: REGISTERS ASSOCIATED WITH SPI OPERATION**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 07h, Unbanked | INTSTA | PEIF | T0CKIF | T0IF | INTF | PEIE | T0CKIE | T0IE | INTE | 0000 0000 | 0000 0000 |
| 10h, Bank 4 | PIR2 | SSPIF | BCLIF | ADIF | — | CA4IF | CA3IF | TX2IF | RC2IF | 000- 0010 | 000- 0010 |
| 11h, Bank 4 | PIE2 | SSPIE | BCLIE | ADIE | — | CA4IE | CA3IE | TX2IE | RC2IE | 000- 0000 | 000- 0000 |
| 14h, Bank 6 | SSPBUF | Synchronous Serial Port Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 11h, Bank 6 | SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 0000 0000 | 0000 0000 |
| 13h, Bank 6 | SSPSTAT | SMP | CKE | D/$\overline{\text{A}}$ | P | S | R/$\overline{\text{W}}$ | UA | BF | 0000 0000 | 0000 0000 |

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the SSP in SPI mode.


© 2000 Microchip Technology Inc.

## 15.2 MSSP I²C Operation

The MSSP module in I²C mode fully implements all master and slave functions (including general call support) and provides interrupts on START and STOP bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications as well as 7-bit and 10-bit addressing.

Refer to Application Note AN578, *"Use of the SSP Module in the I²C Multi-Master Environment."*

A "glitch" filter is on the SCL and SDA pins when the pin is an input. This filter operates in both the 100 kHz and 400 kHz modes. In the 100 kHz mode, when these pins are an output, there is a slew rate control of the pin that is independent of device frequency.

**FIGURE 15-10: I²C SLAVE MODE BLOCK DIAGRAM**

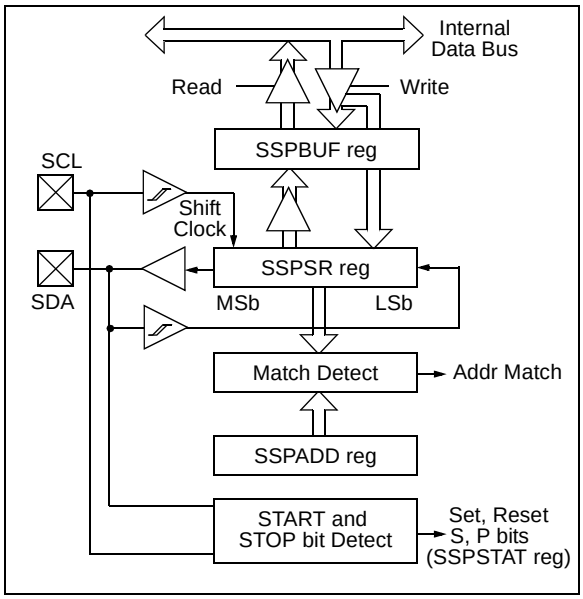
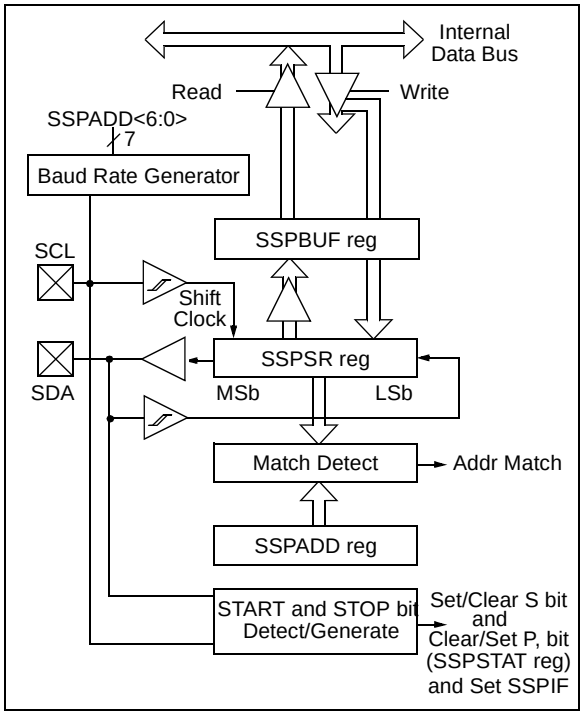



**FIGURE 15-11: I²C MASTER MODE BLOCK DIAGRAM**




Two pins are used for data transfer. These are the SCL pin, which is the clock and the SDA pin, which is the data. The SDA and SCL pins are automatically configured when the I²C mode is enabled. The SSP module functions are enabled by setting SSP Enable bit SSPEN (SSPCON1<5>).

The MSSP module has six registers for I²C operation. These are the:

- SSP Control Register1 (SSPCON1)
- SSP Control Register2 (SSPCON2)
- SSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer (SSPBUF)
- SSP Shift Register (SSPSR) - Not directly accessible
- SSP Address Register (SSPADD)

The SSPCON1 register allows control of the I²C operation. Four mode selection bits (SSPCON1<3:0>) allow one of the following I²C modes to be selected:

- I²C Slave mode (7-bit address)
- I²C Slave mode (10-bit address)
- I²C Master mode, clock = OSC/4 (SSPADD +1)

Before selecting any I²C mode, the SCL and SDA pins must be programmed to inputs by setting the appropriate DDR bits. Selecting an I²C mode, by setting the SSPEN bit, enables the SCL and SDA pins to be used as the clock and data lines in I²C mode.

The SSPSTAT register gives the status of the data transfer. This information includes detection of a START or STOP bit, specifies if the received byte was data or address if the next byte is the completion of 10-bit address and if this will be a read or write data transfer.

The SSPBUF is the register to which transfer data is written to or read from. The SSPSR register shifts the data in or out of the device. In receive operations, the SSPBUF and SSPSR create a doubled buffered receiver. This allows reception of the next byte to begin before reading the last byte of received data. When the complete byte is received, it is transferred to the SSPBUF register and flag bit SSPIF is set. If another complete byte is received before the SSPBUF register is read, a receiver overflow has occurred and bit SSPOV (SSPCON1<6>) is set and the byte in the SSPSR is lost.

The SSPADD register holds the slave address. In 10-bit mode, the user needs to write the high byte of the address (1111 0 A9 A8 0). Following the high byte address match, the low byte of the address needs to be loaded (A7:A0).

### 15.2.1 SLAVE MODE

In Slave mode, the SCL and SDA pins must be configured as inputs. The MSSP module will override the input state with the output data when required (slave-transmitter).

When an address is matched or the data transfer after an address match is received, the hardware automatically will generate the acknowledge ($\overline{ACK}$) pulse and then load the SSPBUF register with the received value currently in the SSPSR register.

There are certain conditions that will cause the MSSP module not to give this $\overline{ACK}$ pulse. These are if either (or both):

a) The buffer full bit BF (SSPSTAT<0>) was set before the transfer was received.

b) The overflow bit SSPOV (SSPCON1<6>) was set before the transfer was received.

If the BF bit is set, the SSPSR register value is not loaded into the SSPBUF, but bit SSPIF and SSPOV are set. Table 15-2 shows what happens when a data transfer byte is received, given the status of bits BF and SSPOV. The shaded cells show the condition where user software did not properly clear the overflow condition. Flag bit BF is cleared by reading the SSPBUF register, while bit SSPOV is cleared through software.

The SCL clock input must have a minimum high and low time for proper operation. The high and low times of the $I^2C$ specification, as well as the requirement of the MSSP module, are shown in timing parameter #100 and parameter #101 of the Electrical Specifications.


© 2000 Microchip Technology Inc.

#### 15.2.1.1 Addressing

Once the MSSP module has been enabled, it waits for a START condition to occur. Following the START condition, the 8-bits are shifted into the SSPSR register. All incoming bits are sampled with the rising edge of the clock (SCL) line. The value of register SSPSR<7:1> is compared to the value of the SSPADD register. The address is compared on the falling edge of the eighth clock (SCL) pulse. If the addresses match and the BF and SSPOV bits are clear, the following events occur:

a) The SSPSR register value is loaded into the SSPBUF register on the falling edge of the 8th SCL pulse.
b) The buffer full bit, BF, is set on the falling edge of the 8th SCL pulse.
c) An $\overline{ACK}$ pulse is generated.
d) SSP interrupt flag bit, SSPIF (PIR2<7>), is set (interrupt is generated if enabled) - on the falling edge of the 9th SCL pulse.

In 10-bit address mode, two address bytes need to be received by the slave. The five Most Significant bits (MSbs) of the first address byte specify if this is a 10-bit address. Bit R/$\overline{W}$ (SSPSTAT<2>) must specify a write so the slave device will receive the second address byte. For a 10-bit address, the first byte would equal '1111 0 A9 A8 0', where A9 and A8 are the two MSbs of the address. The sequence of events for a 10-bit address is as follows, with steps 7- 9 for slave-transmitter:

1. Receive first (high) byte of Address (bits SSPIF, BF and bit UA (SSPSTAT<1>) are set).
2. Update the SSPADD register with second (low) byte of Address (clears bit UA and releases the SCL line).
3. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.
4. Receive second (low) byte of Address (bits SSPIF, BF and UA are set).
5. Update the SSPADD register with the first (high) byte of Address. This will clear bit UA and release the SCL line.
6. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.
7. Receive Repeated Start condition.
8. Receive first (high) byte of Address (bits SSPIF and BF are set).
9. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.

| | |
|---|---|
| **Note:** | Following the Repeated Start condition (step 7) in 10-bit mode, the user only needs to match the first 7-bit address. The user does not update the SSPADD for the second half of the address. |

#### 15.2.1.2 Slave Reception

When the R/$\overline{W}$ bit of the address byte is clear and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register.

When the address byte overflow condition exists, then no acknowledge ($\overline{ACK}$) pulse is given. An overflow condition is defined as either bit BF (SSPSTAT<0>) is set, or bit SSPOV (SSPCON1<6>) is set.

An SSP interrupt is generated for each data transfer byte. Flag bit SSPIF (PIR2<7>) must be cleared in software. The SSPSTAT register is used to determine the status of the received byte.

| | |
|---|---|
| **Note:** | The SSPBUF will be loaded if the SSPOV bit is set and the BF flag is cleared. If a read of the SSPBUF was performed, but the user did not clear the state of the SSPOV bit before the next receive occurred, the $\overline{ACK}$ is not sent and the SSPBUF is updated. |

**TABLE 15-2: DATA TRANSFER RECEIVED BYTE ACTIONS**

| Status Bits as Data Transfer is Received | | SSPSR → SSPBUF | Generate $\overline{ACK}$ Pulse | Set bit SSPIF (SSP Interrupt occurs if enabled) |
|---|---|---|---|---|
| BF | SSPOV | | | |
| 0 | 0 | Yes | Yes | Yes |
| 1 | 0 | No | No | Yes |
| 1 | 1 | No | No | Yes |
| 0 | 1 | Yes | No | Yes |

**Note 1:** Shaded cells show the conditions where the user software did not properly clear the overflow condition.

### 15.2.1.3 Slave Transmission

When the R/$\overline{W}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register. The $\overline{ACK}$ pulse will be sent on the ninth bit, and the SCL pin is held low. The transmit data must be loaded into the SSPBUF register, which also loads the SSPSR register. Then SCL pin should be enabled by setting bit CKP (SSPCON1<4>). The master must monitor the SCL pin prior to asserting another clock pulse. The slave devices may be holding off the master by stretching the clock. The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time (Figure 15-13).

An SSP interrupt is generated for each data transfer byte. The SSPIF flag bit must be cleared in software, and the SSPSTAT register is used to determine the status of the byte transfer. The SSPIF flag bit is set on the falling edge of the ninth clock pulse.

As a slave-transmitter, the $\overline{ACK}$ pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. If the SDA line was high (not $\overline{ACK}$), then the data transfer is complete. When the not $\overline{ACK}$ is latched by the slave, the slave logic is reset and the slave then monitors for another occurrence of the START bit. If the SDA line was low ($\overline{ACK}$), the transmit data must be loaded into the SSPBUF register, which also loads the SSPSR register. Then, the SCL pin should be enabled by setting the CKP bit.

**FIGURE 15-12: I²C WAVEFORMS FOR RECEPTION (7-BIT ADDRESS)**

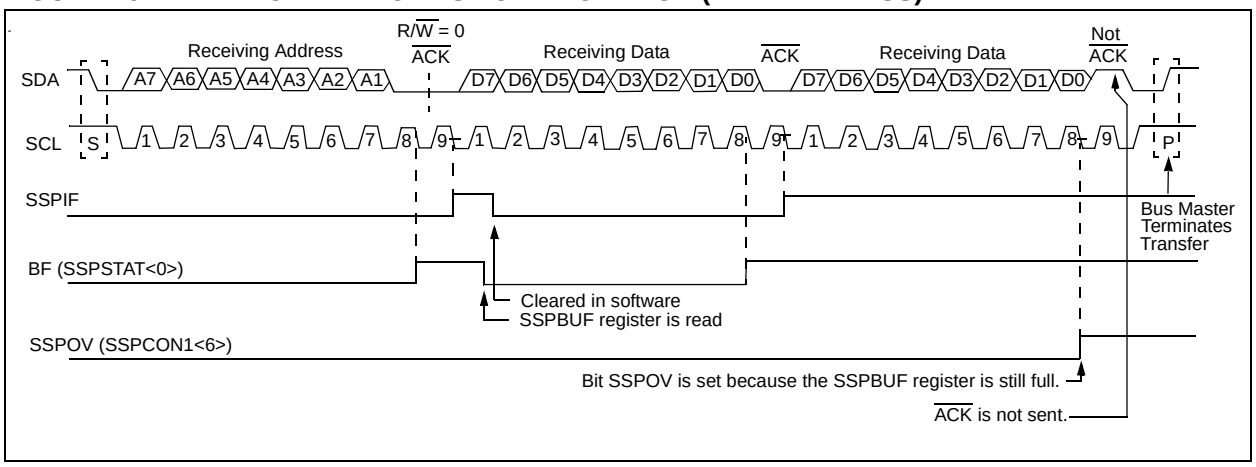
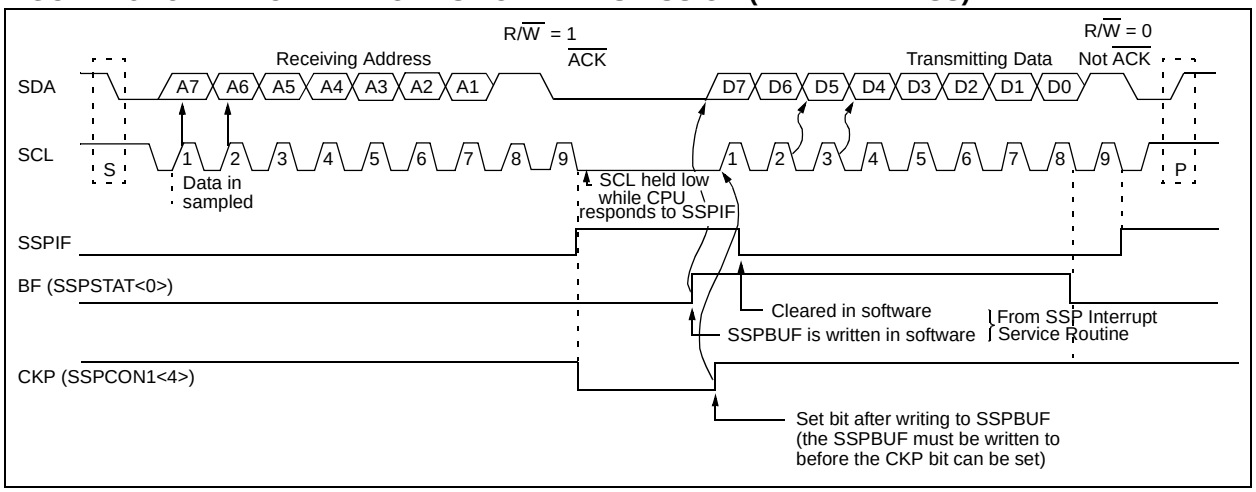



**FIGURE 15-13: I²C WAVEFORMS FOR TRANSMISSION (7-BIT ADDRESS)**





© 2000 Microchip Technology Inc.

**FIGURE 15-14: $I^2C$ SLAVE-TRANSMITTER (10-BIT ADDRESS)**

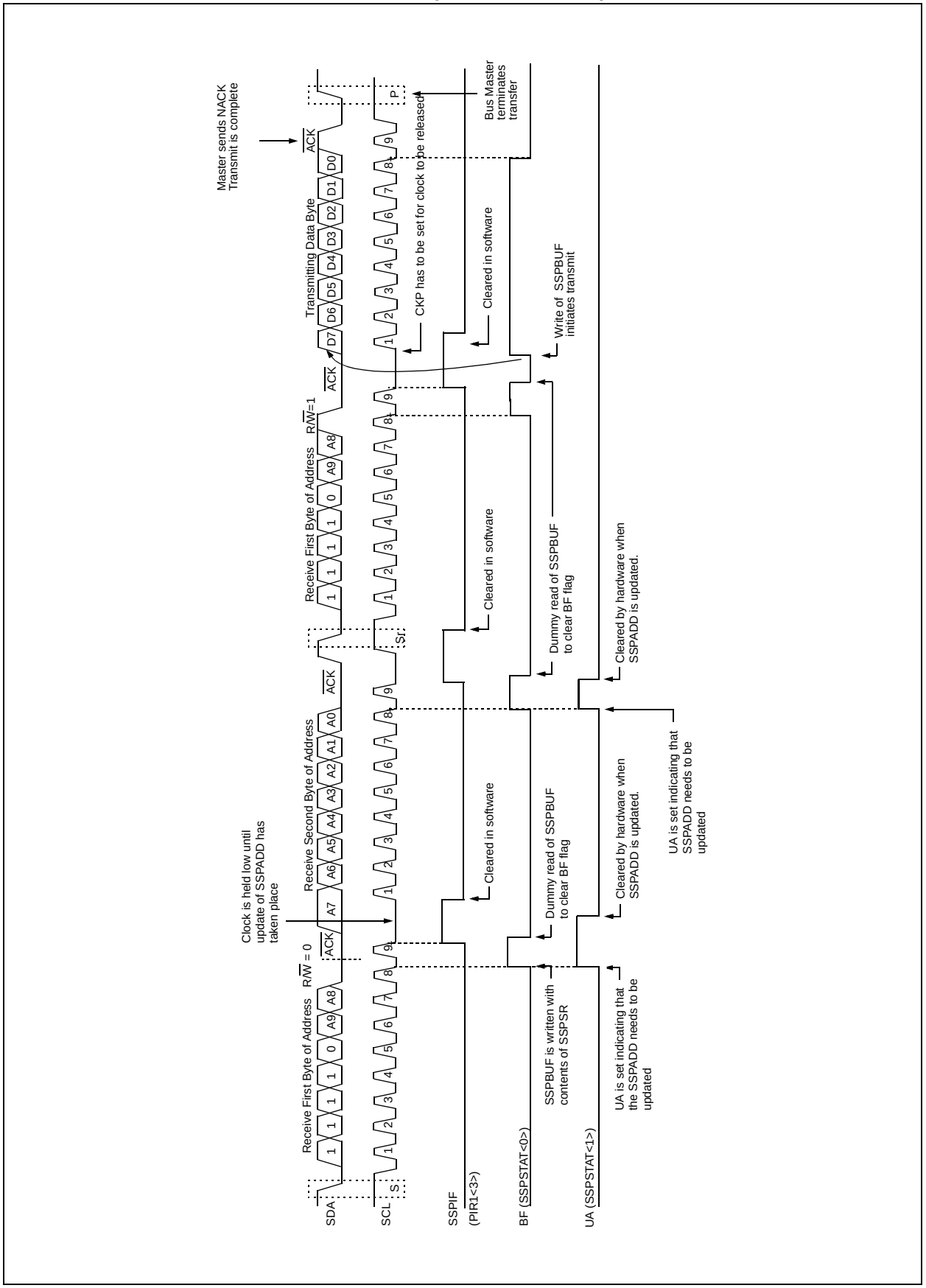
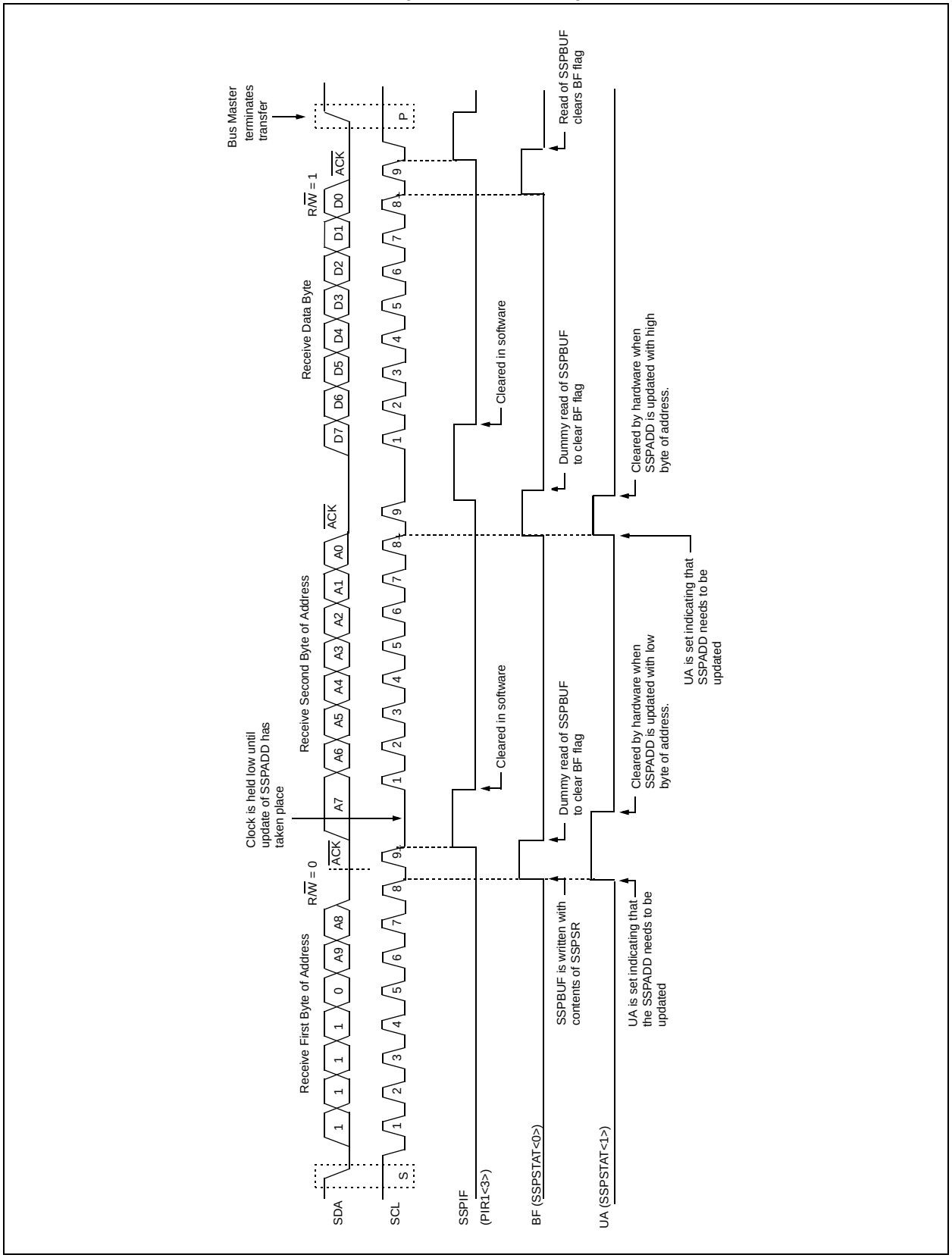


**FIGURE 15-15: $I^2C$ SLAVE-RECEIVER (10-BIT ADDRESS)**





© 2000 Microchip Technology Inc.

### 15.2.2 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the $I^2C$ bus is such that the first byte after the START condition usually determines which device will be the slave addressed by the master. The exception is the general call address, which can address all devices. When this address is used, all devices should, in theory, respond with an acknowledge.

The general call address is one of eight addresses reserved for specific purposes by the $I^2C$ protocol. It consists of all 0's with R/$\overline{W}$ = 0.

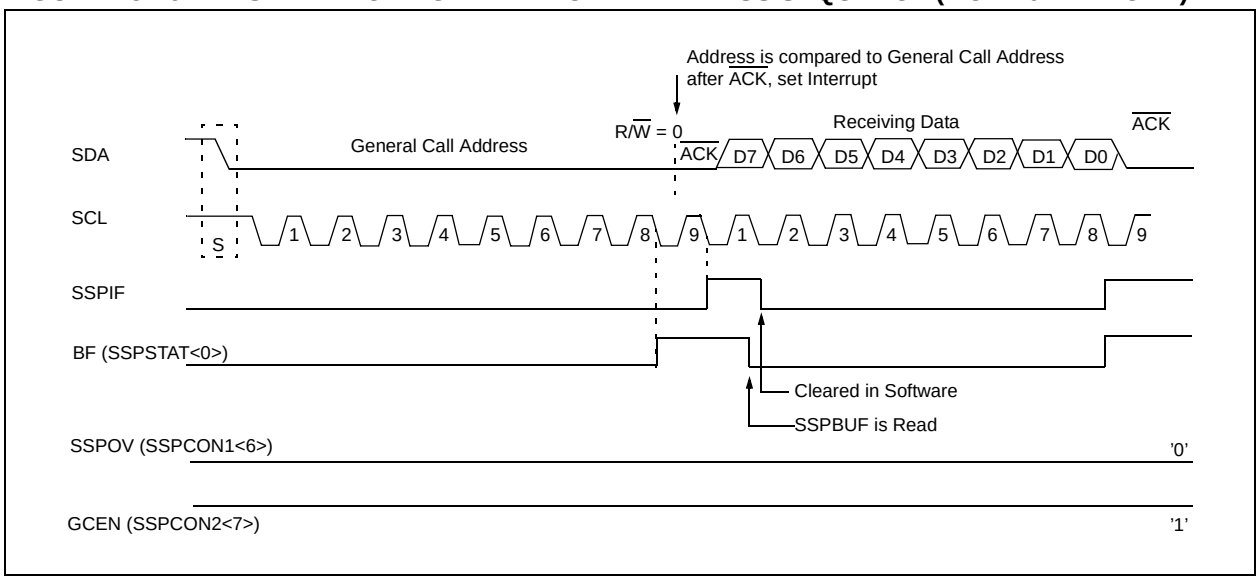
The general call address is recognized when the General Call Enable bit (GCEN) is enabled (SSPCON2<7> is set). Following a START bit detect, 8-bits are shifted into SSPSR and the address is compared against SSPADD and is also compared to the general call address, fixed in hardware.

If the general call address matches, the SSPSR is transferred to the SSPBUF, the BF flag is set (eighth bit) and on the falling edge of the ninth bit ($\overline{ACK}$ bit), the SSPIF flag is set.

When the interrupt is serviced, the source for the interrupt can be checked by reading the contents of the SSPBUF to determine if the address was device specific, or a general call address.

In 10-bit mode, the SSPADD is required to be updated for the second half of the address to match and the UA bit is set (SSPSTAT<1>). If the general call address is sampled when GCEN is set, while the slave is configured in 10-bit address mode, then the second half of the address is not necessary, the UA bit will not be set and the slave will begin receiving data after the acknowledge (Figure 15-16).

**FIGURE 15-16: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE (7 OR 10-BIT MODE)**



### 15.2.3 SLEEP OPERATION

While in SLEEP mode, the $I^2C$ module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from SLEEP (if the SSP interrupt is enabled).

### 15.2.4 EFFECTS OF A RESET

A RESET disables the SSP module and terminates the current transfer.

**TABLE 15-3: REGISTERS ASSOCIATED WITH $I^2C$ OPERATION**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | POR, BOR | $\overline{MCLR}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 07h, Unbanked | INTSTA | PEIF | T0CKIF | T0IF | INTF | PEIE | T0CKIE | T0IE | INTE | 0000 0000 | 0000 0000 |
| 10h, Bank 4 | PIR2 | SSPIF | BCLIF | ADIF | — | CA4IF | CA3IF | TX2IF | RC2IF | 000- 0000 | 000- 0000 |
| 11h, Bank 4 | PIE2 | SSPIE | BCLIE | ADIE | — | CA4IE | CA3IE | TX2IE | RC2IE | 000- 0000 | 000- 0000 |
| 10h. Bank 6 | SSPADD | Synchronous Serial Port ($I^2C$ mode) Address Register | | | | | | | | 0000 0000 | 0000 0000 |
| 14h, Bank 6 | SSPBUF | Synchronous Serial Port Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 11h, Bank 6 | SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 0000 0000 | 0000 0000 |
| 12h, Bank 6 | SSPCON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 0000 0000 | 0000 0000 |
| 13h, Bank 6 | SSPSTAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 0000 0000 | 0000 0000 |

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the SSP in $I^2C$ mode.


© 2000 Microchip Technology Inc.

### 15.2.5 MASTER MODE

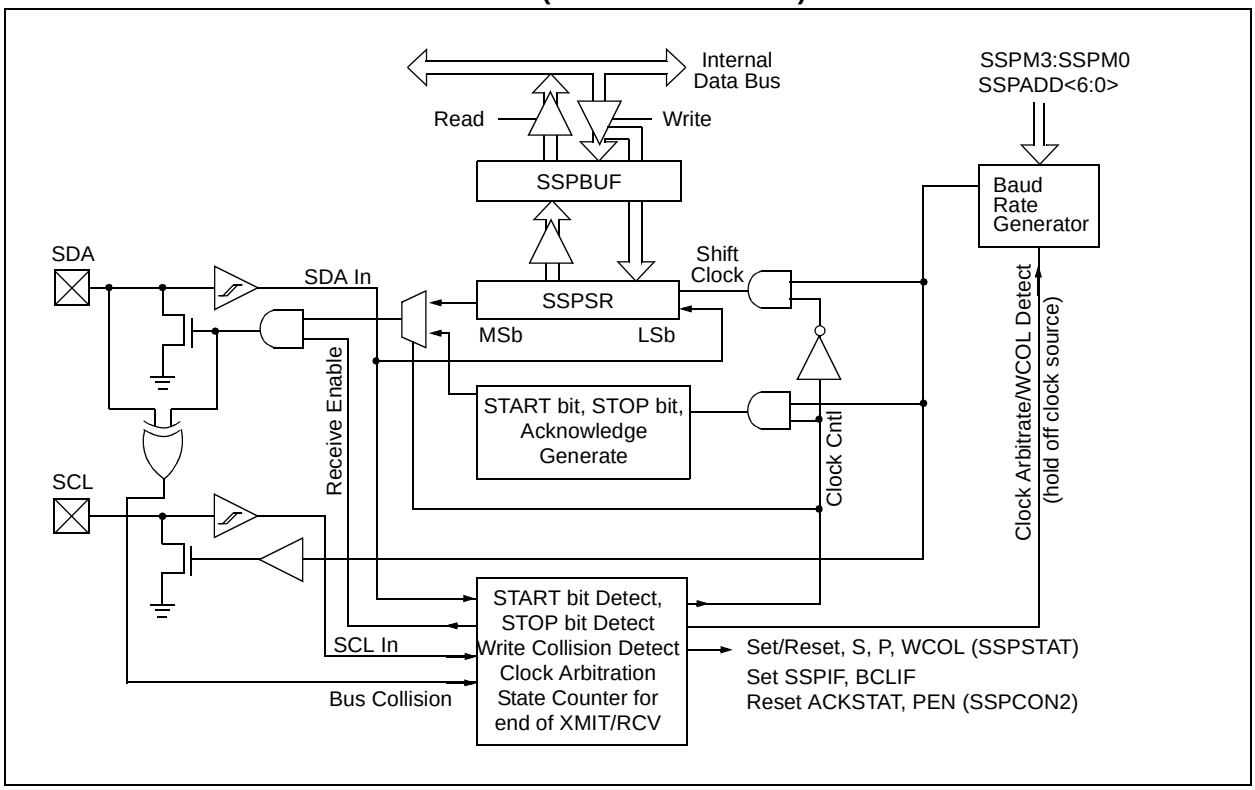
Master mode of operation is supported by interrupt generation on the detection of the START and STOP conditions. The STOP (P) and START (S) bits are cleared from a RESET, or when the MSSP module is disabled. Control of the $I^2C$ bus may be taken when the P bit is set, or the bus is idle, with both the S and P bits clear.

In Master mode, the SCL and SDA lines are manipulated by the MSSP hardware.

The following events will cause SSP Interrupt Flag bit, SSPIF, to be set (SSP Interrupt if enabled):

- START condition
- STOP condition
- Data transfer byte transmitted/received
- Acknowledge transmit
- Repeated Start

**FIGURE 15-17: SSP BLOCK DIAGRAM ($I^2C$ MASTER MODE)**



### 15.2.6 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the START and STOP conditions allows the determination of when the bus is free. The STOP (P) and START (S) bits are cleared from a RESET, or when the MSSP module is disabled. Control of the $I^2C$ bus may be taken when bit P (SSPSTAT<4>) is set, or the bus is idle, with both the S and P bits clear. When the bus is busy, enabling the SSP interrupt will generate the interrupt when the STOP condition occurs.

In Multi-Master operation, the SDA line must be monitored for arbitration, to see if the signal level is the expected output level. This check is performed in hardware, with the result placed in the BCLIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A START Condition
- A Repeated Start Condition
- An Acknowledge Condition

### 15.2.7 $I^2C$ MASTER MODE SUPPORT

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPCON1 and by setting the SSPEN bit. Once Master mode is enabled, the user has six options.

- Assert a START condition on SDA and SCL.
- Assert a Repeated Start condition on SDA and SCL.
- Write to the SSPBUF register initiating transmission of data/address.
- Generate a STOP condition on SDA and SCL.
- Configure the $I^2C$ port to receive data.
- Generate an Acknowledge condition at the end of a received byte of data.

| | |
|---|---|
| **Note:** | The MSSP Module, when configured in $I^2C$ Master mode, does not allow queueing of events. For instance: The user is not allowed to initiate a START condition and immediately write the SSPBUF register to initiate transmission before the START condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur. |

#### 15.2.7.1 $I^2C$ Master Mode Operation

The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the $I^2C$ bus will not be released.

In Master Transmitter mode, serial data is output through SDA, while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/$\overline{W}$) bit. In this case, the R/$\overline{W}$ bit will be logic '0'. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/$\overline{W}$ bit. In this case, the R/$\overline{W}$ bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address, followed by a '1' to indicate receive bit. Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions indicate the beginning and end of transmission.

The baud rate generator used for SPI mode operation is now used to set the SCL clock frequency for either 100 kHz, 400 kHz, or 1 MHz $I^2C$ operation. The baud rate generator reload value is contained in the lower 7 bits of the SSPADD register. The baud rate generator will automatically begin counting on a write to the SSPBUF. Once the given operation is complete (i.e., transmission of the last data bit is followed by ACK), the internal clock will automatically stop counting and the SCL pin will remain in its last state


© 2000 Microchip Technology Inc.

A typical transmit sequence would go as follows:

a) The user generates a START Condition by setting the START enable bit (SEN) in SSPCON2.
b) SSPIF is set. The module will wait the required START time before any other operation takes place.
c) The user loads the SSPBUF with address to transmit.
d) Address is shifted out the SDA pin until all 8 bits are transmitted.
e) The MSSP Module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
f) The module generates an interrupt at the end of the ninth clock cycle by setting SSPIF.
g) The user loads the SSPBUF with eight bits of data.
h) DATA is shifted out the SDA pin until all 8 bits are transmitted.
i) The MSSP Module shifts in the ACK bit from the slave device, and writes its value into the SSPCON2 register (SSPCON2<6>).
j) The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
k) The user generates a STOP condition by setting the STOP enable bit PEN in SSPCON2.
l) Interrupt is generated once the STOP condition is complete.

### 15.2.8 BAUD RATE GENERATOR

In $I^2C$ Master mode, the reload value for the BRG is located in the lower 7 bits of the SSPADD register (Figure 15-18). When the BRG is loaded with this value, the BRG counts down to 0 and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle ($T_{CY}$), on the Q2 and Q4 clock.

In $I^2C$ Master mode, the BRG is reloaded automatically. If Clock Arbitration is taking place, for instance, the BRG will be reloaded when the SCL pin is sampled high (Figure 15-19).

**FIGURE 15-18: BAUD RATE GENERATOR BLOCK DIAGRAM**

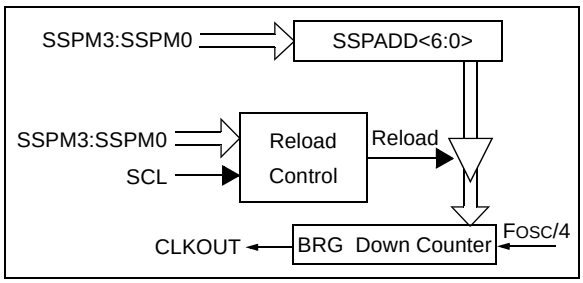
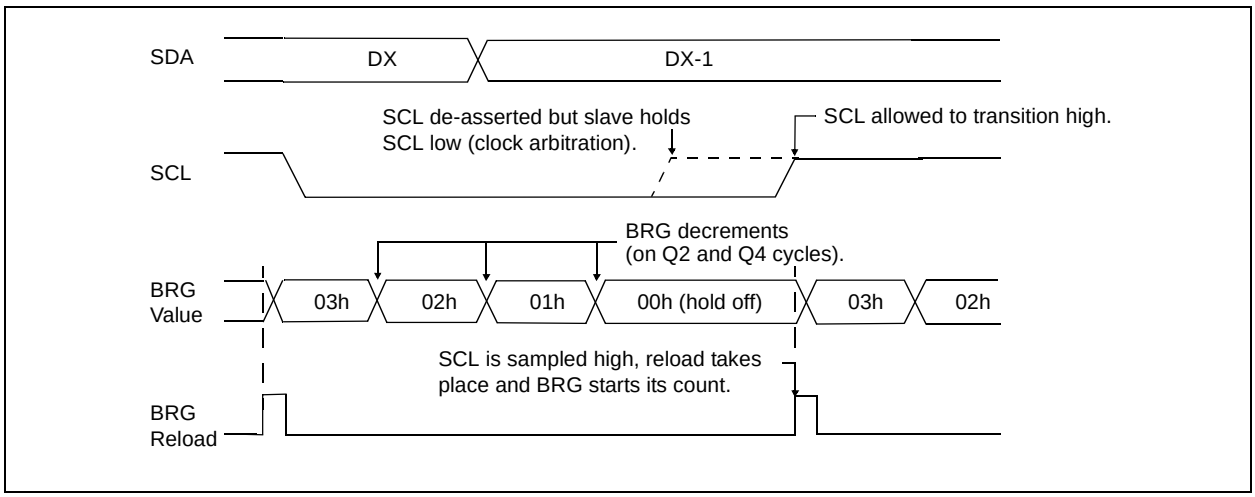



**FIGURE 15-19: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION**



### 15.2.9 I²C MASTER MODE START CONDITION TIMING

To initiate a START condition, the user sets the START condition enable bit, SEN (SSPCON2<0>). If the SDA and SCL pins are sampled high, the baud rate generator is reloaded with the contents of SSPADD<6:0> and starts its count. If SCL and SDA are both sampled high when the baud rate generator times out (TBRG), the SDA pin is driven low. The action of the SDA being driven low while SCL is high is the START condition and causes the S bit (SSPSTAT<3>) to be set. Following this, the baud rate generator is reloaded with the contents of SSPADD<6:0> and resumes its count. When the baud rate generator times out (TBRG), the SEN bit (SSPCON2<0>) will be automatically cleared by hardware, the baud rate generator is suspended, leaving the SDA line held low and the START condition is complete.

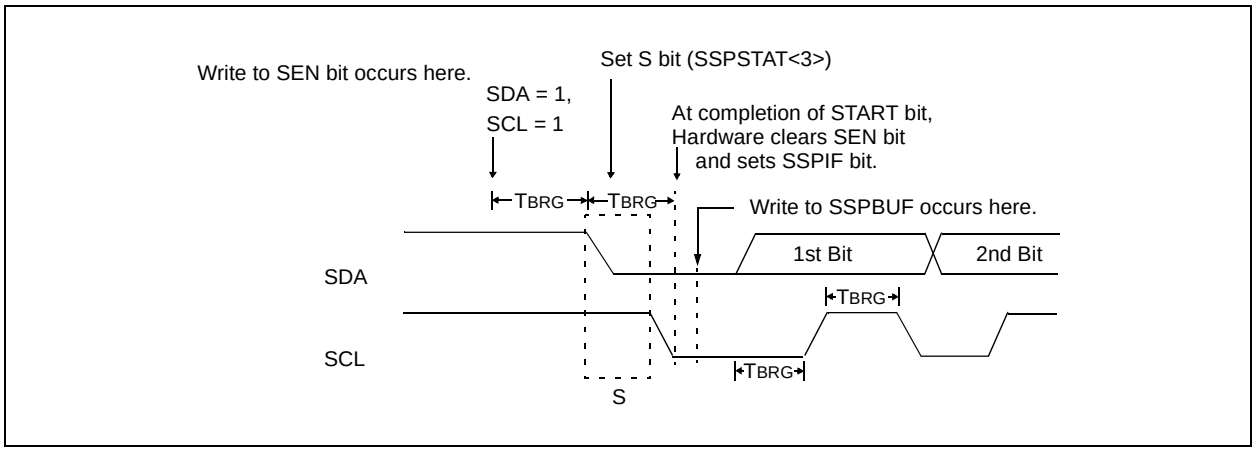
| **Note:** | If at the beginning of START condition, the SDA and SCL pins are already sampled low, or if during the START condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs. The Bus Collision Interrupt Flag (BCLIF) is set, the START condition is aborted and the I²C module is reset into its IDLE state. |
|---|---|

#### 15.2.9.1 WCOL Status Flag

If the user writes the SSPBUF when a START sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

| **Note:** | Because queueing of events is not allowed, writing to the lower 5 bits of SSPCON2 is disabled until the START condition is complete. |
|---|---|

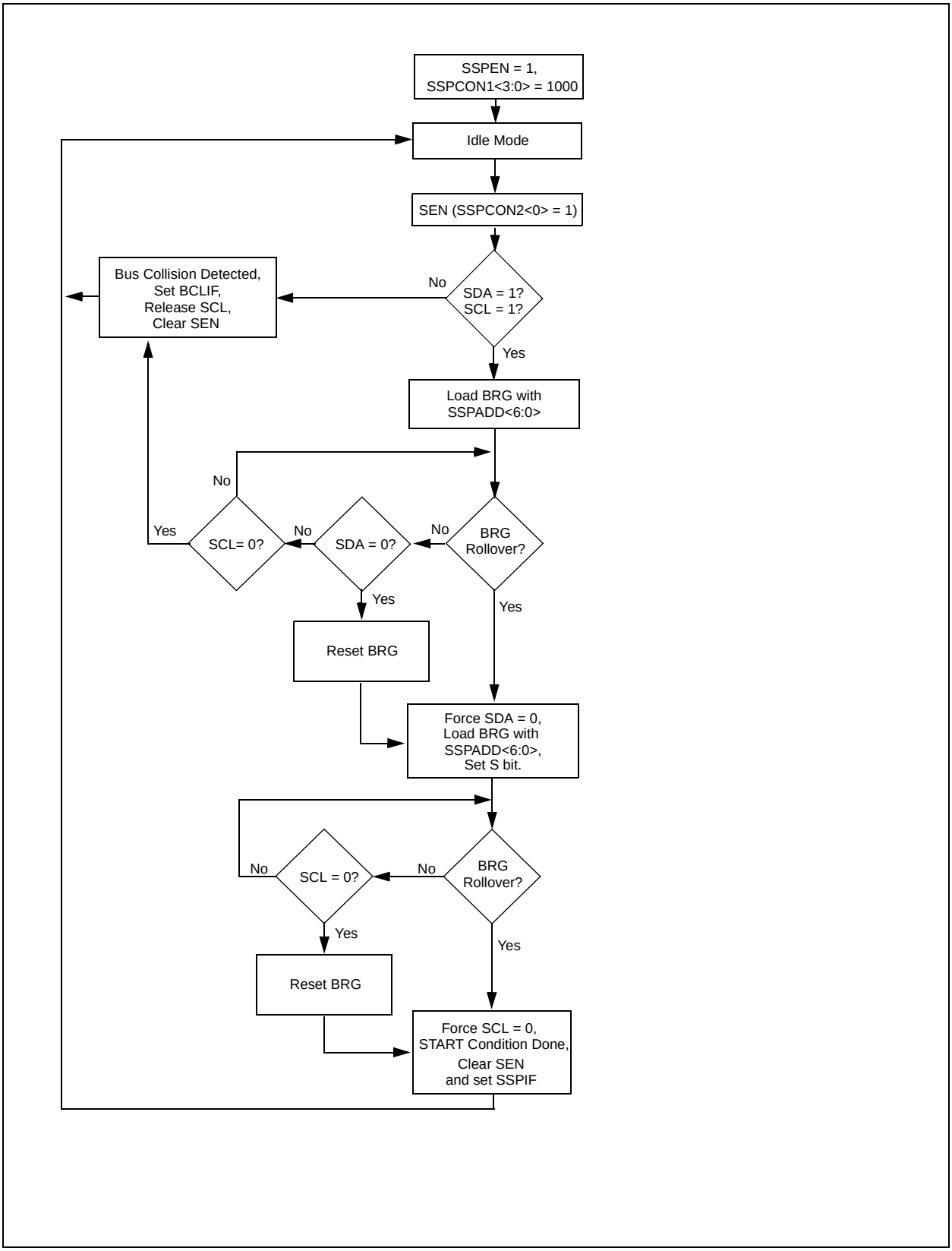
**FIGURE 15-20: FIRST START BIT TIMING**





© 2000 Microchip Technology Inc.

**FIGURE 15-21: START CONDITION FLOW CHART**



### 15.2.10 I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit (SSPCON2<1>) is programmed high and the I²C module is in the idle state. When the RSEN bit is set, the SCL pin is asserted low. When the SCL pin is sampled low, the baud rate generator is loaded with the contents of SSPADD<6:0> and begins counting. The SDA pin is released (brought high) for one baud rate generator count (TBRG). When the baud rate generator times out, if SDA is sampled high, the SCL pin will be de-asserted (brought high). When SCL is sampled high the baud rate generator is reloaded with the contents of SSPADD<6:0> and begins counting. SDA and SCL must be sampled high for one TBRG. This action is then followed by assertion of the SDA pin (SDA is low) for one TBRG while SCL is high. Following this, the RSEN bit in the SSPCON2 register will be automatically cleared and the baud rate generator is not reloaded, leaving the SDA pin held low. As soon as a START condition is detected on the SDA and SCL pins, the S bit (SSPSTAT<3>) will be set. The SSPIF bit will not be set until the baud rate generator has timed out.

**Note 1:** If the RSEN is programmed while any other event is in progress, it will not take effect.

**2:** A bus collision during the Repeated Start condition occurs if:

- SDA is sampled low when SCL goes from low to high.
- SCL goes low before SDA is asserted low. This may indicate that another master is attempting to transmit a data "1".

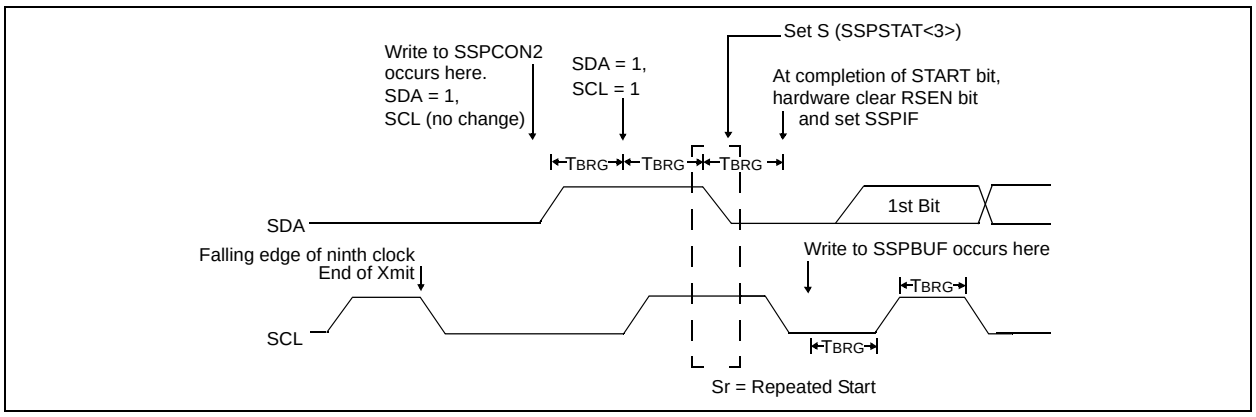
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode), or eight bits of data (7-bit mode).

#### 15.2.10.1 WCOL status flag

If the user writes the SSPBUF when a Repeated Start sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

**Note:** Because queueing of events is not allowed, writing of the lower 5 bits of SSPCON2 is disabled until the Repeated Start condition is complete.

**FIGURE 15-22: REPEAT START CONDITION WAVEFORM**





© 2000 Microchip Technology Inc.

**FIGURE 15-23: REPEATED START CONDITION FLOW CHART (PAGE 1)**

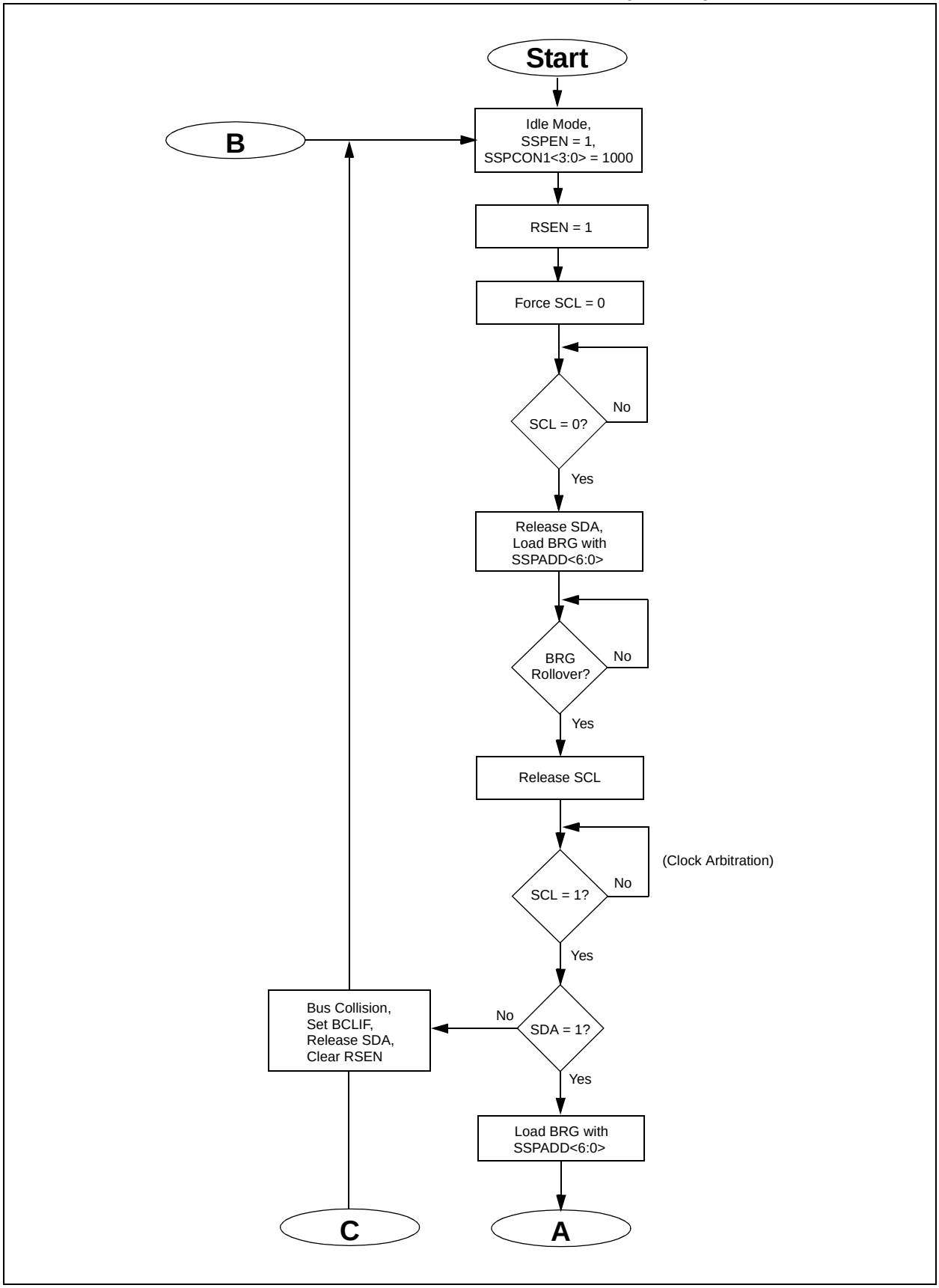
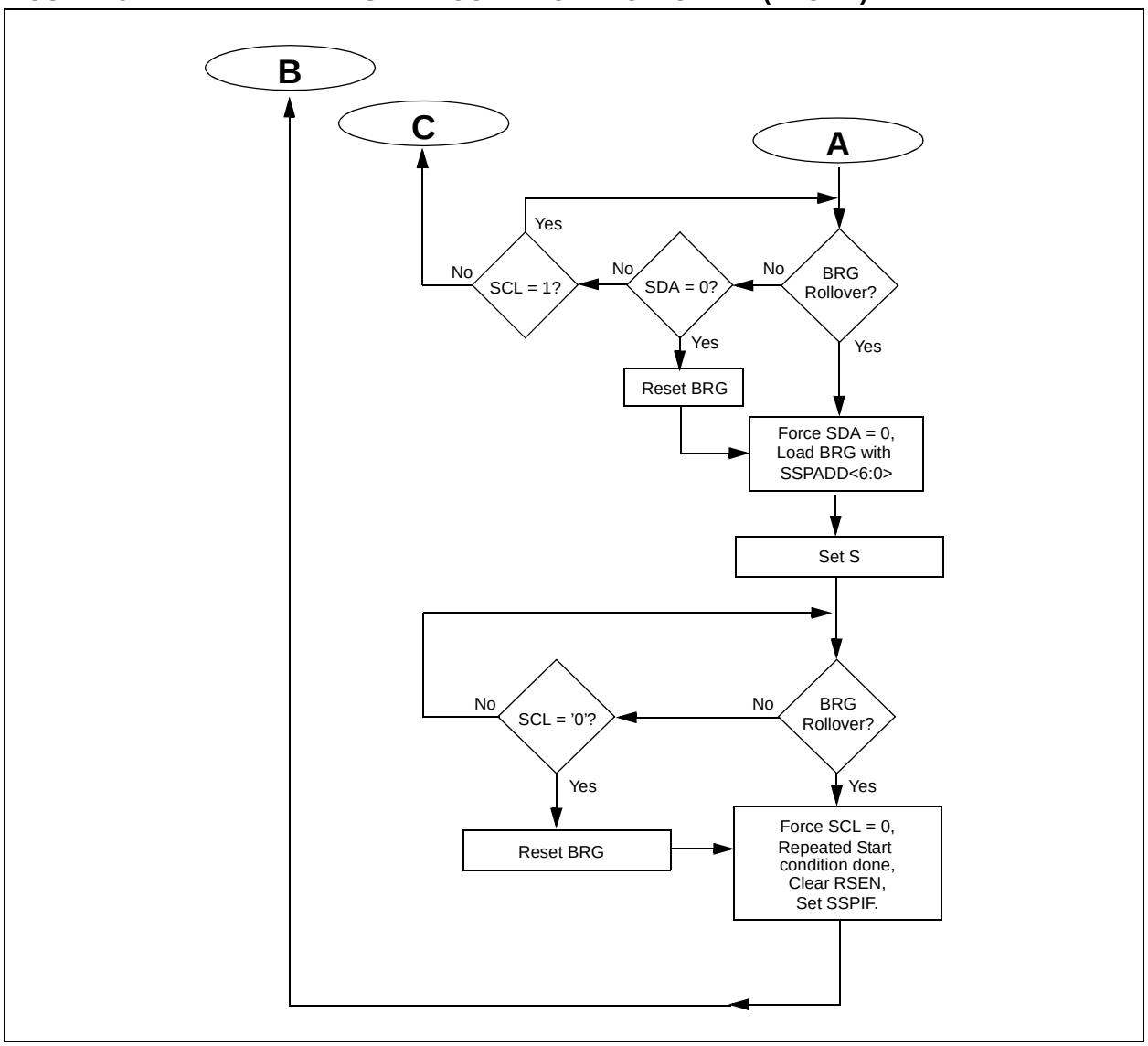


**FIGURE 15-24: REPEATED START CONDITION FLOW CHART (PAGE 2)**





© 2000 Microchip Technology Inc.

### 15.2.11 $I^2C$ MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address, or either half of a 10-bit address, is accomplished by simply writing a value to SSPBUF register. This action will set the buffer full flag (BF) and allow the baud rate generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted (see data hold time spec). SCL is held low for one baud rate generator roll over count (TBRG). Data should be valid before SCL is released high (see Data setup time spec). When the SCL pin is released high, it is held that way for TBRG, the data on the SDA pin must remain stable for that duration and some hold time after the next falling edge of SCL. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDA, allowing the slave device being addressed to respond with an $\overline{ACK}$ bit during the ninth bit time, if an address match occurs or if data was received properly. The status of $\overline{ACK}$ is read into the ACKDT on the falling edge of the ninth clock. If the master receives an acknowledge, the acknowledge status bit (AKSTAT) is cleared. If not, the bit is set. After the ninth clock, the SSPIF is set and the master clock (baud rate generator) is suspended until the next data byte is loaded into the SSPBUF, leaving SCL low and SDA unchanged (Figure 15-26).

After the write to the SSPBUF, each bit of address will be shifted out on the falling edge of SCL until all seven address bits and the R/$\overline{W}$ bit are completed. On the falling edge of the eighth clock, the master will de-assert the SDA pin, allowing the slave to respond with an acknowledge. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the ACK bit is loaded into the ACKSTAT status bit (SSPCON2<6>). Following the falling edge of the ninth clock transmission of the address, the SSPIF is set, the BF flag is cleared and the baud rate generator is turned off until another write to the SSPBUF takes place, holding SCL low and allowing SDA to float.

#### 15.2.11.1 BF Status Flag

In Transmit mode, the BF bit (SSPSTAT<0>) is set when the CPU writes to SSPBUF and is cleared when all 8 bits are shifted out.

#### 15.2.11.2 WCOL Status Flag

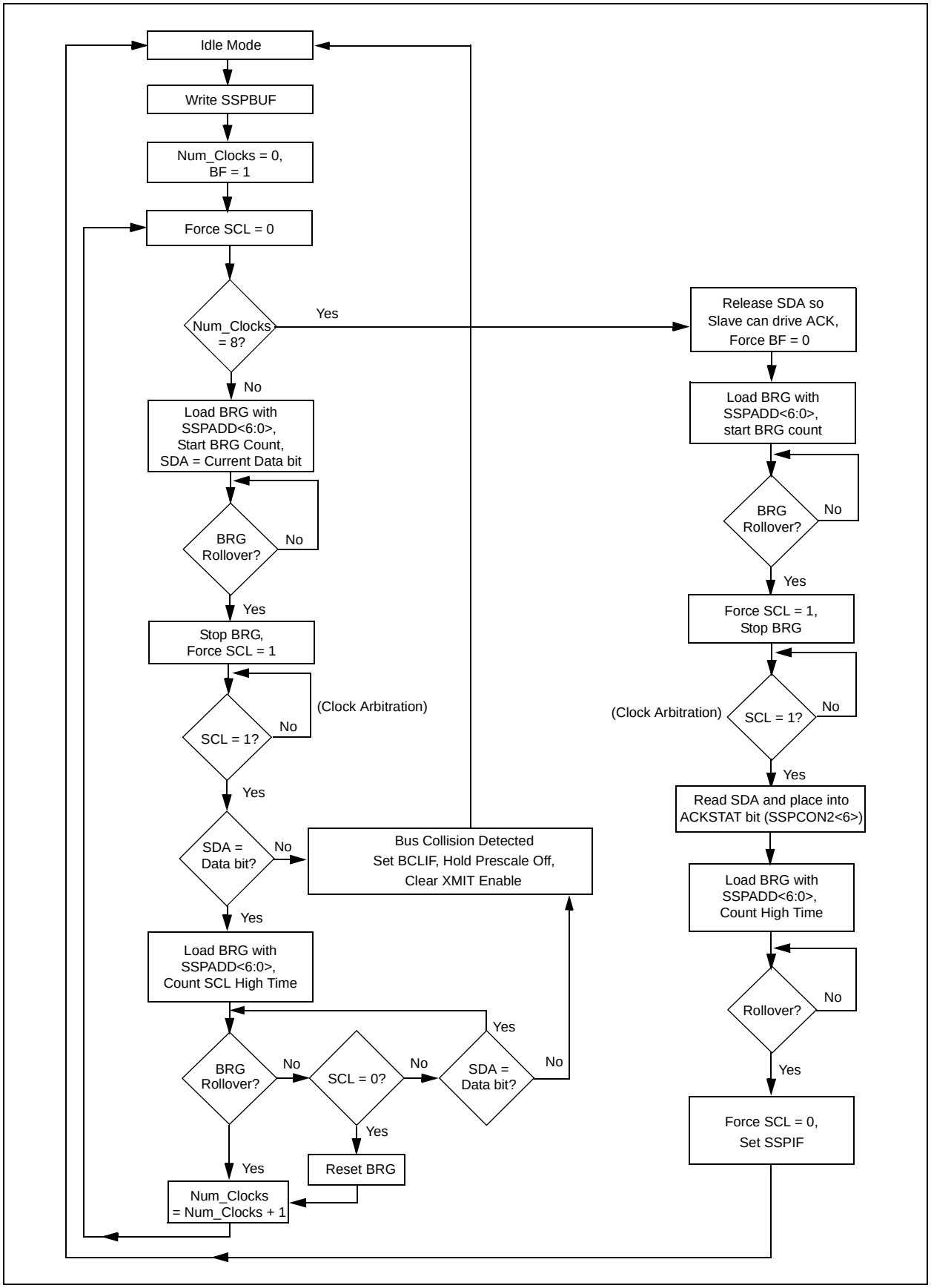
If the user writes the SSPBUF when a transmit is already in progress (i.e., SSPSR is still shifting out a data byte), then WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

WCOL must be cleared in software.

#### 15.2.11.3 AKSTAT Status Flag

In Transmit mode, the AKSTAT bit (SSPCON2<6>) is cleared when the slave has sent an acknowledge ($\overline{ACK}$ = 0) and is set when the slave does not acknowledge ($\overline{ACK}$ = 1). A slave sends an acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

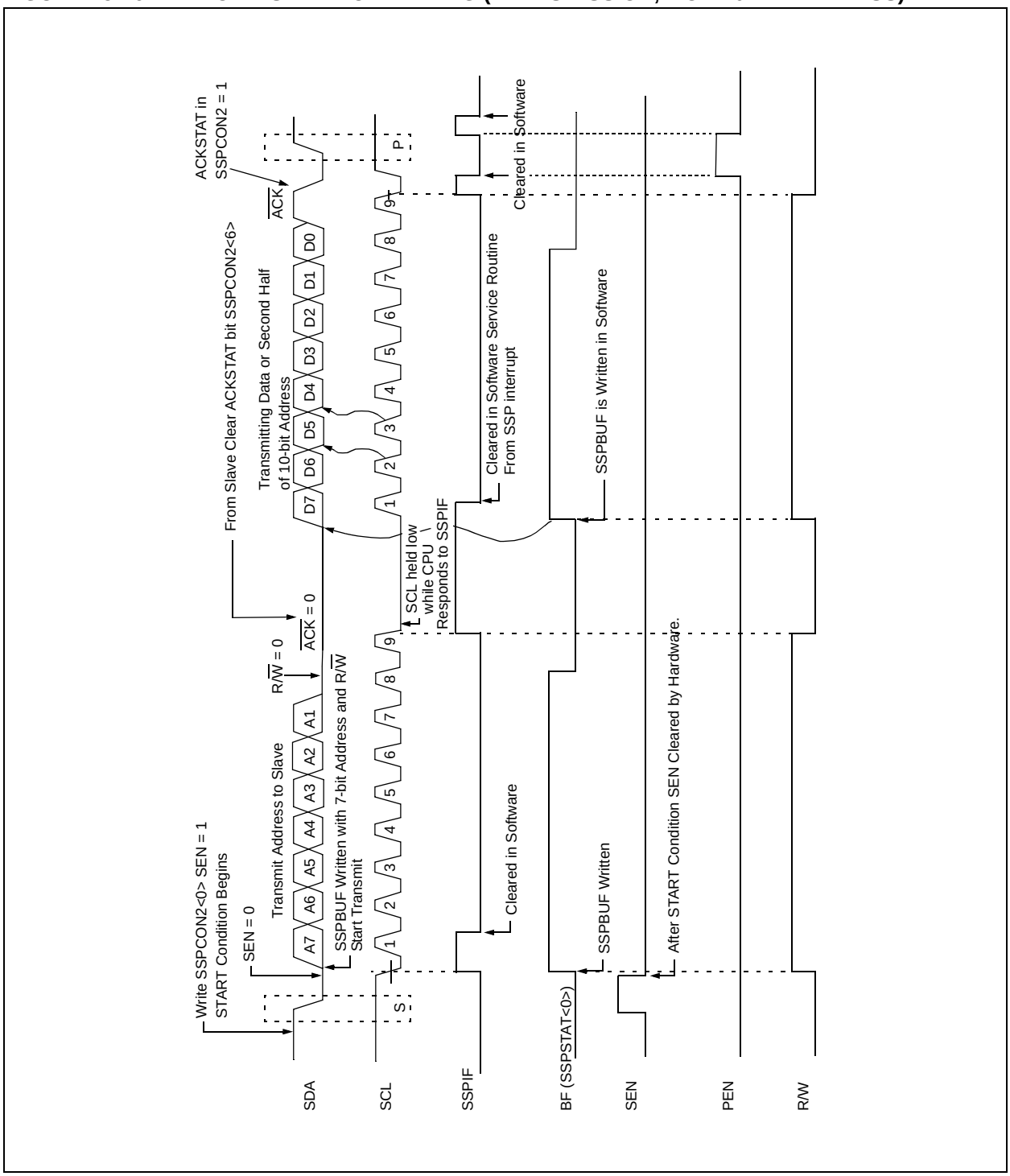
**FIGURE 15-25: MASTER TRANSMIT FLOW CHART**





© 2000 Microchip Technology Inc.

**FIGURE 15-26: I²C MASTER MODE TIMING (TRANSMISSION, 7 OR 10-BIT ADDRESS)**



### 15.2.12 $I^2C$ MASTER MODE RECEPTION

Master mode reception is enabled by programming the receive enable bit, RCEN (SSPCON2<3>).

| | |
|---|---|
| **Note:** | The SSP Module must be in an IDLE STATE before the RCEN bit is set, or the RCEN bit will be disregarded. |

The baud rate generator begins counting and on each rollover, the state of the SCL pin changes (high to low/ low to high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF flag is set, the SSPIF is set and the baud rate generator is suspended from counting, holding SCL low. The SSP is now in IDLE state, awaiting the next command. When the buffer is read by the CPU, the BF flag is automatically cleared. The user can then send an acknowledge bit at the end of reception, by setting the acknowledge sequence enable bit, ACKEN (SSPCON2<4>).

#### 15.2.12.1 BF Status Flag

In receive operation, BF is set when an address or data byte is loaded into SSPBUF from SSPSR. It is cleared when SSPBUF is read.

#### 15.2.12.2 SSPOV Status Flag

In receive operation, SSPOV is set when 8 bits are received into the SSPSR, and the BF flag is already set from a previous reception.

#### 15.2.12.3 WCOL Status Flag

If the user writes the SSPBUF when a receive is already in progress (i.e., SSPSR is still shifting in a data byte), then WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).


© 2000 Microchip Technology Inc.

**FIGURE 15-27: MASTER RECEIVER FLOW CHART**

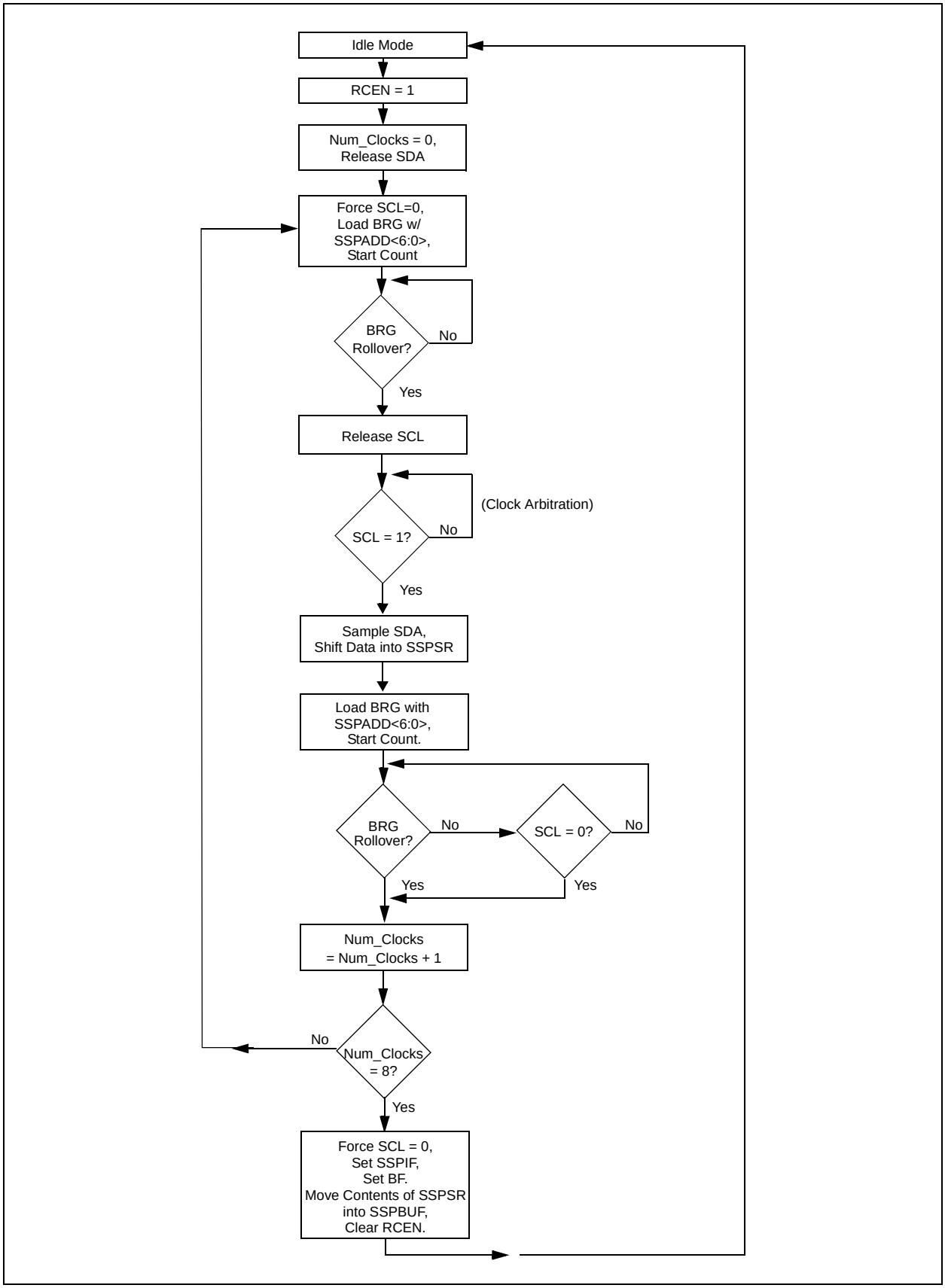
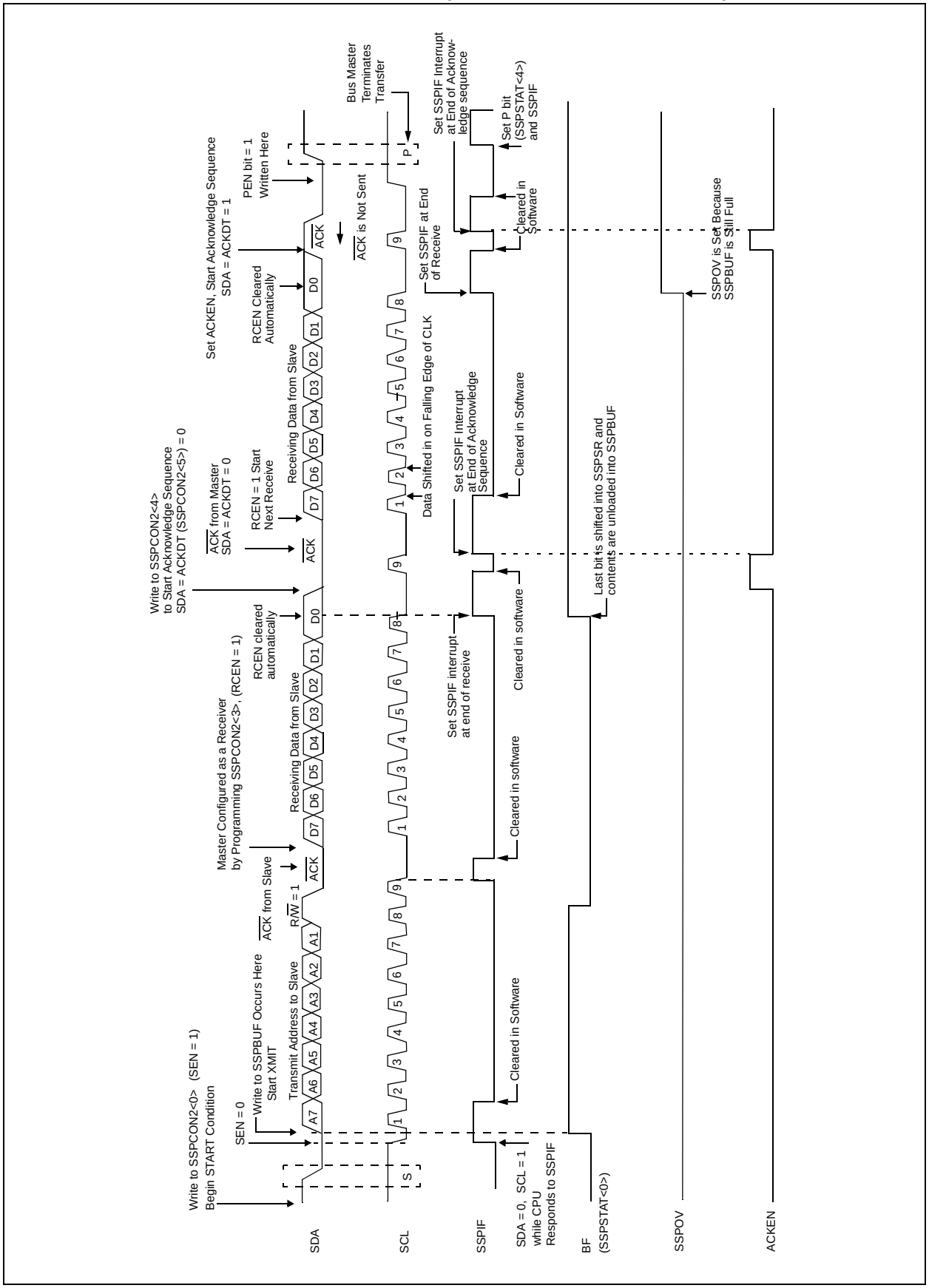


**FIGURE 15-28: I²C MASTER MODE TIMING (RECEPTION 7-BIT ADDRESS)**





© 2000 Microchip Technology Inc.

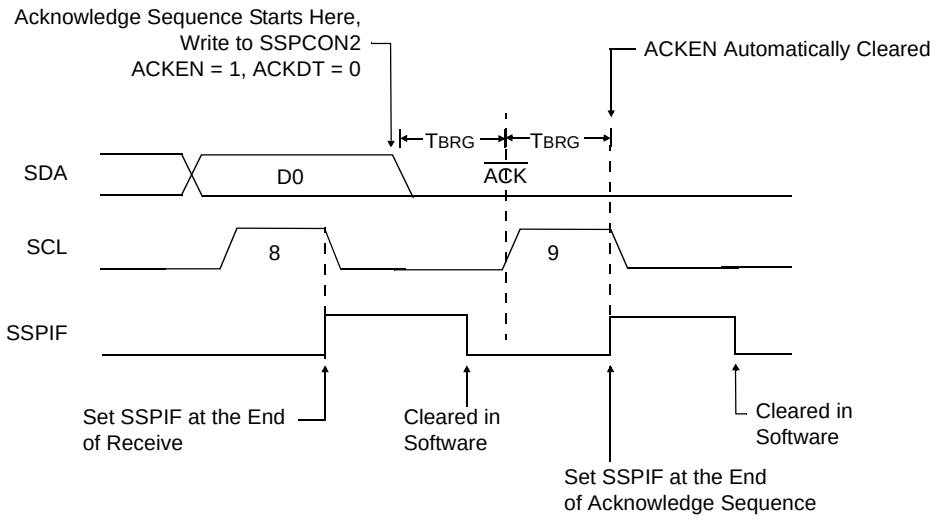
### 15.2.13 ACKNOWLEDGE SEQUENCE TIMING

An acknowledge sequence is enabled by setting the acknowledge sequence enable bit, ACKEN (SSPCON2<4>). When this bit is set, the SCL pin is pulled low and the contents of the acknowledge data bit is presented on the SDA pin. If the user wishes to generate an acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an acknowledge sequence. The baud rate generator then counts for one rollover period (TBRG), and the SCL pin is de-asserted (pulled high). When the SCL pin is sampled high (clock arbitration), the baud rate generator counts for TBRG. The SCL pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the baud rate generator is turned off and the SSP module then goes into IDLE mode (Figure 15-29).

#### 15.2.13.1 WCOL Status Flag

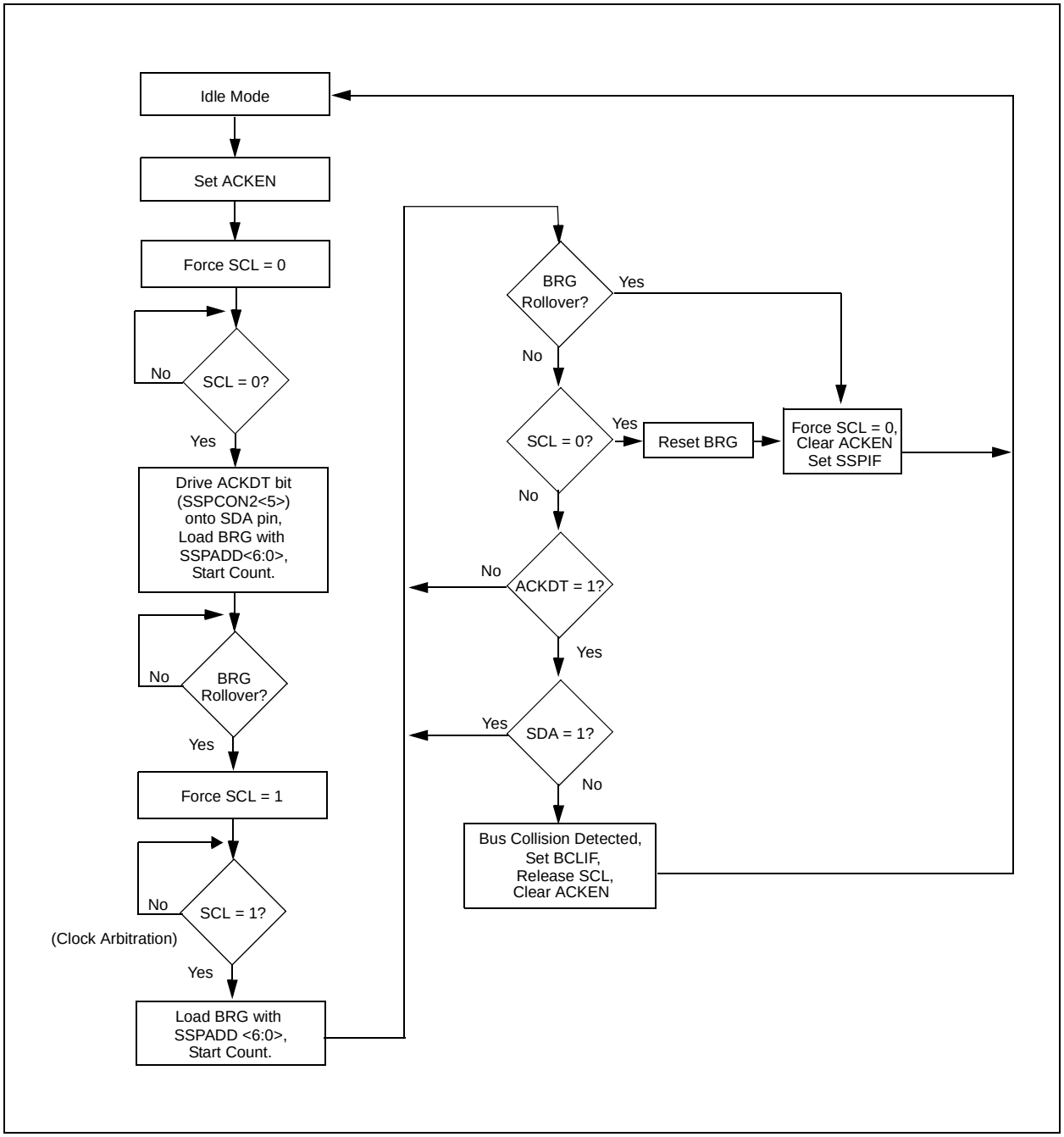
If the user writes the SSPBUF when an acknowledge sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

**FIGURE 15-29: ACKNOWLEDGE SEQUENCE WAVEFORM**




**Note:** TBRG = one baud rate generator period.

**FIGURE 15-30: ACKNOWLEDGE FLOW CHART**





© 2000 Microchip Technology Inc.

### 15.2.14 STOP CONDITION TIMING

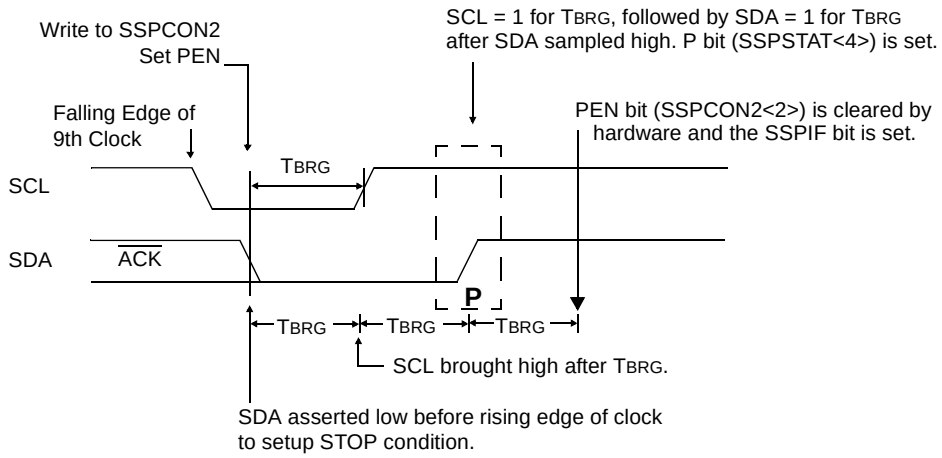
A STOP bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Sequence Enable bit PEN (SSPCON2<2>). At the end of a receive/transmit the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the baud rate generator is reloaded and counts down to '0'. When the baud rate generator times out, the SCL pin will be brought high and one TBRG (baud rate generator rollover count) later, the SDA pin will be de-asserted. When the SDA pin is sampled high while SCL is high, the P bit (SSPSTAT<4>) is set. A TBRG later, the PEN bit is cleared and the SSPIF bit is set (Figure 15-31).

Whenever the firmware decides to take control of the bus, it will first determine if the bus is busy by checking the S and P bits in the SSPSTAT register. If the bus is busy, then the CPU can be interrupted (notified) when a STOP bit is detected (i.e., bus is free).

#### 15.2.14.1 WCOL Status Flag

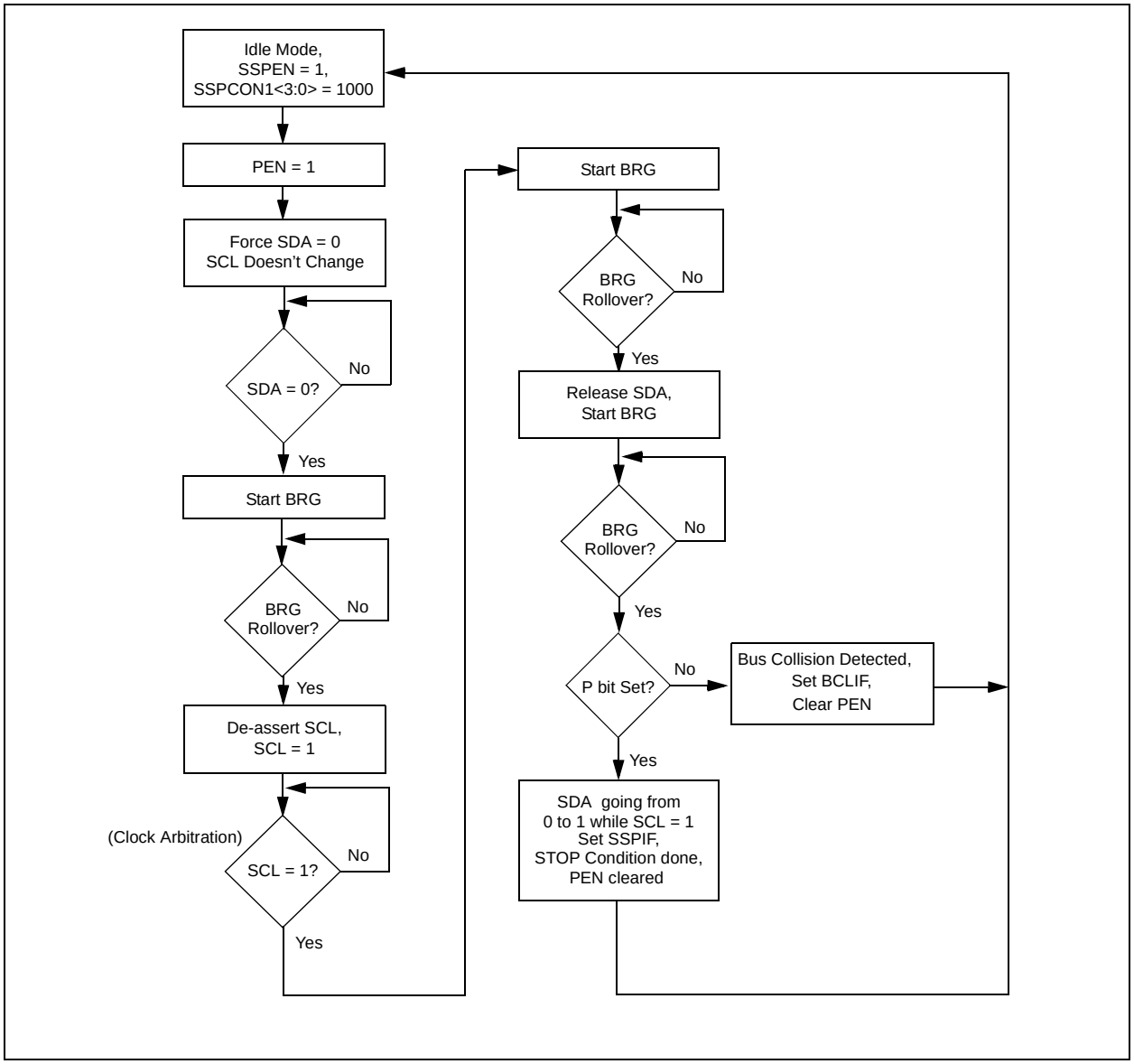
If the user writes the SSPBUF when a STOP sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

**FIGURE 15-31: STOP CONDITION RECEIVE OR TRANSMIT MODE**




**Note:** TBRG = one baud rate generator period.

**FIGURE 15-32: STOP CONDITION FLOW CHART**





© 2000 Microchip Technology Inc.

### 15.2.15 CLOCK ARBITRATION

Clock arbitration occurs when the master, during any receive, transmit, or Repeated Start/Stop condition, deasserts the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the baud rate generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the SCL pin is sampled high, the baud rate generator is reloaded with the contents of SSPADD<6:0> and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count, in the event that the clock is held low by an external device (Figure 15-33).

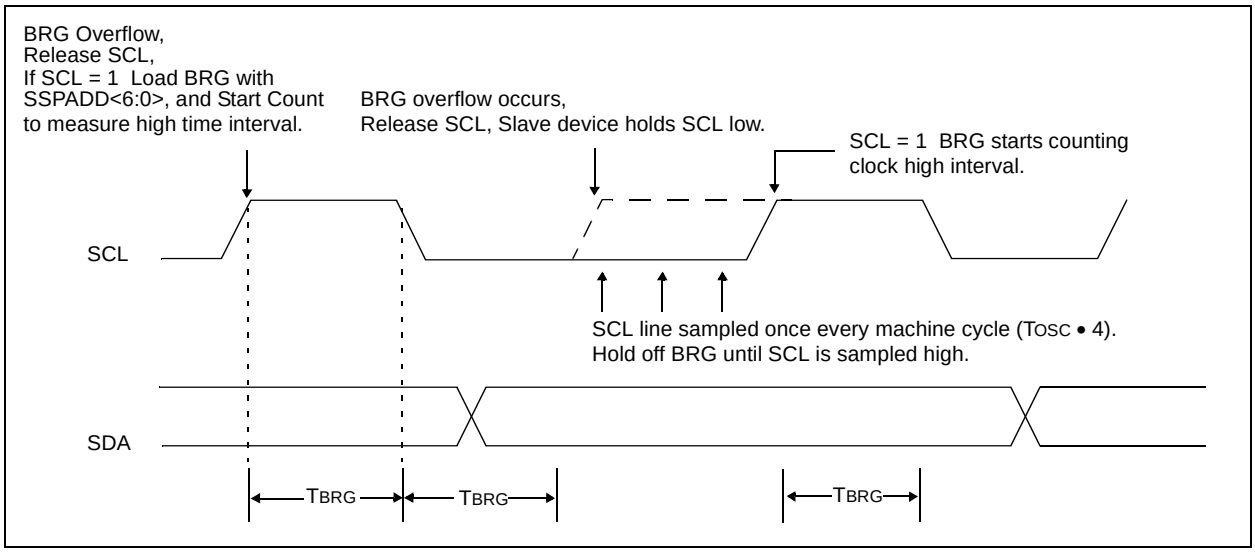
### 15.2.16 SLEEP OPERATION

While in SLEEP mode, the $I^2C$ module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from SLEEP (if the SSP interrupt is enabled).

### 15.2.17 EFFECTS OF A RESET

A RESET disables the SSP module and terminates the current transfer.

**FIGURE 15-33: CLOCK ARBITRATION TIMING IN MASTER TRANSMIT MODE**



### 15.2.18 MULTI -MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA, by letting SDA float high and another master asserts a '0'. When the SCL pin floats high, data should be stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin = '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLIF and reset the $I^2C$ port to its IDLE state (Figure 15-34).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDA and SCL lines are de-asserted and the SSPBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the $I^2C$ bus is free, the user can resume communication by asserting a START condition.

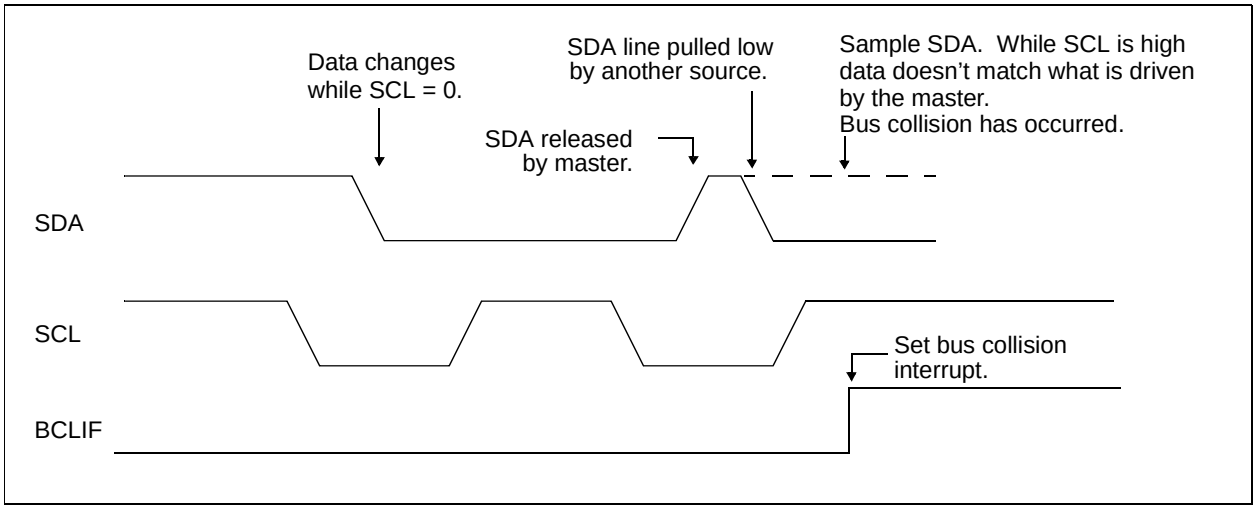
If a START, Repeated Start, STOP, or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are de-asserted and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine, and if the $I^2C$ bus is free, the user can resume communication by asserting a START condition.

The master will continue to monitor the SDA and SCL pins and if a STOP condition occurs, the SSPIF bit will be set.

A write to the SSPBUF will start the transmission of data at the first data bit, regardless of where the transmitter left off when bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of START and STOP conditions allows the determination of when the bus is free. Control of the $I^2C$ bus can be taken when the P bit is set in the SSPSTAT register, or the bus is idle and the S and P bits are cleared.

**FIGURE 15-34: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE**





© 2000 Microchip Technology Inc.

#### 15.2.18.1 Bus Collision During a START Condition

During a START condition, a bus collision occurs if:

a) SDA or SCL are sampled low at the beginning of the START condition (Figure 15-35).

b) SCL is sampled low before SDA is asserted low (Figure 15-36).

During a START condition, both the SDA and the SCL pins are monitored.

If:

the SDA pin is already low
or the SCL pin is already low,

then:

the START condition is aborted,
and the BCLIF flag is set,
and the SSP module is reset to its IDLE state (Figure 15-35).

The START condition begins with the SDA and SCL pins de-asserted. When the SDA pin is sampled high, the baud rate generator is loaded from SSPADD<6:0> and counts down to '0'. If the SCL pin is sampled low while SDA is high, a bus collision occurs, because it is assumed that another master is attempting to drive a data '1' during the START condition.

If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early (Figure 15-37). If, however, a '1' is sampled on the SDA pin, the SDA pin is asserted low at the end of the BRG count. The baud rate generator is then reloaded and counts down to 0 and during this time, if the SCL pin is sampled as '0', a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

**Note:** The reason that bus collision is not a factor during a START condition is that no two bus masters can assert a START condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the START condition and if the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start, or Stop conditions.

**FIGURE 15-35: BUS COLLISION DURING START CONDITION (SDA ONLY)**

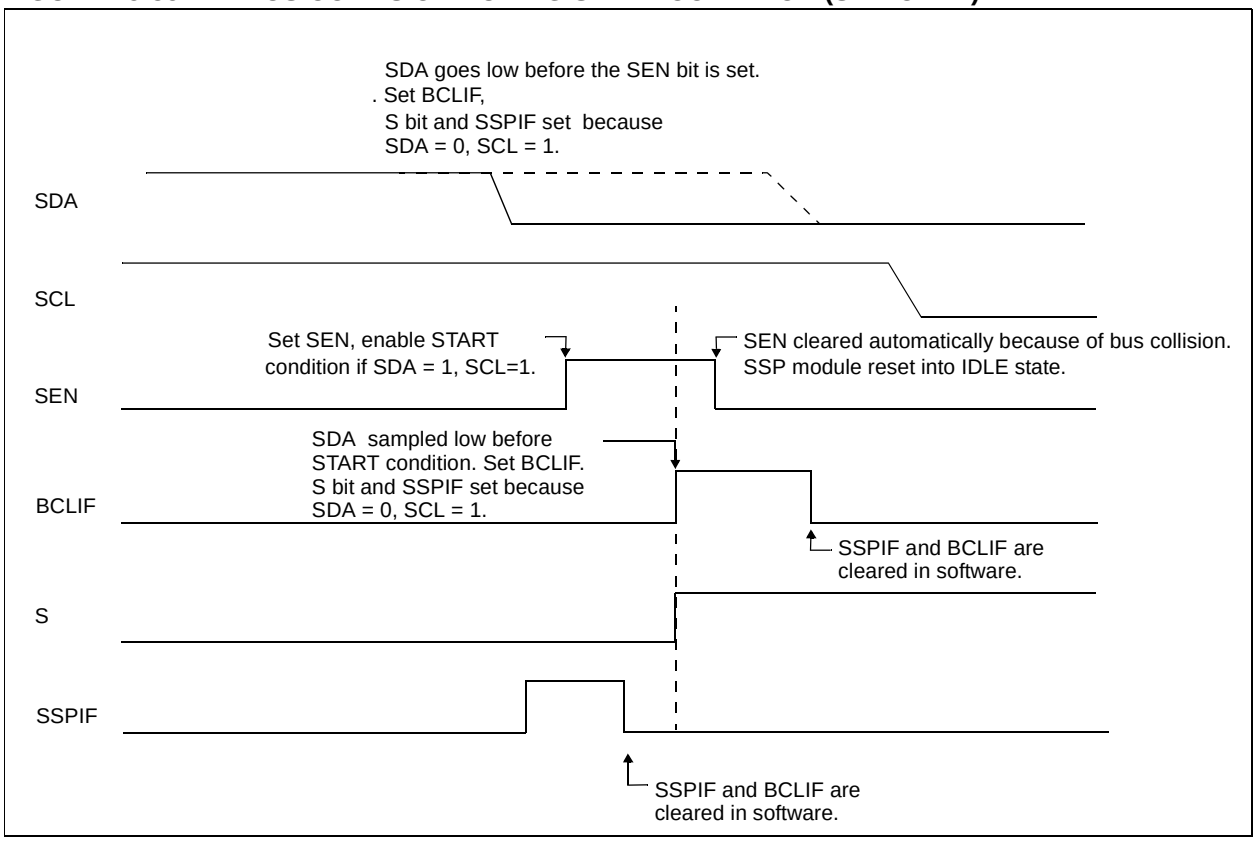


**FIGURE 15-36: BUS COLLISION DURING START CONDITION (SCL = 0)**

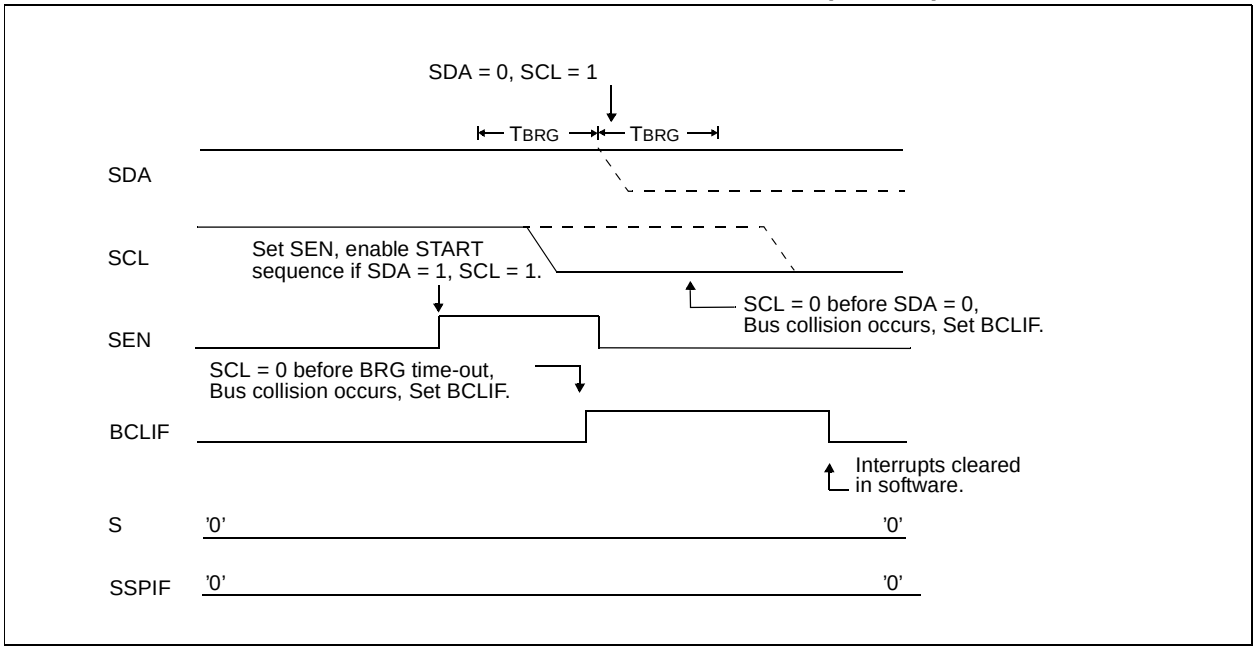
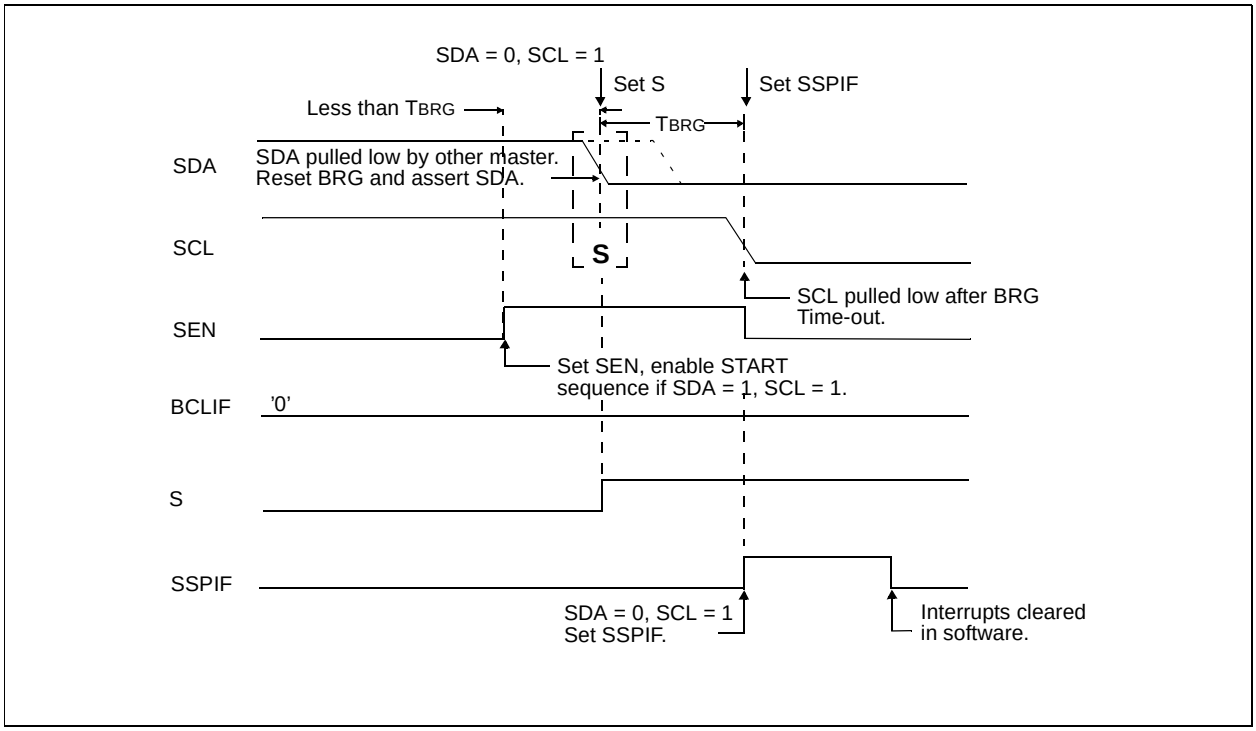



**FIGURE 15-37: BRG RESET DUE TO SDA COLLISION DURING START CONDITION**





© 2000 Microchip Technology Inc.

#### 15.2.18.2 Bus Collision During a Repeated Start Condition

During a Repeated Start condition, a bus collision occurs if:

a) A low level is sampled on SDA when SCL goes from low level to high level.

b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user de-asserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD<6:0> and counts down to '0'. The SCL pin is then de-asserted and when sampled high, the SDA pin is sampled. If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0'). If, however, SDA is sampled high, then the BRG is reloaded and begins counting. If SDA goes from high to low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

If, however, SCL goes from high to low before the BRG times out and SDA has not already been asserted, then a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition.

If, at the end of the BRG time-out, both SCL and SDA are still high, the SDA pin is driven low, the BRG is reloaded and begins counting. At the end of the count, regardless of the status of the SCL pin, the SCL pin is driven low and the Repeated Start condition is complete (Figure 15-38).

**FIGURE 15-38: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)**

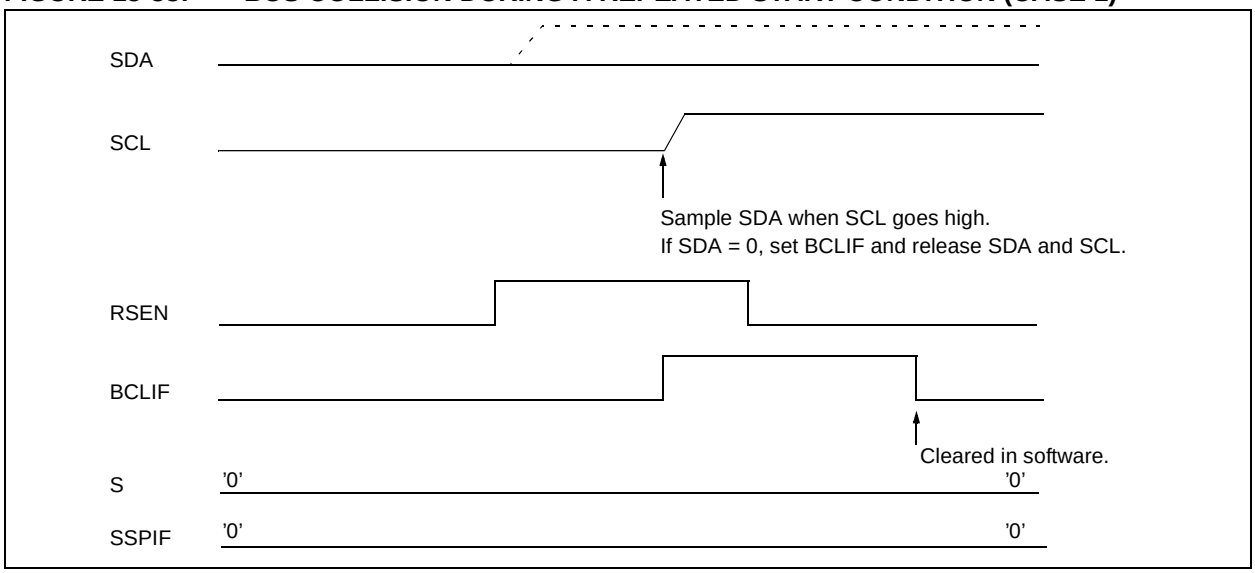
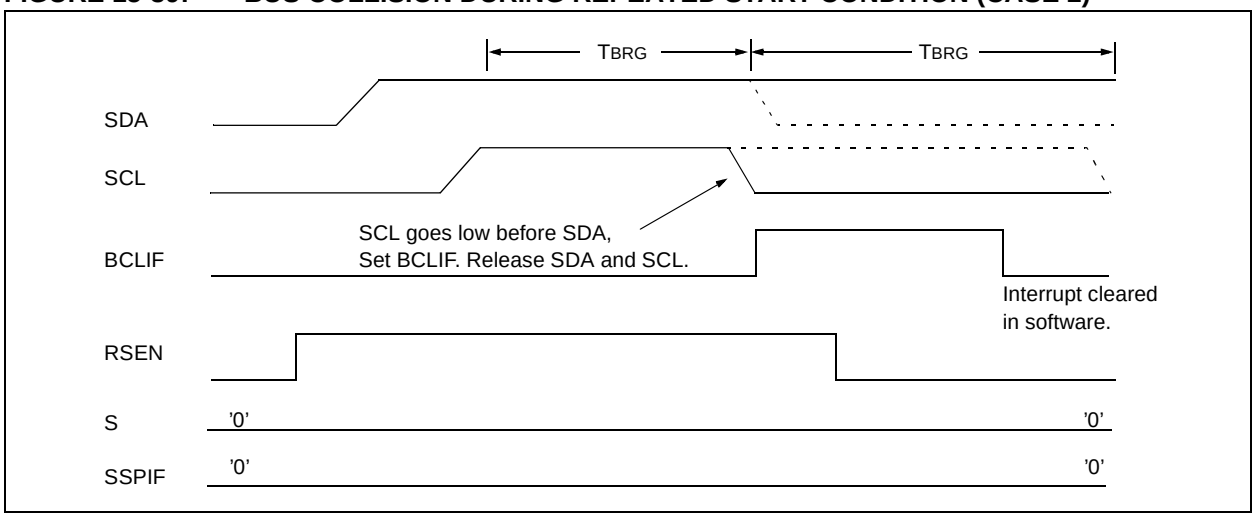



**FIGURE 15-39: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)**



#### 15.2.18.3 Bus Collision During a STOP Condition

Bus collision occurs during a STOP condition if:

a) After the SDA pin has been de-asserted and allowed to float high, SDA is sampled low after the BRG has timed out.

b) After the SCL pin is de-asserted, SCL is sampled low before SDA goes high.

The STOP condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the baud rate generator is loaded with SSPADD<6:0> and counts down to '0'. After the BRG times out, SDA is sampled. If SDA is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0'. If the SCL pin is sampled low before SDA is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (Figure 15-40).

**FIGURE 15-40: BUS COLLISION DURING A STOP CONDITION (CASE 1)**

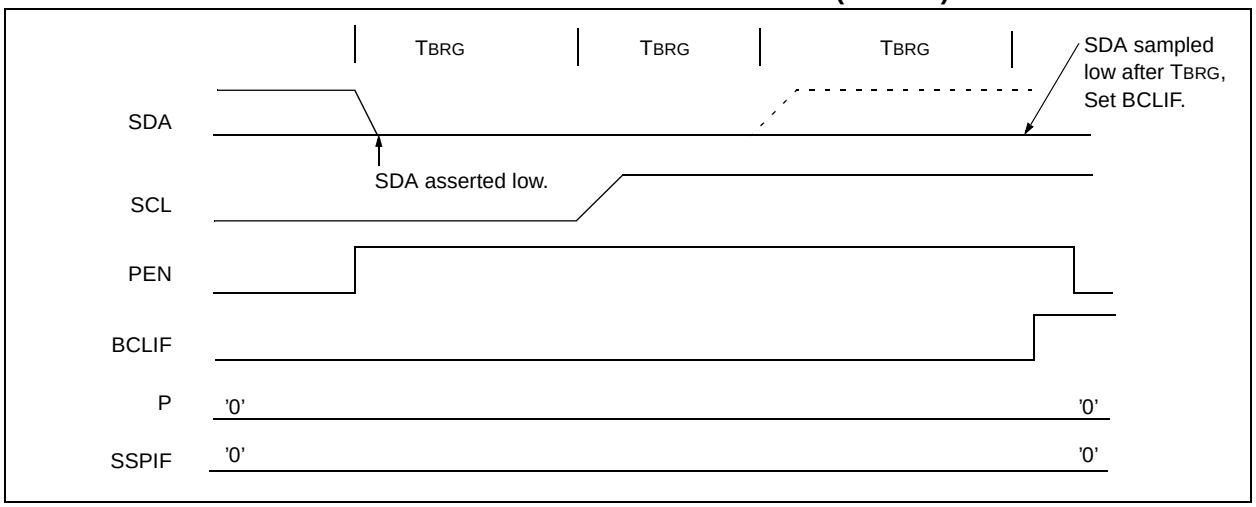
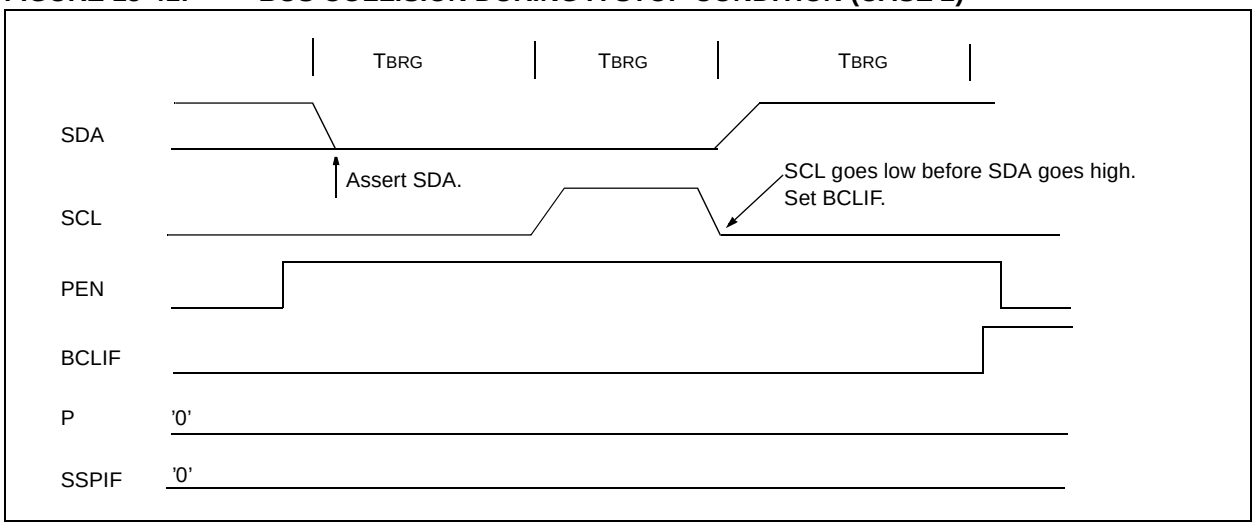



**FIGURE 15-41: BUS COLLISION DURING A STOP CONDITION (CASE 2)**





© 2000 Microchip Technology Inc.

## 15.3 Connection Considerations for I²C Bus

For standard mode I²C bus devices, the values of resistors $R_p$ $R_s$ in Figure 15-42 depends on the following parameters:

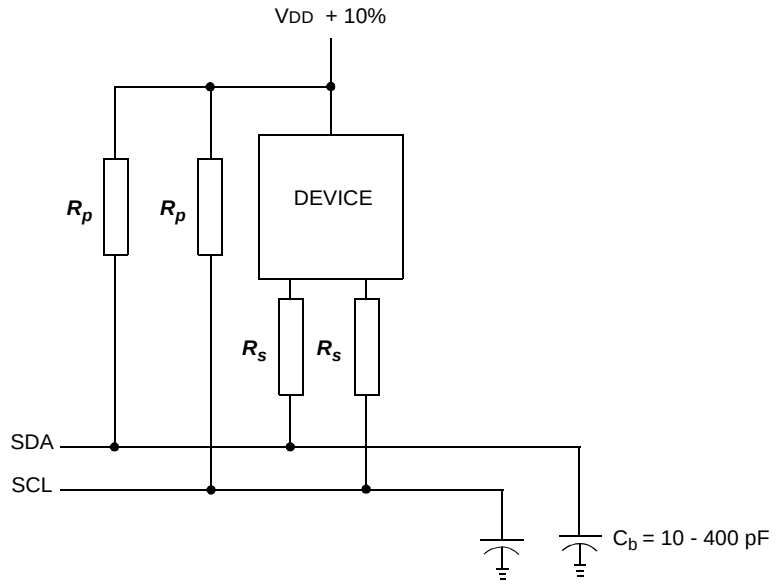
- Supply voltage
- Bus capacitance
- Number of connected devices (input current + leakage current)

The supply voltage limits the minimum value of resistor $R_p$ due to the specified minimum sink current of 3 mA at VOL max = 0.4V for the specified output stages. For example, with a supply voltage of VDD = 5V ±10% and VOL max = 0.4V at 3 mA, $R_p$ min = (5.5-0.4)/0.003 = 1.7 kΩ. VDD as a function of $R_p$ is shown in Figure 15-42. The desired noise margin of 0.1 VDD for the low level, limits the maximum value of $R_s$. Series resistors are optional and used to improve ESD susceptibility.

The bus capacitance is the total capacitance of wire, connections and pins. This capacitance limits the maximum value of $R_p$ due to the specified rise time (Figure 15-42).

The SMP bit is the slew rate control enabled bit. This bit is in the SSPSTAT register and controls the slew rate of the I/O pins when in I²C mode (master or slave).

**FIGURE 15-42: SAMPLE DEVICE CONFIGURATION FOR I²C BUS**




**Note:** I²C devices with input levels related to VDD must have one common supply line to which the pull-up resistor is also connected.

## 15.4 Example Program

Example 15-2 shows MPLAB® C17 'C' code for using the $I^2C$ module in Master mode to communicate with a 24LC01B serial EEPROM. This example uses the PICmicro® 'C' libraries included with MPLAB C17.

**EXAMPLE 15-2: INTERFACING TO A 24LC01B SERIAL EEPROM (USING MPLAB C17)**

```
// Include necessary header files
#include <p17c756.h>     // Processor header file
#include <delays.h>      // Delay routines header file
#include <stdlib.h>      // Standard Library header file
#include <i2c16.h>       // I2C routines header file

#define CONTROL 0xa0     // Control byte definition for 24LC01B

// Function declarations
void main(void);
void WritePORTD(static unsigned char data);
void ByteWrite(static unsigned char address,static unsigned char data);
unsigned char ByteRead(static unsigned char address);
void ACKPoll(void);

// Main program
void main(void)
{
static unsigned char address;    // I2C address of 24LC01B
static unsigned char datao;      // Data written to 24LC01B
static unsigned char datai;      // Data read from 24LC01B

    address = 0;                 // Preset address to 0
    OpenI2C(MASTER,SLEW_ON);     // Configure I2C Module Master mode, Slew rate control on
    SSPADD = 39;                 // Configure clock for 100KHz

    while(address<128)           // Loop 128 times, 24LC01B is 128x8
    {
        datao = PORTB;
        do
        {
            ByteWrite(address,datao);   // Write data to EEPROM
            ACKPoll();                  // Poll the 24LC01B for state
            datai = ByteRead(address);  // Read data from EEPROM into SSPBUF
        } while(datai != datao);        // Loop as long as data not correctly
                                        //    written to 24LC01B

        address++;                      // Increment address
    }
    while(1)                            // Done writing 128 bytes to 24LC01B, Loop forever
    {
        Nop();
    }
}
```


© 2000 Microchip Technology Inc.

**EXAMPLE 15-2: INTERFACING TO A 24LC01B SERIAL EEPROM (USING MPLAB C17)**

```
// Writes the byte data to 24LC01B at the specified address
void ByteWrite(static unsigned char address, static unsigned char data)
{
    StartI2C();                      // Send start bit
    IdleI2C();                       // Wait for idle condition
    WriteI2C(CONTROL);               // Send control byte
    IdleI2C();                       // Wait for idle condition
    if (!SSPCON2bits.ACKSTAT)        // If 24LC01B ACKs
    {
        WriteI2C(address);           // Send control byte
        IdleI2C();                   // Wait for idle condition

        if (!SSPCON2bits.ACKSTAT)    // If 24LC01B ACKs
            WriteI2C(data);          // Send data
    }
    IdleI2C();                       // Wait for idle condition
    StopI2C();                       // Send stop bit
    IdleI2C();                       // Wait for idle condition
    return;
}


// Reads a byte of data from 24LC01B at the specified address
unsigned char ByteRead(static unsigned char address)
{
    StartI2C();                      // Send start bit
    IdleI2C();                       // Wait for idle condition
    WriteI2C(CONTROL);               // Send control byte
    IdleI2C();                       // Wait for idle condition
    if (!SSPCON2bits.ACKSTAT)        // If the 24LC01B ACKs
    {
        WriteI2C(address);           // Send address
        IdleI2C();                   // Wait for idle condition
        if (!SSPCON2bits.ACKSTAT)    // If the 24LC01B ACKs
        {
            RestartI2C();            // Send restart
            IdleI2C();               // Wait for idle condition
            WriteI2C(CONTROL+1);     // Send control byte with R/W set
            IdleI2C();               // Wait for idle condition
            if (!SSPCON2bits.ACKSTAT)    // If the 24LC01B ACKs
            {
                getcI2C();               // Read a byte of data from 24LC01B
                IdleI2C();               // Wait for idle condition
                NotAckI2C();             // Send a NACK to 24LC01B
                IdleI2C();               // Wait for idle condition
                StopI2C();               // Send stop bit
                IdleI2C();               // Wait for idle condition
             }
        }
    }
    return(SSPBUF);
}
```

**EXAMPLE 15-2: INTERFACING TO A 24LC01B SERIAL EEPROM (USING MPLAB C17)**

```
void ACKPoll(void)
{
        StartI2C();                     // Send start bit
        IdleI2C();                      // Wait for idle condition
        WriteI2C(CONTROL);              // Send control byte
        IdleI2C();                      // Wait for idle condition
        // Poll the ACK bit coming from the 24LC01B
        // Loop as long as the 24LC01B NACKs
        while (SSPCON2bits.ACKSTAT)
        {
                RestartI2C();           // Send a restart bit
                IdleI2C();              // Wait for idle condition
                WriteI2C(CONTROL);      // Send control byte
                IdleI2C();              // Wait for idle condition
        }
        IdleI2C();                      // Wait for idle condition
        StopI2C();                      // Send stop bit
        IdleI2C();                      // Wait for idle condition
        return;
}
```


© 2000 Microchip Technology Inc.

## 16.0 ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The analog-to-digital (A/D) converter module has twelve analog inputs for the PIC17C75X devices and sixteen for the PIC17C76X devices.

The analog input charges a sample and hold capacitor. The output of the sample and hold capacitor is the input into the converter. The converter then generates a digital result of this analog level via successive approximation. This A/D conversion of the analog input signal, results in a corresponding 10-bit digital number.

The analog reference voltages (positive and negative supply) are software selectable to either the device's supply voltages ($AV_{DD}$, AVss), or the voltage level on the RG3/AN0/$V_{REF}$+ and RG2/AN1/$V_{REF}$- pins.

The A/D converter has a unique feature of being able to operate while the device is in SLEEP mode. To operate in SLEEP, the A/D clock must be derived from the A/D's internal RC oscillator.

The A/D module has four registers. These registers are:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register0 (ADCON0)
- A/D Control Register1 (ADCON1)

The ADCON0 register, shown in Register 16-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 16-2, configures the functions of the port pins. The port pins can be configured as analog inputs (RG3 and RG2 can also be the voltage references), or as digital I/O.

**REGISTER 16-1: ADCON0 REGISTER (ADDRESS: 14h, BANK 5)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | U-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CHS3 | CHS2 | CHS1 | CHS0 | — | GO/$\overline{\text{DONE}}$ | — | $\overline{\text{ADON}}$ |
| bit 7 | | | | | | | bit 0 |

bit 7-4 **CHS3:CHS0**: Analog Channel Select bits
0000 = channel 0, (AN0)
0001 = channel 1, (AN1)
0010 = channel 2, (AN2)
0011 = channel 3, (AN3)
0100 = channel 4, (AN4)
0101 = channel 5, (AN5)
0110 = channel 6, (AN6)
0111 = channel 7, (AN7)
1000 = channel 8, (AN8)
1001 = channel 9, (AN9)
1010 = channel 10, (AN10)
1011 = channel 11, (AN11)
1100 = channel 12, (AN12) (PIC17C76X only)
1101 = channel 13, (AN13) (PIC17C76X only)
1110 = channel 14, (AN14) (PIC17C76X only)
1111 = channel 15, (AN15) (PIC17C76X only)
11xx = **RESERVED**, do not select (PIC17C75X only)

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

bit 2 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

If ADON = 1:
1 = A/D conversion in progress (setting this bit starts the A/D conversion, which is automatically cleared by hardware when the A/D conversion is complete)
0 = A/D conversion not in progress

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

bit 0 **ADON**: A/D On bit
1 = A/D converter module is operating
0 = A/D converter module is shut-off and consumes no operating current

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

**REGISTER 16-2: ADCON1 REGISTER (ADDRESS 15h, BANK 5)**

| R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ADCS1 | ADCS0 | ADFM | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 |
| bit 7 | | | | | | | bit 0 |

bit 7-6 **ADCS1:ADCS0**: A/D Conversion Clock Select bits
00 = FOSC/8
01 = FOSC/32
10 = FOSC/64
11 = FRC (clock derived from an internal RC oscillator)

bit 5 **ADFM**: A/D Result Format Select
1 = Right justified. 6 Most Significant bits of ADRESH are read as '0'.
0 = Left justified. 6 Least Significant bits of ADRESL are read as '0'.

bit 4 **Unimplemented**: Read as '0'

bit 3-1 **PCFG3:PCFG1**: A/D Port Configuration Control bits

bit 0 **PCFG0**: A/D Voltage Reference Select bit
1 = A/D reference is the VREF+ and VREF- pins
0 = A/D reference is AVDD and AVSS

**Note:** When this bit is set, ensure that the A/D voltage reference specifications are met.

| PCFG3:PCFG0 | AN15 | AN14 | AN13 | AN12 | AN11 | AN10 | AN9 | AN8 | AN7 | AN6 | AN5 | AN4 | AN3 | AN2 | AN1 | AN0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000x | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A |
| 001x | D | A | A | A | A | A | A | A | D | A | A | A | A | A | A | A |
| 010x | D | D | A | A | A | A | A | A | D | D | A | A | A | A | A | A |
| 011x | D | D | D | A | A | A | A | A | D | D | D | A | A | A | A | A |
| 100x | D | D | D | D | A | A | A | A | D | D | D | D | A | A | A | A |
| 101x | D | D | D | D | D | A | A | A | D | D | D | D | D | A | A | A |
| 110x | D | D | D | D | D | D | A | A | D | D | D | D | D | D | A | A |
| 111x | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D | D |

A = Analog input D = Digital I/O

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| - n = Value at POR Reset | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |


© 2000 Microchip Technology Inc.

The ADRESH:ADRESL registers contain the 10-bit result of the A/D conversion. When the A/D conversion is complete, the result is loaded into this A/D result register pair, the GO/$\overline{\text{DONE}}$ bit (ADCON0<2>) is cleared and A/D interrupt flag bit, ADIF is set. The block diagrams of the A/D module are shown in Figure 16-1.

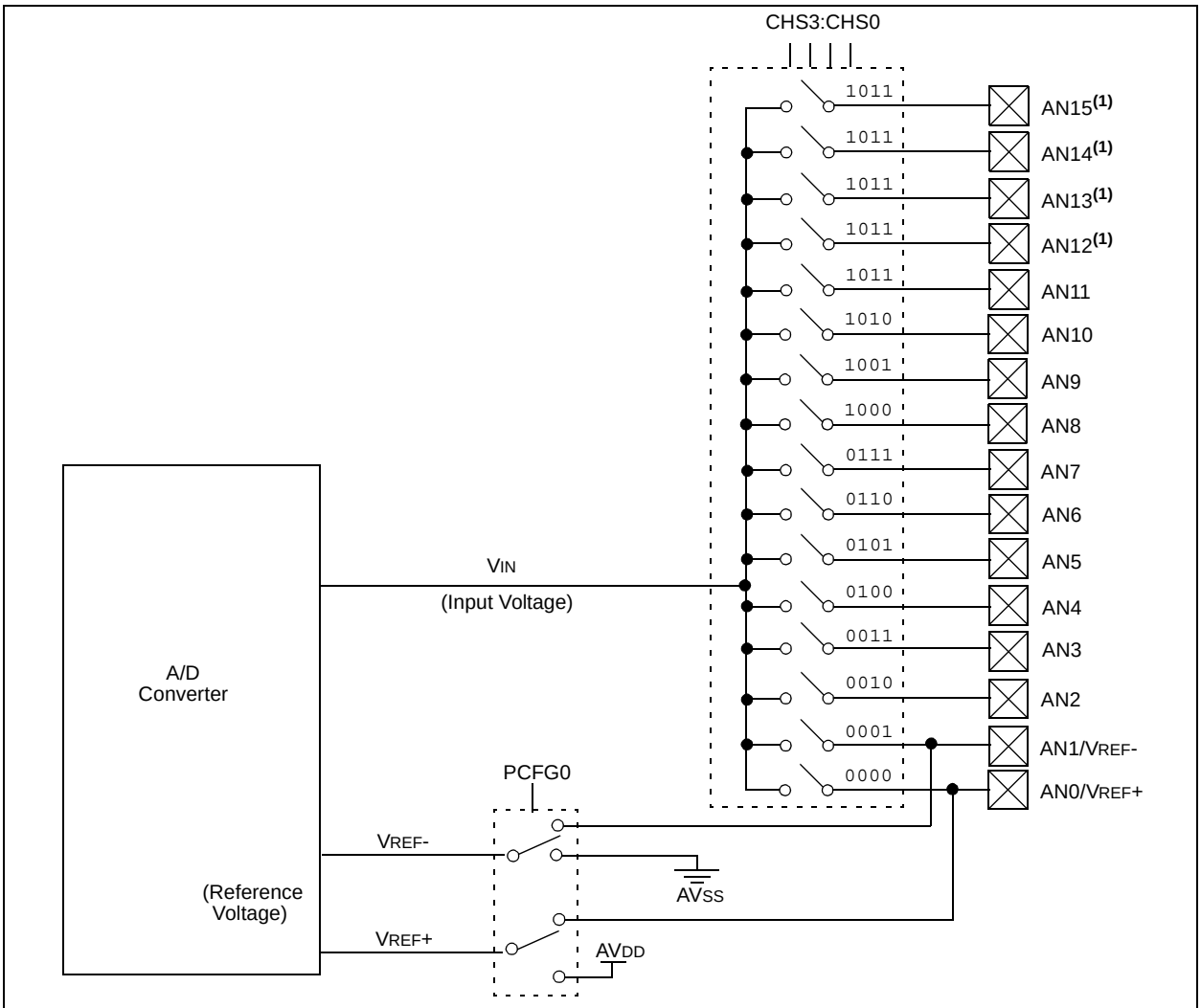
After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding DDR bits selected as inputs. To determine sample time, see Section 16.1. After this acquisition time has elapsed, the A/D conversion can be started. The following steps should be followed for doing an A/D conversion:

1. Configure the A/D module:
   a) Configure analog pins/voltage reference/ and digital I/O (ADCON1)
   b) Select A/D input channel (ADCON0)
   c) Select A/D conversion clock (ADCON0)
   d) Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   a) Clear ADIF bit
   b) Set ADIE bit
   c) Clear GLINTD bit
3. Wait the required acquisition time.
4. Start conversion:
   a) Set GO/$\overline{\text{DONE}}$ bit (ADCON0)
5. Wait for A/D conversion to complete, by either:
   a) Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared

   OR

   b) Waiting for the A/D interrupt
6. Read A/D Result register pair (ADRESH:ADRESL), clear bit ADIF, if required.
7. For next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 2TAD is required before next acquisition starts.

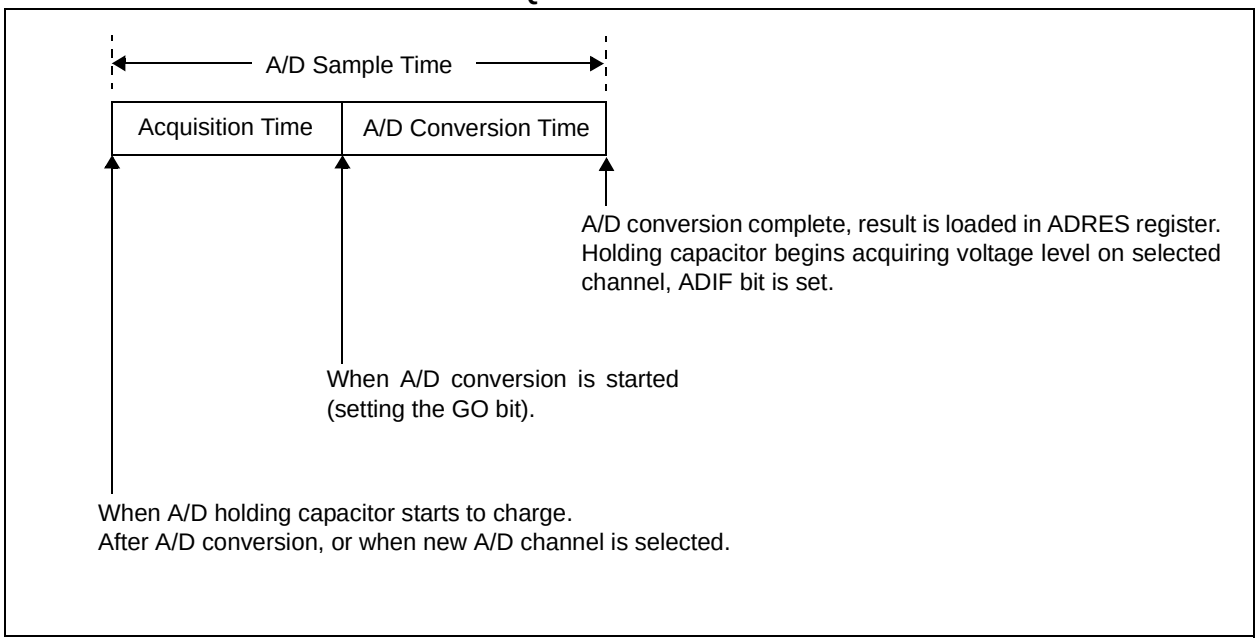
**FIGURE 16-1: A/D BLOCK DIAGRAM**




**Note 1**: These channels are only available on PIC16C76X devices.

Figure 16-2 shows the conversion sequence and the terms that are used. Acquisition time is the time that the A/D module's holding capacitor is connected to the external voltage level. Then, there is the conversion time of 12 TAD, which is started when the GO bit is set. The sum of these two times is the sampling time. There is a minimum acquisition time to ensure that the holding capacitor is charged to a level that will give the desired accuracy for the A/D conversion.

**FIGURE 16-2: A/D CONVERSION SEQUENCE**





© 2000 Microchip Technology Inc.

## 16.1 A/D Acquisition Requirements

For the A/D converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 16-3. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), Figure 16-3. **The maximum recommended impedance for analog sources is 10 kΩ**. As the impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (changed) this acquisition must be done before the conversion can be started.

To calculate the minimum acquisition time, Equation 16-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

Example 16-1 shows the calculation of the minimum required acquisition time (TACQ). This is based on the following application system assumptions.

| | | |
|---|---|---|
| CHOLD | = | 120 pF |
| Rs | = | 10 kΩ |
| Conversion Error | ≤ | 1/2 LSb |
| VDD | = | 5V → Rss = 7 kΩ (see graph in Figure 16-3) |
| Temperature | = | 50°C (system max.) |
| VHOLD | = | 0V @ time = 0 |

**EQUATION 16-1: ACQUISITION TIME**

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \end{aligned}$$

**EQUATION 16-2: A/D MINIMUM CHARGING TIME**

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \cdot (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$

*or*

$$T_C = -(120\ pF)(1\ k\Omega + R_{SS} + R_S)\ \ln(1/2047)$$

**EXAMPLE 16-1: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME**

$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$

Temperature coefficient is only required for temperatures > 25°C.

$T_{ACQ} = 2\ \mu s + T_C + [(Temp - 25°C)(0.05\ \mu s/°C)]$

$T_C = -C_{HOLD}(R_{IC} + R_{SS} + R_S)\ \ln(1/2047)$
$-120\ pF\ (1\ k\Omega + 7\ k\Omega + 10\ k\Omega)\ \ln(0.0004885)$
$-120\ pF\ (18\ k\Omega)\ \ln(0.0004885)$
$-2.16\ \mu s\ (-7.6241)$
$16.47\ \mu s$

$T_{ACQ} = 2\ \mu s + 16.47\ \mu s + [(50 \times C - 25°C)(0.05\ \mu s/°C)]$
$18.447\ \mu s + 1.25\ \mu s$
$19.72\ \mu s$

**Note 1:** The reference voltage (VREF) has no effect on the equation since it cancels itself out.

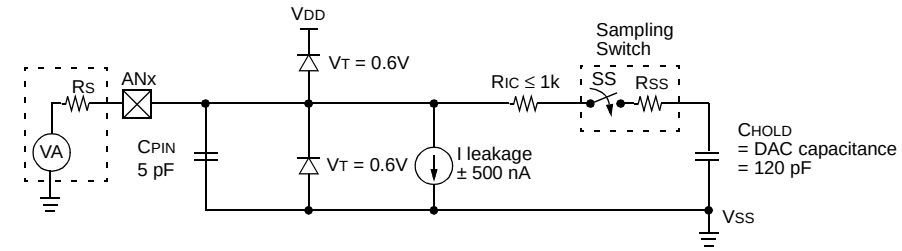
**2:** The charge holding capacitor (CHOLD) is not discharged after each conversion.

**3:** The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

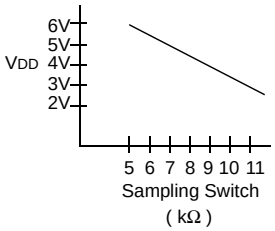
**4:** After a conversion has completed, a 2.0 TAD delay must complete before acquisition can begin again. During this time, the holding capacitor is not connected to the selected A/D input channel.

© 2000 Microchip Technology Inc.

**FIGURE 16-3: ANALOG INPUT MODEL**




| Legend | | |
|---|---|---|
| | $C_{PIN}$ | = input capacitance |
| | $V_T$ | = threshold voltage |
| | I leakage | = leakage current at the pin due to various junctions |
| | $R_{IC}$ | = interconnect resistance |
| | SS | = sampling switch |
| | $C_{HOLD}$ | = sample/hold capacitance (from DAC) |





© 2000 Microchip Technology Inc.

## 16.2 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires a minimum 12TAD per 10-bit conversion. The source of the A/D conversion clock is software selected. The four possible options for TAD are:

- 8TOSC
- 32TOSC
- 64TOSC
- Internal RC oscillator

For correct A/D conversions, the A/D conversion clock (TAD) must be selected to ensure a minimum TAD time of 1.6 µs.

Table 16-1 and Table 16-2 show the resultant TAD times derived from the device operating frequencies and the A/D clock source selected. These times are for standard voltage range devices.

**TABLE 16-1: TAD vs. DEVICE OPERATING FREQUENCIES (STANDARD DEVICES (C))**

| AD Clock Source (TAD) | | Max FOSC (MHz) |
|---|---|---|
| Operation | ADCS1:ADCS0 | |
| 8TOSC | 00 | 5 |
| 32TOSC | 01 | 20 |
| 64TOSC | 10 | 33 |
| RC | 11 | — |

**Note:** When the device frequency is greater than 1 MHz, the RC A/D conversion clock source is only recommended for SLEEP operation.

**TABLE 16-2: TAD vs. DEVICE OPERATING FREQUENCIES (EXTENDED VOLTAGE DEVICES (LC))**

| AD Clock Source (TAD) | | Max FOSC (MHz) |
|---|---|---|
| Operation | ADCS1:ADCS0 | |
| 8TOSC | 00 | 2.67 |
| 32TOSC | 01 | 10.67 |
| 64TOSC | 10 | 21.33 |
| RC | 11 | — |

**Note:** When the device frequency is greater than 1 MHz, the RC A/D conversion clock source is only recommended for SLEEP operation.

## 16.3 Configuring Analog Port Pins

The ADCON1, and DDR registers control the operation of the A/D port pins. The port pins that are desired as analog inputs must have their corresponding DDR bits set (input). If the DDR bit is cleared (output), the digital output level ($V_{OH}$ or $V_{OL}$) will be converted.

The A/D operation is independent of the state of the CHS2:CHS0 bits and the DDR bits.

**Note 1:** When reading the port register, any pin configured as an analog input channel will read as cleared (a low level). Pins configured as digital inputs, will convert an analog input. Analog levels on a digitally configured input will not affect the conversion accuracy.

**2:** Analog levels on any pin that is defined as a digital input (including the AN15:AN0 pins), may cause the input buffer to consume current that is out of the devices specification.

## 16.4 A/D Conversions

Example 16-2 shows how to perform an A/D conversion. The PORTF and lower four PORTG pins are configured as analog inputs. The analog references ($V_{REF+}$ and $V_{REF-}$) are the device $AV_{DD}$ and $AV_{SS}$. The A/D interrupt is enabled, and the A/D conversion clock is FRC. The conversion is performed on the RG3/AN0 pin (channel 0).

**Note:** The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D.

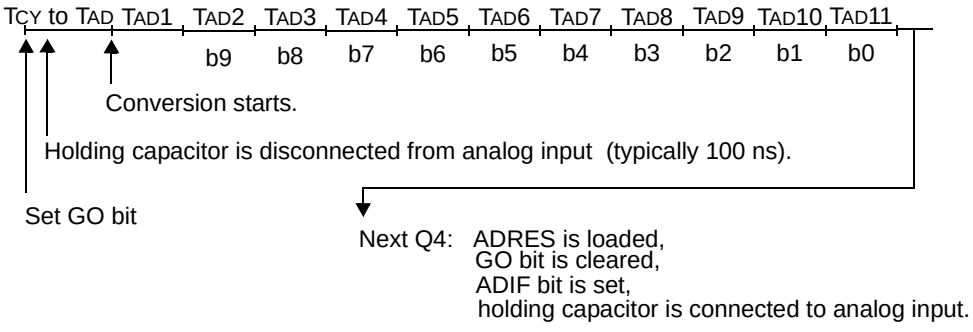
Clearing the GO/$\overline{\text{DONE}}$ bit during a conversion will abort the current conversion. The A/D result register pair will NOT be updated with the partially completed A/D conversion sample. That is, the ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers). After the A/D conversion is aborted, a $2T_{AD}$ wait is required before the next acquisition is started. After this $2T_{AD}$ wait, acquisition on the selected channel is automatically started.

In Figure 16-4, after the GO bit is set, the first time segment has a minimum of $T_{CY}$ and a maximum of $T_{AD}$.

**EXAMPLE 16-2: A/D CONVERSION**

```
     MOVLB   5                ; Bank 5
     CLRF    ADCON1, F        ; Configure A/D inputs, All analog, TAD = Fosc/8, left just.
     MOVLW   0x01             ; A/D is on, Channel 0 is selected
     MOVWF   ADCON0           ;
     MOVLB   4                ; Bank 4
     BCF     PIR2, ADIF       ; Clear A/D interrupt flag bit
     BSF     PIE2, ADIE       ; Enable A/D interrupts
     BSF     INTSTA, PEIE     ; Enable peripheral interrupts
     BCF     CPUSTA, GLINTD   ; Enable all interrupts
;
;  Ensure that the required sampling time for the selected input channel has elapsed.
;  Then the conversion may be started.
;
     MOVLB   5                ; Bank 5
     BSF     ADCON0, GO       ; Start A/D Conversion
       :                      ;   The ADIF bit will be set and the GO/DONE bit
       :                      ;   is cleared upon completion of the A/D Conversion
```

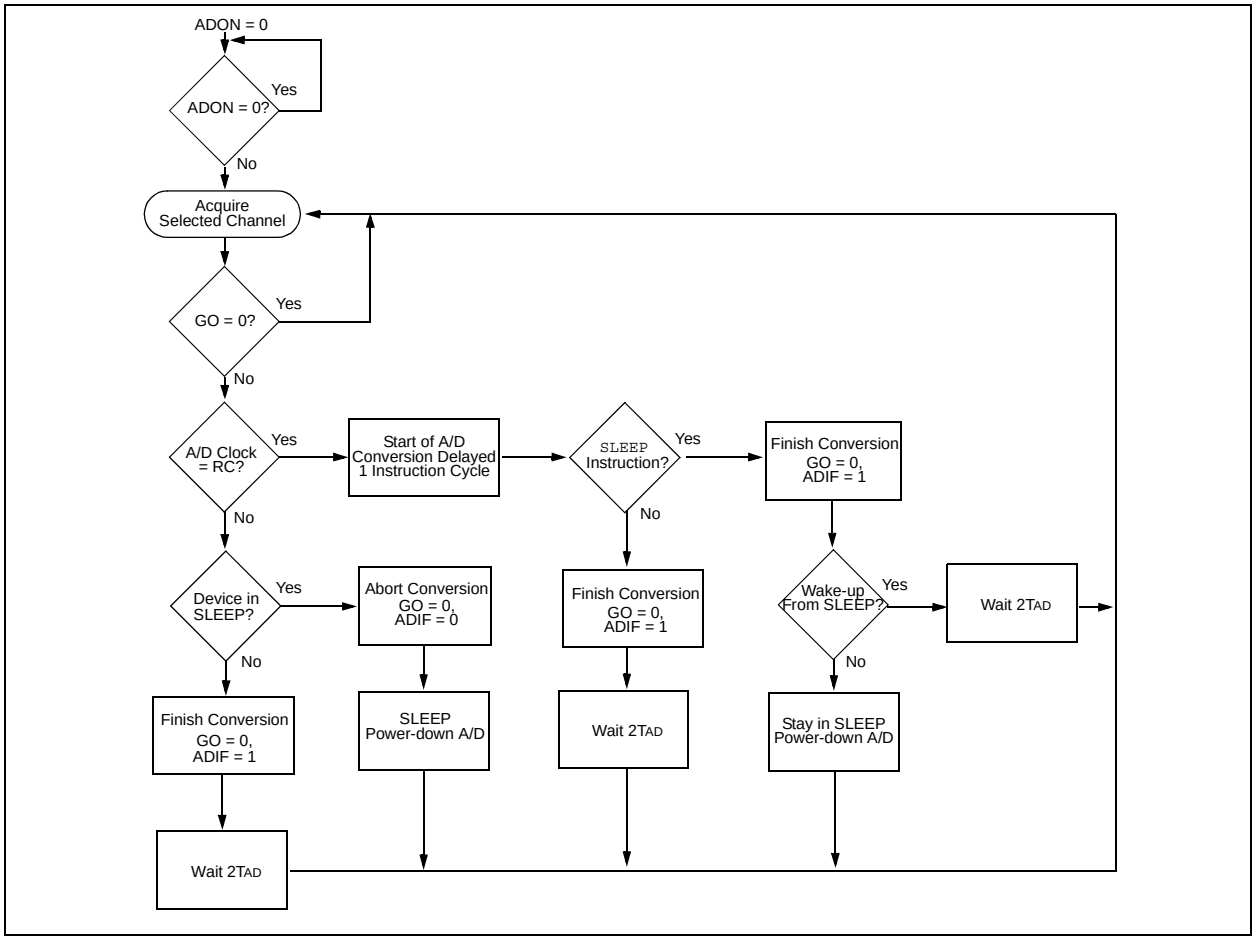
**FIGURE 16-4: A/D CONVERSION $T_{AD}$ CYCLES**





© 2000 Microchip Technology Inc.

**FIGURE 16-5: FLOW CHART OF A/D OPERATION**



### 16.4.1 A/D RESULT REGISTERS

The ADRESH:ADRESL register pair is the location where the 10-bit A/D result is loaded at the completion of the A/D conversion. This register pair is 16-bits wide. The A/D module gives the flexibility to left or right justify the 10-bit result in the 16-bit result register. The A/D Format Select bit (ADFM) controls this justification. Figure 16-6 shows the operation of the A/D result justification. The extra bits are loaded with '0's'. When an A/D result will not overwrite these locations (A/D disable), these registers may be used as two general purpose 8-bit registers.

## 16.5 A/D Operation During SLEEP

The A/D module can operate during SLEEP mode. This requires that the A/D clock source be set to RC (ADCS1:ADCS0 = 11). When the RC clock source is selected, the A/D module waits one instruction cycle before starting the conversion. This allows the SLEEP instruction to be executed, which eliminates all digital switching noise from the conversion. When the conversion is completed, the GO/$\overline{\text{DONE}}$ bit will be cleared, and the result loaded into the ADRES register. If the A/D interrupt is enabled, the device will wake-up from SLEEP. If the A/D interrupt is not enabled, the A/D module will then be turned off, although the ADON bit will remain set.

When the A/D clock source is another clock option (not RC), a SLEEP instruction will cause the present conversion to be aborted and the A/D module to be turned off, though the ADON bit will remain set.

Turning off the A/D places the A/D module in its lowest current consumption state.

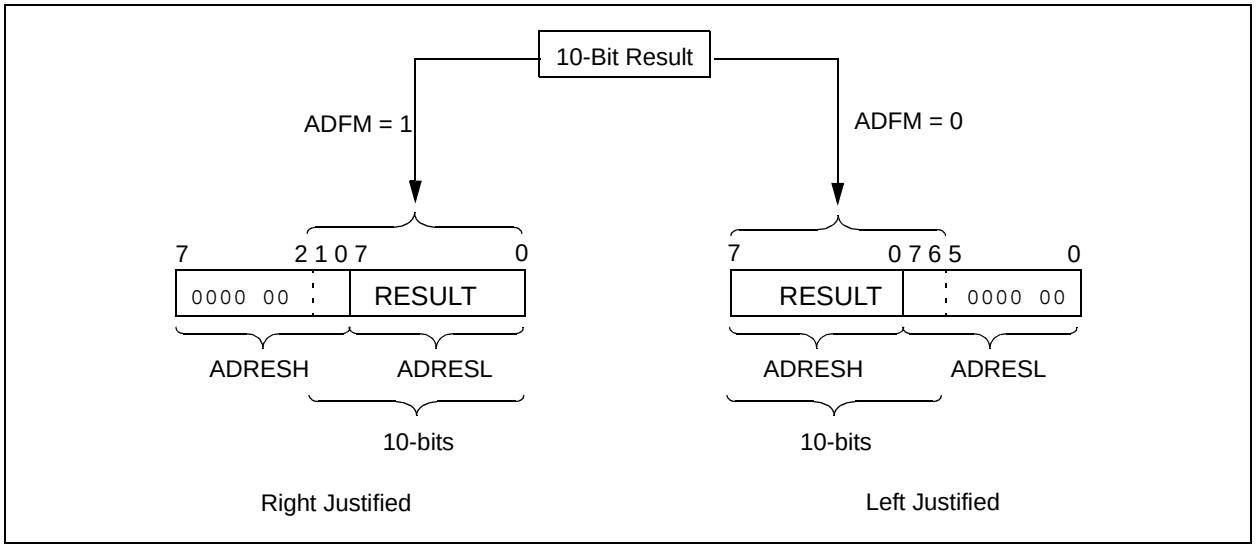
| **Note:** | For the A/D module to operate in SLEEP, the A/D clock source must be set to RC (ADCS1:ADCS0 = 11). To allow the conversion to occur during SLEEP, ensure the SLEEP instruction immediately follows the instruction that sets the GO/$\overline{\text{DONE}}$ bit. |
|---|---|

## 16.6 Effects of a RESET

A device RESET forces all registers to their RESET state. This forces the A/D module to be turned off, and any conversion is aborted.

The value that is in the ADRESH:ADRESL registers is not modified for a Power-on Reset. The ADRESH:ADRESL registers will contain unknown data after a Power-on Reset.

**FIGURE 16-6: A/D RESULT JUSTIFICATION**





© 2000 Microchip Technology Inc.

## 16.7 A/D Accuracy/Error

In systems where the device frequency is low, use of the A/D RC clock is preferred. At moderate to high frequencies, TAD should be derived from the device oscillator.

The absolute accuracy specified for the A/D converter includes the sum of all contributions for quantization error, integral error, differential error, full scale error, offset error, and monotonicity. It is defined as the maximum deviation from an actual transition versus an ideal transition for any code. The absolute error of the A/D converter is specified at < ±1 LSb for VDD = VREF (over the device's specified operating range). However, the accuracy of the A/D converter will degrade as VREF diverges from VDD.

For a given range of analog inputs, the output digital code will be the same. This is due to the quantization of the analog input to a digital code. Quantization error is typically ± 1/2 LSb and is inherent in the analog to digital conversion process. The only way to reduce quantization error is to increase the resolution of the A/D converter or oversample.

Offset error measures the first actual transition of a code versus the first ideal transition of a code. Offset error shifts the entire transfer function. Offset error can be calibrated out of a system or introduced into a system through the interaction of the total leakage current and source impedance at the analog input.

Gain error measures the maximum deviation of the last actual transition and the last ideal transition adjusted for offset error. This error appears as a change in slope of the transfer function. The difference in gain error to full scale error is that full scale does not take offset error into account. Gain error can be calibrated out in software.

Linearity error refers to the uniformity of the code changes. Linearity errors cannot be calibrated out of the system. Integral non-linearity error measures the actual code transition versus the ideal code transition, adjusted by the gain error for each code.

Differential non-linearity measures the maximum actual code width versus the ideal code width. This measure is unadjusted.

The maximum pin leakage current is specified in the Device Data Sheet electrical specification (Table 20-2, parameter #D060).

In systems where the device frequency is low, use of the A/D RC clock is preferred. At moderate to high frequencies, TAD should be derived from the device oscillator. TAD must not violate the minimum and should be minimized to reduce inaccuracies due to noise and sampling capacitor bleed off.

In systems where the device will enter SLEEP mode after the start of the A/D conversion, the RC clock source selection is required. In this mode, the digital noise from the modules in SLEEP are stopped. This method gives high accuracy.

## 16.8 Connection Considerations

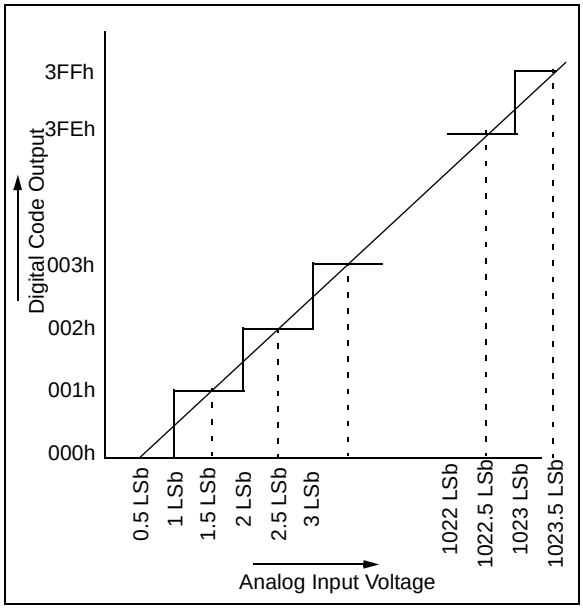
If the input voltage exceeds the rail values (VSS or VDD) by greater than 0.3V, then the accuracy of the conversion is out of specification.

An external RC filter is sometimes added for anti-aliasing of the input signal. The R component should be selected to ensure that the total source impedance is kept under the 10 kΩ recommended specification. Any external components connected (via hi-impedance) to an analog input pin (capacitor, zener diode, etc.) should have very little leakage current at the pin.

## 16.9 Transfer Function

The transfer function of the A/D converter is as follows: the first transition occurs when the analog input voltage (VAIN) equals Analog VREF / 1024 (Figure 16-7).

**FIGURE 16-7: A/D TRANSFER FUNCTION**



## 16.10 References

A good reference for understanding A/D converter is the "Analog-Digital Conversion Handbook" third edition, published by Prentice Hall (ISBN 0-13-03-2848-0).

### TABLE 16-3: REGISTERS/BITS ASSOCIATED WITH A/D

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | POR, BOR | $\overline{\text{MCLR}}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 06h, unbanked | CPUSTA | — | — | STAKAV | GLINTD | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | --11 1100 | --11 qq11 |
| 07h, unbanked | INTSTA | PEIF | T0CKIF | T0IF | INTF | PEIE | T0CKIE | T0IE | INTE | 0000 0000 | 0000 0000 |
| 10h, Bank 4 | PIR2 | SSPIF | BCLIF | ADIF | — | CA4IF | CA3IF | TX2IF | RC2IF | 000- 0010 | 000- 0010 |
| 11h, Bank 4 | PIE2 | SSPIE | BCLIE | ADIE | — | CA4IE | CA3IE | TX2IE | RC2IE | 000- 0000 | 000- 0000 |
| 10h, Bank 5 | DDRF | Data Direction Register for PORTF | | | | | | | | 1111 1111 | 1111 1111 |
| 11h, Bank 5 | PORTF | RF7/ AN11 | RF6/ AN10 | RF5/ AN9 | RF4/ AN8 | RF3/ AN7 | RF2/ AN6 | RF1/ AN5 | RF0/ AN4 | 0000 0000 | 0000 0000 |
| 12h, Bank 5 | DDRG | Data Direction register for PORTG | | | | | | | | 1111 1111 | 1111 1111 |
| 13h, Bank 5 | PORTG | RG7/ TX2/CK2 | RG6/ RX2/DT2 | RG5/ PWM3 | RG4/ CAP3 | RG3/ AN0/$V_{REF+}$ | RG2/ AN1/$V_{REF-}$ | RG1/ AN2 | RG0/ AN3 | xxxx 0000 | uuuu 0000 |
| 14h, Bank 5 | ADCON0 | CHS3 | CHS2 | CHS1 | CHS0 | — | GO/$\overline{\text{DONE}}$ | — | ADON | 0000 -0-0 | 0000 -0-0 |
| 15h, Bank 5 | ADCON1 | ADCS1 | ADCS0 | ADFM | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 000- 0000 | 000- 0000 |
| 16h, Bank 5 | ADRESL | A/D Result Low Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 17h, Bank 5 | ADRESH | A/D Result High Register | | | | | | | | xxxx xxxx | uuuu uuuu |

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.

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


© 2000 Microchip Technology Inc.

## 17.0 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 PIC17CXXX family has a host of such features intended to maximize system reliability, minimize cost through elimination of external components, provide power saving operating modes and offer code protection. These are:

- Oscillator Selection (Section 4.0)
- RESET (Section 5.0)
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts (Section 6.0)
- Watchdog Timer (WDT)
- SLEEP mode
- Code protection

The PIC17CXXX has a Watchdog Timer which can be shut-off only through EPROM bits. It runs off its own RC oscillator for added reliability. There are two timers that offer necessary delays on POR and BOR. One is the Oscillator Start-up Timer (OST), intended to keep the chip in RESET until the crystal oscillator is stable. The other is the Power-up Timer (PWRT), which provides a fixed delay of 96 ms (nominal) on power-up only, designed to keep the part in RESET while the power supply stabilizes. With these two timers on-chip, most applications need no external RESET circuitry.

The SLEEP mode is designed to offer a very low current power-down mode. The user can wake from SLEEP through external RESET, Watchdog Timer Reset, or through an interrupt. Several oscillator options are also made available to allow the part to fit the application. The RC oscillator option saves system cost, while the LF crystal option saves power. Configuration bits are used to select various options. This configuration word has the format shown in Figure 17-1.

**REGISTER 17-1: CONFIGURATION WORDS**

| High (H) Table Read Addr. | U-x | R/P-1 | R/P-1 | U-x | U-x | U-x | U-x | U-x | U-x |
|---|---|---|---|---|---|---|---|---|---|
| FE0Fh - FE08h | — | PM2 | BODEN | — | — | — | — | — | — |
| | bit 15 bit 8 | bit 7 | | | | | | | bit 0 |

| Low (L) Table Read Addr. | U-x | U-x | R/P-1 | U-x | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|---|---|
| FE07h - FE00h | — | — | PM1 | — | PM0 | WDTPS1 | WDTPS0 | FOSC1 | FOSC0 |
| | bit 15 bit 8 | bit 7 | | | | | | | bit 0 |

bits 7H, 6L, 4L — **PM2, PM1, PM0:** Processor Mode Select bits
111 = Microprocessor mode
110 = Microcontroller mode
101 = Extended Microcontroller mode
000 = Code Protected Microcontroller mode

bit 6H — **BODEN:** Brown-out Detect Enable
1 = Brown-out Detect circuitry is enabled
0 = Brown-out Detect circuitry is disabled

bits 3L:2L — **WDTPS1:WDTPS0**: WDT Postscaler Select bits
11 = WDT enabled, postscaler = 1
10 = WDT enabled, postscaler = 256
01 = WDT enabled, postscaler = 64
00 = WDT disabled, 16-bit overflow timer

bits 1L:0L — **FOSC1:FOSC0**: Oscillator Select bits
11 = EC oscillator
10 = XT oscillator
01 = RC oscillator
00 = LF oscillator

Shaded bits (—) — **Reserved**

© 2000 Microchip Technology Inc.

## 17.1 Configuration Bits

The PIC17CXXX has eight configuration locations (Table 17-1). These locations can be programmed (read as '0'), or left unprogrammed (read as '1') to select various device configurations. Any write to a configuration location, regardless of the data, will program that configuration bit. A TABLWT instruction and raising the $\overline{MCLR}$/VPP pin to the programming voltage are both required to write to program memory locations. The configuration bits can be read by using the TABLRD instructions. Reading any configuration location between FE00h and FE07h will read the low byte of the configuration word (Figure 17-1) into the TABLATL register. The TABLATH register will be FFh. Reading a configuration location between FE08h and FE0Fh will read the high byte of the configuration word into the TABLATL register. The TABLATH register will be FFh.

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

**TABLE 17-1: CONFIGURATION LOCATIONS**

| Bit | Address |
|---|---|
| FOSC0 | FE00h |
| FOSC1 | FE01h |
| WDTPS0 | FE02h |
| WDTPS1 | FE03h |
| PM0 | FE04h |
| PM1 | FE06h |
| BODEN | FE0Eh |
| PM2 | FE0Fh |

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

## 17.2 Oscillator Configurations

### 17.2.1 OSCILLATOR TYPES

The PIC17CXXX can be operated in four different oscillator modes. The user can program two configuration bits (FOSC1:FOSC0) to select one of these four modes:

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

For information on the different oscillator types and how to use them, please refer to Section 4.0.


© 2000 Microchip Technology Inc.

## 17.3 Watchdog Timer (WDT)

The Watchdog Timer's function is to recover from software malfunction, or to reset the device while in SLEEP mode. The WDT uses an internal free running on-chip RC oscillator for its clock source. This does not require any external components. This RC oscillator is separate from the RC oscillator of the OSC1/CLKIN pin. That means that the WDT will run even if the clock on the OSC1/CLKIN and OSC2/CLKOUT pins has been stopped, for example, by execution of a SLEEP instruction. During normal operation, a WDT time-out generates a device RESET. The WDT can be permanently disabled by programming the configuration bits WDTPS1:WDTPS0 as '00' (Section 17.1).

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

### 17.3.1 WDT PERIOD

The WDT has a nominal time-out period of 12 ms (with postscaler = 1). The time-out periods vary with temperature, VDD and process variations from part to part (see DC specs). If longer time-out periods are desired, configuration bits should be used to enable the WDT with a greater prescale. Thus, typical time-out periods up to 3.0 seconds can be realized.

The CLRWDT and SLEEP instructions clear the WDT and its postscale setting and prevent it from timing out, thus generating a device RESET condition.

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

### 17.3.2 CLEARING THE WDT AND POSTSCALER

The WDT and postscaler are cleared when:

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

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

### 17.3.3 WDT PROGRAMMING CONSIDERATIONS

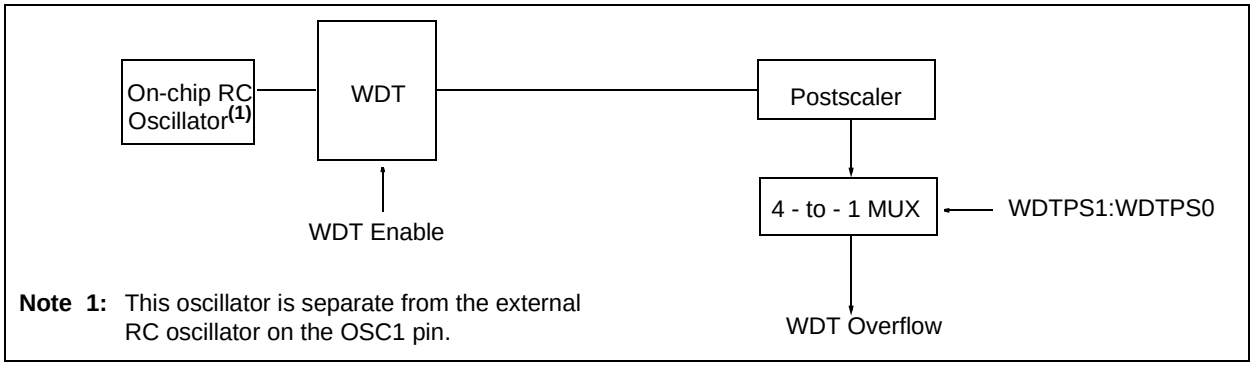
It should also be taken in account that under worst case conditions (VDD = Min., Temperature = Max., Max. WDT postscaler), it may take several seconds before a WDT time-out occurs.

The WDT and postscaler become the Power-up Timer whenever the PWRT is invoked.

### 17.3.4 WDT AS NORMAL TIMER

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

**FIGURE 17-1: WATCHDOG TIMER BLOCK DIAGRAM**




**Note 1:** This oscillator is separate from the external RC oscillator on the OSC1 pin.

**TABLE 17-2: REGISTERS/BITS ASSOCIATED WITH THE WATCHDOG TIMER**

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | $\overline{MCLR}$, WDT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| — | Config | See Figure 17-1 for location of WDTPSx bits in Configuration Word. | | | | | | | | (Note 1) | (Note 1) |
| 06h, Unbanked | CPUSTA | — | — | STKAV | GLINTD | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | --11 11qq | --11 qquu |

Legend: - = unimplemented, read as '0', q = value depends on condition. Shaded cells are not used by the WDT.
**Note 1:** This value will be as the device was programmed, or if unprogrammed, will read as all '1's.

## 17.4 Power-down Mode (SLEEP)

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

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

### 17.4.1 WAKE-UP FROM SLEEP

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

- Power-on Reset
- Brown-out Reset
- External RESET input on $\overline{MCLR}/V_{PP}$ pin
- WDT Reset (if WDT was enabled)
- Interrupt from RA0/INT pin, RB port change, T0CKI interrupt, or some peripheral interrupts

The following peripheral interrupts can wake the device from SLEEP:

- Capture interrupts
- USART synchronous slave transmit interrupts
- USART synchronous slave receive interrupts
- A/D conversion complete
- SPI slave transmit/receive complete
- $I^2C$ slave receive

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

Any RESET event will cause a device RESET. Any interrupt event is considered a continuation of program execution. The $\overline{TO}$ and $\overline{PD}$ bits in the CPUSTA register can be used to determine the cause of a device RESET. The $\overline{PD}$ bit, which is set on power-up, is cleared when SLEEP is invoked. The $\overline{TO}$ bit is cleared if WDT time-out occurred (and caused a RESET).

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

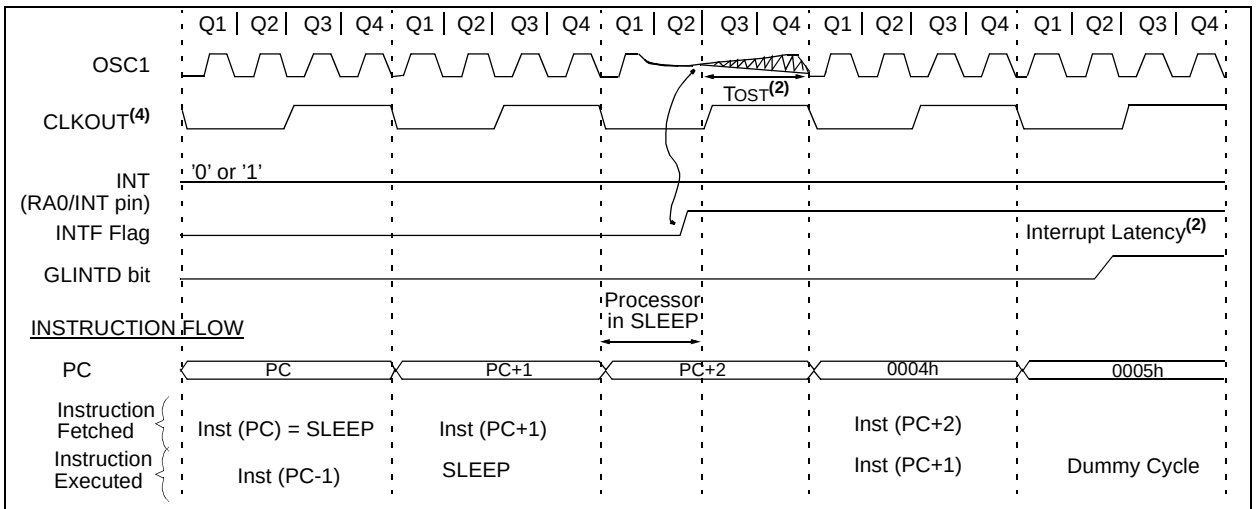
| **Note:** | If the global interrupt is disabled (GLINTD is set), but any interrupt source has both its interrupt enable bit and the corresponding interrupt flag bit set, the device will immediately wake-up from SLEEP. The $\overline{TO}$ bit is set and the $\overline{PD}$ bit is cleared. |
|---|---|

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

#### 17.4.1.1 Wake-up Delay

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

**FIGURE 17-2: WAKE-UP FROM SLEEP THROUGH INTERRUPT**




**Note 1:** XT or LF oscillator mode assumed.
**2:** TOST = 1024TOSC (drawing not to scale). This delay will not be there for RC osc mode.
**3:** When GLINTD = 0, processor jumps to interrupt routine after wake-up. If GLINTD = 1, execution will continue in line.
**4:** CLKOUT is not available in these osc modes, but shown here for timing reference.


© 2000 Microchip Technology Inc.

### 17.4.2 MINIMIZING CURRENT CONSUMPTION

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

## 17.5 Code Protection

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

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

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

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

## 17.6 In-Circuit Serial Programming

The PIC17C7XX group of the high-end family (PIC17CXXX) has an added feature that allows serial programming 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.

Devices may be serialized to make the product unique; “special” variants of the product may be offered and code updates are possible. This allows for increased design flexibility.

To place the device into the Serial Programming Test mode, two pins will need to be placed at $V_{IHH}$. These are the TEST pin and the $\overline{MCLR}/V_{PP}$ pin. Also, a sequence of events must occur as follows:

1. The TEST pin is placed at $V_{IHH}$.
2. The $\overline{MCLR}/V_{PP}$ pin is placed at $V_{IHH}$.

There is a setup time between step 1 and step 2 that must be met.

After this sequence, the Program Counter is pointing to program memory address 0xFF60. This location is in the Boot ROM. The code initializes the USART/SCI so that it can receive commands. For this, the device must be clocked. The device clock source in this mode is the RA1/T0CKI pin. After delaying to allow the USART/SCI to initialize, commands can be received. The flow is shown in these 3 steps:

1. The device clock source starts.
2. Wait 80 device clocks for Boot ROM code to configure the USART/SCI.
3. Commands may now be sent.

For complete details of serial programming, please refer to the PIC17C7XX Programming Specification. (Contact your local Microchip Technology Sales Office for availability.)

**FIGURE 17-3: TYPICAL IN-CIRCUIT SERIAL PROGRAMMING CONNECTION**

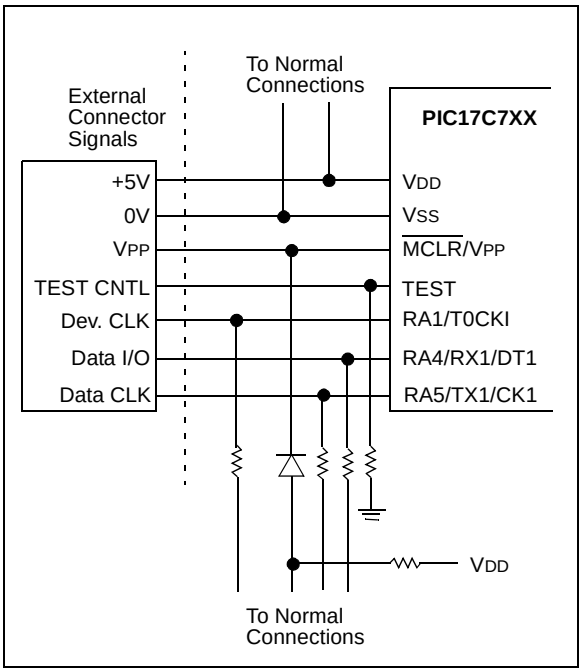



**TABLE 17-3: ICSP INTERFACE PINS**

| Name | During Programming | | |
|---|---|---|---|
| | Function | Type | Description |
| RA4/RX1/DT1 | DT | I/O | Serial Data |
| RA5/TX1/CK1 | CK | I | Serial Clock |
| RA1/T0CKI | OSCI | I | Device Clock Source |
| TEST | TEST | I | Test mode selection control input, force to $V_{IHH}$ |
| $\overline{MCLR}/V_{PP}$ | $\overline{MCLR}/V_{PP}$ | P | Master Clear Reset and Device Programming Voltage |
| $V_{DD}$ | $V_{DD}$ | P | Positive supply for logic and I/O pins |
| $V_{SS}$ | $V_{SS}$ | P | Ground reference for logic and I/O pins |


© 2000 Microchip Technology Inc.

## 18.0 INSTRUCTION SET SUMMARY

The PIC17CXXX instruction set consists of 58 instructions. Each instruction is a 16-bit word divided into an OPCODE and one or more operands. The opcode specifies the instruction type, while the operand(s) further specify the operation of the instruction. The PIC17CXXX instruction set can be grouped into three types:

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

These formats are shown in Figure 18-1.

Table 18-1 shows the field descriptions for the opcodes. These descriptions are useful for understanding the opcodes in Table 18-2 and in each specific instruction descriptions.

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

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

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 13-bit constant or literal value.

The instruction set is highly orthogonal and is grouped into:

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

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

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

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

**TABLE 18-1: OPCODE FIELD DESCRIPTIONS**

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

Table 18-2 lists the instructions recognized by the MPASM assembler.

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

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

0xhh

where h signifies a hexadecimal digit.

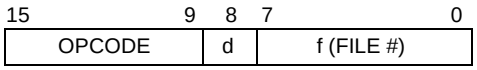
To represent a binary number:

`0000 0100b`

where b signifies a binary string.

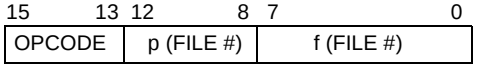
**FIGURE 18-1: GENERAL FORMAT FOR INSTRUCTIONS**

**Byte-oriented** file register operations



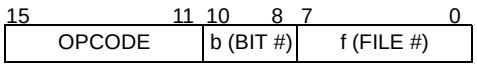

d = 0 for destination WREG
d = 1 for destination f
f = 8-bit file register address

**Byte to Byte** move operations



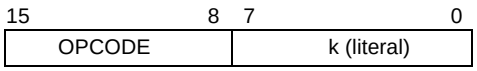

p = peripheral register file address
f = 8-bit file register address

**Bit-oriented** file register operations



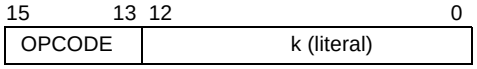

b = 3-bit address
f = 8-bit file register address

**Literal and control** operations




k = 8-bit immediate value

**CALL and GOTO** operations




k = 13-bit immediate value

## 18.1 Special Function Registers as Source/Destination

The PIC17C7XX's orthogonal instruction set allows read and write of all file registers, including special function registers. There are some special situations the user should be aware of:

### 18.1.1 ALUSTA AS DESTINATION

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

### 18.1.2 PCL AS SOURCE OR DESTINATION

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

| | |
|---|---|
| Read PC: | PCH → PCLATH; PCL → dest |
| Write PCL: | PCLATH → PCH;<br>8-bit destination value → PCL |
| Read-Modify-Write: | PCL→ ALU operand<br>PCLATH → PCH;<br>8-bit result → PCL |

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

### 18.1.3 BIT MANIPULATION

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

| | |
|---|---|
| **Note:** | Status bits that are manipulated by the device (including the interrupt flag bits) are set or cleared in the Q1 cycle. So, there is no issue on doing R-M-W instructions on registers which contain these bits |


© 2000 Microchip Technology Inc.

## 18.2 Q Cycle Activity

Each instruction cycle (TCY) is comprised of four Q cycles (Q1-Q4). The Q cycle is the same as the device oscillator cycle (TOSC). The Q cycles provide the timing/ designation for the Decode, Read, Process Data, Write, etc., of each instruction cycle. The following diagram shows the relationship of the Q cycles to the instruction cycle.

The four Q cycles that make up an instruction cycle (TCY) can be generalized as:

Q1: Instruction Decode Cycle or forced No operation

Q2: Instruction Read Cycle or No operation

Q3: Process the Data

Q4: Instruction Write Cycle or No operation

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

**FIGURE 18-2: Q CYCLE ACTIVITY**

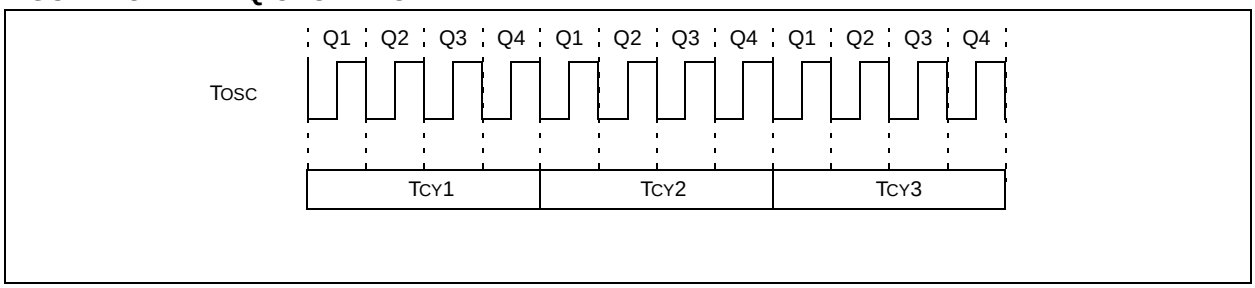


**TABLE 18-2: PIC17CXXX INSTRUCTION SET**

| Mnemonic, Operands | | Description | Cycles | 16-bit Opcode MSb LSb | Status Affected | Notes |
|---|---|---|---|---|---|---|
| **BYTE-ORIENTED FILE REGISTER OPERATIONS** | | | | | | |
| **ADDWF** | **f,d** | ADD WREG to f | 1 | 0000 111d ffff ffff | OV,C,DC,Z | |
| **ADDWFC** | **f,d** | ADD WREG and Carry bit to f | 1 | 0001 000d ffff ffff | OV,C,DC,Z | |
| **ANDWF** | **f,d** | AND WREG with f | 1 | 0000 101d ffff ffff | Z | |
| **CLRF** | **f,s** | Clear f, or Clear f and Clear WREG | 1 | 0010 100s ffff ffff | None | 3 |
| **COMF** | **f,d** | Complement f | 1 | 0001 001d ffff ffff | Z | |
| **CPFSEQ** | **f** | Compare f with WREG, skip if f = WREG | 1 (2) | 0011 0001 ffff ffff | None | 6,8 |
| **CPFSGT** | **f** | Compare f with WREG, skip if f > WREG | 1 (2) | 0011 0010 ffff ffff | None | 2,6,8 |
| **CPFSLT** | **f** | Compare f with WREG, skip if f < WREG | 1 (2) | 0011 0000 ffff ffff | None | 2,6,8 |
| **DAW** | **f,s** | Decimal Adjust WREG Register | 1 | 0010 111s ffff ffff | C | 3 |
| **DECF** | **f,d** | Decrement f | 1 | 0000 011d ffff ffff | OV,C,DC,Z | |
| **DECFSZ** | **f,d** | Decrement f, skip if 0 | 1 (2) | 0001 011d ffff ffff | None | 6,8 |
| **DCFSNZ** | **f,d** | Decrement f, skip if not 0 | 1 (2) | 0010 011d ffff ffff | None | 6,8 |
| **INCF** | **f,d** | Increment f | 1 | 0001 010d ffff ffff | OV,C,DC,Z | |
| **INCFSZ** | **f,d** | Increment f, skip if 0 | 1 (2) | 0001 111d ffff ffff | None | 6,8 |
| **INFSNZ** | **f,d** | Increment f, skip if not 0 | 1 (2) | 0010 010d ffff ffff | None | 6,8 |
| **IORWF** | **f,d** | Inclusive OR WREG with f | 1 | 0000 100d ffff ffff | Z | |
| **MOVFP** | **f,p** | Move f to p | 1 | 011p pppp ffff ffff | None | |
| **MOVPF** | **p,f** | Move p to f | 1 | 010p pppp ffff ffff | Z | |
| **MOVWF** | **f** | Move WREG to f | 1 | 0000 0001 ffff ffff | None | |
| **MULWF** | **f** | Multiply WREG with f | 1 | 0011 0100 ffff ffff | None | |
| **NEGW** | **f,s** | Negate WREG | 1 | 0010 110s ffff ffff | OV,C,DC,Z | 1,3 |
| **NOP** | **—** | No Operation | 1 | 0000 0000 0000 0000 | None | |
| **RLCF** | **f,d** | Rotate left f through Carry | 1 | 0001 101d ffff ffff | C | |
| **RLNCF** | **f,d** | Rotate left f (no carry) | 1 | 0010 001d ffff ffff | None | |
| **RRCF** | **f,d** | Rotate right f through Carry | 1 | 0001 100d ffff ffff | C | |
| **RRNCF** | **f,d** | Rotate right f (no carry) | 1 | 0010 000d ffff ffff | None | |
| **SETF** | **f,s** | Set f | 1 | 0010 101s ffff ffff | None | 3 |
| **SUBWF** | **f,d** | Subtract WREG from f | 1 | 0000 010d ffff ffff | OV,C,DC,Z | 1 |
| **SUBWFB** | **f,d** | Subtract WREG from f with Borrow | 1 | 0000 001d ffff ffff | OV,C,DC,Z | 1 |
| **SWAPF** | **f,d** | Swap f | 1 | 0001 110d ffff ffff | None | |
| **TABLRD** | **t,i,f** | Table Read | 2 (3) | 1010 10ti ffff ffff | None | 7 |
| **TABLWT** | **t,i,f** | Table Write | 2 | 1010 11ti ffff ffff | None | 5 |
| **TLRD** | **t,f** | Table Latch Read | 1 | 1010 00tx ffff ffff | None | |
| **TLWT** | **t,f** | Table Latch Write | 1 | 1010 01tx ffff ffff | None | |

Legend: Refer to Table 18-1 for opcode field descriptions.

**Note 1:** 2's Complement method.

**2:** Unsigned arithmetic.

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

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

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

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

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

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


© 2000 Microchip Technology Inc.

**TABLE 18-2: PIC17CXXX INSTRUCTION SET (CONTINUED)**

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

Legend: Refer to Table 18-1 for opcode field descriptions.

**Note 1:** 2's Complement method.

**2:** Unsigned arithmetic.

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

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

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

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

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

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

© 2000 Microchip Technology Inc.

**ADDLW** **ADD Literal to WREG**

Syntax: [ *label* ] ADDLW k

Operands: $0 \le k \le 255$

Operation: (WREG) + k → (WREG)

Status Affected: OV, C, DC, Z

Encoding:

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

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

**Words:** **1**

**Cycles:** **1**

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to WREG |

Example: `ADDLW 0x15`

Before Instruction
WREG = 0x10

After Instruction
WREG = 0x25

**ADDWF** **ADD WREG to f**

Syntax: [ *label* ] ADDWF f,d

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

Operation: (WREG) + (f) → (dest)

Status Affected: OV, C, DC, Z

Encoding:

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

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `ADDWF REG, 0`

Before Instruction
WREG = 0x17
REG = 0xC2

After Instruction
WREG = 0xD9
REG = 0xC2


© 2000 Microchip Technology Inc.

**ADDWFC** **ADD WREG and Carry bit to f**

Syntax: [ *label* ] ADDWFC f,d

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

Operation: (WREG) + (f) + C → (dest)

Status Affected: OV, C, DC, Z

Encoding:

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

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `ADDWFC REG 0`

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

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

**ANDLW** **And Literal with WREG**

Syntax: [ *label* ] ANDLW k

Operands: $0 \le k \le 255$

Operation: (WREG) .AND. (k) → (WREG)

Status Affected: Z

Encoding:

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

Description: The contents of WREG are AND'ed with the 8-bit literal 'k'. The result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to WREG |

Example: `ANDLW 0x5F`

Before Instruction
WREG = 0xA3

After Instruction
WREG = 0x03

**ANDWF** **AND WREG with f**

Syntax: [ *label* ] ANDWF f,d

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

Operation: (WREG) .AND. (f) → (dest)

Status Affected: Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 101d | ffff | ffff |

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `ANDWF REG, 1`

Before Instruction
WREG = 0x17
REG = 0xC2

After Instruction
WREG = 0x17
REG = 0x02

**BCF** **Bit Clear f**

Syntax: [ *label* ] BCF f,b

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

Operation: 0 → (f<b>)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1000 | 1bbb | ffff | ffff |

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: `BCF FLAG_REG, 7`

Before Instruction
FLAG_REG = 0xC7

After Instruction
FLAG_REG = 0x47


© 2000 Microchip Technology Inc.

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

Syntax: [ *label* ] BSF f,b

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7

Operation: 1 → (f<b>)

Status Affected: None

Encoding:

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

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: `BSF FLAG_REG, 7`

Before Instruction
FLAG_REG = 0x0A

After Instruction
FLAG_REG = 0x8A

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

Syntax: [ *label* ] BTFSC f,b

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7

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

Status Affected: None

Encoding:

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

Description: If bit 'b' in register 'f' is 0, then the next instruction is skipped.
If bit 'b' is 0, then the next instruction fetched during the current instruction execution is discarded and a `NOP` is executed instead, making this a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

Example:
```
HERE    BTFSC   FLAG,1
FALSE   :
TRUE    :
```

Before Instruction
PC = address `(HERE)`

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

**BTFSS** **Bit Test, skip if Set**

Syntax: [ *label* ] BTFSS f,b

Operands: 0 ≤ f ≤ 127
0 ≤ b < 7

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

Status Affected: None

Encoding:

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

Description: If bit 'b' in register 'f' is 1, then the next instruction is skipped.
If bit 'b' is 1, then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

Example:
```
HERE    BTFSS   FLAG,1
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)

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

**BTG** **Bit Toggle f**

Syntax: [ *label* ] BTG f,b

Operands: 0 ≤ f ≤ 255
0 ≤ b < 7

Operation: ($\overline{f<b>}$) → (f<b>)

Status Affected: None

Encoding:

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

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example:
```
BTG     PORTC,  4
```

Before Instruction:
PORTC = 0111 0101 [0x75]

After Instruction:
PORTC = 0110 0101 [0x65]


© 2000 Microchip Technology Inc.

**CALL** **Subroutine Call**

Syntax: [ *label* ] CALL k

Operands: $0 \le k \le 8191$

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

Status Affected: None

Encoding:

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

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

See LCALL for calls outside 8K memory space.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k'<7:0>, Push PC to stack | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

Example: HERE CALL THERE

Before Instruction
PC = Address(HERE)

After Instruction
PC = Address(THERE)
TOS = Address (HERE + 1)

**CLRF** **Clear f**

Syntax: [*label*] CLRF f,s

Operands: $0 \le f \le 255$

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

Status Affected: None

Encoding:

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

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' and if specified WREG |

Example: CLRF FLAG_REG, 1

Before Instruction
FLAG_REG = 0x5A
WREG = 0x01

After Instruction
FLAG_REG = 0x00
WREG = 0x01

**CLRWDT** **Clear Watchdog Timer**

Syntax: [ *label* ] CLRWDT

Operands: None

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

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

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0100 |

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | No operation |

Example: CLRWDT

Before Instruction
WDT counter = ?

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

**COMF** **Complement f**

Syntax: [ *label* ] COMF f,d

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

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

Status Affected: Z

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 001d | ffff | ffff |

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: COMF REG1,0

Before Instruction
REG1 = 0x13

After Instruction
REG1 = 0x13
WREG = 0xEC


© 2000 Microchip Technology Inc.

**CPFSEQ** — **Compare f with WREG, skip if f = WREG**

Syntax: [ *label* ] CPFSEQ f

Operands: $0 \le f \le 255$

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

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 0001 | ffff | ffff |

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

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

Words: 1

Cycles: 1 (2)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

Example:

```
HERE    CPFSEQ REG
NEQUAL  :
EQUAL   :
```

Before Instruction
PC Address = HERE
WREG = ?
REG = ?

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

**CPFSGT** — **Compare f with WREG, skip if f > WREG**

Syntax: [ *label* ] CPFSGT f

Operands: $0 \le f \le 255$

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

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 0010 | ffff | ffff |

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

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

Words: 1

Cycles: 1 (2)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

Example:

```
HERE      CPFSGT REG
NGREATER  :
GREATER   :
```

Before Instruction
PC = Address (HERE)
WREG = ?

After Instruction
If REG > WREG;
PC = Address (GREATER)
If REG £ WREG;
PC = Address (NGREATER)

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

Syntax: [ *label* ] CPFSLT f

Operands: $0 \le f \le 255$

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

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 0000 | ffff | ffff |

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

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

Words: 1

Cycles: 1 (2)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

Example:

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

Before Instruction
PC = Address (HERE)
W = ?

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

**DAW** — **Decimal Adjust WREG Register**

Syntax: [*label*] DAW f,s

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

Operation:
If [ [WREG<7:4> > 9].OR.[C = 1] ].AND. [WREG<3:0> > 9]
then
WREG<7:4> + 7→ f<7:4>, s<7:4>;

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

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

Status Affected: C

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 111s | ffff | ffff |

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

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

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' and other specified register |

Example:

```
DAW   REG1, 0
```

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

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


© 2000 Microchip Technology Inc.

**DECF** **Decrement f**

Syntax: [ *label* ] DECF f,d

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

Operation: (f) – 1 → (dest)

Status Affected: OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 011d | ffff | ffff |

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `DECF CNT, 1`

Before Instruction
CNT = 0x01
Z = 0

After Instruction
CNT = 0x00
Z = 1

**DECFSZ** **Decrement f, skip if 0**

Syntax: [ *label* ] DECFSZ f,d

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

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

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 011d | ffff | ffff |

Description: The contents of register 'f' are decremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f'.

If the result is 0, the next instruction, which is already fetched is discarded and a `NOP` is executed instead, making it a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

Example:
```
HERE    DECFSZ  CNT, 1
        GOTO    HERE
NZERO
ZERO
```

Before Instruction
PC = Address `(HERE)`

After Instruction
CNT = CNT - 1
If CNT = 0;
PC = Address `(HERE)`
If CNT ≠ 0;
PC = Address `(NZERO)`

© 2000 Microchip Technology Inc.

**DCFSNZ** **Decrement f, skip if not 0**

Syntax: [*label*] DCFSNZ f,d

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

Operation: (f) – 1 → (dest);
skip if not 0

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 011d | ffff | ffff |

Description: The contents of register 'f' are decremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f'.

If the result is not 0, the next instruction, which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

Example:
```
HERE    DCFSNZ  TEMP, 1
ZERO    :
NZERO   :
```

Before Instruction
TEMP_VALUE = ?

After Instruction
TEMP_VALUE = TEMP_VALUE - 1,
If TEMP_VALUE = 0;
PC = Address (ZERO)
If TEMP_VALUE ≠ 0;
PC = Address (NZERO)

**GOTO** **Unconditional Branch**

Syntax: [ *label* ] GOTO k

Operands: $0 \le k \le 8191$

Operation: k → PC<12:0>;
k<12:8> → PCLATH<4:0>,
PC<15:13> → PCLATH<7:5>

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 110k | kkkk | kkkk | kkkk |

Description: GOTO allows an unconditional branch anywhere within an 8K page boundary. The thirteen-bit immediate value is loaded into PC bits <12:0>. Then the upper eight bits of PC are loaded into PCLATH. GOTO is always a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

Example: GOTO THERE

After Instruction
PC = Address (THERE)


© 2000 Microchip Technology Inc.

**INCF** **Increment f**

Syntax: [ *label* ] INCF f,d

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

Operation: (f) + 1 → (dest)

Status Affected: OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 010d | ffff | ffff |

Description: The contents of register 'f' are incremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `INCF CNT, 1`

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

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

**INCFSZ** **Increment f, skip if 0**

Syntax: [ *label* ] INCFSZ f,d

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

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

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 111d | ffff | ffff |

Description: The contents of register 'f' are incremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f'.
If the result is 0, the next instruction, which is already fetched is discarded and a `NOP` is executed instead, making it a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

Example:
```
HERE    INCFSZ  CNT, 1
NZERO   :
ZERO    :
```

Before Instruction
PC = Address `(HERE)`

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

© 2000 Microchip Technology Inc.

**INFSNZ** **Increment f, skip if not 0**

Syntax: [*label*] INFSNZ f,d

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

Operation: (f) + 1 → (dest),
skip if not 0

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 010d | ffff | ffff |

Description: The contents of register 'f' are incremented. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f'.

If the result is not 0, the next instruction, which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

Example:
```
HERE    INFSNZ  REG, 1
ZERO
NZERO
```

Before Instruction
REG = REG

After Instruction
REG = REG + 1
If REG = 1;
PC = Address (ZERO)
If REG = 0;
PC = Address (NZERO)

**IORLW** **Inclusive OR Literal with WREG**

Syntax: [ *label* ] IORLW k

Operands: $0 \le k \le 255$

Operation: (WREG) .OR. (k) → (WREG)

Status Affected: Z

Encoding:

| | | | |
|---|---|---|---|
| 1011 | 0011 | kkkk | kkkk |

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to WREG |

Example:
```
IORLW   0x35
```

Before Instruction
WREG = 0x9A

After Instruction
WREG = 0xBF


© 2000 Microchip Technology Inc.

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

Syntax: [ *label* ] IORWF f,d

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

Operation: (WREG) .OR. (f) → (dest)

Status Affected: $\overline{Z}$

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 100d | ffff | ffff |

Description: Inclusive OR WREG with register 'f'. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `IORWF RESULT, 0`

Before Instruction
RESULT = 0x13
WREG = 0x91

After Instruction
RESULT = 0x13
WREG = 0x93

**LCALL** **Long Call**

Syntax: [ *label* ] LCALL k

Operands: $0 \le k \le 255$

Operation: PC + 1 → TOS;
k → PCL, (PCLATH) → PCH

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1011 | 0111 | kkkk | kkkk |

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

First, the return address (PC + 1) is pushed onto the stack. A 16-bit destination address is then loaded into the program counter. The lower 8-bits of the destination address are embedded in the instruction. The upper 8-bits of PC are loaded from PC high holding latch, PCLATH.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write register PCL |
| No operation | No operation | No operation | No operation |

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

Before Instruction
SUBROUTINE = 16-bit Address
PC = ?

After Instruction
PC = Address (SUBROUTINE)

**MOVFP** **Move f to p**

Syntax: [*label*] MOVFP f,p

Operands: 0 ≤ f ≤ 255
0 ≤ p ≤ 31

Operation: (f) → (p)

Status Affected: None

Encoding:

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

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

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

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'p' |

Example: MOVFP REG1, REG2

Before Instruction
REG1 = 0x33,
REG2 = 0x11

After Instruction
REG1 = 0x33,
REG2 = 0x33

**MOVLB** **Move Literal to low nibble in BSR**

Syntax: [ *label* ] MOVLB k

Operands: 0 ≤ k ≤ 15

Operation: k → (BSR<3:0>)

Status Affected: None

Encoding:

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

Description: The four-bit literal 'k' is loaded in the Bank Select Register (BSR). Only the low 4-bits of the Bank Select Register are affected. The upper half of the BSR is unchanged. The assembler will encode the “u” fields as '0'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write literal 'k' to BSR<3:0> |

Example: MOVLB 5

Before Instruction
BSR register = 0x22

After Instruction
BSR register = 0x25 (Bank 5)


© 2000 Microchip Technology Inc.

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

Syntax: [ *label* ] MOVLR k

Operands: $0 \le k \le 15$

Operation: $k \rightarrow (BSR<7:4>)$

Status Affected: None

Encoding:

| 1011 | 101x | kkkk | uuuu |
|---|---|---|---|

Description: The 4-bit literal 'k' is loaded into the most significant 4-bits of the Bank Select Register (BSR). Only the high 4-bits of the Bank Select Register are affected. The lower half of the BSR is unchanged. The assembler will encode the “u” fields as 0.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write literal 'k' to BSR<7:4> |

Example: `MOVLR 5`

Before Instruction
BSR register = 0x22

After Instruction
BSR register = 0x52

**MOVLW** — **Move Literal to WREG**

Syntax: [ *label* ] MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow (WREG)$

Status Affected: None

Encoding:

| 1011 | 0000 | kkkk | kkkk |
|---|---|---|---|

Description: The eight-bit literal 'k' is loaded into WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to WREG |

Example: `MOVLW 0x5A`

After Instruction
WREG = 0x5A

**MOVPF** **Move p to f**

Syntax: [*label*] MOVPF p,f

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

Operation: (p) → (f)

Status Affected: Z

Encoding:

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

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

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

MOVPF is particularly useful for transferring a peripheral register (e.g. the timer or an I/O port) to a data memory location. Both 'f' and 'p' can be indirectly addressed.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'p' | Process Data | Write register 'f' |

Example: MOVPF REG1, REG2

Before Instruction
REG1 = 0x11
REG2 = 0x33

After Instruction
REG1 = 0x11
REG2 = 0x11

**MOVWF** **Move WREG to f**

Syntax: [ *label* ] MOVWF f

Operands: $0 \le f \le 255$

Operation: (WREG) → (f)

Status Affected: None

Encoding:

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

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: MOVWF REG

Before Instruction
WREG = 0x4F
REG = 0xFF

After Instruction
WREG = 0x4F
REG = 0x4F


© 2000 Microchip Technology Inc.

**MULLW** **Multiply Literal with WREG**

Syntax: [ *label* ] MULLW k

Operands: $0 \le k \le 255$

Operation: (k x WREG) → PRODH:PRODL

Status Affected: None

Encoding:

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

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write registers PRODH: PRODL |

Example: `MULLW 0xC4`

Before Instruction
WREG = 0xE2
PRODH = ?
PRODL = ?

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

**MULWF** **Multiply WREG with f**

Syntax: [ *label* ] MULWF f

Operands: $0 \le f \le 255$

Operation: (WREG x f) → PRODH:PRODL

Status Affected: None

Encoding:

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

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write registers PRODH: PRODL |

Example: `MULWF REG`

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

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

© 2000 Microchip Technology Inc.

**NEGW** **Negate W**

Syntax: [*label*] NEGW f,s

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

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

Status Affected: OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 110s | ffff | ffff |

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' and other specified register |

Example: `NEGW REG,0`

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

After Instruction
WREG = 1100 0110 [0xC6]
REG = 1100 0110 [0xC6]

**NOP** **No Operation**

Syntax: [ *label* ] NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0000 |

Description: No operation.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |

Example:

None.


© 2000 Microchip Technology Inc.

**RETFIE** **Return from Interrupt**

Syntax: [ *label* ] RETFIE

Operands: None

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

Status Affected: GLINTD

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0101 |

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

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Clear GLINTD | POP PC from stack |
| No operation | No operation | No operation | No operation |

Example: RETFIE

After Interrupt
PC = TOS
GLINTD = 0

**RETLW** **Return Literal to WREG**

Syntax: [ *label* ] RETLW k

Operands: 0 ≤ k ≤ 255

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

Status Affected: None

Encoding:

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

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

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | POP PC from stack, Write to WREG |
| No operation | No operation | No operation | No operation |

Example:

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

Before Instruction
WREG = 0x07

After Instruction
WREG = value of k7

**RETURN** **Return from Subroutine**

Syntax: [ *label* ] RETURN

Operands: None

Operation: TOS → PC;

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0010 |

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

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | POP PC from stack |
| No operation | No operation | No operation | No operation |

Example: `RETURN`

After Interrupt
PC = TOS

**RLCF** **Rotate Left f through Carry**

Syntax: [ *label* ] RLCF f,d

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

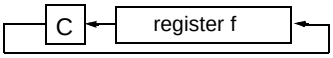
Operation: f<n> → d<n+1>;
f<7> → C;
C → d<0>

Status Affected: C

Encoding:

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

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is stored back in register 'f'.




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `RLCF REG,0`

Before Instruction
REG = 1110 0110
C = 0

After Instruction
REG = 1110 0110
WREG = 1100 1100
C = 1


© 2000 Microchip Technology Inc.

**RLNCF** **Rotate Left f (no carry)**

Syntax: [ *label* ] RLNCF f,d

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

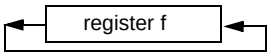
Operation: f<n> → d<n+1>;
f<7> → d<0>

Status Affected: None

Encoding:

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

Description: The contents of register 'f' are rotated one bit to the left. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is stored back in register 'f'.




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: RLNCF REG, 1

Before Instruction
C = 0
REG = 1110 1011

After Instruction
C =
REG = 1101 0111

**RRCF** **Rotate Right f through Carry**

Syntax: [ *label* ] RRCF f,d

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

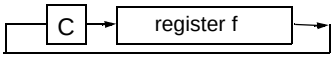
Operation: f<n> → d<n-1>;
f<0> → C;
C → d<7>

Status Affected: C

Encoding:

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

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f'.




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: RRCF REG1,0

Before Instruction
REG1 = 1110 0110
C = 0

After Instruction
REG1 = 1110 0110
WREG = 0111 0011
C = 0

**RRNCF** **Rotate Right f (no carry)**

Syntax: [ *label* ] RRNCF f,d

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

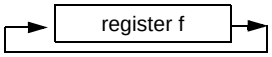
Operation: f<n> → d<n-1>;
f<0> → d<7>

Status Affected: None

Encoding:

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

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f'.




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1: `RRNCF REG, 1`

Before Instruction
WREG = ?
REG = 1101 0111

After Instruction
WREG = 0
REG = 1110 1011

Example 2: `RRNCF REG, 0`

Before Instruction
WREG = ?
REG = 1101 0111

After Instruction
WREG = 1110 1011
REG = 1101 0111

**SETF** **Set f**

Syntax: [ *label* ] SETF f,s

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

Operation: FFh → f;
FFh → d

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 101s | ffff | ffff |

Description: If 's' is 0, both the data memory location 'f' and WREG are set to FFh. If 's' is 1, only the data memory location 'f' is set to FFh.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' and other specified register |

Example1: `SETF REG, 0`

Before Instruction
REG = 0xDA
WREG = 0x05

After Instruction
REG = 0xFF
WREG = 0xFF

Example2: `SETF REG, 1`

Before Instruction
REG = 0xDA
WREG = 0x05

After Instruction
REG = 0xFF
WREG = 0x05


© 2000 Microchip Technology Inc.

**SLEEP** **Enter SLEEP mode**

Syntax: [ *label* ] SLEEP

Operands: None

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

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

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0011 |

Description: The power-down status bit ($\overline{PD}$) is cleared. The time-out status bit ($\overline{TO}$) is set. Watchdog Timer and its postscaler are cleared.
The processor is put into SLEEP mode with the oscillator stopped.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | Go to sleep |

Example: `SLEEP`

Before Instruction
$\overline{TO}$ = ?
$\overline{PD}$ = ?

After Instruction
$\overline{TO}$ = 1 †
$\overline{PD}$ = 0

† If WDT causes wake-up, this bit is cleared

**SUBLW** **Subtract WREG from Literal**

Syntax: [ *label* ] SUBLW k

Operands: 0 ≤ k ≤ 255

Operation: k – (WREG) → (WREG)

Status Affected: OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 1011 | 0010 | kkkk | kkkk |

Description: WREG is subtracted from the eight-bit literal 'k'. The result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to WREG |

Example 1: `SUBLW 0x02`

Before Instruction
WREG = 1
C = ?

After Instruction
WREG = 1
C = 1 ; result is positive
Z = 0

Example 2:

Before Instruction
WREG = 2
C = ?

After Instruction
WREG = 0
C = 1 ; result is zero
Z = 1

Example 3:

Before Instruction
WREG = 3
C = ?

After Instruction
WREG = FF ; (2's complement)
C = 0 ; result is negative
Z = 0

© 2000 Microchip Technology Inc.

**SUBWF** **Subtract WREG from f**

Syntax: [ *label* ] SUBWF f,d

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

Operation: (f) – (W) → (dest)

Status Affected: OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 010d | ffff | ffff |

Description: Subtract WREG from register 'f' (2's complement method). If 'd' is 0, the result is stored in WREG. If 'd' is 1, the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1: `SUBWF REG1, 1`

Before Instruction
REG1 = 3
WREG = 2
C = ?

After Instruction
REG1 = 1
WREG = 2
C = 1 ; result is positive
Z = 0

Example 2:

Before Instruction
REG1 = 2
WREG = 2
C = ?

After Instruction
REG1 = 0
WREG = 2
C = 1 ; result is zero
Z = 1

Example 3:

Before Instruction
REG1 = 1
WREG = 2
C = ?

After Instruction
REG1 = FF
WREG = 2
C = 0 ; result is negative
Z = 0

**SUBWFB** **Subtract WREG from f with Borrow**

Syntax: [ *label* ] SUBWFB f,d

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

Operation: (f) – (W) – $\overline{C}$ → (dest)

Status Affected: OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 001d | ffff | ffff |

Description: Subtract WREG and the carry flag (borrow) from register 'f' (2's complement method). If 'd' is 0, the result is stored in WREG. If 'd' is 1, the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1: `SUBWFB REG1, 1`

Before Instruction
REG1 = 0x19 (0001 1001)
WREG = 0x0D (0000 1101)
C = 1

After Instruction
REG1 = 0x0C (0000 1011)
WREG = 0x0D (0000 1101)
C = 1 ; result is positive
Z = 0

Example2: `SUBWFB REG1,0`

Before Instruction
REG1 = 0x1B (0001 1011)
WREG = 0x1A (0001 1010)
C = 0

After Instruction
REG1 = 0x1B (0001 1011)
WREG = 0x00
C = 1 ; result is zero
Z = 1

Example3: `SUBWFB REG1,1`

Before Instruction
REG1 = 0x03 (0000 0011)
WREG = 0x0E (0000 1101)
C = 1

After Instruction
REG1 = 0xF5 (1111 0100) [2's comp]
WREG = 0x0E (0000 1101)
C = 0 ; result is negative
Z = 0


© 2000 Microchip Technology Inc.

**SWAPF** **Swap f**

Syntax: [ *label* ] SWAPF f,d

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

Operation: f<3:0> → dest<7:4>;
f<7:4> → dest<3:0>

Status Affected: None

Encoding:

| 0001 | 110d | ffff | ffff |
|---|---|---|---|

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `SWAPF REG, 0`

Before Instruction
REG = 0x53

After Instruction
REG = 0x35

**TABLRD** **Table Read**

Syntax: [ *label* ] TABLRD t,i,f

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

Operation: If t = 1,
TBLATH → f;
If t = 0,
TBLATL → f;
Prog Mem (TBLPTR) → TBLAT;
If i = 1,
TBLPTR + 1 → TBLPTR
If i = 0,
TBLPTR is unchanged

Status Affected: None

Encoding:

| 1010 | 10ti | ffff | ffff |
|---|---|---|---|

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

Words: 1

Cycles: 2 (3-cycle if f = PCL)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register TBLATH or TBLATL | Process Data | Write register 'f' |
| No operation | No operation (Table Pointer on Address bus) | No operation | No operation (OE goes low) |

**TABLRD** **Table Read**

Example1: `TABLRD 1, 1, REG ;`

Before Instruction
- REG = 0x53
- TBLATH = 0xAA
- TBLATL = 0x55
- TBLPTR = 0xA356
- MEMORY(TBLPTR) = 0x1234

After Instruction (table write completion)
- REG = 0xAA
- TBLATH = 0x12
- TBLATL = 0x34
- TBLPTR = 0xA357
- MEMORY(TBLPTR) = 0x5678

Example2: `TABLRD 0, 0, REG ;`

Before Instruction
- REG = 0x53
- TBLATH = 0xAA
- TBLATL = 0x55
- TBLPTR = 0xA356
- MEMORY(TBLPTR) = 0x1234

After Instruction (table write completion)
- REG = 0x55
- TBLATH = 0x12
- TBLATL = 0x34
- TBLPTR = 0xA356
- MEMORY(TBLPTR) = 0x1234

**TABLWT** **Table Write**

Syntax: [ *label* ] TABLWT t,i,f

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

Operation: If t = 0,
f → TBLATL;
If t = 1,
f → TBLATH;
TBLAT → Prog Mem (TBLPTR);
If i = 1,
TBLPTR + 1 → TBLPTR
If i = 0,
TBLPTR is unchanged

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1010 | 11ti | ffff | ffff |

Description:
1. Load value in 'f' into 16-bit table latch (TBLAT)
   If t = 1: load into high byte;
   If t = 0: load into low byte
2. The contents of TBLAT are written to the program memory location pointed to by TBLPTR.
   If TBLPTR points to external program memory location, then the instruction takes two-cycle.
   If TBLPTR points to an internal EPROM location, then the instruction is terminated when an interrupt is received.

**Note:** The $\overline{MCLR}/V_{PP}$ pin must be at the programming voltage for successful programming of internal memory.
If $\overline{MCLR}/V_{PP} = V_{DD}$
the programming sequence of internal memory will be interrupted. A short write will occur (2 $T_{CY}$). The internal memory location will not be affected.

3. The TBLPTR can be automatically incremented
   If i = 1; TBLPTR is not incremented
   If i = 0; TBLPTR is incremented

Words: 1

Cycles: 2 (many if write is to on-chip EPROM program memory)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register TBLATH or TBLATL |
| No operation | No operation (Table Pointer on Address bus) | No operation | No operation (Table Latch on Address bus, $\overline{WR}$ goes low) |


© 2000 Microchip Technology Inc.

**TABLWT** **Table Write**

Example1: `TABLWT 1, 1, REG`

Before Instruction

| | | |
|---|---|---|
| REG | = | 0x53 |
| TBLATH | = | 0xAA |
| TBLATL | = | 0x55 |
| TBLPTR | = | 0xA356 |
| MEMORY(TBLPTR) | = | 0xFFFF |

After Instruction (table write completion)

| | | |
|---|---|---|
| REG | = | 0x53 |
| TBLATH | = | 0x53 |
| TBLATL | = | 0x55 |
| TBLPTR | = | 0xA357 |
| MEMORY(TBLPTR - 1) | = | 0x5355 |

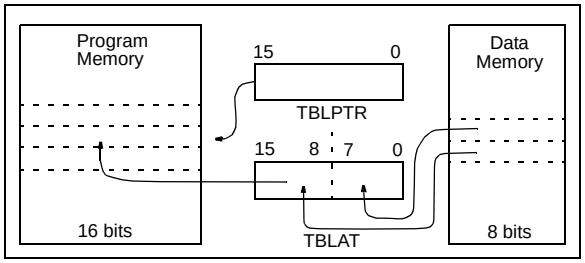
Example 2: `TABLWT 0, 0, REG`

Before Instruction

| | | |
|---|---|---|
| REG | = | 0x53 |
| TBLATH | = | 0xAA |
| TBLATL | = | 0x55 |
| TBLPTR | = | 0xA356 |
| MEMORY(TBLPTR) | = | 0xFFFF |

After Instruction (table write completion)

| | | |
|---|---|---|
| REG | = | 0x53 |
| TBLATH | = | 0xAA |
| TBLATL | = | 0x53 |
| TBLPTR | = | 0xA356 |
| MEMORY(TBLPTR) | = | 0xAA53 |




**TLRD** **Table Latch Read**

Syntax: [ *label* ] TLRD t,f

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

Operation: If t = 0,
TBLATL → f;
If t = 1,
TBLATH → f

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1010 | 00tx | ffff | ffff |

Description: Read data from 16-bit table latch (TBLAT) into file register 'f'. Table Latch is unaffected.

If t = 1; high byte is read

If t = 0; low byte is read

This instruction is used in conjunction with `TABLRD` to transfer data from program memory to data memory.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register TBLATH or TBLATL | Process Data | Write register 'f' |

Example: `TLRD t, RAM`

Before Instruction

| | | | |
|---|---|---|---|
| t | = | 0 | |
| RAM | = | ? | |
| TBLAT | = | 0x00AF | (TBLATH = 0x00) |
| | | | (TBLATL = 0xAF) |

After Instruction

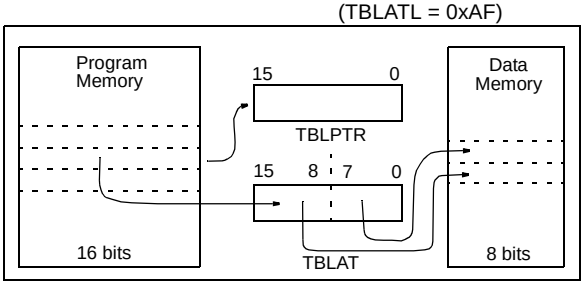
| | | | |
|---|---|---|---|
| RAM | = | 0xAF | |
| TBLAT | = | 0x00AF | (TBLATH = 0x00) |
| | | | (TBLATL = 0xAF) |

Before Instruction

| | | | |
|---|---|---|---|
| t | = | 1 | |
| RAM | = | ? | |
| TBLAT | = | 0x00AF | (TBLATH = 0x00) |
| | | | (TBLATL = 0xAF) |

After Instruction

| | | | |
|---|---|---|---|
| RAM | = | 0x00 | |
| TBLAT | = | 0x00AF | (TBLATH = 0x00) |
| | | | (TBLATL = 0xAF) |




© 2000 Microchip Technology Inc.

**TLWT** **Table Latch Write**

Syntax: [ *label* ] TLWT t,f

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

Operation: If t = 0,
f → TBLATL;
If t = 1,
f → TBLATH

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1010 | 01tx | ffff | ffff |

Description: Data from file register 'f' is written into the 16-bit table latch (TBLAT).
If t = 1; high byte is written
If t = 0; low byte is written
This instruction is used in conjunction with TABLWT to transfer data from data memory to program memory.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register TBLATH or TBLATL |

Example: TLWT t, RAM

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

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

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

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

**TSTFSZ** **Test f, skip if 0**

Syntax: [ *label* ] TSTFSZ f

Operands: $0 \le f \le 255$

Operation: skip if f = 0

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 0011 | ffff | ffff |

Description: If 'f' = 0, the next instruction, fetched during the current instruction execution, is discarded and a NOP is executed, making this a two-cycle instruction.

Words: 1

Cycles: 1 (2)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

Example:
```
HERE    TSTFSZ  CNT
NZERO     :
ZERO      :
```

Before Instruction
PC = Address (HERE)

After Instruction
If CNT = 0x00,
PC = Address (ZERO)
If CNT ¼ 0x00,
PC = Address (NZERO)


© 2000 Microchip Technology Inc.

**XORLW** **Exclusive OR Literal with WREG**

Syntax: [ *label* ] XORLW k

Operands: $0 \le k \le 255$

Operation: (WREG) .XOR. k → (WREG)

Status Affected: Z

Encoding:

| | | | |
|---|---|---|---|
| 1011 | 0100 | kkkk | kkkk |

Description: The contents of WREG are XOR'ed with the 8-bit literal 'k'. The result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to WREG |

Example: `XORLW 0xAF`

Before Instruction
WREG = 0xB5

After Instruction
WREG = 0x1A

**XORWF** **Exclusive OR WREG with f**

Syntax: [ *label* ] XORWF f,d

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

Operation: (WREG) .XOR. (f) → (dest)

Status Affected: Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 110d | ffff | ffff |

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

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `XORWF REG, 1`

Before Instruction
REG = 0xAF 1010 1111
WREG = 0xB5 1011 0101

After Instruction
REG = 0x1A 0001 1010
WREG = 0xB5

**NOTES:**


© 2000 Microchip Technology Inc.

## 19.0 DEVELOPMENT SUPPORT

The PICmicro® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM™ Assembler
  - MPLAB C17 and MPLAB C18 C Compilers
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - ICEPIC™ In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD for PIC16F87X
- Device Programmers
  - PRO MATE® II Universal Device Programmer
  - PICSTART® Plus Entry-Level Development Programmer
- Low Cost Demonstration Boards
  - PICDEM™ 1 Demonstration Board
  - PICDEM 2 Demonstration Board
  - PICDEM 3 Demonstration Board
  - PICDEM 17 Demonstration Board
  - KEELOQ® Demonstration Board

### 19.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8-bit microcontroller market. The MPLAB IDE is a Windows®-based application that contains:

- An interface to debugging tools
  - simulator
  - programmer (sold separately)
  - emulator (sold separately)
  - in-circuit debugger (sold separately)
- A full-featured editor
- A project manager
- Customizable toolbar and key mapping
- A status bar
- On-line help

The MPLAB IDE allows you to:

- Edit your source files (either assembly or ‘C’)
- One touch assemble (or compile) and download to PICmicro emulator and simulator tools (automatically updates all project information)
- Debug using:
  - source files
  - absolute listing file
  - machine code

The ability to use MPLAB IDE with multiple debugging tools allows users to easily switch from the cost-effective simulator to a full-featured emulator with minimal retraining.

### 19.2 MPASM Assembler

The MPASM assembler is a full-featured universal macro assembler for all PICmicro MCU’s.

The MPASM assembler has a command line interface and a Windows shell. It can be used as a stand-alone application on a Windows 3.x or greater system, or it can be used through MPLAB IDE. The MPASM assembler generates relocatable object files for the MPLINK object linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, an absolute LST file that contains source lines and generated machine code, and a COD file for debugging.

The MPASM assembler features include:

- Integration into MPLAB IDE projects.
- User-defined macros to streamline assembly code.
- Conditional assembly for multi-purpose source files.
- Directives that allow complete control over the assembly process.

### 19.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI ‘C’ compilers for Microchip’s PIC17CXXX and PIC18CXXX family of microcontrollers, respectively. These compilers provide powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compilers provide symbol information that is compatible with the MPLAB IDE memory display.

© 2000 Microchip Technology Inc.

## 19.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK object linker combines relocatable objects created by the MPASM assembler and the MPLAB C17 and MPLAB C18 C compilers. It can also link relocatable objects from pre-compiled libraries, using directives from a linker script.

The MPLIB object librarian is a librarian for pre-compiled code to be used with the MPLINK object linker. When a routine from a library is called from another source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications. The MPLIB object librarian manages the creation and modification of library files.

The MPLINK object linker features include:

- Integration with MPASM assembler and MPLAB C17 and MPLAB C18 C compilers.
- Allows all memory areas to be defined as sections to provide link-time flexibility.

The MPLIB object librarian features include:

- Easier linking because single libraries can be included instead of many smaller files.
- Helps keep code maintainable by grouping related modules together.
- Allows libraries to be created and modules to be added, listed, replaced, deleted or extracted.

## 19.5 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC-hosted environment by simulating the PICmicro series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user-defined key press, to any of the pins. The execution can be performed in single step, execute until break, or trace mode.

The MPLAB SIM simulator fully supports symbolic debugging using the MPLAB C17 and the MPLAB C18 C compilers and the MPASM assembler. The software simulator offers the flexibility to develop and debug code outside of the laboratory environment, making it an excellent multi-project software development tool.

## 19.6 MPLAB ICE High Performance Universal In-Circuit Emulator with MPLAB IDE

The MPLAB ICE universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PICmicro microcontrollers (MCUs). Software control of the MPLAB ICE in-circuit emulator is provided by the MPLAB Integrated Development Environment (IDE), which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the MPLAB ICE in-circuit emulator allows expansion to support new PICmicro microcontrollers.

The MPLAB ICE in-circuit 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 platform and Microsoft® Windows environment were chosen to best make these features available to you, the end user.

## 19.7 ICEPIC In-Circuit Emulator

The ICEPIC low cost, in-circuit emulator is a solution for the Microchip Technology PIC16C5X, PIC16C6X, PIC16C7X and PIC16CXXX families of 8-bit One-Time-Programmable (OTP) microcontrollers. The modular system can support different subsets of PIC16C5X or PIC16CXXX products through the use of interchangeable personality modules, or daughter boards. The emulator is capable of emulating without target application circuitry being present.


© 2000 Microchip Technology Inc.

## 19.8 MPLAB ICD In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD, is a powerful, low cost, run-time development tool. This tool is based on the FLASH PIC16F87X and can be used to develop for this and other PICmicro microcontrollers from the PIC16CXXX family. The MPLAB ICD utilizes the in-circuit debugging capability built into the PIC16F87X. This feature, along with Microchip's In-Circuit Serial Programming™ protocol, offers cost-effective in-circuit FLASH debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by watching variables, single-stepping and setting break points. Running at full speed enables testing hardware in real-time.

## 19.9 PRO MATE II Universal Device Programmer

The PRO MATE II universal device programmer is a full-featured programmer, capable of operating in stand-alone mode, as well as PC-hosted mode. The PRO MATE II device programmer is CE compliant.

The PRO MATE II device programmer has programmable $V_{DD}$ and $V_{PP}$ supplies, which allow it to verify programmed memory at $V_{DD}$ min and $V_{DD}$ max for maximum reliability. It has an LCD display for instructions and 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 device programmer can read, verify, or program PICmicro devices. It can also set code protection in this mode.

## 19.10 PICSTART Plus Entry Level Development Programmer

The PICSTART Plus development programmer is an easy-to-use, low cost, prototype programmer. It connects to the PC via a COM (RS-232) port. MPLAB Integrated Development Environment software makes using the programmer simple and efficient.

The PICSTART Plus development programmer supports all PICmicro devices with up to 40 pins. Larger pin count devices, such as the PIC16C92X and PIC17C76X, may be supported with an adapter socket. The PICSTART Plus development programmer is CE compliant.

## 19.11 PICDEM 1 Low Cost PICmicro Demonstration Board

The PICDEM 1 demonstration board 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 user can program the sample microcontrollers provided with the PICDEM 1 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer, and easily test firmware. The user can also connect the PICDEM 1 demonstration board to the MPLAB ICE in-circuit emulator and download the firmware to the emulator for testing. A 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.

## 19.12 PICDEM 2 Low Cost PIC16CXX Demonstration Board

The PICDEM 2 demonstration board 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 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer, and easily test firmware. The MPLAB ICE in-circuit emulator may also be used with the PICDEM 2 demonstration board to test firmware. A 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 $I^2C$™ bus and separate headers for connection to an LCD module and a keypad.

© 2000 Microchip Technology Inc.

## 19.13 PICDEM 3 Low Cost PIC16CXXX Demonstration Board

The PICDEM 3 demonstration board 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 an 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 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer with an adapter socket, and easily test firmware. The MPLAB ICE in-circuit emulator may also be used with the PICDEM 3 demonstration board to test firmware. A prototype area has been provided to the user for adding 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 thermistor and separate headers for connection to an external LCD module and a keypad. Also provided on the PICDEM 3 demonstration board is a LCD panel, with 4 commons and 12 segments, that is capable of displaying time, temperature and day of the week. The PICDEM 3 demonstration board provides an additional RS-232 interface and Windows 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.

## 19.14 PICDEM 17 Demonstration Board

The PICDEM 17 demonstration board is an evaluation board that demonstrates the capabilities of several Microchip microcontrollers, including PIC17C752, PIC17C756A, PIC17C762 and PIC17C766. All necessary hardware is included to run basic demo programs, which are supplied on a 3.5-inch disk. A programmed sample is included and the user may erase it and program it with the other sample programs using the PRO MATE II device programmer, or the PICSTART Plus development programmer, and easily debug and test the sample code. In addition, the PICDEM 17 demonstration board supports downloading of programs to and executing out of external FLASH memory on board. The PICDEM 17 demonstration board is also usable with the MPLAB ICE in-circuit emulator, or the PICMASTER emulator and all of the sample programs can be run and modified using either emulator. Additionally, a generous prototype area is available for user hardware.

## 19.15 KEELOQ Evaluation and Programming Tools

KEELOQ evaluation and programming tools support Microchip's HCS Secure Data Products. The HCS evaluation kit includes a LCD display to show changing codes, a decoder to decode transmissions and a programming interface to program test transmitters.


© 2000 Microchip Technology Inc.

**TABLE 19-1: DEVELOPMENT TOOLS FROM MICROCHIP**

| | | PIC12CXXX | PIC14000 | PIC16C5X | PIC16C6X | PIC16CXXX | PIC16F62X | PIC16C7X | PIC16C7XX | PIC16C8X | PIC16F8XX | PIC16C9XX | PIC17C4X | PIC17C7XX | PIC18CXX2 | 24CXX/ 25CXX/ 93CXX | HCSXXX | MCRFXXX | MCP2510 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Software Tools | MPLAB® Integrated Development Environment | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | | |
| | MPLAB® C17 C Compiler | | | | | | | | | | | | ✓ | ✓ | | | | | |
| | MPLAB® C18 C Compiler | | | | | | | | | | | | | | ✓ | | | | |
| | MPASM™ Assembler/ MPLINK™ Object Linker | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| Emulators | MPLAB® ICE In-Circuit Emulator | ✓ | ✓ | ✓ | ✓ | ✓ | ✓** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | | |
| | ICEPIC™ In-Circuit Emulator | ✓ | | ✓ | ✓ | ✓ | | ✓ | ✓ | | | ✓ | | | | | | | |
| Debugger | MPLAB® ICD In-Circuit Debugger | | | | ✓* | | | ✓* | | | ✓ | | | | | | | | |
| Programmers | PICSTART® Plus Entry Level Development Programmer | ✓ | ✓ | ✓ | ✓ | ✓ | ✓** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | | |
| | PRO MATE® II Universal Device Programmer | ✓ | ✓ | ✓ | ✓ | ✓ | ✓** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | | |
| Demo Boards and Eval Kits | PICDEM™ 1 Demonstration Board | | | ✓ | | ✓ | | ✓† | | ✓ | | | ✓ | | | | | | |
| | PICDEM™ 2 Demonstration Board | | | | ✓† | | | ✓† | | | | | | | ✓ | | | | |
| | PICDEM™ 3 Demonstration Board | | | | | | | | | | | ✓ | | | | | | | |
| | PICDEM™ 14A Demonstration Board | | ✓ | | | | | | | | | | | | | | | | |
| | PICDEM™ 17 Demonstration Board | | | | | | | | | | | | | ✓ | | | | | |
| | KEELOQ® Evaluation Kit | | | | | | | | | | | | | | | | ✓ | | |
| | KEELOQ® Transponder Kit | | | | | | | | | | | | | | | | ✓ | | |
| | microID™ Programmer's Kit | | | | | | | | | | | | | | | | | ✓ | |
| | 125 kHz microID™ Developer's Kit | | | | | | | | | | | | | | | | | ✓ | |
| | 125 kHz Anticollision microID™ Developer's Kit | | | | | | | | | | | | | | | | | ✓ | |
| | 13.56 MHz Anticollision microID™ Developer's Kit | | | | | | | | | | | | | | | | | ✓ | |
| | MCP2510 CAN Developer's Kit | | | | | | | | | | | | | | | | | | ✓ |

* Contact the Microchip Technology Inc. web site at www.microchip.com for information on how to use the MPLAB® ICD In-Circuit Debugger (DV164001) with PIC16C62, 63, 64, 65, 72, 73, 74, 76, 77.

** Contact Microchip Technology Inc. for availability date.

† Development tool is available on select devices.

**NOTES:**


© 2000 Microchip Technology Inc.

## 20.0 PIC17C7XX ELECTRICAL CHARACTERISTICS

**Absolute Maximum Ratings †**

Ambient temperature under bias ..... -55°C to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on $V_{DD}$ with respect to $V_{SS}$ ..... 0 V to +7.5 V
Voltage on $\overline{MCLR}$ with respect to $V_{SS}$ **(Note 2)** ..... -0.3 V to +14 V
Voltage on RA2 and RA3 with respect to $V_{SS}$ ..... -0.3 V to +8.5 V
Voltage on all other pins with respect to $V_{SS}$ ..... -0.3 V to $V_{DD}$ + 0.3 V
Total power dissipation **(Note 1)** ..... 1.0 W
Maximum current out of $V_{SS}$ pin(s) - total (@ 70°C) ..... 500 mA
Maximum current into $V_{DD}$ pin(s) - total (@ 70°C) ..... 500 mA
Input clamp current, $I_{IK}$ ($V_I$ < 0 or $V_I$ > $V_{DD}$) ..... ±20 mA
Output clamp current, $I_{OK}$ ($V_O$ < 0 or $V_O$ > $V_{DD}$) ..... ±20 mA
Maximum output current sunk by any I/O pin (except RA2 and RA3) ..... 35 mA
Maximum output current sunk by RA2 or RA3 pins ..... 60 mA
Maximum output current sourced by any I/O pin ..... 20 mA
Maximum current sunk by PORTA and PORTB (combined) ..... 150 mA
Maximum current sourced by PORTA and PORTB (combined) ..... 100 mA
Maximum current sunk by PORTC, PORTD and PORTE (combined) ..... 150 mA
Maximum current sourced by PORTC, PORTD and PORTE (combined) ..... 100 mA
Maximum current sunk by PORTF and PORTG (combined) ..... 150 mA
Maximum current sourced by PORTF and PORTG (combined) ..... 100 mA
Maximum current sunk by PORTH and PORTJ (combined) ..... 150 mA
Maximum current sourced by PORTH and PORTJ (combined) ..... 100 mA

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

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

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

**FIGURE 20-1: PIC17C7XX-33 VOLTAGE-FREQUENCY GRAPH**

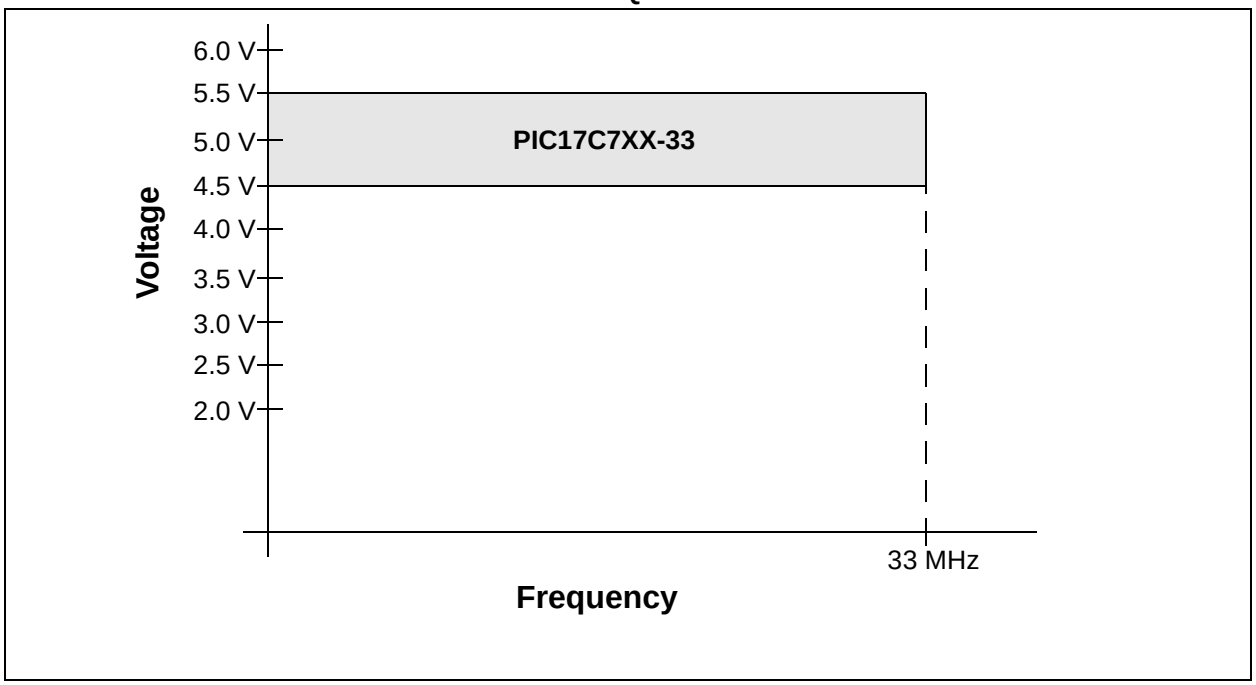
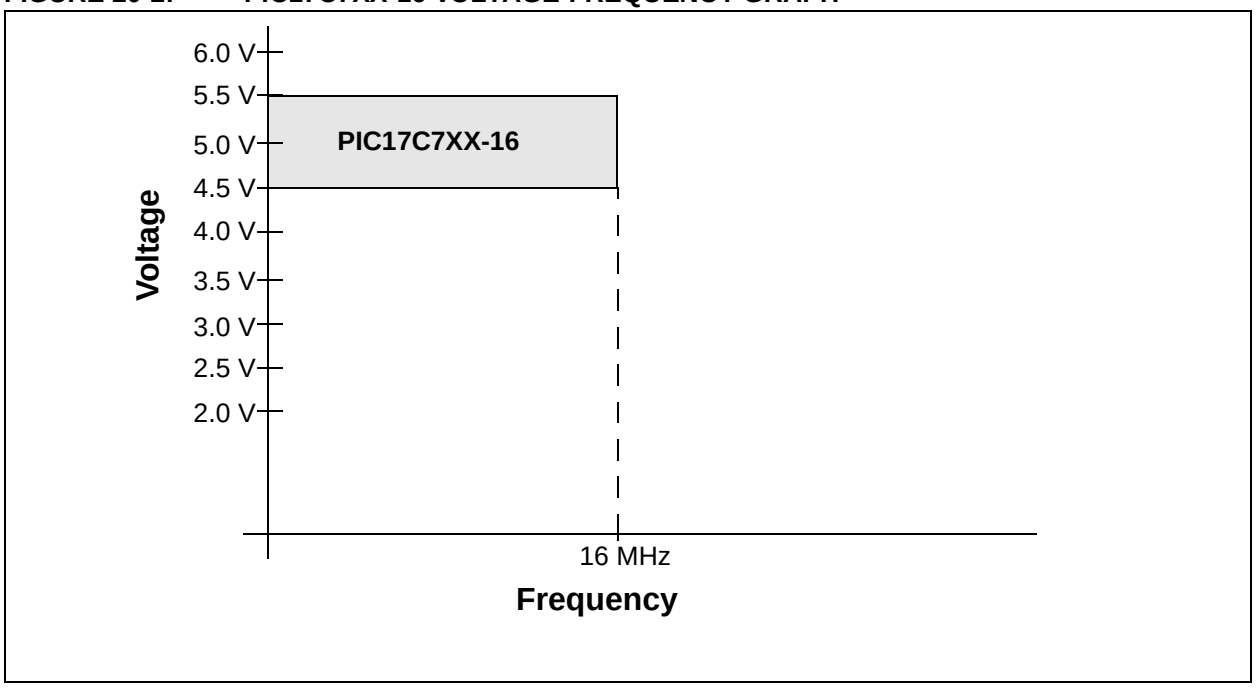



**FIGURE 20-2: PIC17C7XX-16 VOLTAGE-FREQUENCY GRAPH**





© 2000 Microchip Technology Inc.

**FIGURE 20-3: PIC17LC7XX-08 VOLTAGE-FREQUENCY GRAPH**

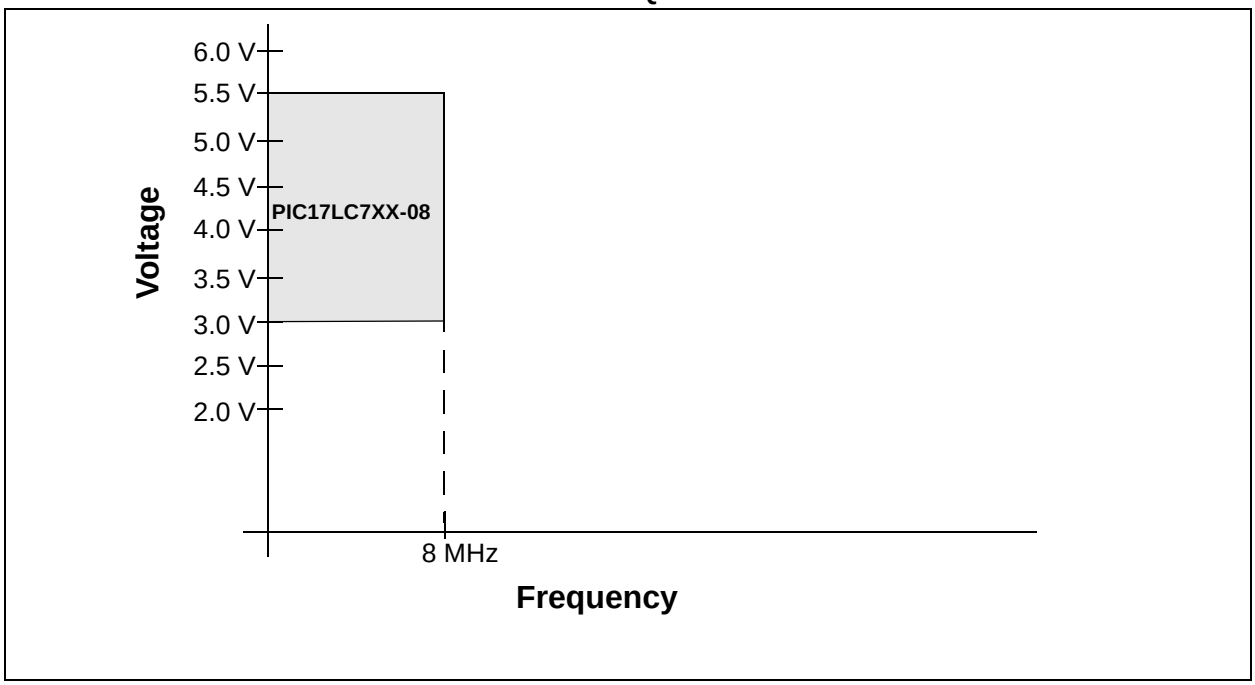
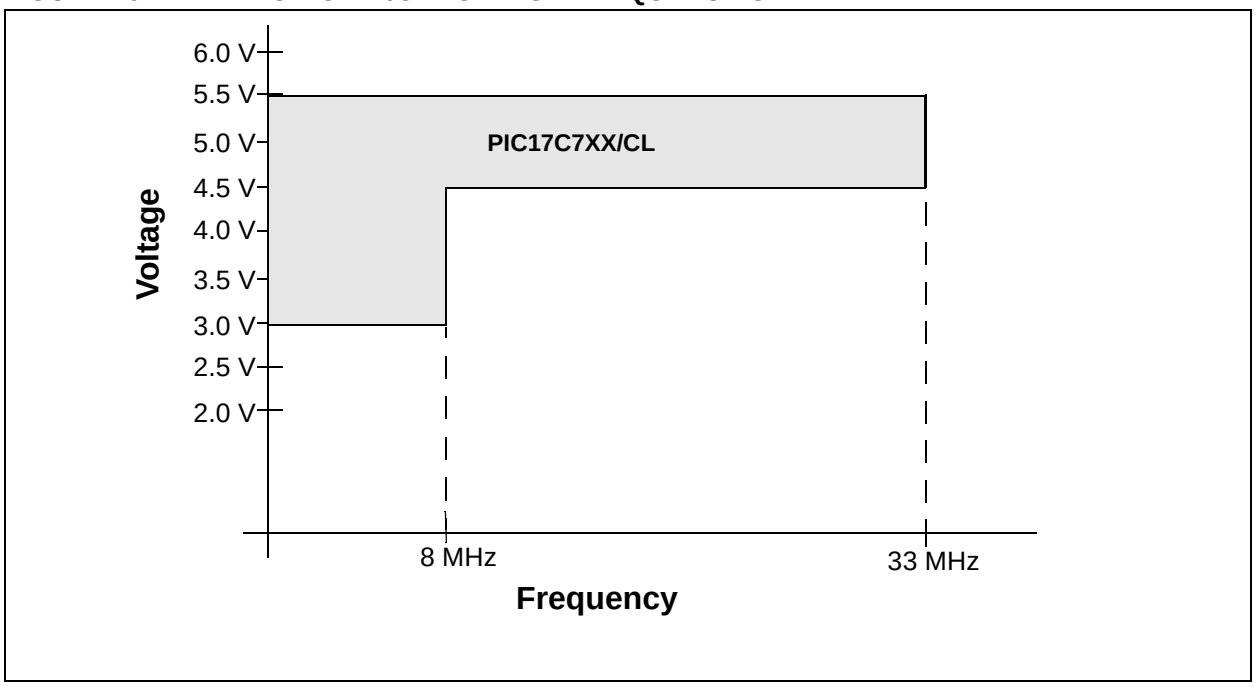



**FIGURE 20-4: PIC17C7XX/CL VOLTAGE-FREQUENCY GRAPH**



## 20.1 DC Characteristics

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC17LC7XX-08**<br>(Commercial, Industrial) | | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature<br>-40°C ≤ TA ≤ +85°C for industrial and<br>0°C ≤ TA ≤ +70°C for commercial | | | | |
| **PIC17C7XX-16**<br>(Commercial, Industrial, Extended)<br>**PIC17C7XX-33**<br>(Commercial, Industrial, Extended) | | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature<br>-40°C ≤ TA ≤ +125°C for extended<br>-40°C ≤ TA ≤ +85°C for industrial<br>0°C ≤ TA ≤ +70°C for commercial | | | | |
| **Param. No.** | **Sym** | **Characteristic** | **Min** | **Typ†** | **Max** | **Units** | **Conditions** |
| D001 | VDD | **Supply Voltage** | | | | | |
| | | PIC17LC7XX | 3.0 | — | 5.5 | V | |
| D001 | | PIC17C7XX-33<br>PIC17C7XX-16 | 4.5<br>VBOR | —<br>— | 5.5<br>5.5 | V<br>V | (BOR enabled) **(Note 5)** |
| D002 | VDR | **RAM Data Retention Voltage (Note 1)** | 1.5 | — | — | V | Device in SLEEP mode |
| D003 | VPOR | **VDD Start Voltage** to ensure internal Power-on Reset signal | — | Vss | — | V | See section on Power-on Reset for details |
| D004 | SVDD | **VDD Rise Rate** to ensure proper operation | | | | | |
| | | PIC17LCXX | 0.010 | — | — | V/ms | See section on Power-on Reset for details |
| D004 | | PIC17CXX | 0.085 | — | — | V/ms | See section on Power-on Reset for details |
| D005 | VBOR | **Brown-out Reset** voltage trip point | 3.65 | — | 4.35 | V | |
| D006 | VPORTP | **Power-on Reset** trip point | — | 2.2 | — | V | VDD = VPORTP |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**Note 1:** This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

**2:** The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail to rail; all I/O pins tri-stated, pulled to VDD or VSS, T0CKI = VDD, MCLR = VDD; WDT disabled.

Current consumed from the oscillator and I/O's driving external capacitive or resistive loads needs to be considered.

For the RC oscillator, the current through the external pull-up resistor (R) can be estimated as: VDD/(2 • R).

For capacitive loads, the current can be estimated (for an individual I/O pin) as (CL • VDD) • f

CL = Total capacitive load on the I/O pin; f = average frequency the I/O pin switches.

The capacitive currents are most significant when the device is configured for external execution (includes Extended Microcontroller mode).

**3:** The power-down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD or VSS.

**4:** For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula IR = VDD/2REXT (mA) with REXT in kOhm.

**5:** This is the voltage where the device enters the Brown-out Reset. When BOR is enabled, the device (-16) will operate correctly to this trip point.


© 2000 Microchip Technology Inc.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC17LC7XX-08**<br>(Commercial, Industrial) | | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature<br>-40°C ≤ TA ≤ +85°C for industrial and<br>0°C ≤ TA ≤ +70°C for commercial | | | | |
| **PIC17C7XX-16**<br>(Commercial, Industrial, Extended)<br>**PIC17C7XX-33**<br>(Commercial, Industrial, Extended) | | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature<br>-40°C ≤ TA ≤ +125°C for extended<br>-40°C ≤ TA ≤ +85°C for industrial<br>0°C ≤ TA ≤ +70°C for commercial | | | | |
| **Param. No.** | **Sym** | **Characteristic** | **Min** | **Typ†** | **Max** | **Units** | **Conditions** |
| D010 | IDD | **Supply Current (Note 2)** | | | | | |
| | | PIC17LC7XX | — | 3 | 6 | mA | FOSC = 4 MHz **(Note 4)** |
| D010 | | PIC17C7XX | — | 3 | 6 | mA | FOSC = 4 MHz **(Note 4)** |
| D011 | | PIC17LC7XX | — | 5 | 10 | mA | FOSC = 8 MHz |
| D011<br>D012 | | PIC17C7XX | —<br>— | 5<br>9 | 10<br>18 | mA<br>mA | FOSC = 8 MHz<br>FOSC = 16 MHz |
| D014 | | PIC17LC7XX | — | 85 | 150 | µA | FOSC = 32 kHz,<br>(EC osc configuration) |
| D015 | | PIC17C7XX | — | 15 | 30 | mA | FOSC = 33 MHz |
| D021 | IPD | **Power-down Current (Note 3)** | | | | | |
| | | PIC17LC7XX | — | <1 | 5 | µA | VDD = 3.0V, WDT disabled |
| D021<br>(commercial, industrial) | | PIC17C7XX | — | <1 | 20 | µA | VDD = 5.5V, WDT disabled |
| D021A<br>(extended) | | | — | 2 | 20 | µA | VDD = 5.5V, WDT disabled |
| | | **Module Differential Current** | | | | | |
| D023 | ΔIBOR | BOR circuitry | – | 75 | 150 | µA | VDD = 4.5V, BODEN enabled |
| D024 | ΔIWDT | Watchdog Timer | – | 10 | 35 | µA | VDD = 5.5V |
| D026 | ΔIAD | A/D converter | – | 1 | – | µA | VDD = 5.5V, A/D not converting |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**Note 1:** This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

**2:** The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail to rail; all I/O pins tri-stated, pulled to VDD or VSS, T0CKI = VDD, MCLR = VDD; WDT disabled.

Current consumed from the oscillator and I/O's driving external capacitive or resistive loads needs to be considered.

For the RC oscillator, the current through the external pull-up resistor (R) can be estimated as: VDD/(2 • R).

For capacitive loads, the current can be estimated (for an individual I/O pin) as (CL • VDD) • f

CL = Total capacitive load on the I/O pin; f = average frequency the I/O pin switches.

The capacitive currents are most significant when the device is configured for external execution (includes Extended Microcontroller mode).

**3:** The power-down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD or VSS.

**4:** For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula IR = VDD/2REXT (mA) with REXT in kOhm.

**5:** This is the voltage where the device enters the Brown-out Reset. When BOR is enabled, the device (-16) will operate correctly to this trip point.

## 20.2 DC Characteristics: PIC17C7XX-16 (Commercial, Industrial, Extended) PIC17C7XX-33 (Commercial, Industrial, Extended) PIC17LC7XX-08 (Commercial, Industrial)

| DC CHARACTERISTICS | | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature<br>-40°C ≤ TA ≤ +125°C for extended<br>-40°C ≤ TA ≤ +85°C for industrial<br>0°C ≤ TA ≤ +70°C for commercial<br>Operating voltage VDD range as described in Section 20.1 | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym** | **Characteristic** | **Min** | **Typ†** | **Max** | **Units** | **Conditions** |
| | | **Input Low Voltage** | | | | | |
| | VIL | I/O ports | | | | | |
| D030 | | with TTL buffer **(Note 6)** | Vss | – | 0.8 | V | 4.5V ≤ VDD ≤ 5.5V |
| | | | Vss | – | 0.2VDD | V | 3.0V ≤ VDD ≤ 4.5V |
| D031 | | with Schmitt Trigger buffer | | | | | |
| | | RA2, RA3 | Vss | – | 0.3VDD | V | $I^2C$ compliant |
| | | All others | Vss | – | 0.2VDD | V | |
| D032 | | $\overline{MCLR}$, OSC1 (in EC and RC mode) | Vss | – | 0.2VDD | V | **(Note 1)** |
| D033 | | OSC1 (in XT, and LF mode) | – | 0.5VDD | – | V | |
| | | **Input High Voltage** | | | | | |
| | VIH | I/O ports | | | | | |
| D040 | | with TTL buffer **(Note 6)** | 2.0 | – | VDD | V | 4.5V ≤ VDD ≤ 5.5V |
| | | | 1 + 0.2VDD | – | VDD | V | 3.0V ≤ VDD ≤ 4.5V |
| D041 | | with Schmitt Trigger buffer | | | | | |
| | | RA2, RA3 | 0.7VDD | – | VDD | V | $I^2C$ compliant |
| | | All others | 0.8VDD | – | VDD | V | |
| D042 | | $\overline{MCLR}$ | 0.8VDD | – | VDD | V | **(Note 1)** |
| D043 | | OSC1 (XT, and LF mode) | – | 0.5VDD | – | V | |
| D050 | VHYS | **Hysteresis of Schmitt Trigger Inputs** | 0.15VDD | – | – | V | |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**Note 1:** In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PIC17CXXX devices be driven with external clock in RC mode.

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

**3:** Negative current is defined as current sourced by the pin.

**4:** These specifications are for the programming of the on-chip program memory EPROM through the use of the table write instructions. The complete programming specifications can be found in: PIC17C7XX Programming Specifications (Literature number DS TBD).

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

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


© 2000 Microchip Technology Inc.

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature<br>-40°C ≤ TA ≤ +125°C for extended<br>-40°C ≤ TA ≤ +85°C for industrial<br>0°C ≤ TA ≤ +70°C for commercial<br>Operating voltage VDD range as described in Section 20.1 | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym** | **Characteristic** | **Min** | **Typ†** | **Max** | **Units** | **Conditions** |
| | | **Input Leakage Current (Notes 2, 3)** | | | | | |
| D060 | IIL | I/O ports (except RA2, RA3) | – | – | ±1 | µA | VSS ≤ VPIN ≤ VDD, I/O Pin (in digital mode) at hi-impedance PORTB weak pull-ups disabled |
| D061 | | $\overline{MCLR}$, TEST | – | – | ±2 | µA | VPIN = Vss or VPIN = VDD |
| D062 | | RA2, RA3 | | | ±2 | µA | VSS ≤ VRA2, VRA3 ≤ 12V |
| D063 | | OSC1 (EC, RC modes) | – | – | ±1 | µA | VSS ≤ VPIN ≤ VDD |
| D063B | | OSC1 (XT, LF modes) | – | – | VPIN | µA | RF ≥ 1 MΩ |
| D064 | | $\overline{MCLR}$, TEST | – | – | 25 | µA | VMCLR = VPP = 12V (when not programming) |
| D070 | IPURB | **PORTB Weak Pull-up Current** | 85 | 130 | 260 | µA | VPIN = VSS, $\overline{RBPU}$ = 0<br>4.5V ≤ VDD ≤ 5.5V |
| | | **Output Low Voltage** | | | | | |
| D080 | VOL | I/O ports | | | | | IOL = VDD/1.250 mA |
| | | | – | – | 0.1VDD | V | 4.5V ≤ VDD ≤ 5.5V |
| | | | – | – | 0.1VDD | V | VDD = 3.0V |
| D081 | | with TTL buffer | – | – | 0.4 | V | IOL = 6 mA, VDD = 4.5V **(Note 6)** |
| D082 | | RA2 and RA3 | – | – | 3.0 | V | IOL = 60.0 mA, VDD = 5.5V |
| | | | – | – | 0.6 | V | IOL = 60.0 mA, VDD = 4.5V |
| D083 | | OSC2/CLKOUT | – | – | 0.4 | V | IOL = 1 mA, VDD = 4.5V |
| D084 | | (RC and EC osc modes) | – | – | 0.1VDD | V | IOL = VDD/5 mA (PIC17LC7XX only) |
| | | **Output High Voltage (Note 3)** | | | | | |
| D090 | VOH | I/O ports (except RA2 and RA3) | | | | | IOH = -VDD/2.5 mA |
| | | | 0.9VDD | – | – | V | 4.5V ≤ VDD ≤ 5.5V |
| | | | 0.9VDD | – | – | V | VDD = 3.0V |
| D091 | | with TTL buffer | 2.4 | – | – | V | IOH = -6.0 mA, VDD = 4.5V **(Note 6)** |
| D093 | | OSC2/CLKOUT | 2.4 | – | – | V | IOH = -5 mA, VDD = 4.5V |
| D094 | | (RC and EC osc modes) | 0.9VDD | – | – | V | IOH = -VDD/5 mA (PIC17LC7XX only) |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**Note 1:** In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PIC17CXXX devices be driven with external clock in RC mode.

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

**3:** Negative current is defined as current sourced by the pin.

**4:** These specifications are for the programming of the on-chip program memory EPROM through the use of the table write instructions. The complete programming specifications can be found in: PIC17C7XX Programming Specifications (Literature number DS TBD).

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

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

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature<br>-40°C ≤ TA ≤ +125°C for extended<br>-40°C ≤ TA ≤ +85°C for industrial<br>0°C ≤ TA ≤ +70°C for commercial<br>Operating voltage VDD range as described in Section 20.1 | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym** | **Characteristic** | **Min** | **Typ†** | **Max** | **Units** | **Conditions** |
| D150 | VOD | **Open Drain High Voltage** | – | – | 8.5 | V | RA2 and RA3 pins only pulled up to externally applied voltage |
| | | **Capacitive Loading Specs on Output Pins** | | | | | |
| D100 | COSC2 | OSC2/CLKOUT pin | – | – | 25 | pF | In EC or RC osc modes, when OSC2 pin is outputting CLKOUT. External clock is used to drive OSC1. |
| D101 | CIO | All I/O pins and OSC2 (in RC mode) | – | – | 50 | pF | |
| D102 | CAD | System Interface Bus (PORTC, PORTD and PORTE) | – | – | 50 | pF | In Microprocessor or Extended Microcontroller mode |
| | | **Internal Program Memory Programming Specs (Note 4)** | | | | | |
| D110 | VPP | Voltage on $\overline{MCLR}$/VPP pin | 12.75 | – | 13.25 | V | **(Note 5)** |
| D111 | VDDP | Supply voltage during programming | 4.75 | 5.0 | 5.25 | V | |
| D112 | IPP | Current into $\overline{MCLR}$/VPP pin | – | 25 | 50 | mA | |
| D113 | IDDP | Supply current during programming | – | – | 30 | mA | |
| D114 | TPROG | Programming pulse width | 100 | – | 1000 | ms | Terminated via internal/ external interrupt or a RESET |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**Note 1:** In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PIC17CXXX devices be driven with external clock in RC mode.

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

**3:** Negative current is defined as current sourced by the pin.

**4:** These specifications are for the programming of the on-chip program memory EPROM through the use of the table write instructions. The complete programming specifications can be found in: PIC17C7XX Programming Specifications (Literature number DS TBD).

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

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

**Note 1:** When using the Table Write for internal programming, the device temperature must be less than 40°C.

**2:** For In-Circuit Serial Programming (ICSP™), refer to the device programming specification.


© 2000 Microchip Technology Inc.

## 20.3 Timing Parameter Symbology

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

1. TppS2ppS
2. TppS
3. TCC:ST (I²C specifications only)
4. Ts (I²C specifications only)

| **T** | | | |
|---|---|---|---|
| F | Frequency | T | Time |

Lowercase symbols (pp) and their meanings:

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

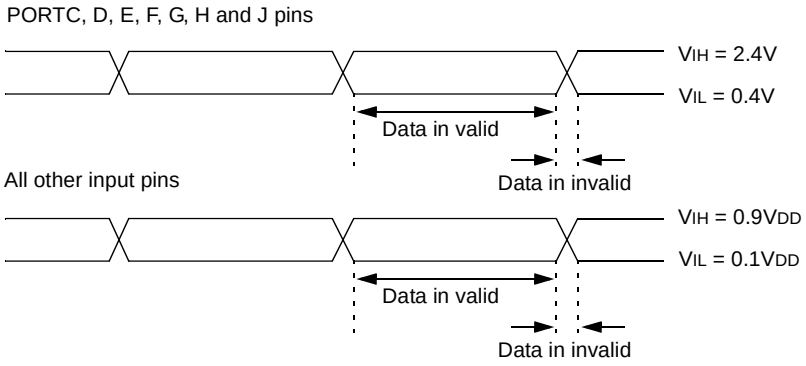
Uppercase symbols and their meanings:

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

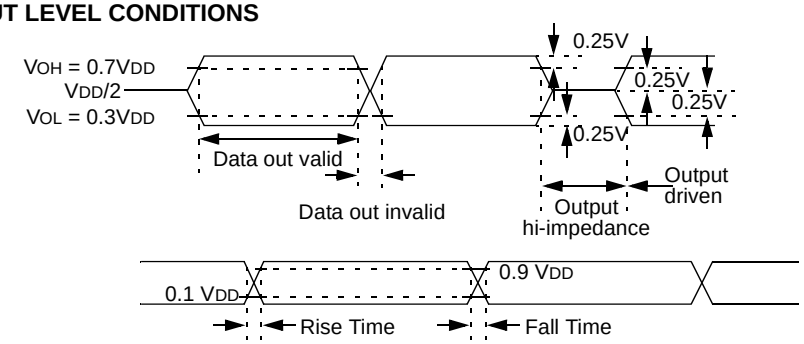
**FIGURE 20-5: PARAMETER MEASUREMENT INFORMATION**

All timings are measured between high and low measurement points as indicated below.

**INPUT LEVEL CONDITIONS**

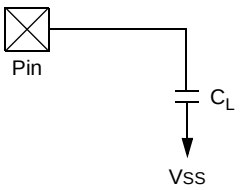



**OUTPUT LEVEL CONDITIONS**




**LOAD CONDITIONS**

Load Condition 1

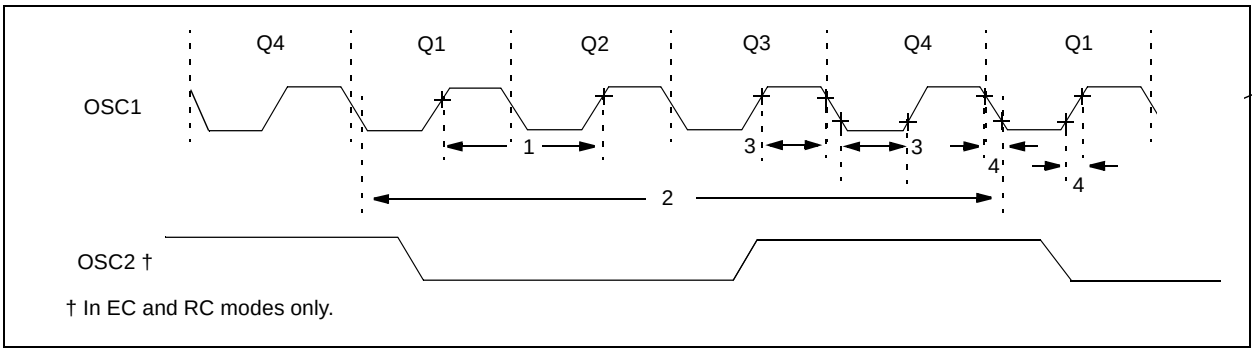



50 pF $\leq C_L$


© 2000 Microchip Technology Inc.

## 20.4 Timing Diagrams and Specifications

**FIGURE 20-6: EXTERNAL CLOCK TIMING**




† In EC and RC modes only.

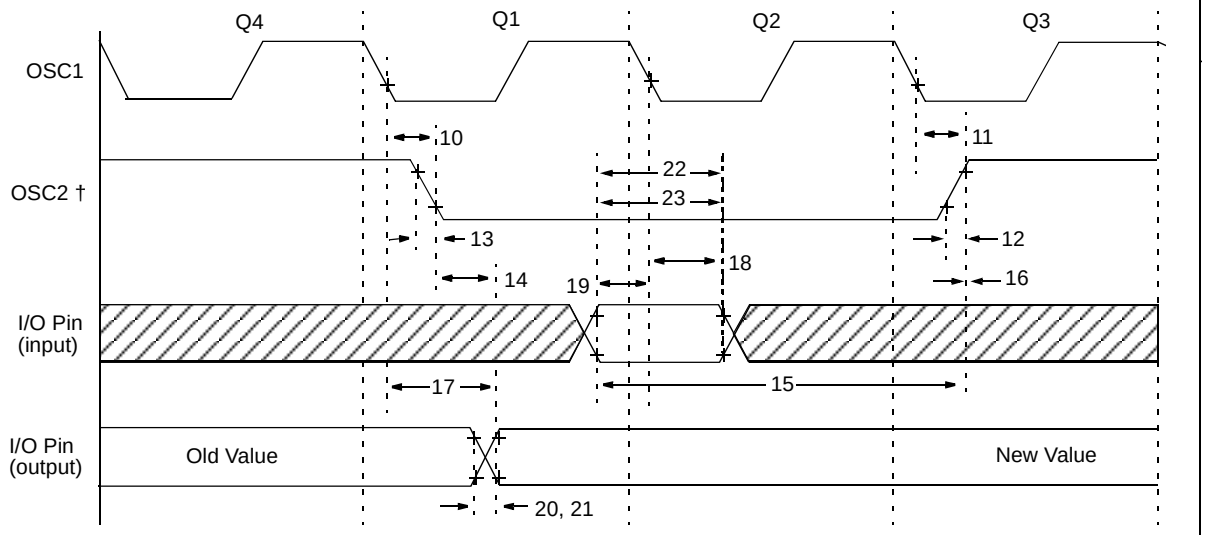
**TABLE 20-1: EXTERNAL CLOCK TIMING REQUIREMENTS**

| Param No. | Sym | Characteristic | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| | FOSC | **External CLKIN Frequency (Note 1)** | DC | — | 8 | MHz | EC osc mode - 08 devices (8 MHz devices) |
| | | | DC | — | 16 | MHz | - 16 devices (16 MHz devices) |
| | | | DC | — | 33 | MHz | - 33 devices (33 MHz devices) |
| | | **Oscillator Frequency (Note 1)** | DC | — | 4 | MHz | RC osc mode |
| | | | 2 | — | 8 | MHz | XT osc mode - 08 devices (8 MHz devices) |
| | | | 2 | — | 16 | MHz | - 16 devices (16 MHz devices) |
| | | | 2 | — | 33 | MHz | - 33 devices (33 MHz devices) |
| | | | DC | — | 2 | MHz | LF osc mode |
| 1 | TOSC | **External CLKIN Period (Note 1)** | 125 | — | — | ns | EC osc mode - 08 devices (8 MHz devices) |
| | | | 62.5 | — | — | ns | - 16 devices (16 MHz devices) |
| | | | 30.3 | — | — | ns | - 33 devices (33 MHz devices) |
| | | **Oscillator Period (Note 1)** | 250 | — | — | ns | RC osc mode |
| | | | 125 | — | 1,000 | ns | XT osc mode - 08 devices (8 MHz devices) |
| | | | 62.5 | — | 1,000 | ns | - 16 devices (16 MHz devices) |
| | | | 30.3 | — | 1,000 | ns | - 33 devices (33 MHz devices) |
| | | | 500 | — | — | ns | LF osc mode |
| 2 | TCY | **Instruction Cycle Time (Note 1)** | 121.2 | 4/FOSC | DC | ns | |
| 3 | TosL, TosH | **Clock in (OSC1) High or Low Time** | 10 | — | — | ns | EC oscillator |
| 4 | TosR, TosF | **Clock in (OSC1) Rise or Fall Time** | — | — | 5 | ns | EC oscillator |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**Note 1:** Instruction cycle period (TCY) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min." values with an external clock applied to the OSC1/CLKIN pin. When an external clock input is used, the "max." cycle time limit is "DC" (no clock) for all devices.

**FIGURE 20-7: CLKOUT AND I/O TIMING**




† In EC and RC modes only.

**TABLE 20-2: CLKOUT AND I/O TIMING REQUIREMENTS**

| Param No. | Sym | Characteristic | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 10 | TosL2ckL | OSC1↓ to CLKOUT↓ | — | 15 | 30 | ns | **(Note 1)** |
| 11 | TosL2ckH | OSC1↓ to CLKOUT↑ | — | 15 | 30 | ns | **(Note 1)** |
| 12 | TckR | CLKOUT rise time | — | 5 | 15 | ns | **(Note 1)** |
| 13 | TckF | CLKOUT fall time | — | 5 | 15 | ns | **(Note 1)** |
| 14 | TckH2ioV | CLKOUT ↑ to Port out valid | — | — | 0.5TCY + 20 | ns | **(Note 1)** |
| 15 | TioV2ckH | Port in valid before CLKOUT↑ | 0.25TCY + 25 | — | — | ns | **(Note 1)** |
| 16 | TckH2ioI | Port in hold after CLKOUT↑ | 0 | — | — | ns | **(Note 1)** |
| 17 | TosL2ioV | OSC1↓ (Q1 cycle) to Port out valid | — | — | 100 | ns | |
| 18 | TosL2ioI | OSC1↓ (Q2 cycle) to Port input invalid (I/O in hold time) | 0 | — | — | ns | |
| 19 | TioV2osL | Port input valid to OSC1↓ (I/O in setup time) | 30 | — | — | ns | |
| 20 | TioR | Port output rise time | — | 10 | 35 | ns | |
| 21 | TioF | Port output fall time | — | 10 | 35 | ns | |
| 22 | TinHL | INT pin high or low time | 25 | — | — | ns | |
| 23 | TrbHL | RB7:RB0 change INT high or low time | 25 | — | — | ns | |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Measurements are taken in EC mode, where CLKOUT output is 4 x TOSC.


© 2000 Microchip Technology Inc.

**FIGURE 20-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER, AND BROWN-OUT RESET TIMING**

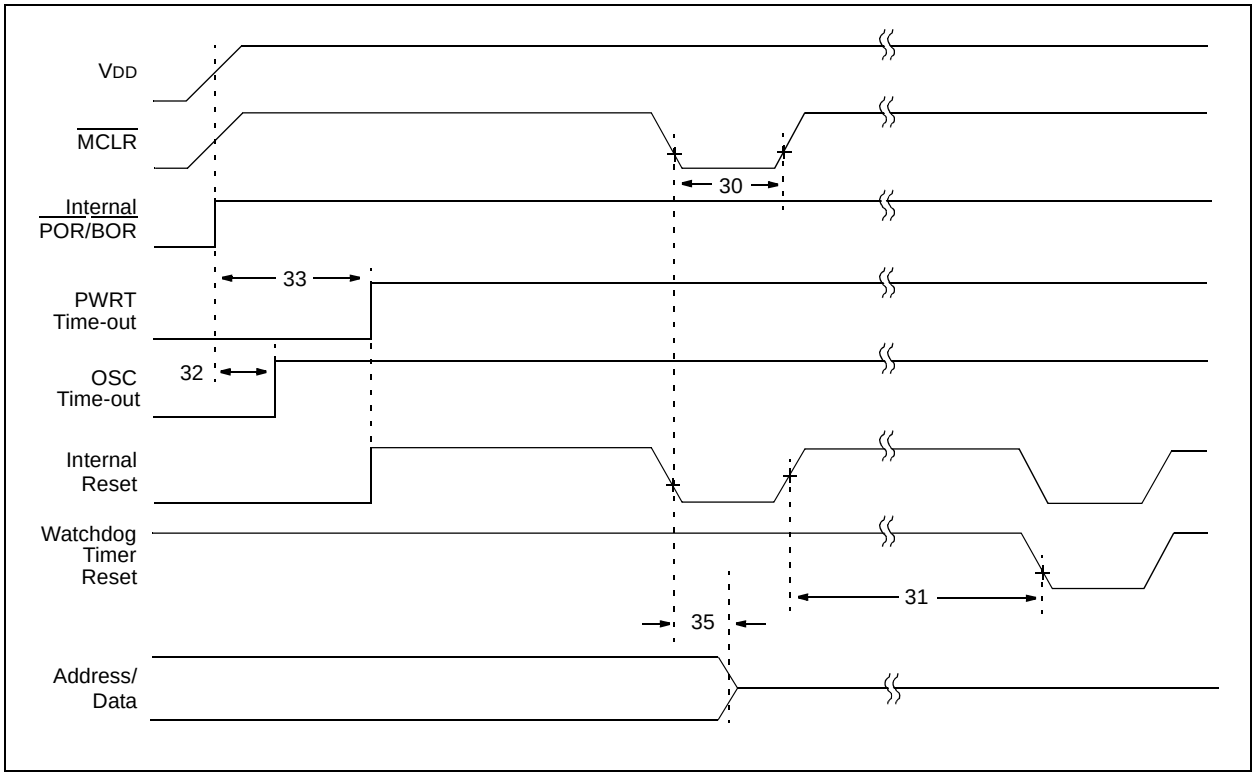



**TABLE 20-3: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER, AND BROWN-OUT RESET REQUIREMENTS**

| Param. No. | Sym | Characteristic | | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 30 | TmcL | $\overline{MCLR}$ Pulse Width (low) | | 100 | — | — | ns | VDD = 5V |
| 31 | TWDT | Watchdog Timer Time-out Period (Postscale = 1) | | 5 | 12 | 25 | ms | VDD = 5V |
| 32 | TOST | Oscillation Start-up Timer Period | | — | 1024TOSC | — | ms | TOSC = OSC1 period |
| 33 | TPWRT | Power-up Timer Period | | 40 | 96 | 200 | ms | VDD = 5V |
| 34 | TIOZ | $\overline{MCLR}$ to I/O hi-impedance | | 100 | — | — | ns | Depends on pin load |
| 35 | TmcL2adl | $\overline{MCLR}$ to System Interface bus (AD15:AD0>) invalid | PIC17**C**7XX | — | — | 100 | ns | |
| | | | PIC17**LC**7XX | — | — | 120 | ns | |
| 36 | TBOR | Brown-out Reset Pulse Width (low) | | 100 | — | — | ns | VDD within VBOR limits (parameter D005) |

† Data in “Typ” column is at 5V, 25°C unless otherwise stated.

.

**FIGURE 20-9: TIMER0 EXTERNAL CLOCK TIMINGS**

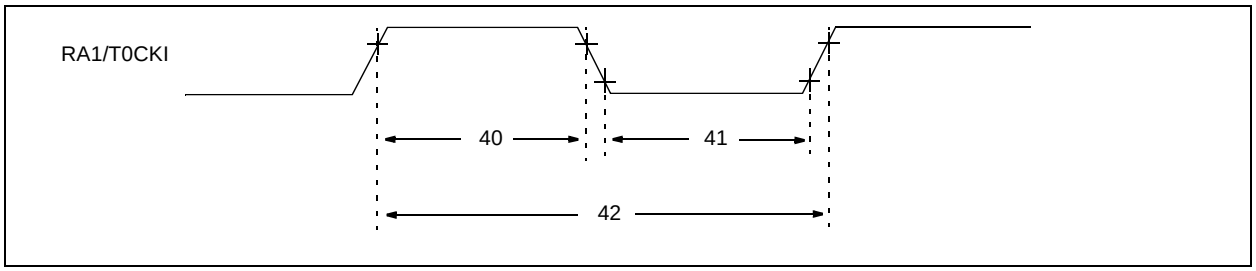



**TABLE 20-4: TIMER0 EXTERNAL CLOCK REQUIREMENTS**

| Param No. | Sym | Characteristic | | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 40 | Tt0H | T0CKI High Pulse Width | No Prescaler | 0.5TCY + 20 | — | — | ns | |
| | | | With Prescaler | 10 | — | — | ns | |
| 41 | Tt0L | T0CKI Low Pulse Width | No Prescaler | 0.5TCY + 20 | — | — | ns | |
| | | | With Prescaler | 10 | — | — | ns | |
| 42 | Tt0P | T0CKI Period | | Greater of: 20 ns or $\frac{T_{CY} + 40}{N}$ | — | — | ns | N = prescale value (1, 2, 4, ..., 256) |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**FIGURE 20-10: TIMER1, TIMER2 AND TIMER3 EXTERNAL CLOCK TIMINGS**

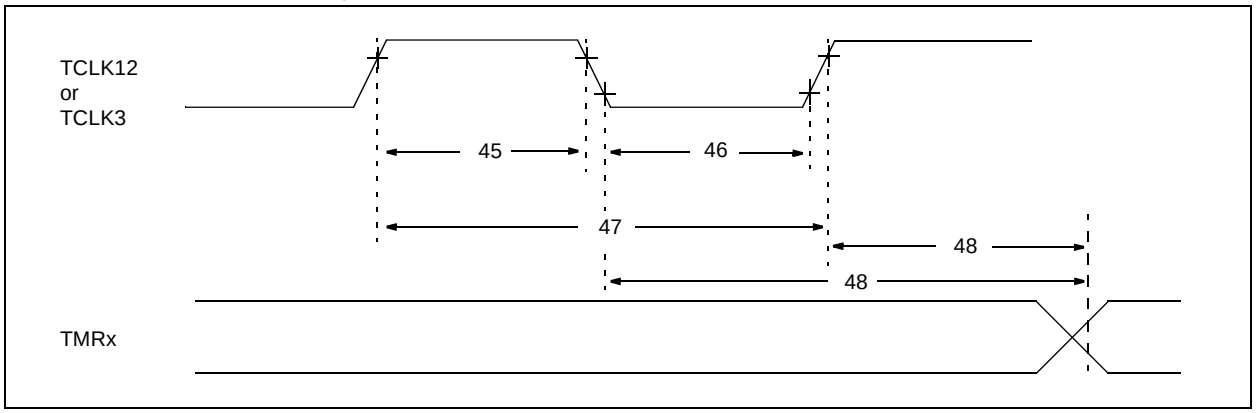



**TABLE 20-5: TIMER1, TIMER2 AND TIMER3 EXTERNAL CLOCK REQUIREMENTS**

| Param No. | Sym | Characteristic | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 45 | Tt123H | TCLK12 and TCLK3 high time | 0.5TCY + 20 | — | — | ns | |
| 46 | Tt123L | TCLK12 and TCLK3 low time | 0.5TCY + 20 | — | — | ns | |
| 47 | Tt123P | TCLK12 and TCLK3 input period | $\frac{T_{CY} + 40}{N}$ | — | — | ns | N = prescale value (1, 2, 4, 8) |
| 48 | TckE2tmrI | Delay from selected External Clock Edge to Timer increment | 2TOSC | — | 6Tosc | — | |

† Data in “Typ” column is at 5V, 25°C unless otherwise stated.


© 2000 Microchip Technology Inc.

**FIGURE 20-11: CAPTURE TIMINGS**

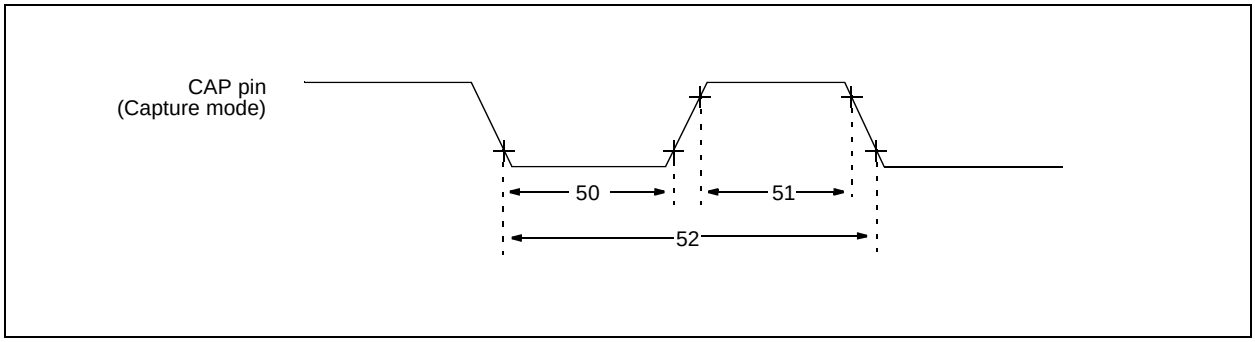



**TABLE 20-6: CAPTURE REQUIREMENTS**

| Param No. | Sym | Characteristic | Min | Typ † | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 50 | TccL | Capture pin input low time | 10 | — | — | ns | |
| 51 | TccH | Capture pin input high time | 10 | — | — | ns | |
| 52 | TccP | Capture pin input period | $\frac{2T_{CY}}{N}$ | — | — | ns | N = prescale value (4 or 16) |

† Data in “Typ” column is at 5V, 25°C unless otherwise stated.

**FIGURE 20-12: PWM TIMINGS**

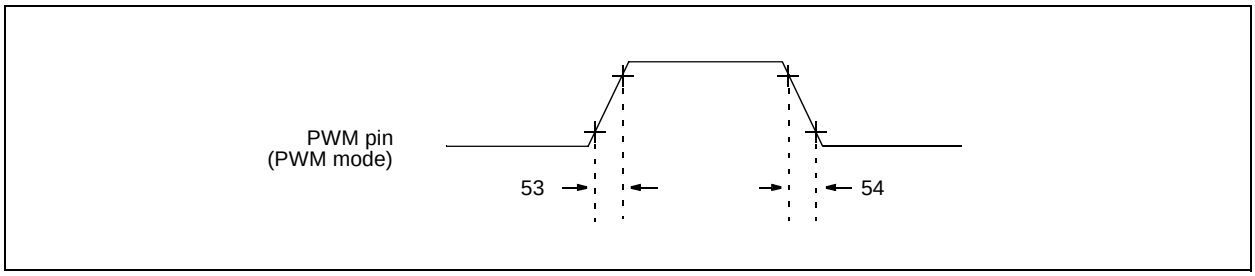
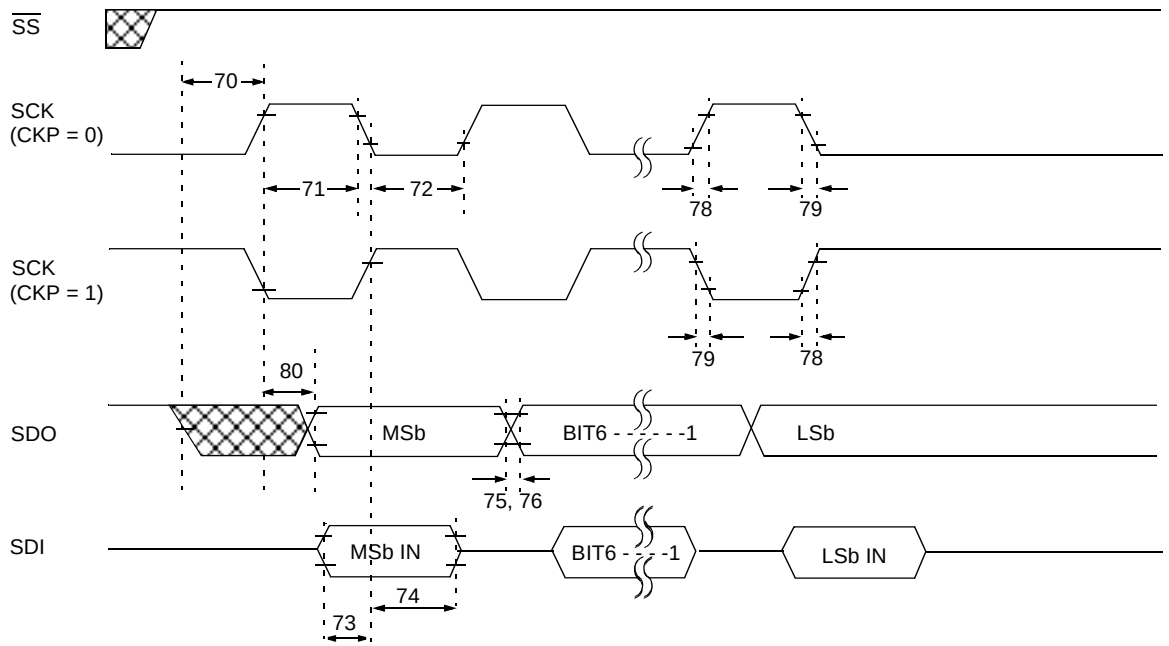



**TABLE 20-7: PWM REQUIREMENTS**

| Param No. | Sym | Characteristic | Min | Typ † | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 53 | TccR | PWM pin output rise time | — | 10 | 35 | ns | |
| 54 | TccF | PWM pin output fall time | — | 10 | 35 | ns | |

† Data in “Typ” column is at 5V, 25°C unless otherwise stated.

**FIGURE 20-13: SPI MASTER MODE TIMING (CKE = 0)**




**Note:** Refer to Figure 20-5 for load conditions.

**TABLE 20-8: SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)**

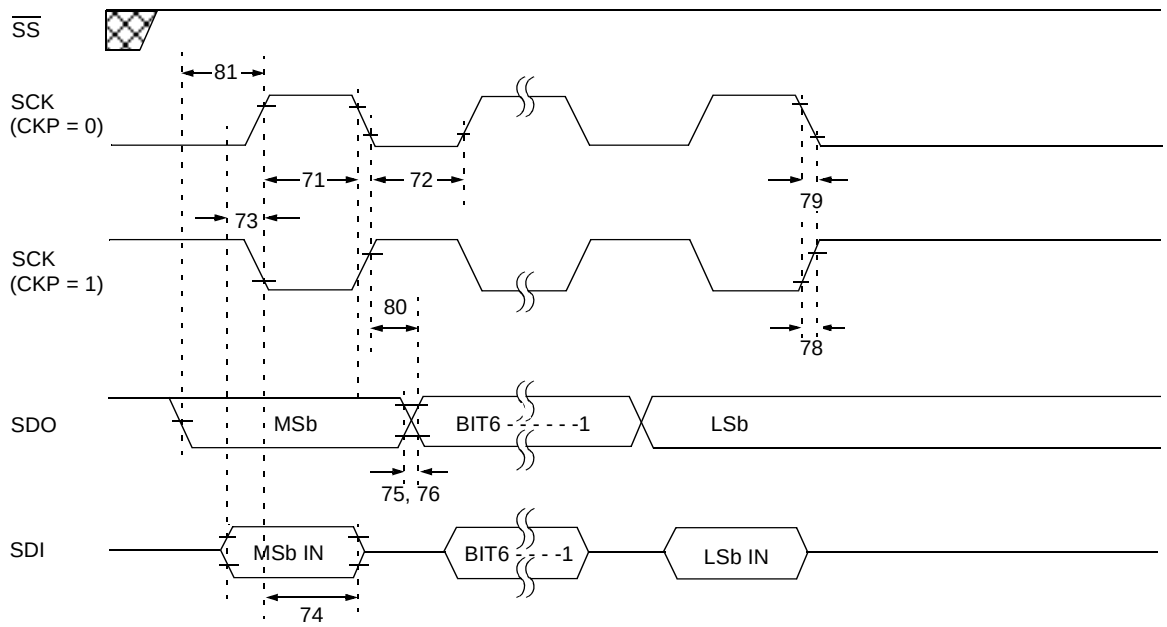
| Param. No. | Symbol | Characteristic | | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 70 | TssL2scH, TssL2scL | $\overline{SS}\downarrow$ to SCK$\downarrow$ or SCK$\uparrow$ input | | Tcy | — | — | ns | |
| 71 | TscH | SCK input high time (Slave mode) | Continuous | 1.25TCY + 30 | — | — | ns | |
| 71A | | | Single Byte | 40 | — | — | ns | **(Note 1)** |
| 72 | TscL | SCK input low time (Slave mode) | Continuous | 1.25TCY + 30 | — | — | ns | |
| 72A | | | Single Byte | 40 | — | — | ns | **(Note 1)** |
| 73 | TdiV2scH, TdiV2scL | Setup time of SDI data input to SCK edge | | 100 | — | — | ns | |
| 73A | TB2B | Last clock edge of Byte1 to the 1st clock edge of Byte2 | | 1.5TCY + 40 | — | — | ns | **(Note 1)** |
| 74 | TscH2diL, TscL2diL | Hold time of SDI data input to SCK edge | | 100 | — | — | ns | |
| 75 | TdoR | SDO data output rise time | | — | 10 | 25 | ns | |
| 76 | TdoF | SDO data output fall time | | — | 10 | 25 | ns | |
| 78 | TscR | SCK output rise time (Master mode) | | — | 10 | 25 | ns | |
| 79 | TscF | SCK output fall time (Master mode) | | — | 10 | 25 | ns | |
| 80 | TscH2doV, TscL2doV | SDO data output valid after SCK edge | | — | — | 50 | ns | |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**Note 1:** Specification 73A is only required if specifications 71A and 72A are used.


© 2000 Microchip Technology Inc.

**FIGURE 20-14: SPI MASTER MODE TIMING (CKE = 1)**




**Note:** Refer to Figure 20-5 for load conditions.

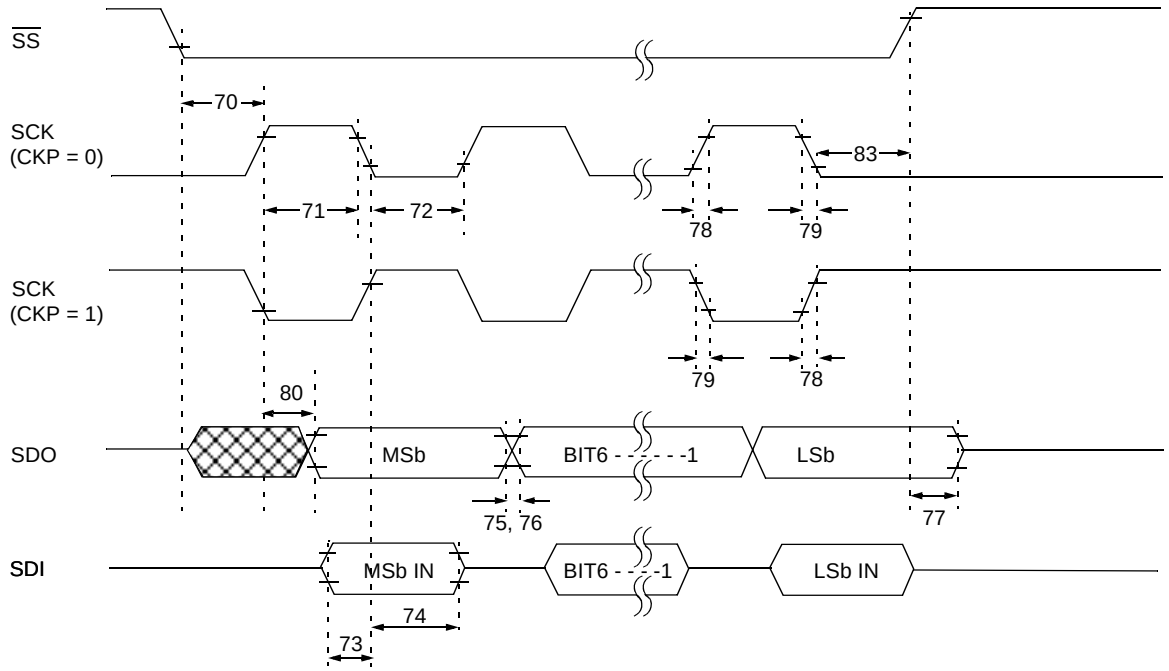
**TABLE 20-9: SPI MODE REQUIREMENTS (MASTER MODE, CKE = 1)**

| Param. No. | Symbol | Characteristic | | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 71 | TscH | SCK input high time (Slave mode) | Continuous | 1.25TCY + 30 | — | — | ns | |
| 71A | | | Single Byte | 40 | — | — | ns | **(Note 1)** |
| 72 | TscL | SCK input low time (Slave mode) | Continuous | 1.25 TCY + 30 | — | — | ns | |
| 72A | | | Single Byte | 40 | — | — | ns | **(Note 1)** |
| 73 | TdiV2scH, TdiV2scL | Setup time of SDI data input to SCK edge | | 100 | — | — | ns | |
| 73A | TB2B | Last clock edge of Byte1 to the 1st clock edge of Byte2 | | 1.5TCY + 40 | — | — | ns | **(Note 1)** |
| 74 | TscH2diL, TscL2diL | Hold time of SDI data input to SCK edge | | 100 | — | — | ns | |
| 75 | TdoR | SDO data output rise time | | — | 10 | 25 | ns | |
| 76 | TdoF | SDO data output fall time | | — | 10 | 25 | ns | |
| 78 | TscR | SCK output rise time (Master mode) | | — | 10 | 25 | ns | |
| 79 | TscF | SCK output fall time (Master mode) | | — | 10 | 25 | ns | |
| 80 | TscH2doV, TscL2doV | SDO data output valid after SCK edge | | — | — | 50 | ns | |
| 81 | TdoV2scH, TdoV2scL | SDO data output setup to SCK edge | | Tcy | — | — | ns | |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**Note 1:** Specification 73A is only required if specifications 71A and 72A are used.

**FIGURE 20-15: SPI SLAVE MODE TIMING (CKE = 0)**




**Note:** Refer to Figure 20-5 for load conditions.

**TABLE 20-10: SPI MODE REQUIREMENTS (SLAVE MODE TIMING, CKE = 0)**

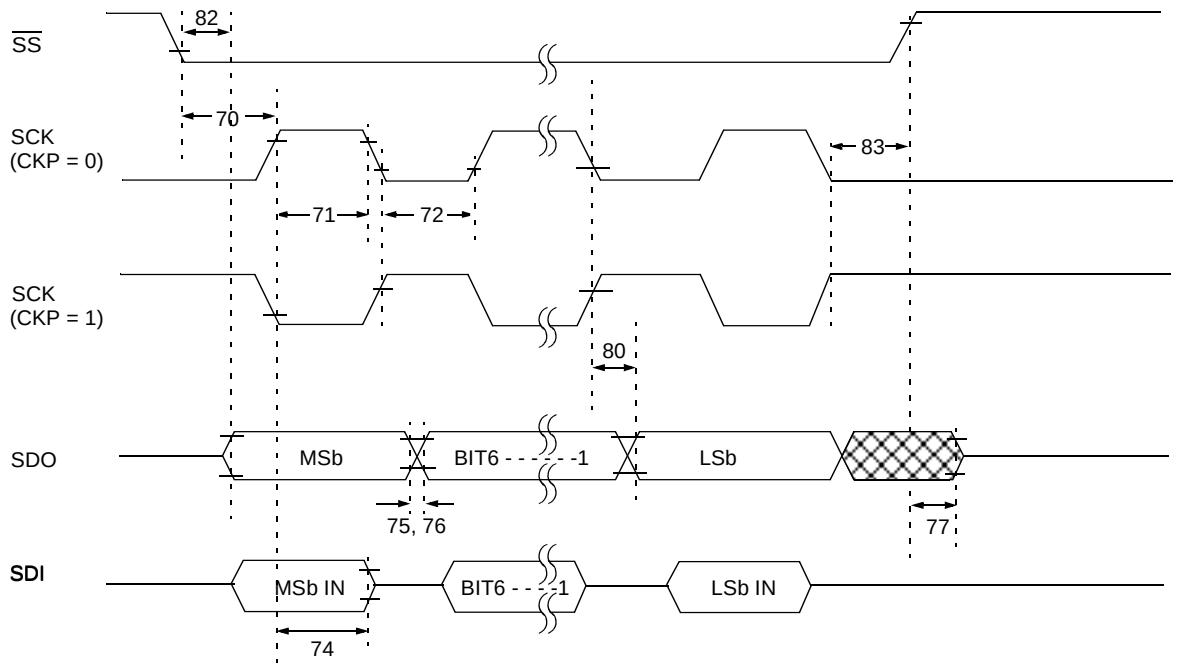
| Param. No. | Symbol | Characteristic | | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 70 | TssL2scH, TssL2scL | $\overline{SS}\downarrow$ to SCK$\downarrow$ or SCK$\uparrow$ input | | Tcy | — | — | ns | |
| 71 | TscH | SCK input high time (Slave mode) | Continuous | 1.25TCY + 30 | — | — | ns | |
| 71A | | | Single Byte | 40 | — | — | ns | **(Note 1)** |
| 72 | TscL | SCK input low time (Slave mode) | Continuous | 1.25TCY + 30 | — | — | ns | |
| 72A | | | Single Byte | 40 | — | — | ns | **(Note 1)** |
| 73 | TdiV2scH, TdiV2scL | Setup time of SDI data input to SCK edge | | 100 | — | — | ns | |
| 73A | TB2B | Last clock edge of Byte1 to the 1st clock edge of Byte2 | | 1.5TCY + 40 | — | — | ns | **(Note 1)** |
| 74 | TscH2diL, TscL2diL | Hold time of SDI data input to SCK edge | | 100 | — | — | ns | |
| 75 | TdoR | SDO data output rise time | | — | 10 | 25 | ns | |
| 76 | TdoF | SDO data output fall time | | — | 10 | 25 | ns | |
| 77 | TssH2doZ | $\overline{SS}\uparrow$ to SDO output hi-impedance | | 10 | — | 50 | ns | |
| 78 | TscR | SCK output rise time (Master mode) | | — | 10 | 25 | ns | |
| 79 | TscF | SCK output fall time (Master mode) | | — | 10 | 25 | ns | |
| 80 | TscH2doV, TscL2doV | SDO data output valid after SCK edge | | — | — | 50 | ns | |
| 83 | TscH2ssH, TscL2ssH | $\overline{SS}\uparrow$ after SCK edge | | 1.5TCY + 40 | — | — | ns | |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**Note 1:** Specification 73A is only required if specifications 71A and 72A are used.


© 2000 Microchip Technology Inc.

**FIGURE 20-16: SPI SLAVE MODE TIMING (CKE = 1)**




**Note:** Refer to Figure 20-5 for load conditions.

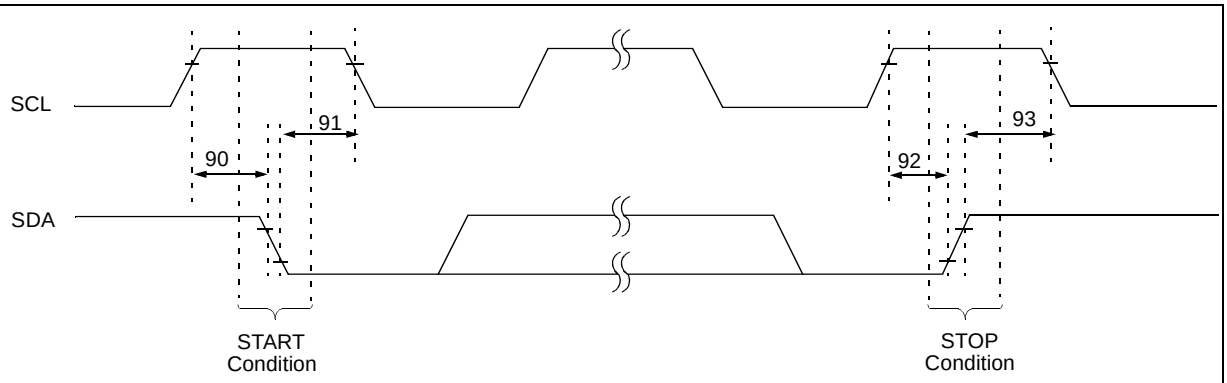
**TABLE 20-11: SPI MODE REQUIREMENTS (SLAVE MODE, CKE = 1)**

| Param. No. | Symbol | Characteristic | | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 70 | TssL2scH, TssL2scL | $\overline{SS}\downarrow$ to SCK↓ or SCK↑ input | | Tcy | — | — | ns | |
| 71 | TscH | SCK input high time (Slave mode) | Continuous | 1.25TCY + 30 | — | — | ns | |
| 71A | | | Single Byte | 40 | — | — | ns | **(Note 1)** |
| 72 | TscL | SCK input low time (Slave mode) | Continuous | 1.25TCY + 30 | — | — | ns | |
| 72A | | | Single Byte | 40 | — | — | ns | **(Note 1)** |
| 73A | TB2B | Last clock edge of Byte1 to the 1st clock edge of Byte2 | | 1.5TCY + 40 | — | — | ns | **(Note 1)** |
| 74 | TscH2diL, TscL2diL | Hold time of SDI data input to SCK edge | | 100 | — | — | ns | |
| 75 | TdoR | SDO data output rise time | | — | 10 | 25 | ns | |
| 76 | TdoF | SDO data output fall time | | — | 10 | 25 | ns | |
| 77 | TssH2doZ | $\overline{SS}\uparrow$ to SDO output hi-impedance | | 10 | — | 50 | ns | |
| 80 | TscH2doV, TscL2doV | SDO data output valid after SCK edge | | — | — | 50 | ns | |
| 82 | TssL2doV | SDO data output valid after $\overline{SS}\downarrow$ edge | | — | — | 50 | ns | |
| 83 | TscH2ssH, TscL2ssH | $\overline{SS}\uparrow$ after SCK edge | | 1.5TCY + 40 | — | — | ns | |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**Note 1:** Specification 73A is only required if specifications 71A and 72A are used.

**FIGURE 20-17: I²C BUS START/STOP BITS TIMING**




**Note:** Refer to Figure 20-5 for load conditions.

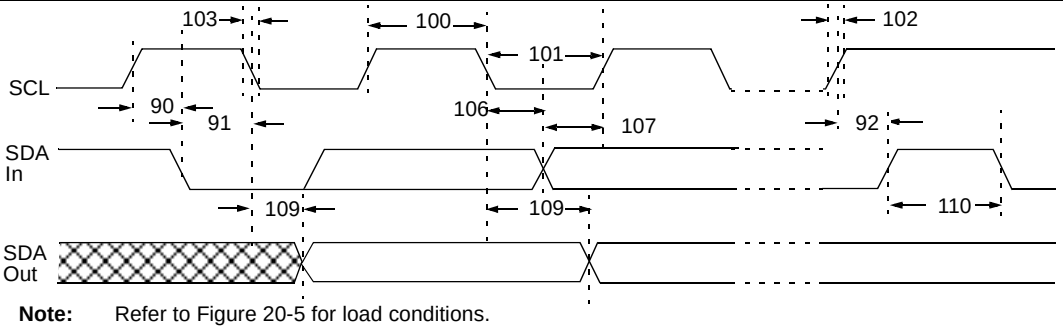
**TABLE 20-12: I²C BUS START/STOP BITS REQUIREMENTS**

| Param. No. | Sym | Characteristic | | Min | Typ | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 90 | Tsu:sta | START condition Setup time | 100 kHz mode | 2(Tosc)(BRG + 1) | — | — | ns | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 2(Tosc)(BRG + 1) | — | — | | |
| | | | 1 MHz mode(1) | 2(Tosc)(BRG + 1) | — | — | | |
| 91 | Thd:sta | START condition Hold time | 100 kHz mode | 2(Tosc)(BRG + 1) | — | — | ns | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 2(Tosc)(BRG + 1) | — | — | | |
| | | | 1 MHz mode(1) | 2(Tosc)(BRG + 1) | — | — | | |
| 92 | Tsu:sto | STOP condition Setup time | 100 kHz mode | 2(Tosc)(BRG + 1) | — | — | ns | |
| | | | 400 kHz mode | 2(Tosc)(BRG + 1) | — | — | | |
| | | | 1 MHz mode(1) | 2(Tosc)(BRG + 1) | — | — | | |
| 93 | Thd:sto | STOP condition Hold time | 100 kHz mode | 2(Tosc)(BRG + 1) | — | — | ns | |
| | | | 400 kHz mode | 2(Tosc)(BRG + 1) | — | — | | |
| | | | 1 MHz mode(1) | 2(Tosc)(BRG + 1) | — | — | | |

**Note 1:** Maximum pin capacitance = 10 pF for all I²C pins.


© 2000 Microchip Technology Inc.

**FIGURE 20-18: I²C BUS DATA TIMING**




**Note:** Refer to Figure 20-5 for load conditions.

**TABLE 20-13: I²C BUS DATA REQUIREMENTS**

| Param No. | Sym | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 100 | Thigh | Clock high time | 100 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(Tosc)(BRG + 1) | — | ms | |
| 101 | Tlow | Clock low time | 100 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(Tosc)(BRG + 1) | — | ms | |
| 102 | Tr | SDA and SCL rise time | 100 kHz mode | — | 1000 | ns | Cb is specified to be from 10 to 400 pF |
| | | | 400 kHz mode | 20 + 0.1Cb | 300 | ns | |
| | | | 1 MHz mode(1) | — | 300 | ns | |
| 103 | Tf | SDA and SCL fall time | 100 kHz mode | — | 300 | ns | Cb is specified to be from 10 to 400 pF |
| | | | 400 kHz mode | 20 + 0.1Cb | 300 | ns | |
| | | | 1 MHz mode(1) | — | 10 | ns | |
| 90 | Tsu:sta | START condition setup time | 100 kHz mode | 2(Tosc)(BRG + 1) | — | ms | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(Tosc)(BRG + 1) | — | ms | |
| 91 | Thd:sta | START condition hold time | 100 kHz mode | 2(Tosc)(BRG + 1) | — | ms | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(Tosc)(BRG + 1) | — | ms | |
| 106 | Thd:dat | Data input hold time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | 0.9 | ms | |
| | | | 1 MHz mode(1) | 0 | — | ns | |
| 107 | Tsu:dat | Data input setup time | 100 kHz mode | 250 | — | ns | **(Note 2)** |
| | | | 400 kHz mode | 100 | — | ns | |
| | | | 1 MHz mode(1) | 100 | — | ns | |
| 92 | Tsu:sto | STOP condition setup time | 100 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(Tosc)(BRG + 1) | — | ms | |
| 109 | Taa | Output valid from clock | 100 kHz mode | — | 3500 | ns | |
| | | | 400 kHz mode | — | 1000 | ns | |
| | | | 1 MHz mode(1) | — | 400 | ns | |

**Note 1:** Maximum pin capacitance = 10 pF for all I²C pins.

**2:** A fast mode (400 KHz) I²C bus device can be used in a standard mode I²C bus system, but the parameter # 107 ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line. Parameter #102 + #107 = 1000 + 250 = 1250 ns (for 100 kHz mode) before the SCL line is released.

**3:** $C_b$ is specified to be from 10-400pF. The minimum specifications are characterized with $C_b$=10pF. The rise time spec ($t_r$) is characterized with $R_p$=$R_p$ min. The minimum fall time specification ($t_f$) is characterized with $C_b$=10pF,and $R_p$=$R_p$ max. These are only valid for fast mode operation ($V_{DD}$=4.5-5.5V) and where the SPM bit (SSPSTAT<7>) =1.)

**4:** Max specifications for these parameters are valid for falling edge only. Specs are characterized with $R_p$=$R_p$ min and $C_b$=400pF for standard mode, 200pF for fast mode, and 10pF for 1MHz mode.

© 2000 Microchip Technology Inc.

| Param No. | Sym | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 110 | Tbuf | Bus free time | 100 kHz mode | 4.7 | — | ms | Time the bus must be free before a new transmission can start |
| | | | 400 kHz mode | 1.3 | — | ms | |
| | | | 1 MHz mode(1) | 0.5 | — | ms | |
| D102 | Cb | Bus capacitive loading | | — | 400 | pF | |

**Note 1:** Maximum pin capacitance = 10 pF for all $I^2C$ pins.

**2:** A fast mode (400 KHz) $I^2C$ bus device can be used in a standard mode $I^2C$ bus system, but the parameter # 107 ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line. Parameter #102 + #107 = 1000 + 250 = 1250 ns (for 100 kHz mode) before the SCL line is released.

**3:** $C_b$ is specified to be from 10-400pF. The minimum specifications are characterized with $C_b$=10pF. The rise time spec ($t_r$) is characterized with $R_p$=$R_p$ min. The minimum fall time specification ($t_f$) is characterized with $C_b$=10pF,and $R_p$=$R_p$ max. These are only valid for fast mode operation ($V_{DD}$=4.5-5.5V) and where the SPM bit (SSPSTAT<7>) =1.)

**4:** Max specifications for these parameters are valid for falling edge only. Specs are characterized with $R_p$=$R_p$ min and $C_b$=400pF for standard mode, 200pF for fast mode, and 10pF for 1MHz mode.

**FIGURE 20-19: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**

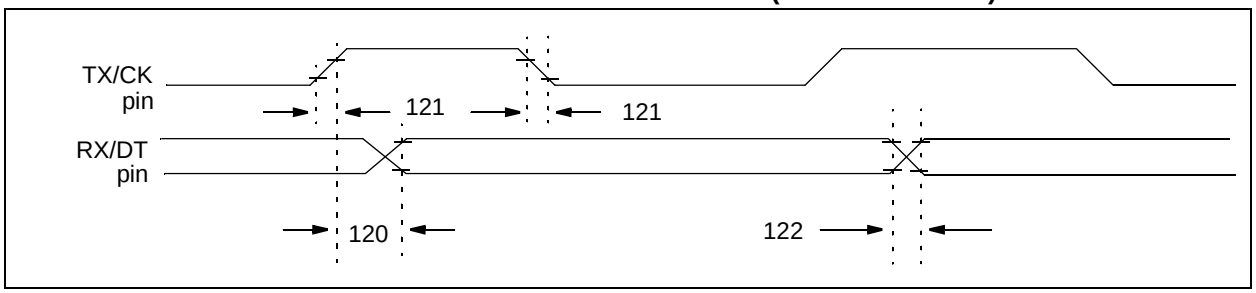



**TABLE 20-14: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS**

| Param No. | Sym | Characteristic | | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 120 | TckH2dtV | SYNC XMIT (MASTER & SLAVE) Clock high to data out valid | PIC17**C**XXX | — | — | 50 | ns | |
| | | | PIC17**LC**XXX | — | — | 75 | ns | |
| 121 | TckRF | Clock out rise time and fall time (Master mode) | PIC17**C**XXX | — | — | 25 | ns | |
| | | | PIC17**LC**XXX | — | — | 40 | ns | |
| 122 | TdtRF | Data out rise time and fall time | PIC17**C**XXX | — | — | 25 | ns | |
| | | | PIC17**LC**XXX | — | — | 40 | ns | |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.


© 2000 Microchip Technology Inc.

**FIGURE 20-20: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**

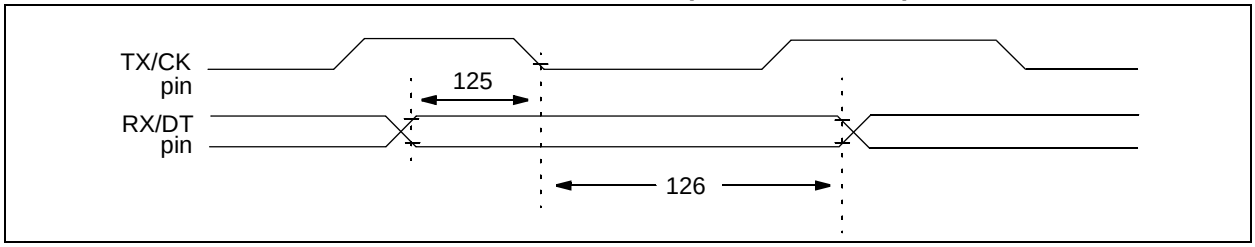



**TABLE 20-15: USART SYNCHRONOUS RECEIVE REQUIREMENTS**

| Param No. | Sym | Characteristic | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 125 | TdtV2ckL | SYNC RCV (MASTER & SLAVE)<br>Data setup before CK↓ (DT setup time) | 15 | — | — | ns | |
| 126 | TckL2dtl | Data hold after CK↓ (DT hold time) | 15 | — | — | ns | |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**FIGURE 20-21: USART ASYNCHRONOUS MODE START BIT DETECT**

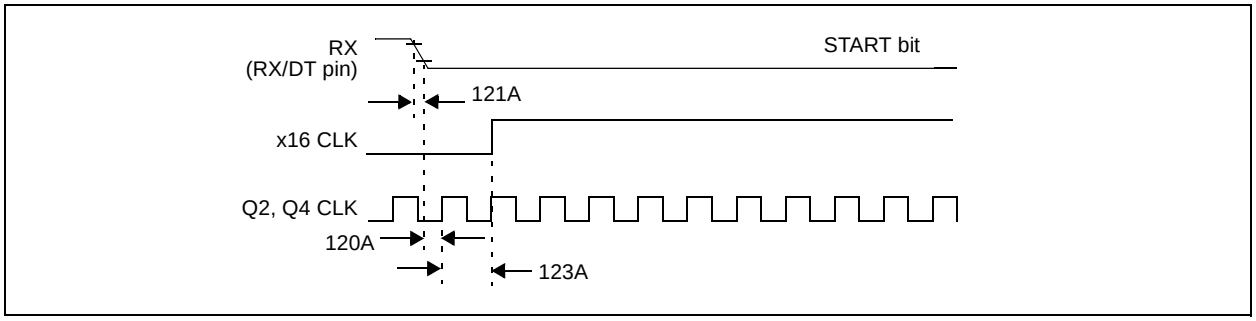



**TABLE 20-16: USART ASYNCHRONOUS MODE START BIT DETECT REQUIREMENTS**

| Param No. | Sym | Characteristic | | Min | Typ | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 120A | TdtL2ckH | Time to ensure that the RX pin is sampled low | | — | — | TCY | ns | |
| 121A | TdtRF | Data rise time and fall time | Receive | — | — | **(Note 1)** | ns | |
| | | | Transmit | — | — | 40 | ns | |
| 123A | TckH2bckL | Time from RX pin sampled low to first rising edge of x16 clock | | — | — | TCY | ns | |

**Note 1:** Schmitt trigger will determine logic level.

**FIGURE 20-22: USART ASYNCHRONOUS RECEIVE SAMPLING WAVEFORM**

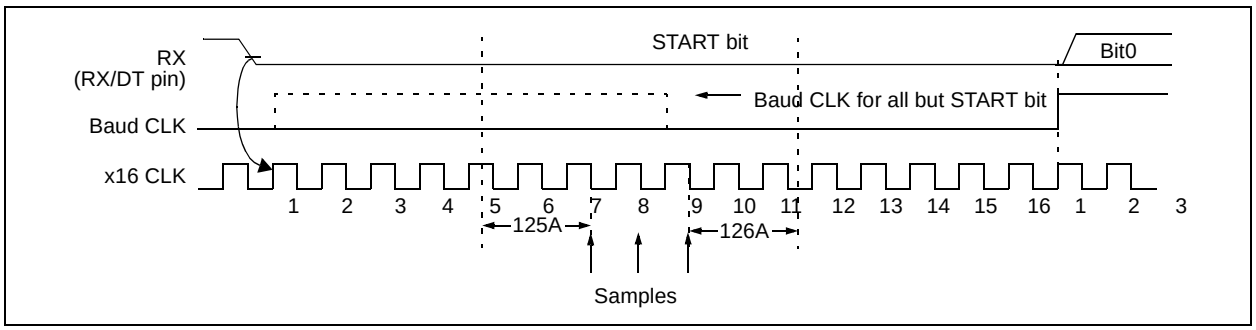



**TABLE 20-17: USART ASYNCHRONOUS RECEIVE SAMPLING REQUIREMENTS**

| Param No. | Sym | Characteristic | Min | Typ | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 125A | TdtL2ckH | Setup time of RX pin to first data sampled | TCY | — | — | ns | |
| 126A | TdtL2ckH | Hold time of RX pin from last data sampled | TCY | — | — | ns | |


© 2000 Microchip Technology Inc.

**TABLE 20-18: A/D CONVERTER CHARACTERISTICS**

| Param. No. | Sym | Characteristic | | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| A01 | NR | Resolution | | — | — | 10 | bit | VREF+ = VDD = 5.12V, VSS ≤ VAIN ≤ VREF+ |
| | | | | — | — | 10 | bit | (VREF+ — VREF-) ≥ 3.0V, VREF- ≤ VAIN ≤ VREF+ |
| A02 | EABS | Absolute error | | — | — | < ±1 | LSb | VREF+ = VDD = 5.12V, VSS ≤ VAIN ≤ VREF+ |
| | | | | — | — | < ±1 | LSb | (VREF+ — VREF-) ≥ 3.0V, VREF- ≤ VAIN ≤ VREF+ |
| A03 | EIL | Integral linearity error | | — | — | < ±1 | LSb | VREF+ = VDD = 5.12V, VSS ≤ VAIN ≤ VREF+ |
| | | | | — | — | < ±1 | LSb | (VREF+ — VREF-) ≥ 3.0V, VREF- ≤ VAIN ≤ VREF+ |
| A04 | EDL | Differential linearity error | | — | — | < ±1 | LSb | VREF+ = VDD = 5.12V, VSS ≤ VAIN ≤ VREF+ |
| | | | | — | — | < ±1 | LSb | (VREF+ — VREF-) ≥ 3.0V, VREF- ≤ VAIN ≤ VREF+ |
| A05 | EFS | Full scale error | | — | — | < ±1 | LSb | VREF+ = VDD = 5.12V, VSS ≤ VAIN ≤ VREF+ |
| | | | | — | — | < ±1 | LSb | (VREF+ — VREF-) ≥ 3.0V, VREF- ≤ VAIN ≤ VREF+ |
| A06 | EOFF | Offset error | | — | — | < ±1 | LSb | VREF+ = VDD = 5.12V, VSS ≤ VAIN ≤ VREF+ |
| | | | | — | — | < ±1 | LSb | (VREF+ — VREF-) ≥ 3.0V, VREF- ≤ VAIN ≤ VREF+ |
| A10 | — | Monotonicity | | — | guaranteed(3) | — | — | VSS ≤ VAIN ≤ VREF |
| A20 | VREF | Reference voltage (VREF+ — VREF-) | | 0V | — | — | V | VREF delta when changing voltage levels on VREF inputs |
| A20A | | | | 3V | — | — | V | Absolute minimum electrical spec. to ensure 10-bit accuracy |
| A21 | VREF+ | Reference voltage high | | AVSS + 3.0V | — | AVDD + 0.3V | V | |
| A22 | VREF- | Reference voltage low | | Avss - 0.3V | — | AVDD - 3.0V | V | |
| A25 | VAIN | Analog input voltage | | AVSS - 0.3V | — | Vref + 0.3V | V | |
| A30 | ZAIN | Recommended impedance of analog voltage source | | — | — | 10.0 | kΩ | |
| A40 | IAD | A/D conversion current (VDD) | PIC17**C**XXX | — | 180 | — | µA | Average current consumption when A/D is on **(Note 1)** |
| | | | PIC17**LC**XXX | — | 90 | — | µA | |
| A50 | IREF | VREF input current **(Note 2)** | | 10 | — | 1000 | µA | During VAIN acquisition. Based on differential of VHOLD to VAIN |
| | | | | — | — | 10 | µA | During A/D conversion cycle |

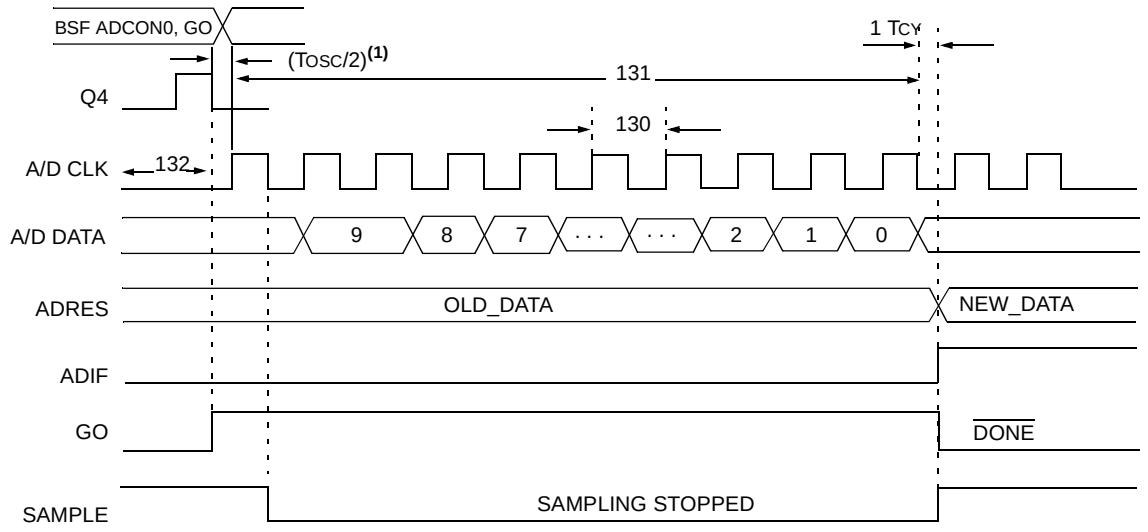
† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**Note 1:** When A/D is off, it will not consume any current other than minor leakage current. The power-down current spec includes any such leakage from the A/D module.

**2:** VREF current is from RG0 and RG1 pins or AVDD and AVSS pins, whichever is selected as reference input.

**3:** The A/D conversion result never decreases with an increase in the Input Voltage and has no missing codes.

**FIGURE 20-23: A/D CONVERSION TIMING**




**Note 1:** If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.

**TABLE 20-19: A/D CONVERSION REQUIREMENTS**

| Param. No. | Sym | Characteristic | | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 130 | TAD | A/D clock period | PIC17**C**XXX | 1.6 | — | — | µs | TOSC based, VREF ≥ 3.0V |
| | | | PIC17**LC**XXX | 3.0 | — | — | µs | TOSC based, VREF full range |
| | | | PIC17**C**XXX | 2.0 | 4.0 | 6.0 | µs | A/D RC mode |
| | | | PIC17**LC**XXX | 3.0 | 6.0 | 9.0 | µs | A/D RC mode |
| 131 | TCNV | Conversion time (not including acquisition time) **(Note 1)** | | 11 | — | 12 | Tad | |
| 132 | TACQ | Acquisition time | | **(Note 2)** | 20 | — | µs | |
| | | | | 10 | — | — | µs | The minimum time is the amplifier settling time. This may be used if the "new" input voltage has not changed by more than 1LSb (i.e., 5 mV @ 5.12V) from the last sampled voltage (as stated on CHOLD). |
| 134 | TGO | Q4 to ADCLK start | | — | Tosc/2 | — | — | If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed. |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**Note 1:** ADRES register may be read on the following TCY cycle.

**2:** See Section 16.1 for minimum conditions when input voltage has changed more than 1 LSb.


© 2000 Microchip Technology Inc.

**FIGURE 20-24: MEMORY INTERFACE WRITE TIMING**

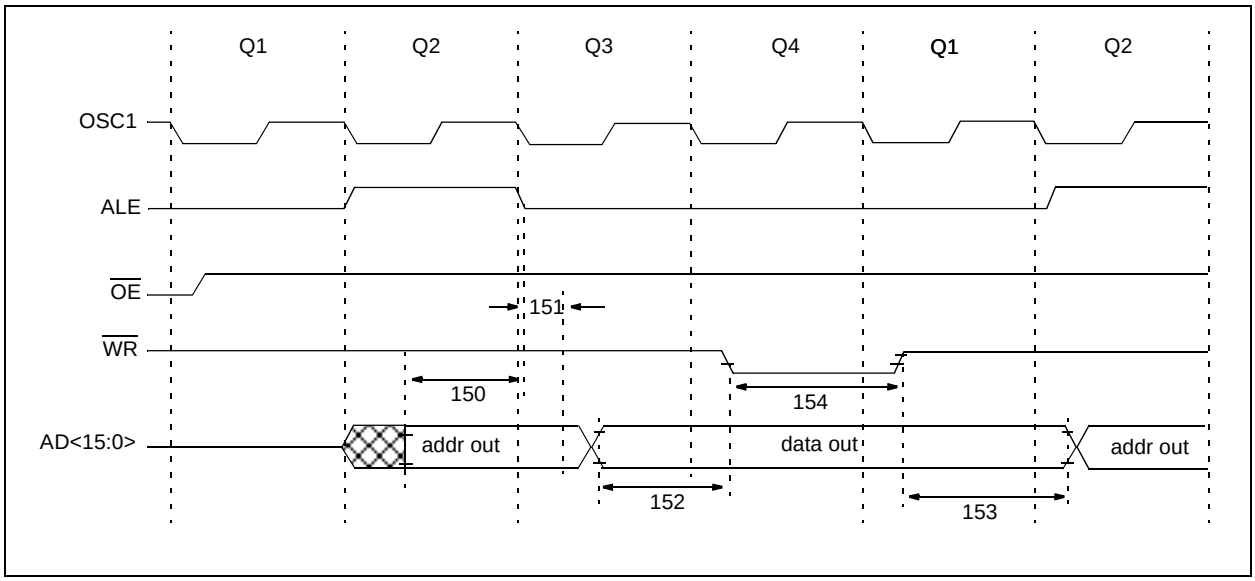



**TABLE 20-20: MEMORY INTERFACE WRITE REQUIREMENTS**

| Param. No. | Sym | Characteristic | | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 150 | TadV2alL | AD<15:0> (address) valid to ALE↓ (address setup time) | PIC17**C**XXX | 0.25TCY - 10 | — | — | ns | |
| | | | PIC17**LC**XXX | 0.25TCY - 10 | — | — | | |
| 151 | TalL2adI | ALE↓ to address out invalid (address hold time) | PIC17**C**XXX | 0 | — | — | ns | |
| | | | PIC17**LC**XXX | 0 | — | — | | |
| 152 | TadV2wrL | Data out valid to $\overline{WR}$↓ (data setup time) | PIC17**C**XXX | 0.25TCY - 40 | — | — | ns | |
| | | | PIC17**LC**XXX | 0.25TCY - 40 | — | — | | |
| 153 | TwrH2adI | $\overline{WR}$↑ to data out invalid (data hold time) | PIC17**C**XXX | — | 0.25TCY | — | ns | |
| | | | PIC17**LC**XXX | — | 0.25TCY | — | | |
| 154 | TwrL | $\overline{WR}$ pulse width | PIC17**C**XXX | — | 0.25TCY | — | ns | |
| | | | PIC17**LC**XXX | — | 0.25TCY | — | | |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

**FIGURE 20-25: MEMORY INTERFACE READ TIMING**

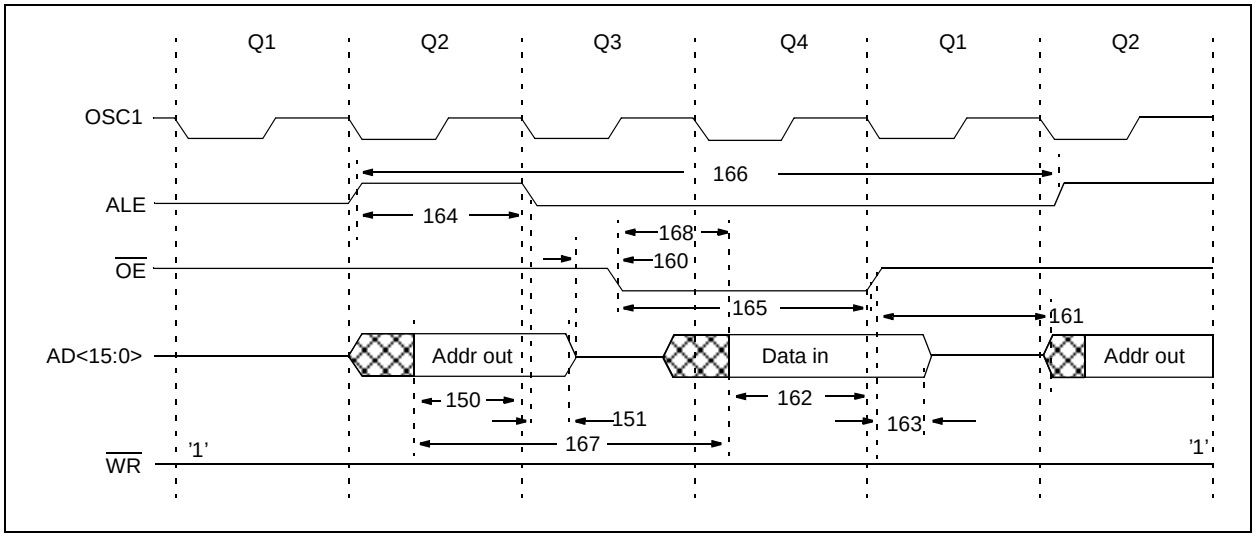



**TABLE 20-21: MEMORY INTERFACE READ REQUIREMENTS**

| Param. No. | Sym | Characteristic | | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 150 | TadV2alL | AD15:AD0 (address) valid to ALE↓ (address setup time) | PIC17**C**XXX | 0.25TCY - 10 | — | — | ns | |
| | | | PIC17**LC**XXX | 0.25TCY - 10 | — | — | | |
| 151 | TalL2adI | ALE↓ to address out invalid (address hold time) | PIC17**C**XXX | 5 | — | — | ns | |
| | | | PIC17**LC**XXX | 5 | — | — | | |
| 160 | TadZ2oeL | AD15:AD0 hi-impedance to $\overline{OE}$↓ | PIC17**C**XXX | 0 | — | — | ns | |
| | | | PIC17**LC**XXX | 0 | — | — | | |
| 161 | ToeH2adD | $\overline{OE}$↑ to AD15:AD0 driven | PIC17**C**XXX | 0.25TCY - 15 | — | — | ns | |
| | | | PIC17**LC**XXX | 0.25TCY - 15 | — | — | | |
| 162 | TadV2oeH | Data in valid before $\overline{OE}$↑ (data setup time) | PIC17**C**XXX | 35 | — | — | ns | |
| | | | PIC17**LC**XXX | 45 | — | — | | |
| 163 | ToeH2adI | $\overline{OE}$↑to data in invalid (data hold time) | PIC17**C**XXX | 0 | — | — | ns | |
| | | | PIC17**LC**XXX | 0 | — | — | | |
| 164 | TalH | ALE pulse width | PIC17**C**XXX | — | 0.25TCY | — | ns | |
| | | | PIC17**LC**XXX | — | 0.25TCY | — | | |
| 165 | ToeL | $\overline{OE}$ pulse width | PIC17**C**XXX | 0.5TCY - 35 | — | — | ns | |
| | | | PIC17**LC**XXX | 0.5TCY - 35 | — | — | | |
| 166 | TalH2alH | ALE↑ to ALE↑(cycle time) | PIC17**C**XXX | — | TCY | — | ns | |
| | | | PIC17**LC**XXX | — | TCY | — | | |
| 167 | Tacc | Address access time | PIC17**C**XXX | — | — | 0.75TCY - 30 | ns | |
| | | | PIC17**LC**XXX | — | — | 0.75TCY - 45 | | |
| 168 | Toe | Output enable access time ($\overline{OE}$ low to data valid) | PIC17**C**XXX | — | — | 0.5TCY - 45 | ns | |
| | | | PIC17**LC**XXX | — | — | 0.5TCY - 75 | | |

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.


© 2000 Microchip Technology Inc.

## 21.0 PIC17C7XX DC AND AC CHARACTERISTICS

The graphs and tables provided in this section are for design guidance and are not tested nor guaranteed. In some graphs or tables the data presented is outside specified operating range (e.g., outside specified VDD range). This is for information only and devices are ensured to operate properly only within the specified range.

The data presented in this section is a statistical summary of data collected on units from different lots over a period of time.

- **Typ** or **Typical** represents the mean of the distribution at 25°C.
- **Max** or **Maximum** represents (mean + 3σ) over the temperature range of -40°C to 85°C.
- **Min** or **Minimum** represents (mean - 3σ) over the temperature range of -40°C to 85°C.

**Note:** Standard deviation is denoted by sigma (σ).

**TABLE 21-1: PIN CAPACITANCE PER PACKAGE TYPE**

| Pin Name | Typical Capacitance (pF) | |
|---|---|---|
| | 68-pin PLCC | 64-pin TQFP |
| All pins, except $\overline{MCLR}$, VDD, and VSS | 10 | 10 |
| $\overline{MCLR}$ pin | 20 | 20 |

**FIGURE 21-1: TYPICAL RC OSCILLATOR FREQUENCY vs. TEMPERATURE**

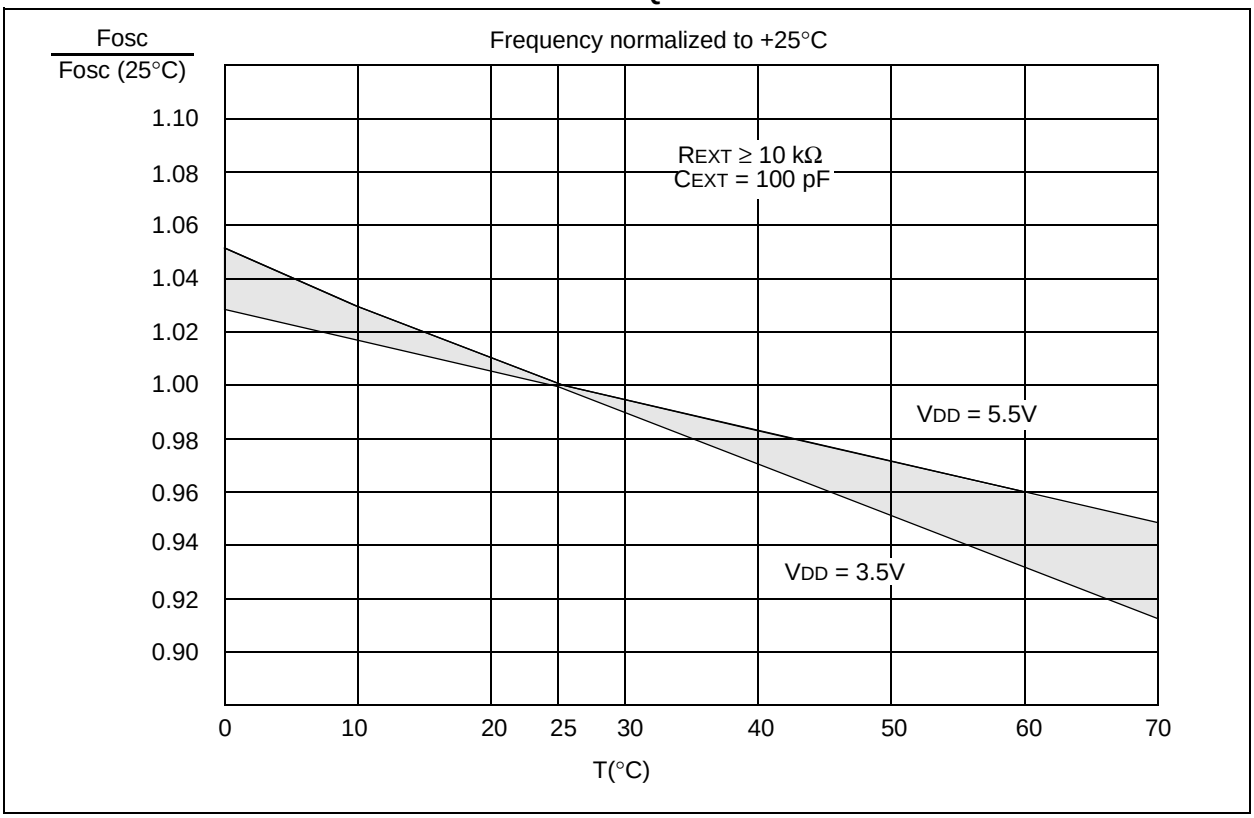


**FIGURE 21-2: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD**

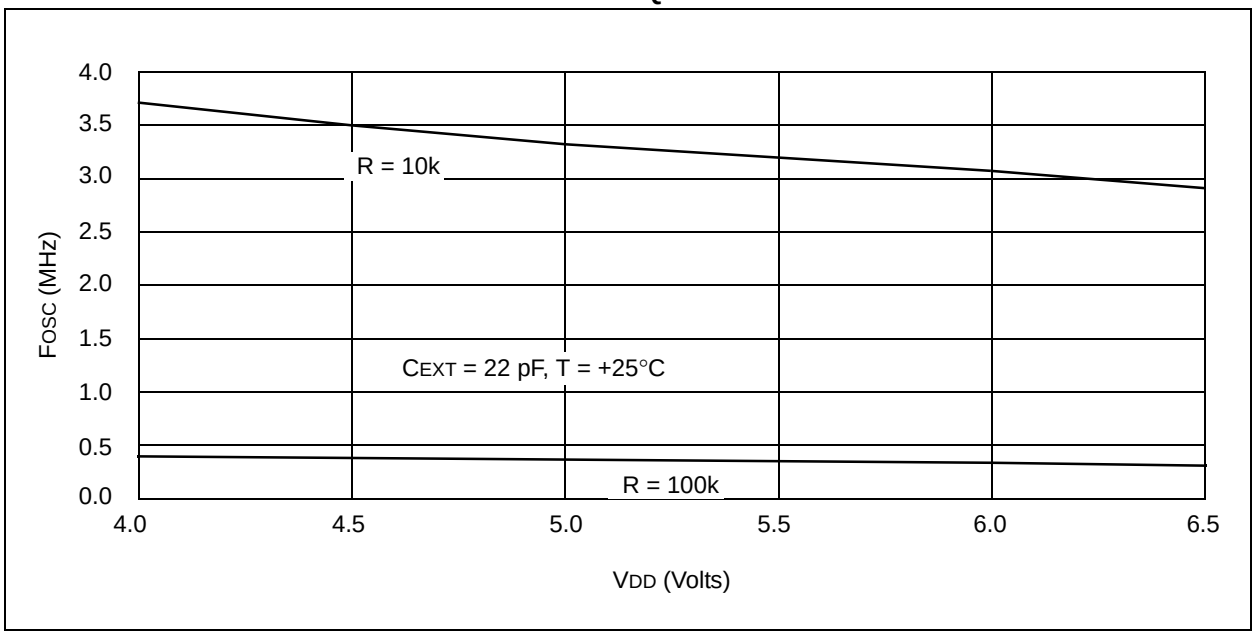
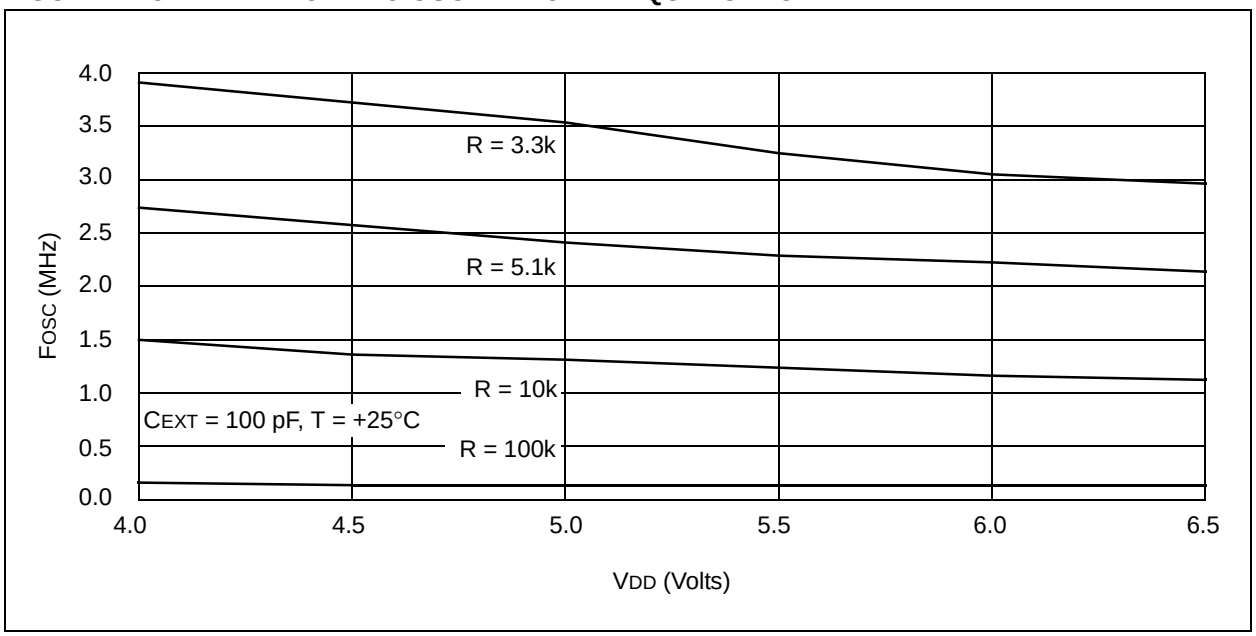



**FIGURE 21-3: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD**





© 2000 Microchip Technology Inc.

**FIGURE 21-4: TYPICAL RC OSCILLATOR FREQUENCY vs. VDD**

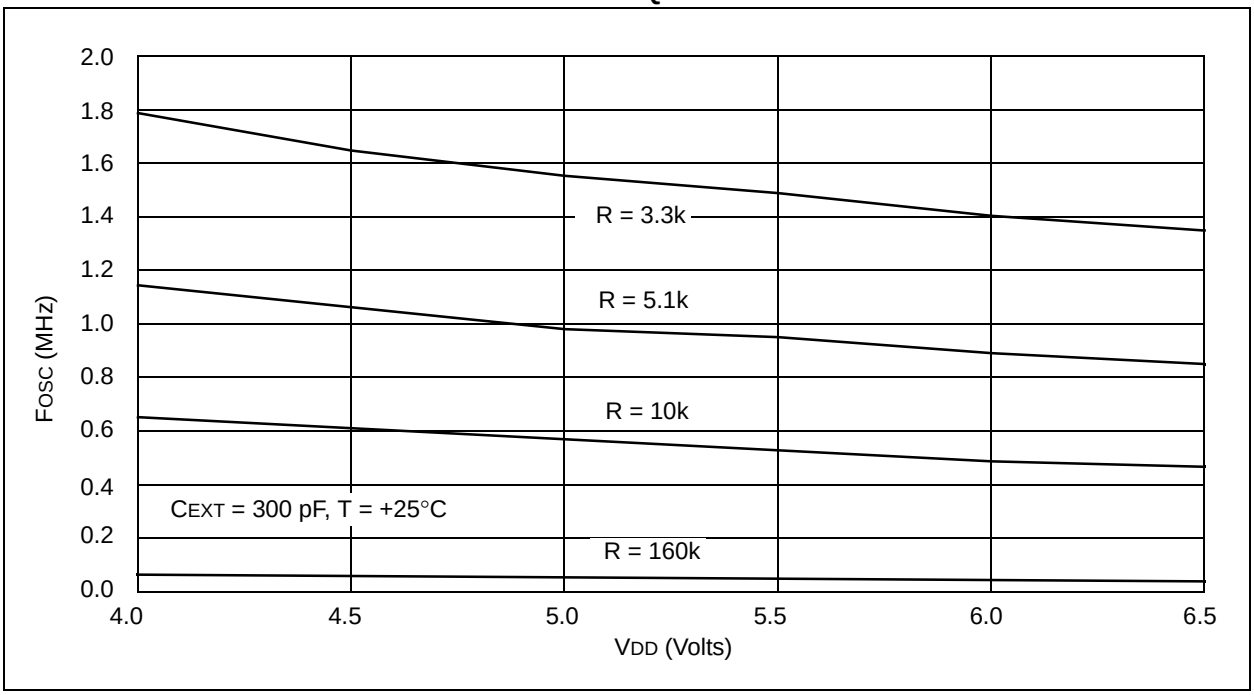



**TABLE 21-2: RC OSCILLATOR FREQUENCIES**

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

© 2000 Microchip Technology Inc.

**FIGURE 21-5: TRANSCONDUCTANCE (gm) OF LF OSCILLATOR vs. VDD**

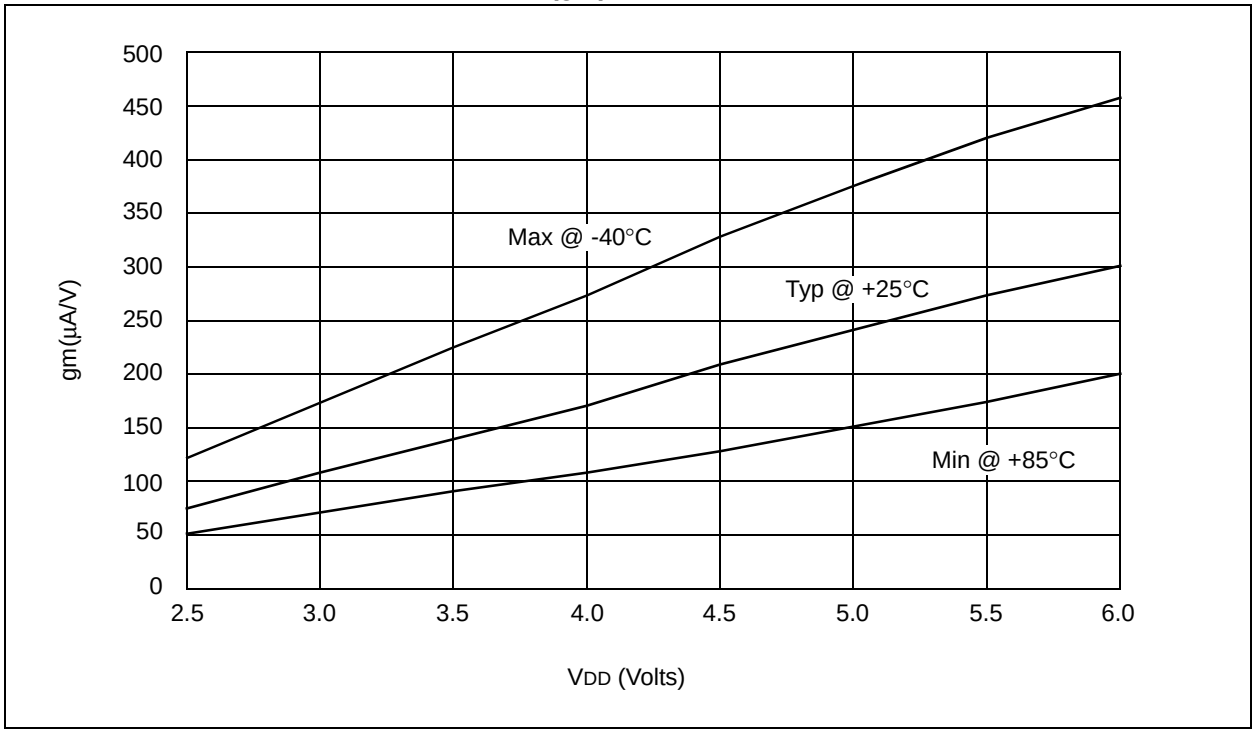
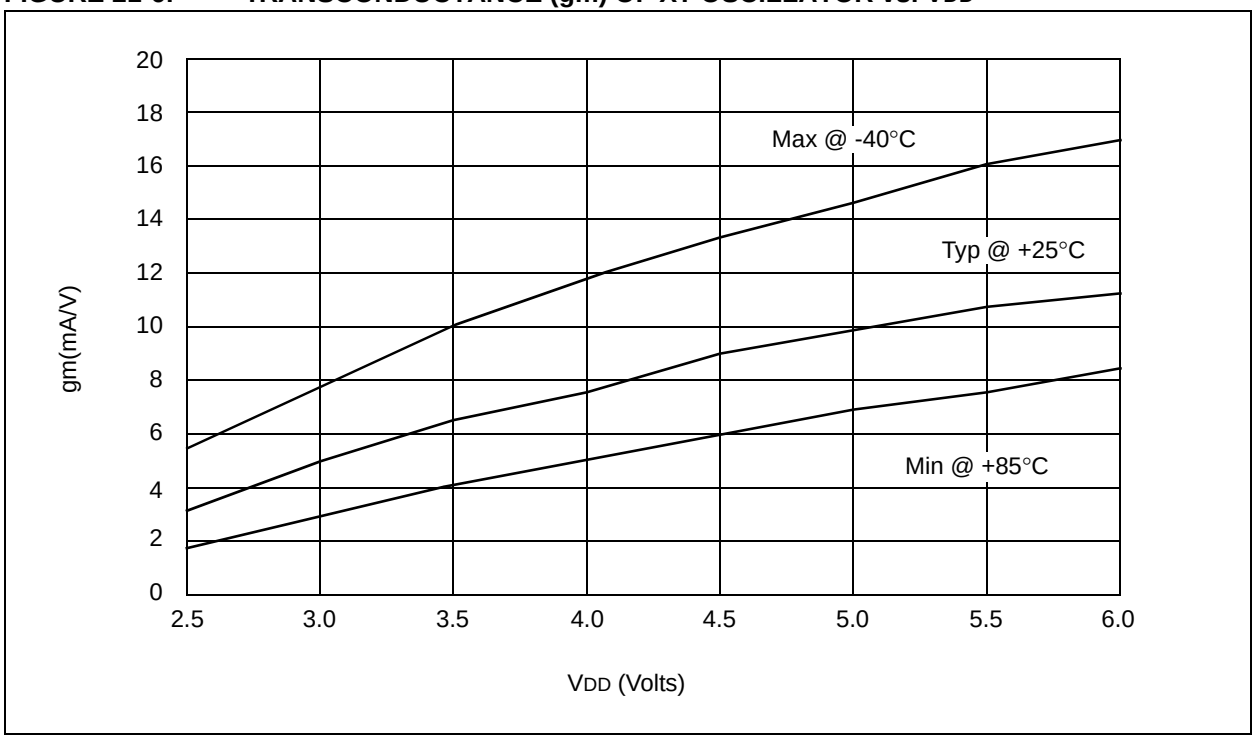



**FIGURE 21-6: TRANSCONDUCTANCE (gm) OF XT OSCILLATOR vs. VDD**





© 2000 Microchip Technology Inc.

**FIGURE 21-7: TYPICAL IDD vs. FOSC OVER VDD (LF MODE)**

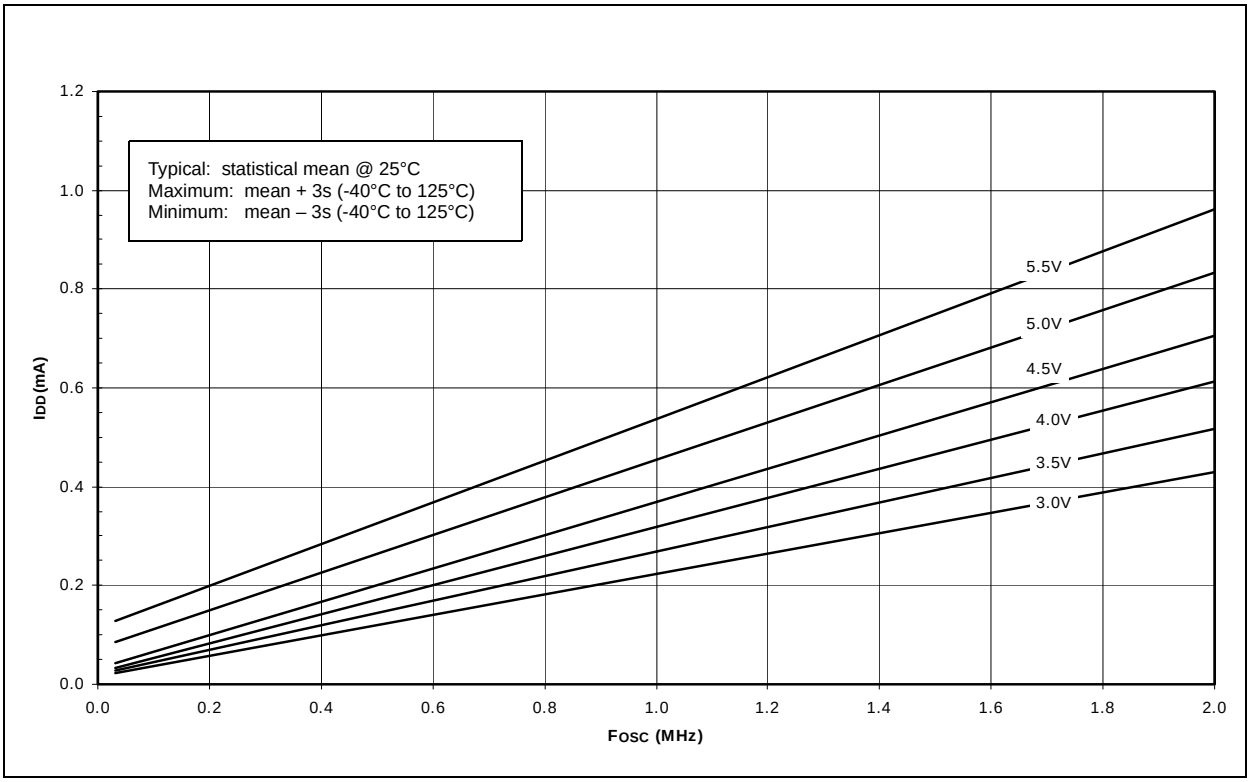



**FIGURE 21-8: MAXIMUM IDD vs. FOSC OVER VDD (LF MODE)**

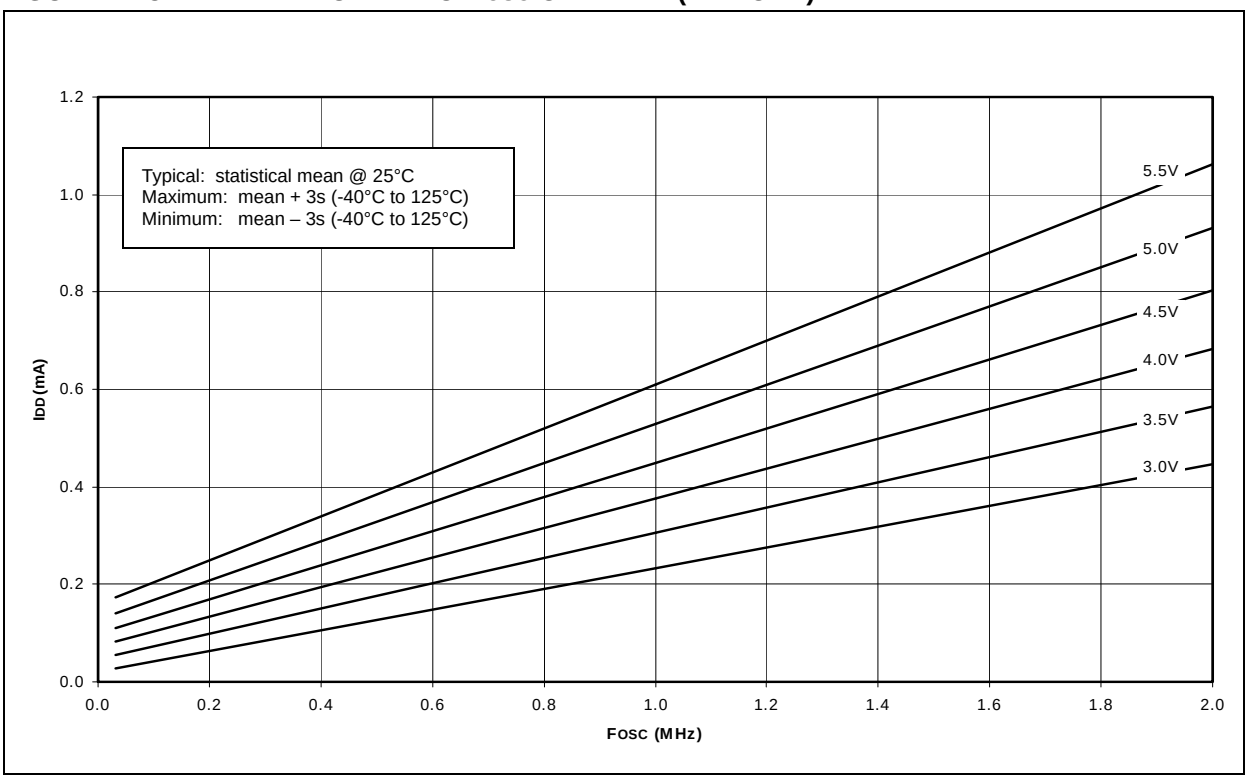


**FIGURE 21-9: TYPICAL IDD vs. FOSC OVER VDD (XT MODE)**

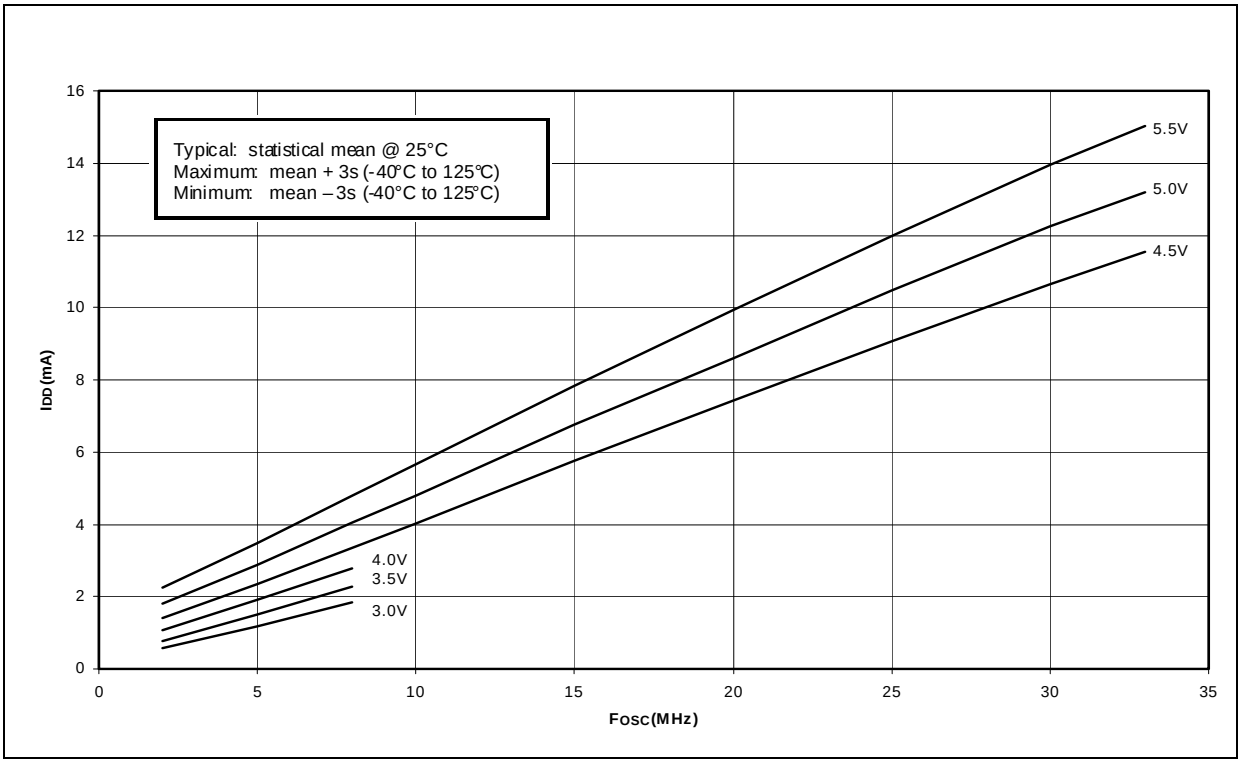
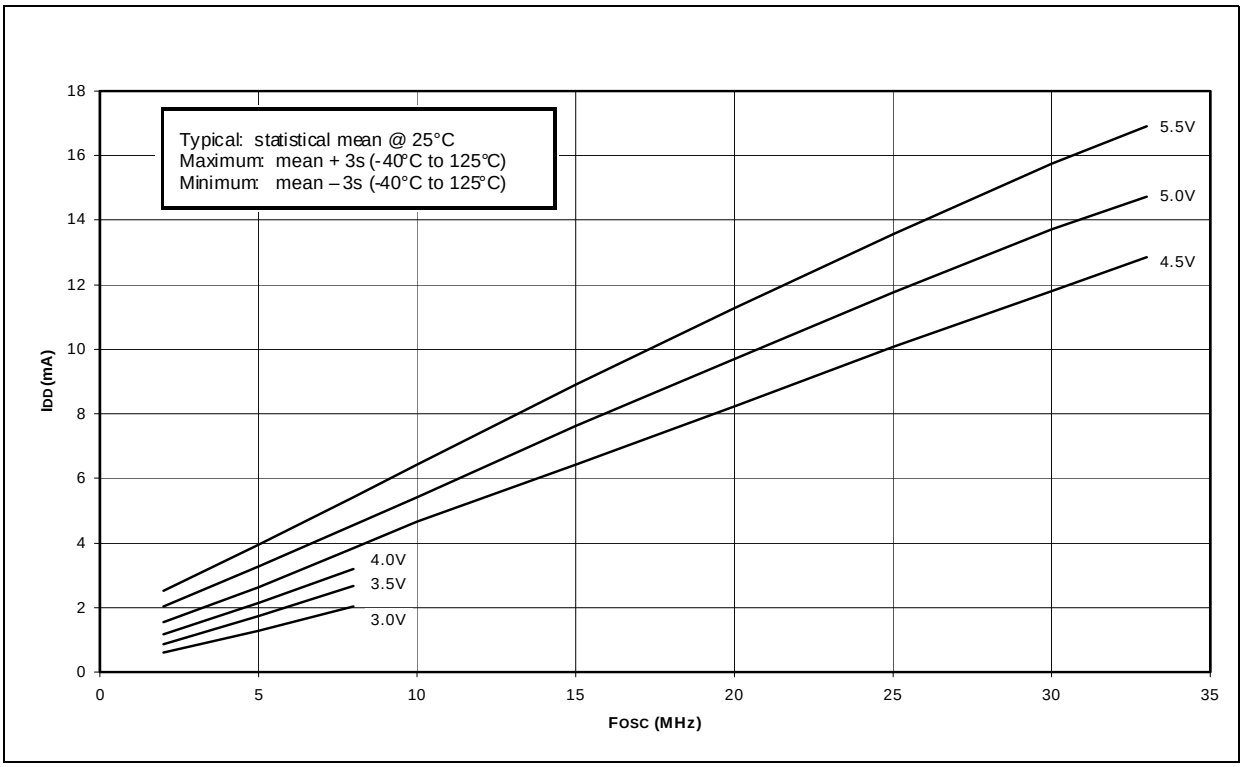



**FIGURE 21-10: MAXIMUM IDD vs. FOSC OVER VDD (XT MODE)**





© 2000 Microchip Technology Inc.

**FIGURE 21-11: TYPICAL AND MAXIMUM IPD vs. VDD (SLEEP MODE, ALL PERIPHERALS DISABLED, -40°C to +125°C)**

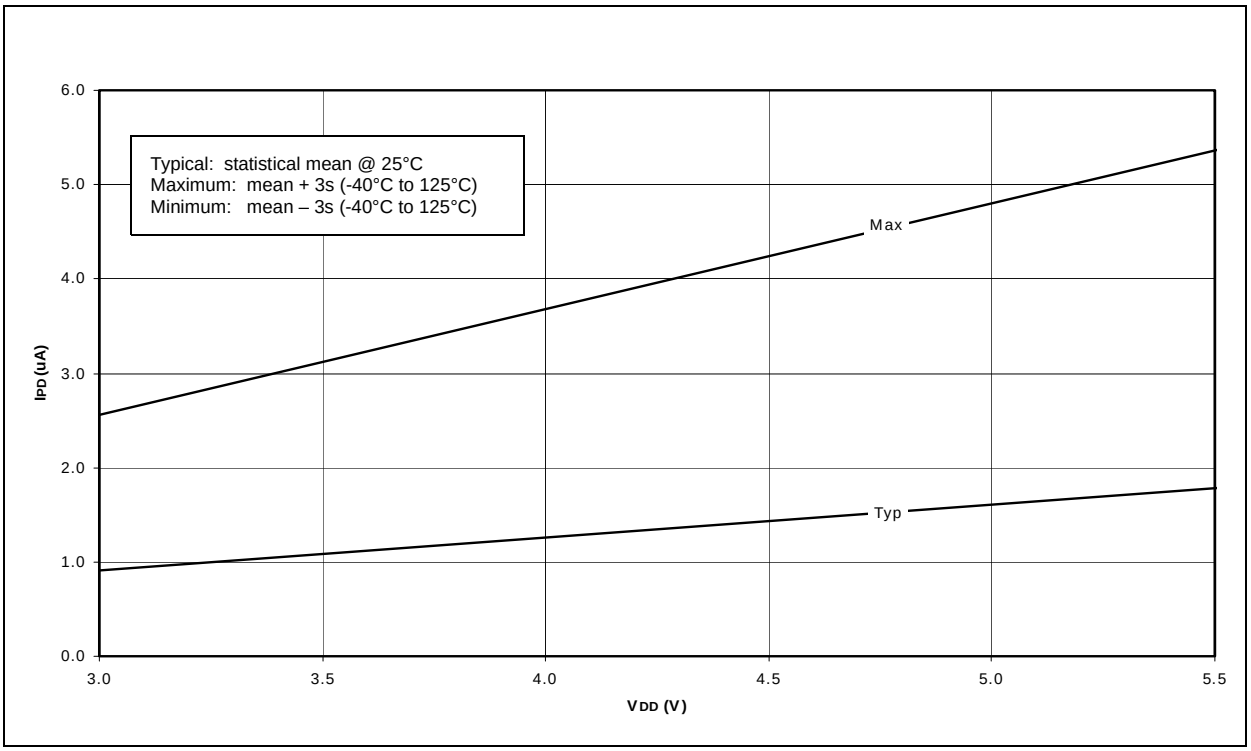



**FIGURE 21-12: TYPICAL AND MAXIMUM IPD vs. VDD (SLEEP MODE, BOR ENABLED, -40°C to +125°C)**

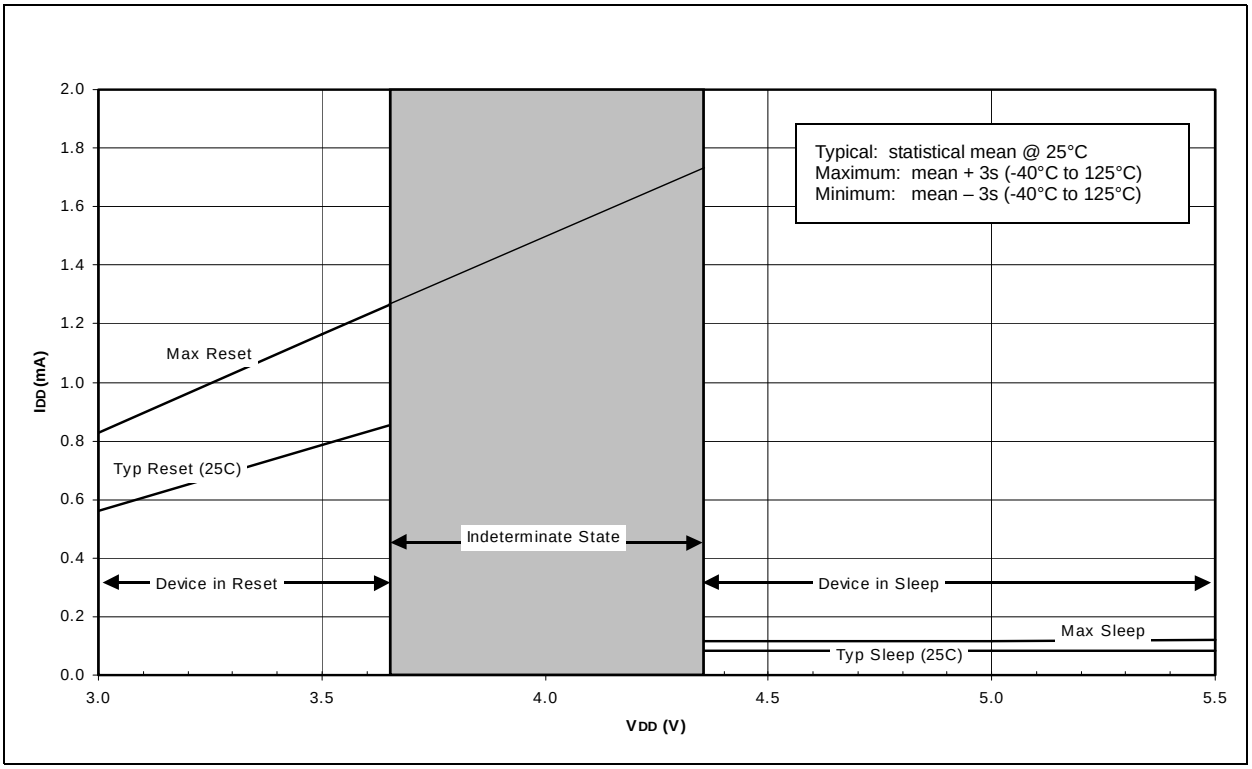


**FIGURE 21-13: TYPICAL AND MAXIMUM ΔIPD vs. VDD (SLEEP MODE, WDT ENABLED, -40°C to +125°C)**

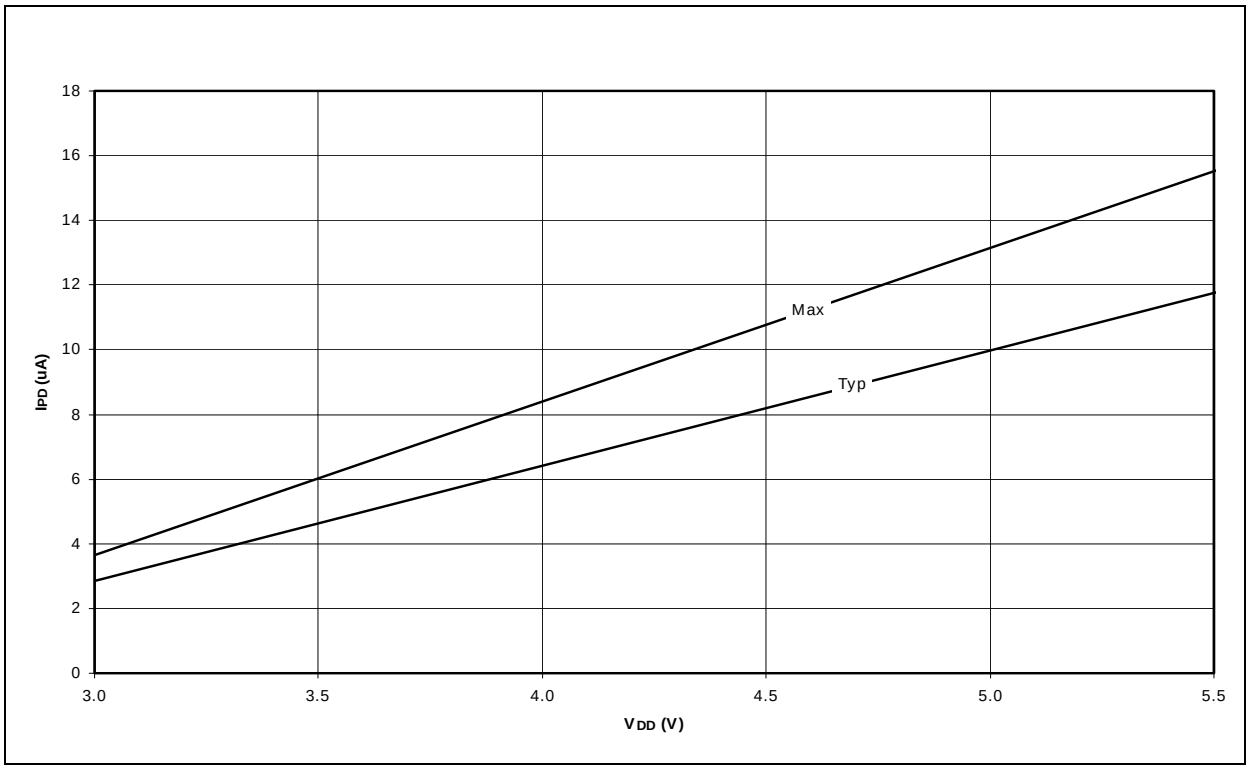
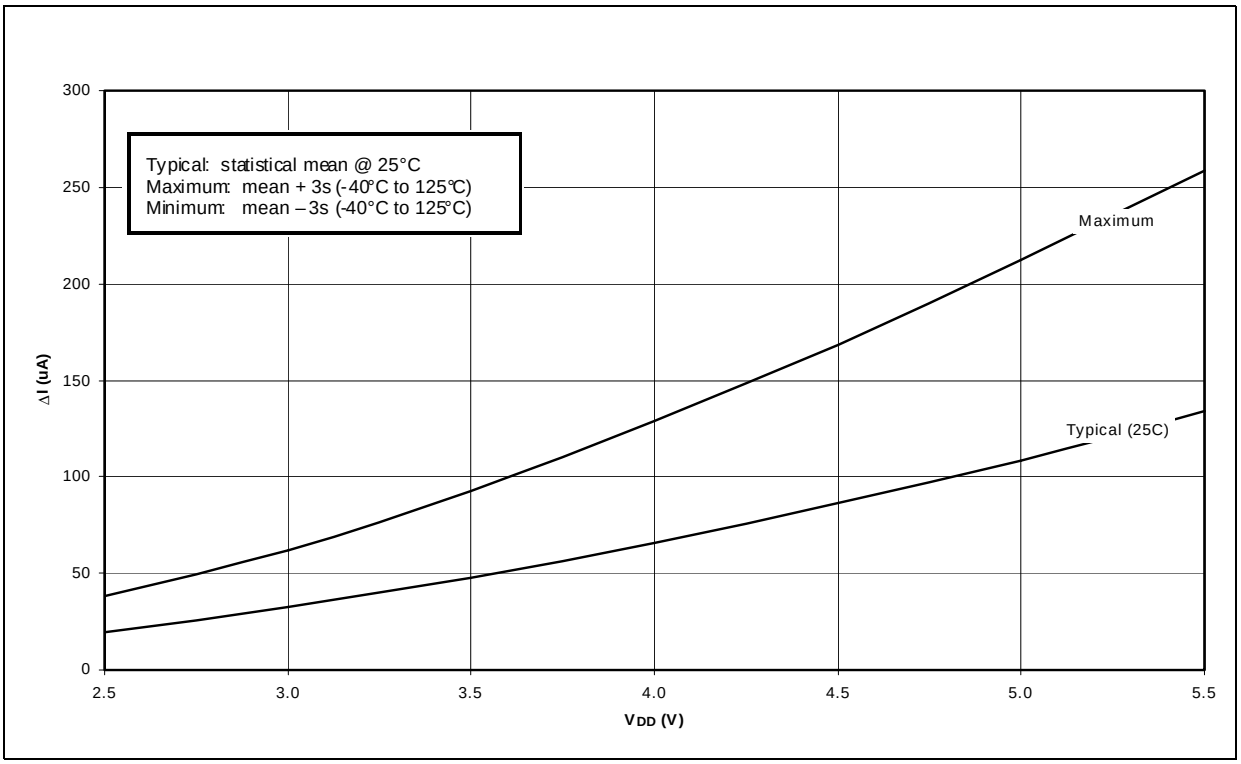



**FIGURE 21-14: TYPICAL AND MAXIMUM ΔIRBPU vs. VDD (MEASURED PER INPUT PIN, -40°C TO +125°C)**





© 2000 Microchip Technology Inc.

**FIGURE 21-15: TYPICAL, MINIMUM AND MAXIMUM WDT PERIOD vs. VDD (-40°C TO +125°C)**

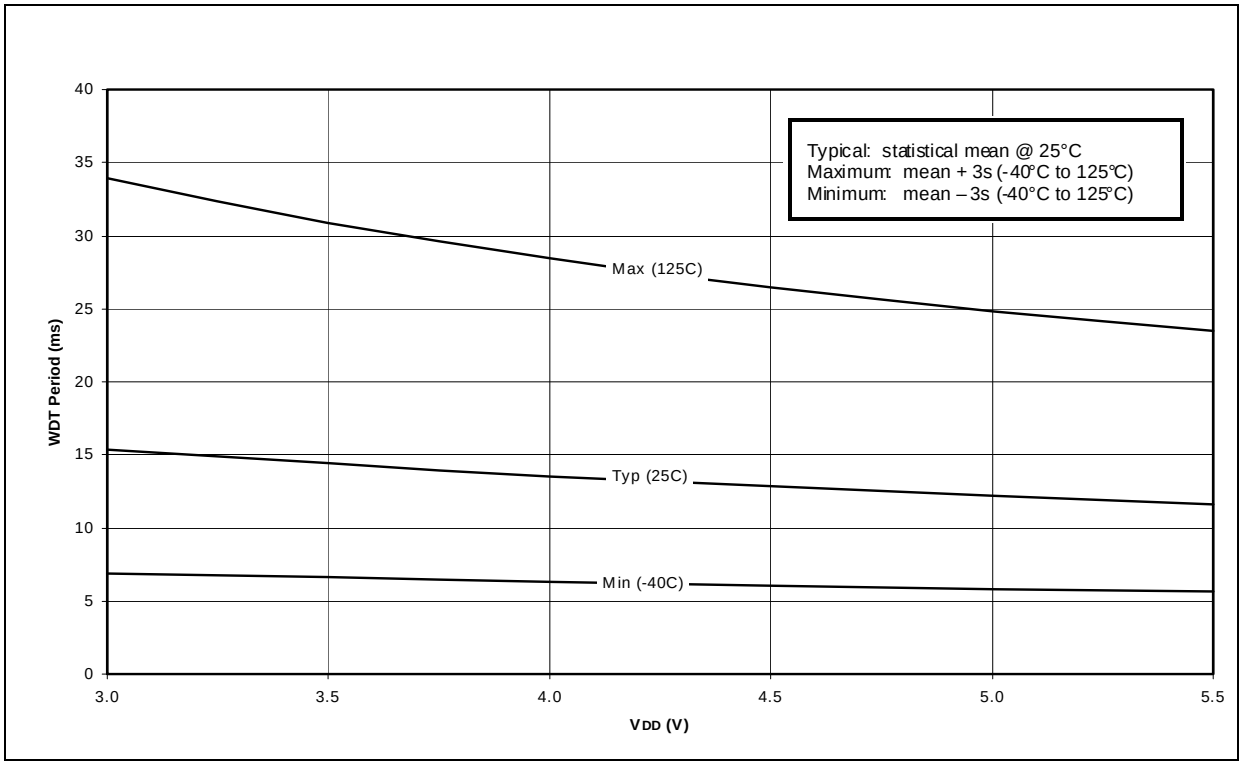



**FIGURE 21-16: TYPICAL WDT PERIOD vs. VDD OVER TEMPERATURE (-40°C TO +125°C)**

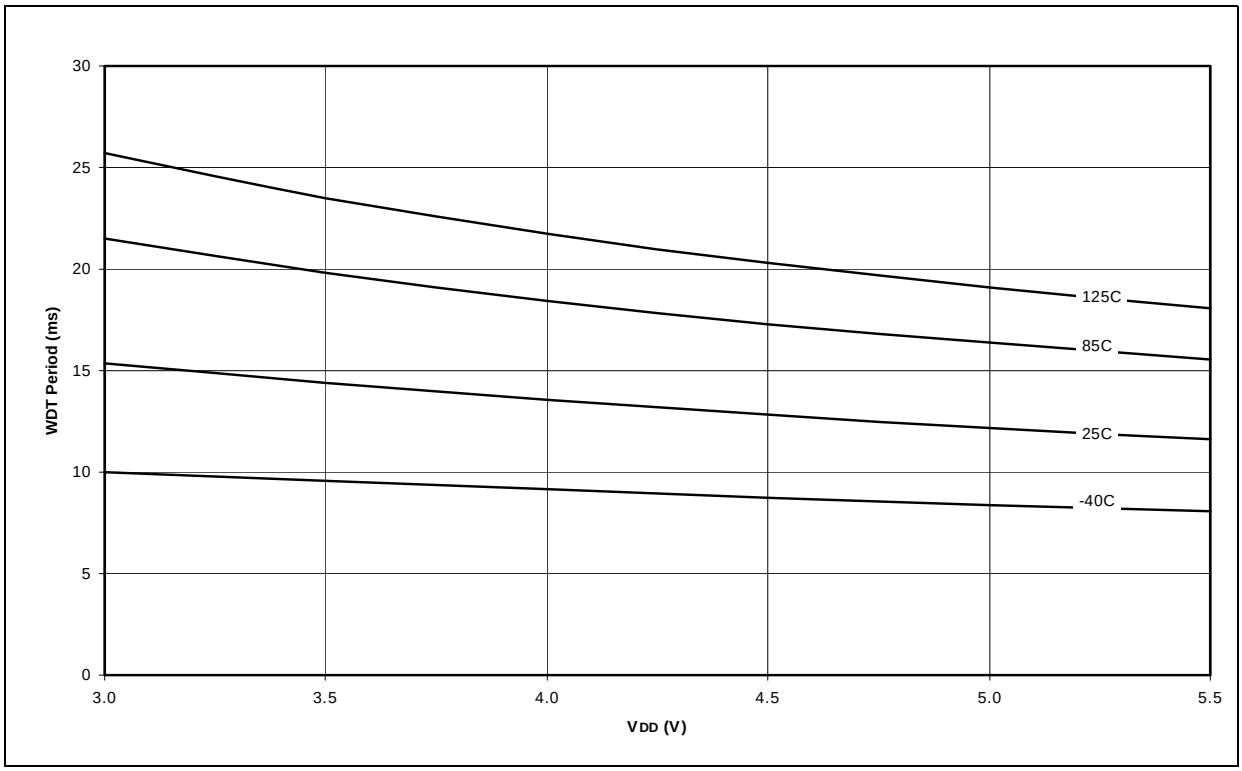


**FIGURE 21-17: TYPICAL, MINIMUM AND MAXIMUM $V_{OH}$ vs. $I_{OH}$ ($V_{DD}$ = 5V, -40°C TO +125°C)**

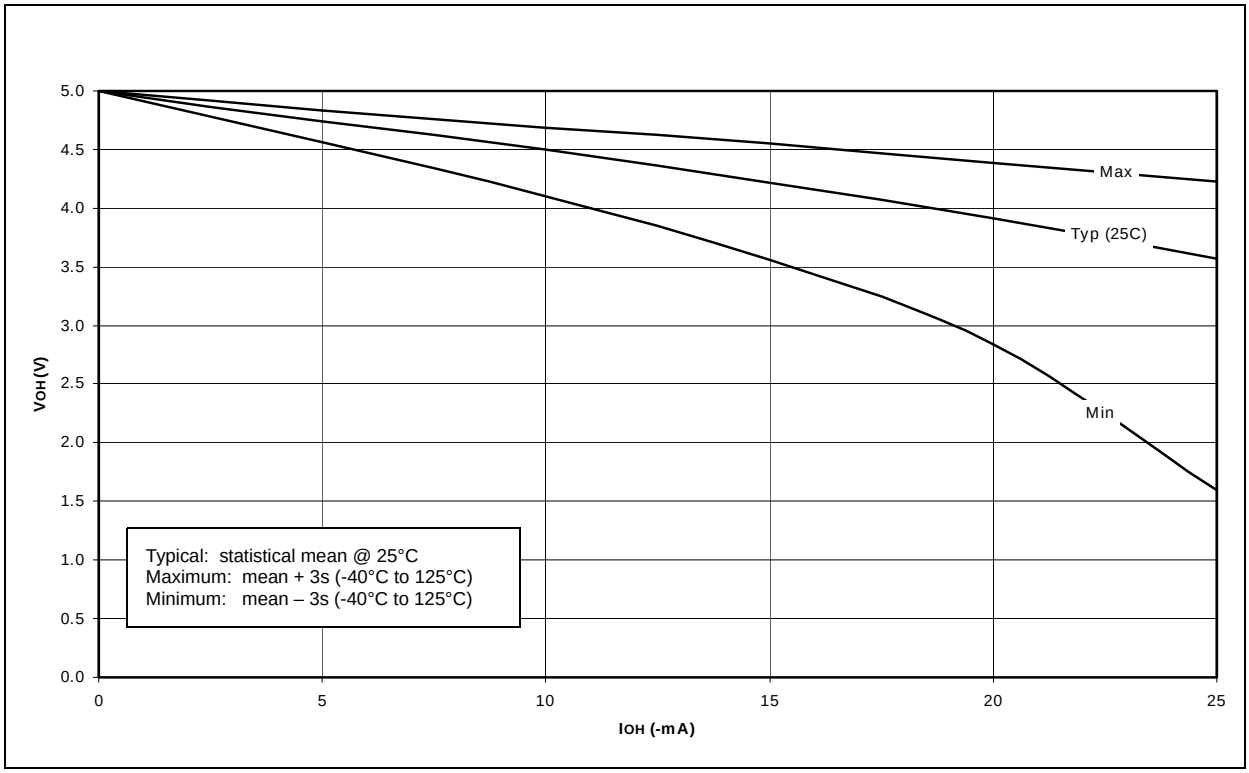
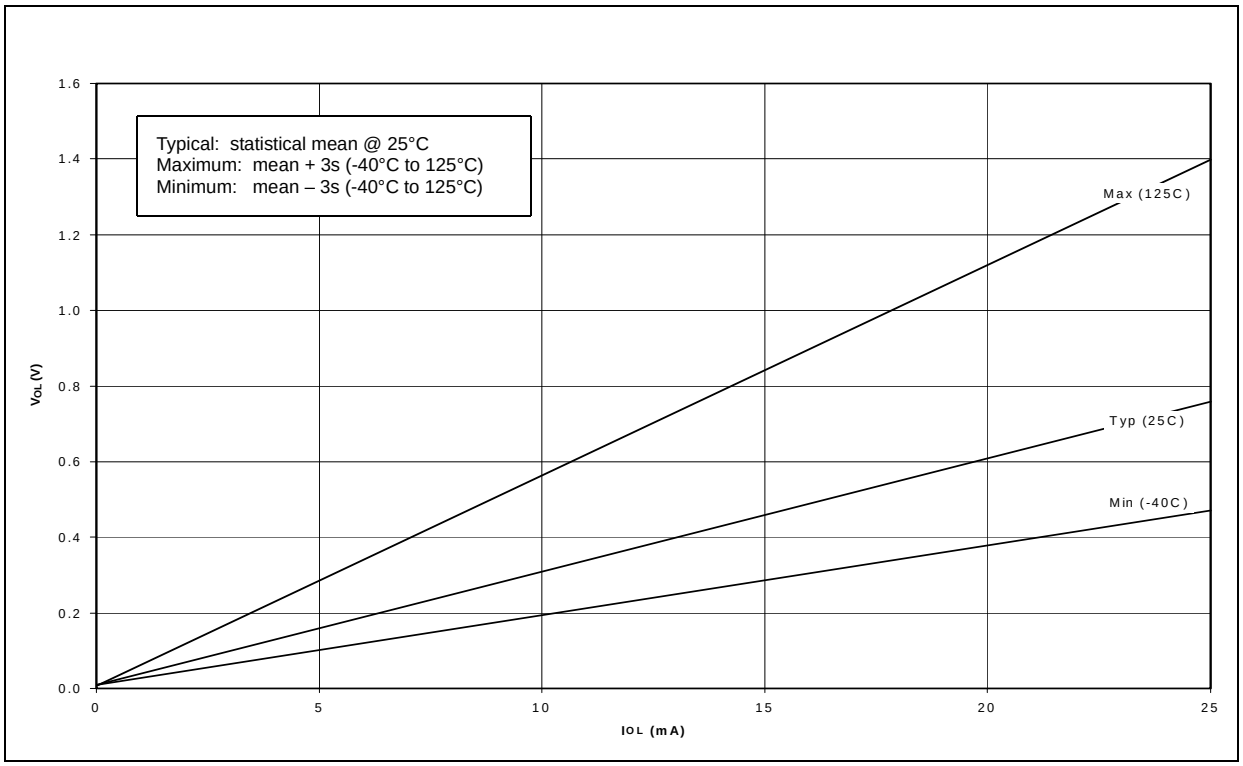



**FIGURE 21-18: TYPICAL, MINIMUM AND MAXIMUM $V_{OL}$ vs. $I_{OL}$ ($V_{DD}$ = 5V, -40°C TO +125°C)**





© 2000 Microchip Technology Inc.

**FIGURE 21-19: TYPICAL, MINIMUM AND MAXIMUM $V_{OH}$ vs. $I_{OH}$ ($V_{DD}$ = 3V, -40°C TO +125°C)**

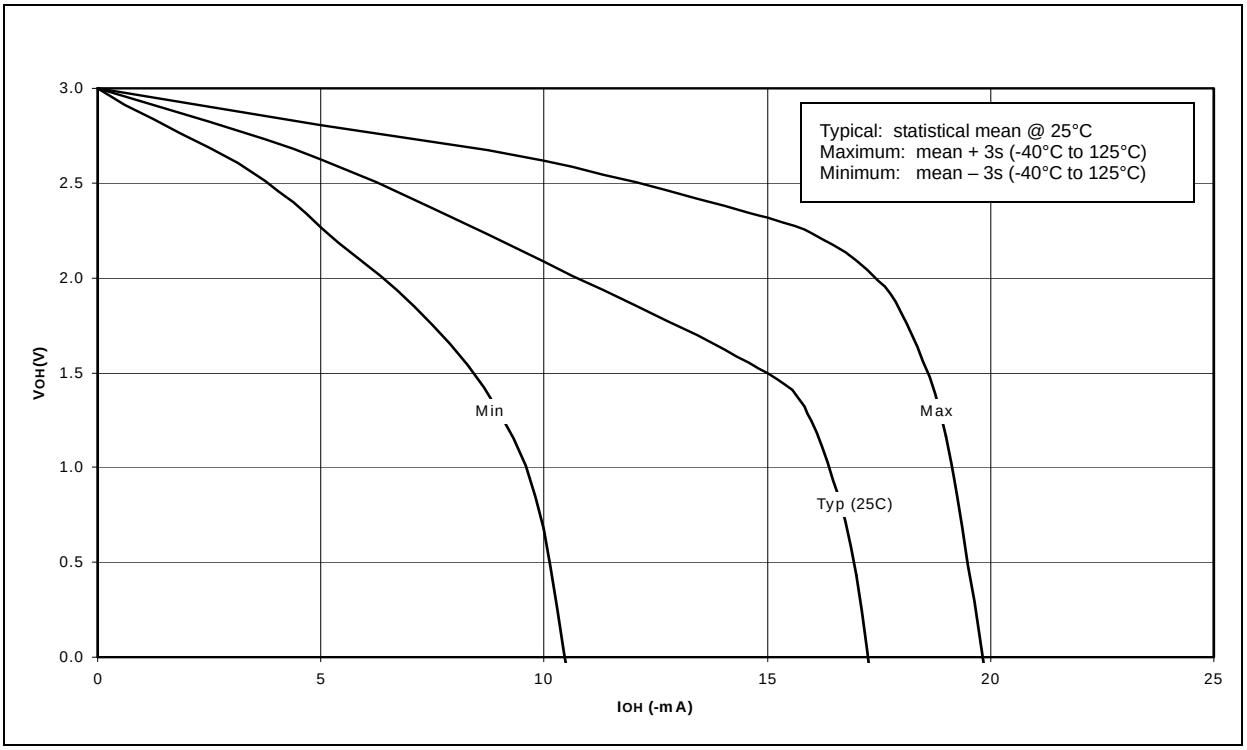



**FIGURE 21-20: TYPICAL, MINIMUM AND MAXIMUM $V_{OL}$ vs. $I_{OL}$ ($V_{DD}$ = 3V, -40°C TO +125°C)**

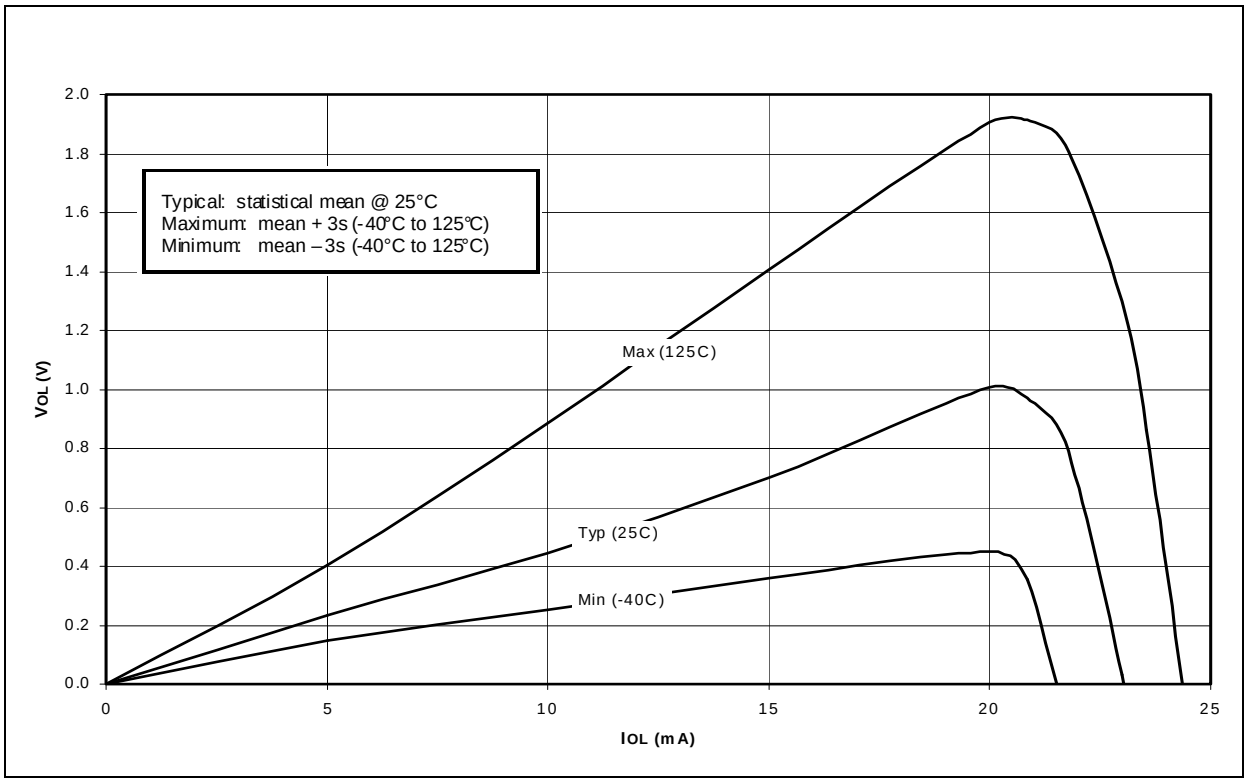


**FIGURE 21-21: TYPICAL, MAXIMUM AND MINIMUM VIN vs. VDD (TTL INPUT, -40°C to 125°C)**

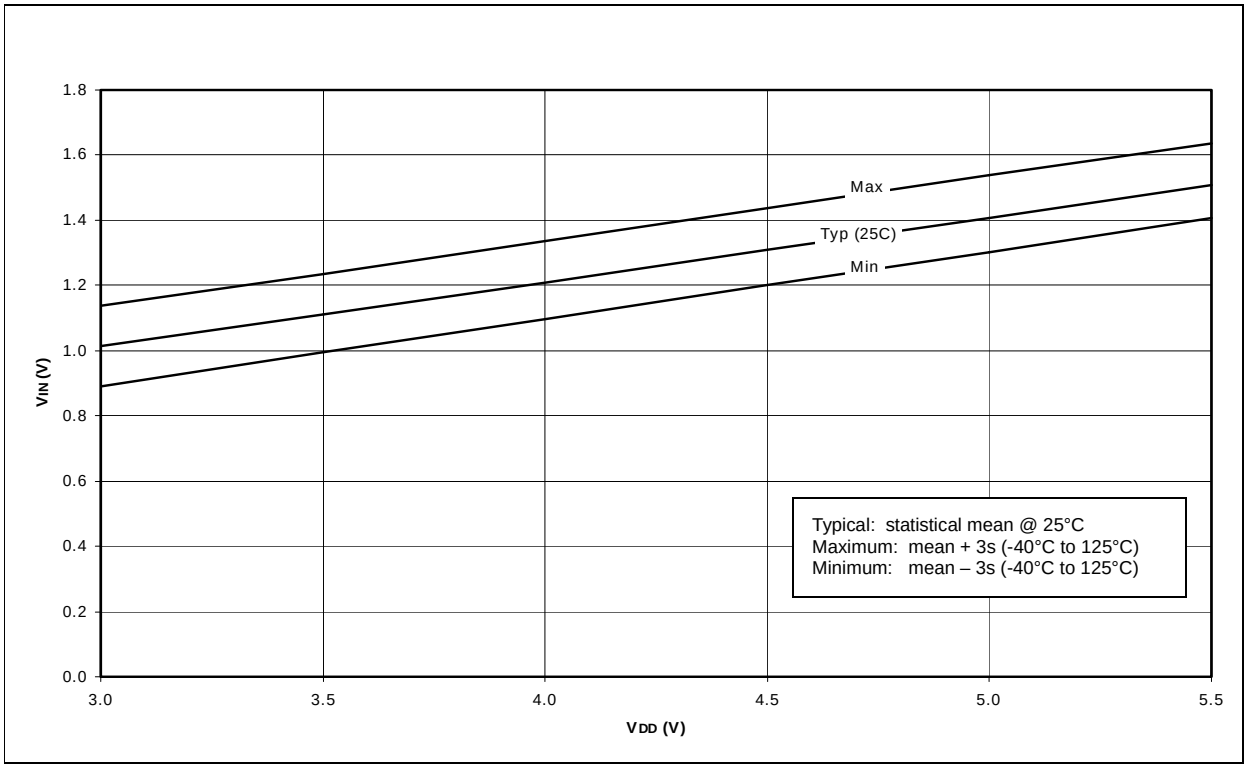
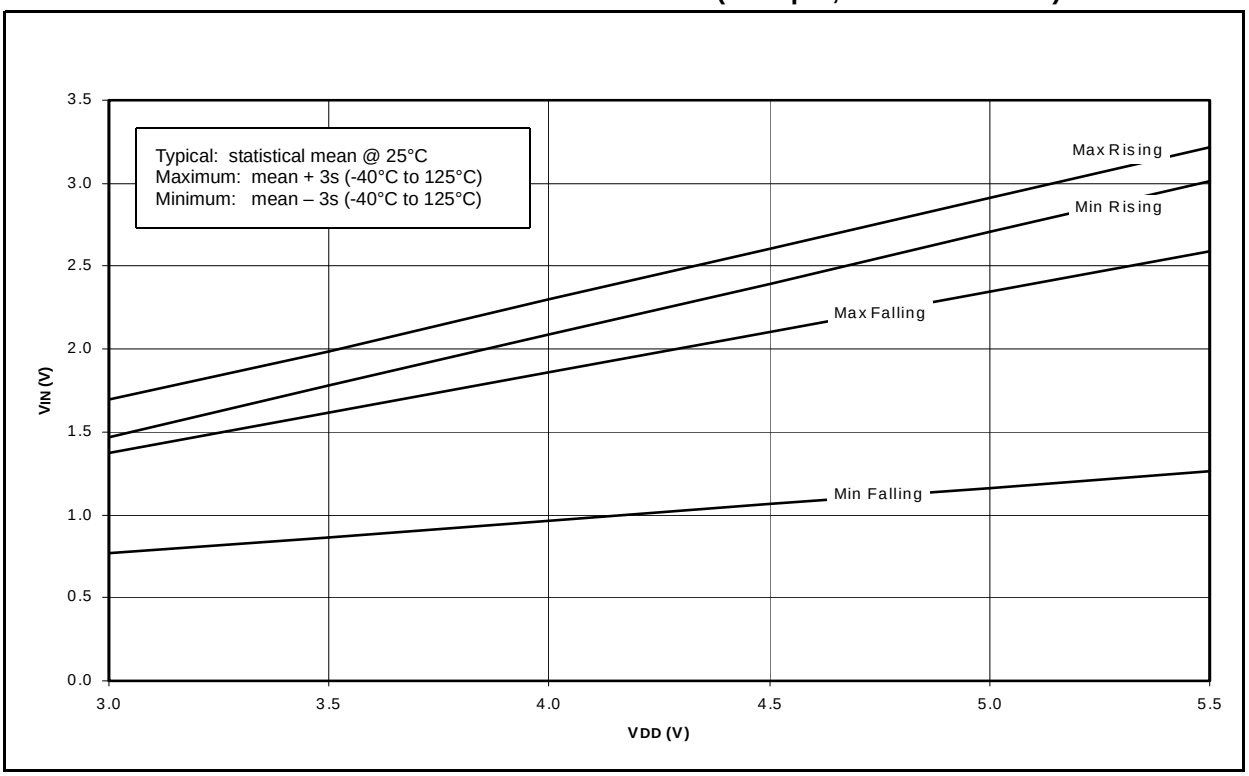


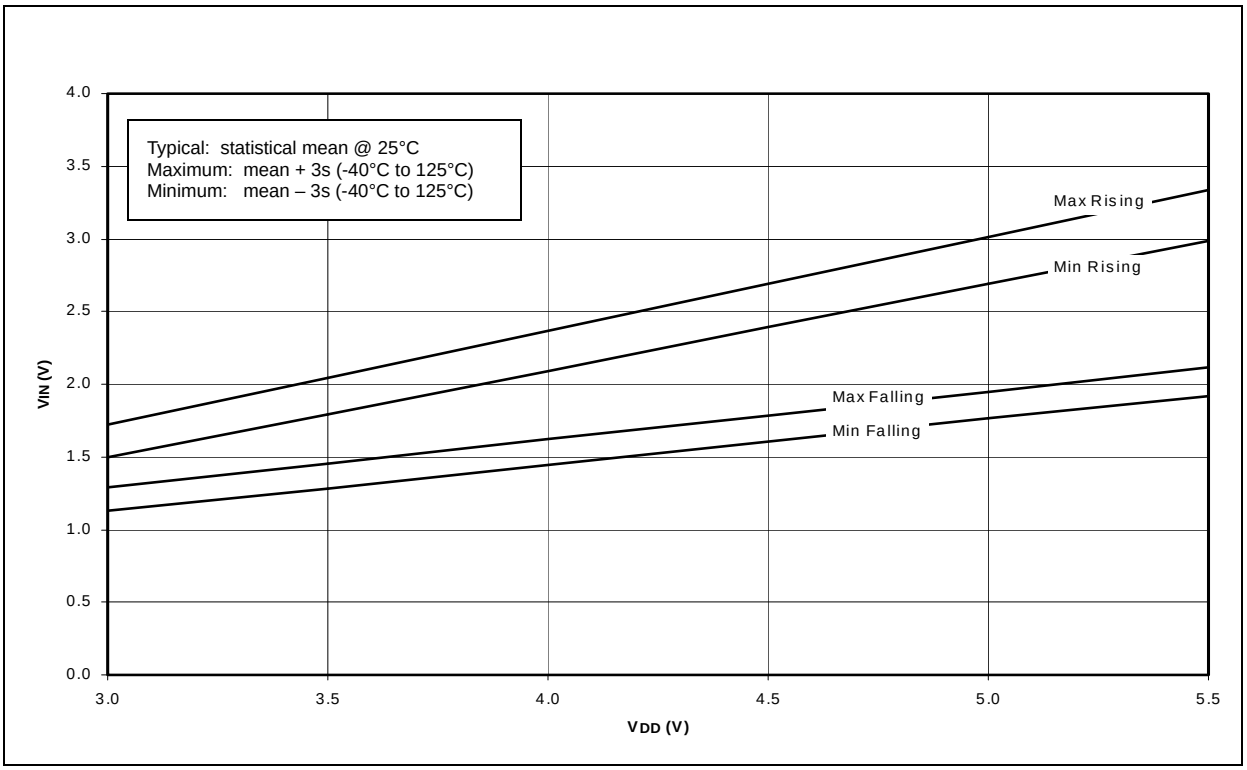

**FIGURE 21-22: MAXIMUM AND MINIMUM VIN vs. VDD (ST Input, -40° C to +125°C)**





© 2000 Microchip Technology Inc.

**FIGURE 21-23: MAXIMUM AND MINIMUM VIN VS. VDD ($I^2C$ Input, -40°C to +125°C)**



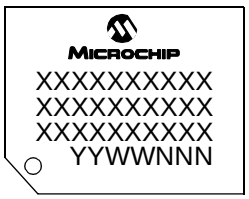
**NOTES:**


© 2000 Microchip Technology Inc.

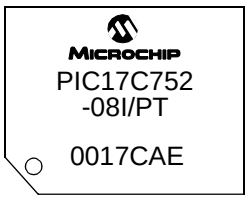
## 22.0 PACKAGING INFORMATION

### 22.1 Package Marking Information

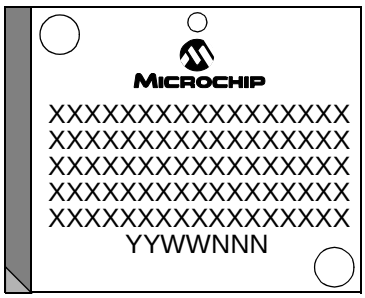
64-Lead TQFP



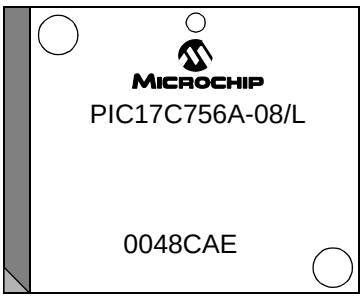

Example



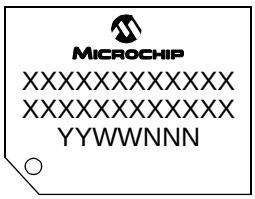

68-Lead PLCC



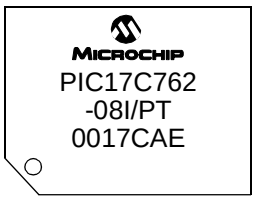

Example




80-Lead TQFP




Example



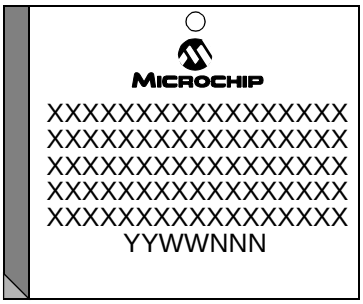

| | | |
|---|---|---|
| **Legend:** | XX...X | Customer specific information* |
| | YY | Year code (last 2 digits of calendar year) |
| | WW | Week code (week of January 1 is week ‘01’) |
| | NNN | Alphanumeric traceability code |

**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.

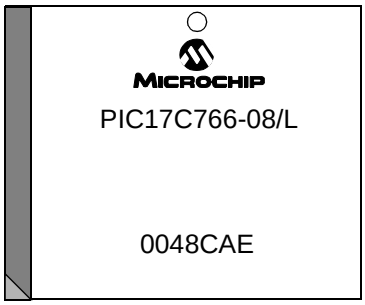
***** Standard OTP marking consists of Microchip part number, year code, week code, facility code, mask rev#, and assembly code. For OTP marking beyond this, certain price adders apply. Please check with your Microchip Sales Office. For QTP devices, any special marking adders are included in QTP price.

**Package Marking Information (Cont.)**

84-Lead PLCC



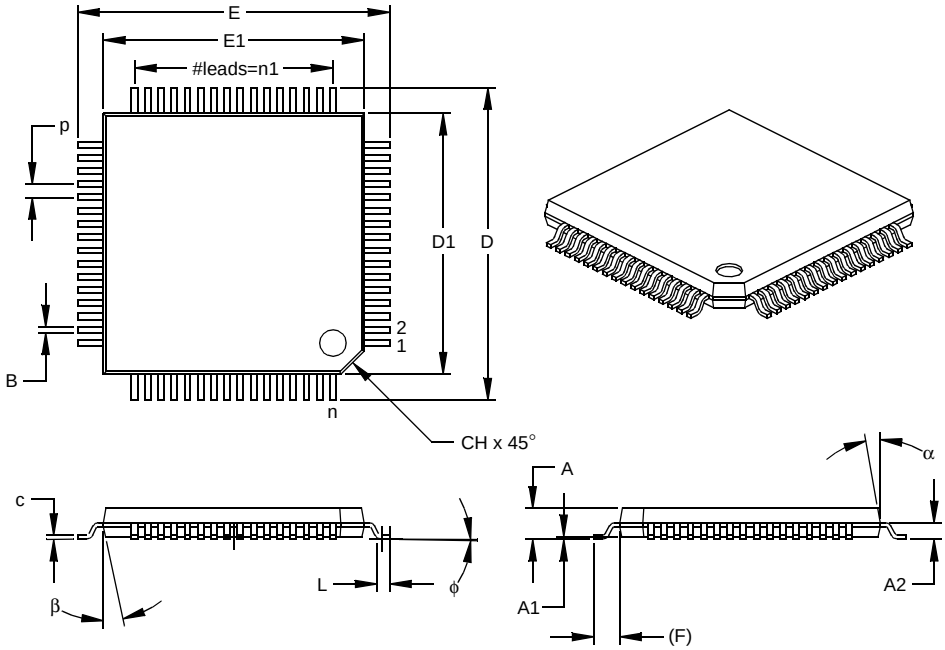

Example





© 2000 Microchip Technology Inc.

## 64-Lead Plastic Thin Quad Flatpack (PT) 10x10x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)

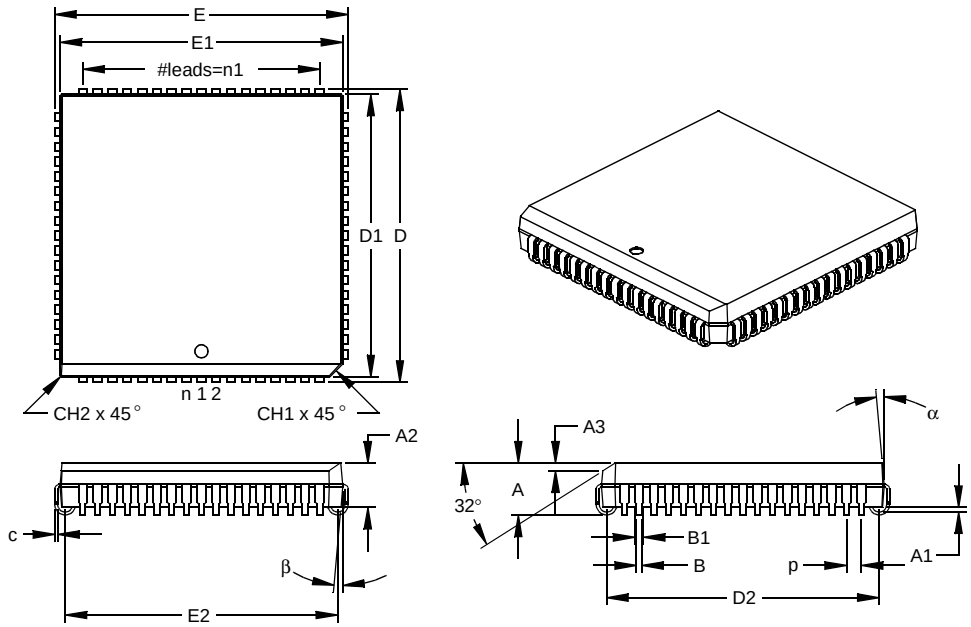



| Units | | INCHES | | | MILLIMETERS* | | |
|---|---|---|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX | MIN | NOM | MAX |
| Number of Pins | n | | 64 | | | 64 | |
| Pitch | p | | .020 | | | 0.50 | |
| Pins per Side | n1 | | 16 | | | 16 | |
| Overall Height | A | .039 | .043 | .047 | 1.00 | 1.10 | 1.20 |
| Molded Package Thickness | A2 | .037 | .039 | .041 | 0.95 | 1.00 | 1.05 |
| Standoff § | A1 | .002 | .006 | .010 | 0.05 | 0.15 | 0.25 |
| Foot Length | L | .018 | .024 | .030 | 0.45 | 0.60 | 0.75 |
| Footprint (Reference) | (F) | | .039 | | | 1.00 | |
| Foot Angle | φ | 0 | 3.5 | 7 | 0 | 3.5 | 7 |
| Overall Width | E | .463 | .472 | .482 | 11.75 | 12.00 | 12.25 |
| Overall Length | D | .463 | .472 | .482 | 11.75 | 12.00 | 12.25 |
| Molded Package Width | E1 | .390 | .394 | .398 | 9.90 | 10.00 | 10.10 |
| Molded Package Length | D1 | .390 | .394 | .398 | 9.90 | 10.00 | 10.10 |
| Lead Thickness | c | .005 | .007 | .009 | 0.13 | 0.18 | 0.23 |
| Lead Width | B | .007 | .009 | .011 | 0.17 | 0.22 | 0.27 |
| Pin 1 Corner Chamfer | CH | .025 | .035 | .045 | 0.64 | 0.89 | 1.14 |
| Mold Draft Angle Top | α | 5 | 10 | 15 | 5 | 10 | 15 |
| Mold Draft Angle Bottom | β | 5 | 10 | 15 | 5 | 10 | 15 |

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MS-026
Drawing No. C04-085

### 68-Lead Plastic Leaded Chip Carrier (L) – Square (PLCC)

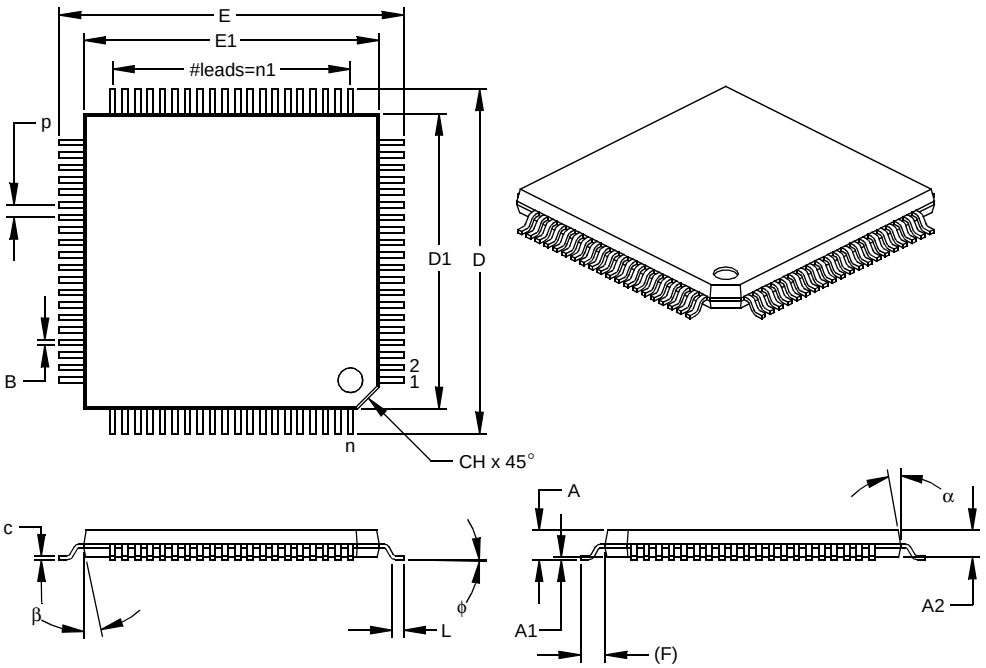



| Units | | INCHES* | | | MILLIMETERS | | |
|---|---|---|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX | MIN | NOM | MAX |
| Number of Pins | n | | 68 | | | 68 | |
| Pitch | p | | .050 | | | 1.27 | |
| Pins per Side | n1 | | 17 | | | 17 | |
| Overall Height | A | .165 | .173 | .180 | 4.19 | 4.39 | 4.57 |
| Molded Package Thickness | A2 | .145 | .153 | .160 | 3.68 | 3.87 | 4.06 |
| Standoff § | A1 | .020 | .028 | .035 | 0.51 | 0.71 | 0.89 |
| Side 1 Chamfer Height | A3 | .024 | .029 | .034 | 0.61 | 0.74 | 0.86 |
| Corner Chamfer 1 | CH1 | .040 | .045 | .050 | 1.02 | 1.14 | 1.27 |
| Corner Chamfer (others) | CH2 | .000 | .005 | .010 | 0.00 | 0.13 | 0.25 |
| Overall Width | E | .985 | .990 | .995 | 25.02 | 25.15 | 25.27 |
| Overall Length | D | .985 | .990 | .995 | 25.02 | 25.15 | 25.27 |
| Molded Package Width | E1 | .950 | .954 | .958 | 24.13 | 24.23 | 24.33 |
| Molded Package Length | D1 | .950 | .954 | .958 | 24.13 | 24.23 | 24.33 |
| Footprint Width | E2 | .890 | .920 | .930 | 22.61 | 23.37 | 23.62 |
| Footprint Length | D2 | .890 | .920 | .930 | 22.61 | 23.37 | 23.62 |
| Lead Thickness | c | .008 | .011 | .013 | 0.20 | 0.27 | 0.33 |
| Upper Lead Width | B1 | .026 | .029 | .032 | 0.66 | 0.74 | 0.81 |
| Lower Lead Width | B | .013 | .020 | .021 | 0.33 | 0.51 | 0.53 |
| Mold Draft Angle Top | α | 0 | 5 | 10 | 0 | 5 | 10 |
| Mold Draft Angle Bottom | β | 0 | 5 | 10 | 0 | 5 | 10 |

* Controlling Parameter
§ Significant Characteristic
Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MO-047
Drawing No. C04-049


© 2000 Microchip Technology Inc.

## 80-Lead Plastic Thin Quad Flatpack (PT) 12x12x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)

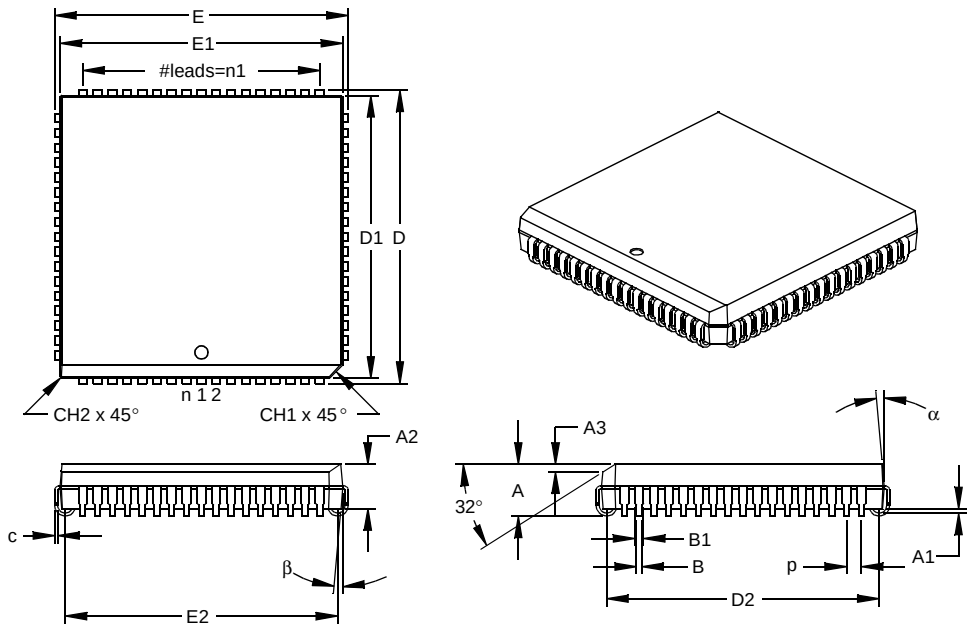



| Units | | INCHES | | | MILLIMETERS* | | |
|---|---|---|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX | MIN | NOM | MAX |
| Number of Pins | n | | 80 | | | 80 | |
| Pitch | p | | .020 | | | 0.50 | |
| Pins per Side | n1 | | 20 | | | 20 | |
| Overall Height | A | .039 | .043 | .047 | 1.00 | 1.10 | 1.20 |
| Molded Package Thickness | A2 | .037 | .039 | .041 | 0.95 | 1.00 | 1.05 |
| Standoff § | A1 | .002 | .004 | .006 | 0.05 | 0.10 | 0.15 |
| Foot Length | L | .018 | .024 | .030 | 0.45 | 0.60 | 0.75 |
| Footprint (Reference) | (F) | | .039 | | | 1.00 | |
| Foot Angle | ϕ | 0 | 3.5 | 7 | 0 | 3.5 | 7 |
| Overall Width | E | .541 | .551 | .561 | 13.75 | 14.00 | 14.25 |
| Overall Length | D | .541 | .551 | .561 | 13.75 | 14.00 | 14.25 |
| Molded Package Width | E1 | .463 | .472 | .482 | 11.75 | 12.00 | 12.25 |
| Molded Package Length | D1 | .463 | .472 | .482 | 11.75 | 12.00 | 12.25 |
| Lead Thickness | c | .004 | .006 | .008 | 0.09 | 0.15 | 0.20 |
| Lead Width | B | .007 | .009 | .011 | 0.17 | 0.22 | 0.27 |
| Pin 1 Corner Chamfer | CH | .025 | .035 | .045 | 0.64 | 0.89 | 1.14 |
| Mold Draft Angle Top | α | 5 | 10 | 15 | 5 | 10 | 15 |
| Mold Draft Angle Bottom | β | 5 | 10 | 15 | 5 | 10 | 15 |

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MS-026
Drawing No. C04-092

### 84-Lead Plastic Leaded Chip Carrier (L) – Square (PLCC)




| Units | | INCHES* | | | MILLIMETERS | | |
|---|---|---|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX | MIN | NOM | MAX |
| Number of Pins | n | | 68 | | | 68 | |
| Pitch | p | | .050 | | | 1.27 | |
| Pins per Side | n1 | | 17 | | | 17 | |
| Overall Height | A | .165 | .173 | .180 | 4.19 | 4.39 | 4.57 |
| Molded Package Thickness | A2 | .145 | .153 | .160 | 3.68 | 3.87 | 4.06 |
| Standoff § | A1 | .020 | .028 | .035 | 0.51 | 0.71 | 0.89 |
| Side 1 Chamfer Height | A3 | .024 | .029 | .034 | 0.61 | 0.74 | 0.86 |
| Corner Chamfer 1 | CH1 | .040 | .045 | .050 | 1.02 | 1.14 | 1.27 |
| Corner Chamfer (others) | CH2 | .000 | .005 | .010 | 0.00 | 0.13 | 0.25 |
| Overall Width | E | .985 | .990 | .995 | 25.02 | 25.15 | 25.27 |
| Overall Length | D | .985 | .990 | .995 | 25.02 | 25.15 | 25.27 |
| Molded Package Width | E1 | .950 | .954 | .958 | 24.13 | 24.23 | 24.33 |
| Molded Package Length | D1 | .950 | .954 | .958 | 24.13 | 24.23 | 24.33 |
| Footprint Width | E2 | .890 | .920 | .930 | 22.61 | 23.37 | 23.62 |
| Footprint Length | D2 | .890 | .920 | .930 | 22.61 | 23.37 | 23.62 |
| Lead Thickness | c | .008 | .011 | .013 | 0.20 | 0.27 | 0.33 |
| Upper Lead Width | B1 | .026 | .029 | .032 | 0.66 | 0.74 | 0.81 |
| Lower Lead Width | B | .013 | .020 | .021 | 0.33 | 0.51 | 0.53 |
| Mold Draft Angle Top | α | 0 | 5 | 10 | 0 | 5 | 10 |
| Mold Draft Angle Bottom | β | 0 | 5 | 10 | 0 | 5 | 10 |

* Controlling Parameter
§ Significant Characteristic
Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MO-047
Drawing No. C04-093


© 2000 Microchip Technology Inc.

## APPENDIX A: MODIFICATIONS

The following is the list of modifications over the PIC16CXX microcontroller family:

1. Instruction word length is increased to 16-bit. This allows larger page sizes, both in program memory (8 Kwords verses 2 Kwords) and register file (256 bytes versus 128 bytes).
2. Four modes of operation: Microcontroller, Protected Microcontroller, Extended Microcontroller, and Microprocessor.
3. 22 new instructions.
   The MOVF, TRIS and OPTION instructions are no longer supported.
4. Four new instructions (TLRD, TLWT, TABLRD, TABLWT) for transferring data between data memory and program memory. They can be used to "self program" the EPROM program memory.
5. Single cycle data memory to data memory transfers possible (MOVPF and MOVFP instructions). These instructions do not affect the Working register (WREG).
6. W register (WREG) is now directly addressable.
7. A PC high latch register (PCLATH) is extended to 8-bits. The PCLATCH register is now both readable and writable.
8. Data memory paging is redefined slightly.
9. DDR registers replace function of TRIS registers.
10. Multiple Interrupt vectors added. This can decrease the latency for servicing interrupts.
11. Stack size is increased to 16 deep.
12. BSR register for data memory paging.
13. Wake-up from SLEEP operates slightly differently.
14. The Oscillator Start-Up Timer (OST) and Power-Up Timer (PWRT) operate in parallel and not in series.
15. PORTB interrupt-on-change feature works on all eight port pins.
16. TMR0 is 16-bit, plus 8-bit prescaler.
17. Second indirect addressing register added (FSR1 and FSR2). Control bits can select the FSR registers to auto-increment, auto-decrement, remain unchanged after an indirect address.
18. Hardware multiplier added (8 x 8 $\rightarrow$ 16-bit).
19. Peripheral modules operate slightly differently.
20. A/D has both $V_{REF}$+ and $V_{REF}$- inputs.
21. USARTs do not implement BRGH feature.
22. Oscillator modes slightly redefined.
23. Control/Status bits and registers have been placed in different registers and the control bit for globally enabling interrupts has inverse polarity.
24. In-circuit serial programming is implemented differently.

## APPENDIX B: COMPATIBILITY

To convert code written for PIC16CXXX to PIC17CXXX, the user should take the following steps:

1. Remove any TRIS and OPTION instructions, and implement the equivalent code.
2. Separate the Interrupt Service Routine into its four vectors.
3. Replace:
   ```
   MOVF    REG1, W
   ```
   with:
   ```
   MOVFP   REG1, WREG
   ```
4. Replace:
   ```
   MOVF    REG1, W
   MOVWF   REG2
   ```
   with:
   ```
   MOVPF   REG1, REG2 ; Addr(REG1)<20h
   ```
   or
   ```
   MOVFP   REG1, REG2 ; Addr(REG2)<20h
   ```

   **Note:** If REG1 and REG2 are both at addresses greater then 20h, two instructions are required.
   ```
   MOVFP   REG1, WREG ;
   MOVPF   WREG, REG2 ;
   ```
5. Ensure that all bit names and register names are updated to new data memory map locations.
6. Verify data memory banking.
7. Verify mode of operation for indirect addressing.
8. Verify peripheral routines for compatibility.
9. Weak pull-ups are enabled on RESET.
10. WDT time-outs always reset the device (in run or SLEEP mode).

### B.1 Upgrading from PIC17C42 Devices

To convert code from the PIC17C42 to all the other PIC17CXXX devices, the user should take the following steps.

1. If the hardware multiply is to be used, ensure that any variables at address 18h and 19h are moved to another address.
2. Ensure that the upper nibble of the BSR was not written with a non-zero value. This may cause unexpected operation since the RAM bank is no longer 0.
3. The disabling of global interrupts has been enhanced, so there is no additional testing of the GLINTD bit after a `BSF CPUSTA, GLINTD` instruction.

© 2000 Microchip Technology Inc.

## APPENDIX C: WHAT'S NEW

This is a new Data Sheet for the Following Devices:

- PIC17C752
- PIC17C756A
- PIC17C762
- PIC17C766

This Data Sheet is based on the PIC17C75X Data Sheet (DS30246A).

## APPENDIX D: WHAT'S CHANGED

Clarified the TAD vs. device maximum operating frequency tables in Section 16.2.

Added device characteristic graphs and charts in Section 21.

Removed the "Preliminary" status from the entire document.


© 2000 Microchip Technology Inc.

## INDEX

© 2000 Microchip Technology Inc.

© 2000 Microchip Technology Inc.

© 2000 Microchip Technology Inc.

© 2000 Microchip Technology Inc.

© 2000 Microchip Technology Inc.

**NOTES:**


© 2000 Microchip Technology Inc.

## ON-LINE SUPPORT

Microchip provides on-line support on the Microchip World Wide Web (WWW) site.

The web site 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**

The file transfer site is available by using an FTP service to connect to:

**ftp://ftp.microchip.com**

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
- Conferences for products, Development Systems, technical information and more
- Listing of seminars and events

### 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

1-480-792-7302 for the rest of the world.

001024

**Trademarks:** The Microchip name, logo, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, KEELOQ, SEEVAL, MPLAB and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.
Total Endurance, ICSP, In-Circuit Serial Programming, FilterLab, MXDEV, microID, *Flex*ROM, *fuzzy*LAB, MPASM, MPLINK, MPLIB, PICDEM, ICEPIC and Migratable Memory are trademarks and SQTP is a service mark of Microchip in the U.S.A.

All other trademarks mentioned herein are the property of their respective companies.

## 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 (480) 792-4150.

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: **PIC17C7XX** Literature Number: **DS30289B**

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?


© 2000 Microchip Technology Inc.

## PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

| PART NO. | X | /XX | XXX |
|---|---|---|---|
| Device | Temperature Range | Package | Pattern |

| | |
|---|---|
| Device | PIC17C756: Standard $V_{DD}$ range<br>PIC17C756T: (Tape and Reel)<br>PIC17LC756: Extended $V_{DD}$ range |
| Temperature Range | - = 0°C to +70°C<br>I = -40°C to +85°C |
| Package | CL = Windowed LCC<br>PT = TQFP<br>L = PLCC |
| Pattern | QTP, SQTP, ROM Code (factory specified) or Special Requirements . Blamk for OTP and Windowed devices. |

**Examples:**

a) PIC17C756 – 16L Commercial Temp., PLCC package, 16 MHz, normal $V_{DD}$ limits

b) PIC17LC756–08/PT Commercial Temp., TQFP package, 8MHz, extended $V_{DD}$ limits

c) PIC17C756–33I/PT Industrial Temp., TQFP package, 33 MHz, normal $V_{DD}$ limits

* JW Devices are UV erasable and can be programmed to any device configuration. JW Devices meet the electrical requirement of each oscillator type.

### Sales and Support

**Data Sheets**
Products supported by a preliminary Data Sheet may 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
2. The Microchip Corporate Literature Center U.S. FAX: (480) 792-7277
3. The Microchip Worldwide Site (www.microchip.com)

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

**New Customer Notification System**
Register on our web site (www.microchip.com/cn) to receive the most current information on our products.

© 2000 Microchip Technology Inc.

**NOTES:**


© 2000 Microchip Technology Inc.

**NOTES:**

# WORLDWIDE SALES AND SERVICE

## AMERICAS

**Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

**Rocky Mountain**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

**Atlanta**
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

**Boston**
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3838 Fax: 978-692-3821

**Chicago**
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

**Dallas**
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

**Dayton**
Two Prestige Place, Suite 130
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

**Detroit**
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

**Los Angeles**
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

**New York**
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

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

**Toronto**
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

## ASIA/PACIFIC

**China - Beijing**
Microchip Technology Beijing Office
Unit 915
New China Hong Kong Manhattan Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

**China - Shanghai**
Microchip Technology Shanghai Office
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

**Hong Kong**
Microchip Asia Pacific
RM 2101, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

**India**
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

**Japan**
Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

**Korea**
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
Tel: 82-2-554-7200 Fax: 82-2-558-5934

## ASIA/PACIFIC (continued)

**Singapore**
Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

**Taiwan**
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

## EUROPE

**Denmark**
Microchip Technology Denmark ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

**France**
Arizona Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

**Germany**
Arizona Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

**Italy**
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

**United Kingdom**
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

10/01/00

DNV Certification, Inc. USA — ANSI-RAB QMS ACCREDITED — DNV — DNV MSC The Netherlands Accredited by the RvA
ISO 9001 / QS-9000 REGISTERED FIRM

*Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.*

All rights reserved. © 2000 Microchip Technology Incorporated. Printed in the USA. 11/00 Printed on recycled paper.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. 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, except as maybe explicitly expressed herein, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.