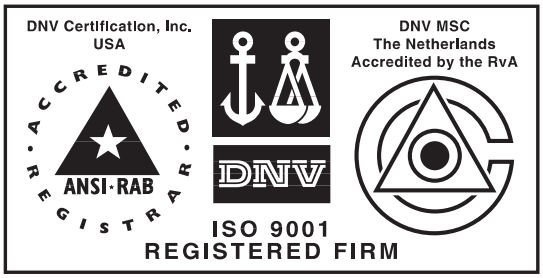
PICmicro™ Mid-Range MCU Family Reference Manual

December 1997 /DS33023A

Internationally Recognized Quality System Certifications

Microchip's Quality System embodies the requirements of ISO9001:1994. Our Microchip Chandler and Tempe Design and Manufacturing facilities have been certified to ISO 9001. The Microchip Kaohsiung Test facility, and primary Assembly houses have been certified to ISO 9002. ISO certification plans are in-process for an estimated certification grant by year-end 1997. In addition, Microchip has received numerous customer certifications, including a Delco issued certificate of compliance to AEC-A100/QS9000.



Microchip received ISO 9001 Quality System certification for its worldwide headquarters, design, and wafer fabrication facilities in January, 1997. Our field-programmable PICmicro™ 8-bit MCUs, Serial EEPROMs, related specialty memory products and development systems conform to the stringent quality standards of the International Standard Organization (ISO).

Trademarks

The Microchip name, logo, PIC, KEELOQ, PICMASTER, PICSTART, PRO MATE, and SEEVAL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

MPLAB, PICmicro, ICSP and In-Circuit Serial Programming are trademarks of Microchip Technology Incorporated.

Serialized Quick-Turn Production is a Service Mark of Microchip Technology Incorporated.

All other trademarks mentioned herein are property of their respective companies.

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Section 1. Introduction

HIGHLIGHTS

This section of the manual contains the following major topics:

1.1 Introduction

Microchip is the Embedded Control Solutions Company®. The company's focus is on products that meet the needs of the embedded control market. We are a leading supplier of:

- 8-bit General Purpose Microcontrollers (PICmicro™ MCUs)
- Speciality and standard non-volatile memory devices
- Security devices (KEELOQ®)
- Application specific standard products

Please request a Microchip Product Line Card for a listing of all the interesting products that we have to offer. This literature can be obtained from your local sales office, or downloaded from the Microchip web site (www.microchip.com).

In the past, 8-bit MCU users were fixed on the traditional MCU model for production, a ROM device was required. Microchip has been the leader in changing this perception by showing that OTP devices can give a better lifetime product cost compared to ROM versions.

Microchip has a strength is in EPROM technology. That made it the memory technology of choice for the PICmicro MCU's program memory. Microchip has minimized the cost difference between EPROM and ROM memory technology, and therefore Microchip can pass these benefits onto our customers. This is not true for other MCU vendors, and is seen in the price difference between their EPROM and ROM versions.

The growth of Microchip's 8-bit MCU market share is a testament to the PICmicro MCUs ability to meet the needs of many. This growth has made the PICmicro architecture one of the top three architectures available in the general market today. This growth was fueled by the Microchip vision of the benefits of a low cost OTP solution. Some of the benefits for the customer include:

- Quick time to market
- Allows code changes to product, during production run
- No Non-Recurring Engineering (NRE) charges for Mask Revisions
- Ability to easily serialize the product
- Ability to store calibration data, without additional hardware
- Better able to maximize PICmicro MCU inventory
- Less risk, since the same device is used for development as well as for production.

Microchip's PICmicro 8-bit MCUs offer a price/performance ratio that allows them to be considered for any traditional 8-bit MCU application as well as some traditional 4-bit applications (Base-Line family), dedicated logic replacement and low-end DSP applications (High-End family). These features and price-performance mix make PICmicro MCUs an attractive solution for most applications.

1.2 Manual Objective

PICmicro devices are grouped by the size of their Instruction Word. The three current PICmicro families are:

1. Base-Line: 12-bit Instruction Word length
2. Mid-Range: 14-bit Instruction Word length
3. High-End: 16-bit Instruction Word length

This manual focuses on the Mid-Range devices, which are also referred to as the PIC16CXXX MCU family.

The operation of the PIC16CXXX MCU family architecture and peripheral modules is explained, but does not cover the specifics of each device. Therefore, it is not intended to replace the device data sheets, but complement them. In other words, this guide supplies the general details and operation of the PICmicro architecture and peripheral modules, while the data sheet s give specific details such as device memory mapping.

Initialization examples are given throughout this manual. These examples sometimes need to be written as device specific as opposed to family generic, though they are valid for most other devices. Some modifications may be required for devices with variations in register file mappings.

Note:	The first few Mid-Range devices have minor device variations when compared to this general description. We have tried to describe these variations throughout this manual. Please refer to the specific device data sheet for complete information on the device.

1.3 Device Structure

Each part of a device can be placed into one of three groups:

1. Core
2. Peripherals
3. Special Features

1.3.1 The Core

The core pertains to the basic features that are required to make the device operate. These include:

1.	Device Oscillator	Revision “DS31002A”
2.	Reset logic	Revision “DS31003A”
3.	CPU (Central Processing Unit) operation	Revision “DS31005A”
4.	ALU (Arithmetic Logical Unit) operation	Revision “DS31005A”
5.	Device memory map organization	Revision “DS31006A”
6.	Interrupt operation	Revision “DS31008A”
7.	Instruction set	Revision “DS31029A”

1.3.2 Peripherals

Peripherals are the features that add a differentiation from a microprocessor. These ease in interfacing to the external world (such as general purpose I/O, LCD drivers, A/D inputs, and PWM outputs), and internal tasks such as keeping different time bases (such as timers). The peripherals that are discussed are:

1.	General purpose I/O	Revision “DS31009A”
2.	Timer0	Revision “DS31011A”
3.	Timer1	Revision “DS31012A”
4.	Timer2	Revision “DS31013A”
5.	Capture, Compare, and PWM (CCP)	Revision “DS31014A”
6.	Synchronous Serial Port (SSP)	Revision “DS31015A”
7.	Basic Synchronous Serial Port (SSP)	Revision “DS31016A”
8.	Master Synchronous Serial Port (MSSP)	Revision “DS31017A”
9.	USART (SCI)	Revision “DS31018A”
10.	Voltage References	Revision “DS31019A”
11.	Comparators	Revision “DS31020A”
12.	8-bit Analog to Digital (A/D)	Revision “DS31021A”
13.	Basic 8-bit Analog to Digital (A/D)	Revision “DS31022A”
14.	10-bit Analog to Digital (A/D)	Revision “DS31023A”
15.	Slope Analog to Digital (A/D) w/ Thermister	Revision “DS31024A”
16.	Liquid Crystal Display (LCD) Drivers	Revision “DS31025A”
17.	Parallel Slave Port (PSP)	Revision “DS31010A”

1.3.3 Special Features

Special features are the unique features that help to do one or more of the following things:

- Decrease system cost
- Increase system reliability
- Increase design flexibility

The Mid-Range PICmicro MCUs offer several features that help achieve these goals. The special features discussed are:

1.	Device Configuration bits	Revision “DS31027A”
2.	On-chip Power-on Reset (POR)	Revision “DS31003A”
3.	Brown-out Reset (BOR) logic	Revision “DS31003A”
4.	Watchdog Timer	Revision “DS31026A”
5.	Low power mode (Sleep)	Revision “DS31026A”
6.	Internal RC device oscillator	Revision “DS31002A”
7.	In-Circuit Serial Programming™ (ICSP™)	Revision “DS31028A”

1.4 Development Support

Microchip offers a wide range of development tools that allow users to efficiently develop and debug application code. Microchip's development tools can be broken down into four categories:

1. Code generation
2. Software debug
3. Device programmer
4. Product evaluation boards

All tools developed by Microchip operate under the MPLAB™ Integrated Development Environment (IDE), while some third party tools may not. The code generation tools include:

- MPASM
- MPLAB-C
- MP-DriveWay™

These software development programs include device header files. Each header file defines the register names (as shown in the device data sheet) to the specified address or bit location. Using the header files eases code migration, and reduces the tediousness of memorizing a register's address or a bit's position in a register.

Note:	Microchip strongly recommends that the supplied header files be used in the source code of your program. This eases code migration as well as increases the quality and depth of the technical support that Microchip can offer.

Tools which ease in debugging software are:

- PICMASTER® In-Circuit Emulator
- ICEPIC In-Circuit Emulator
- MPLAB-SIM Software Simulator

After generating and debugging the application software, the device will need to be programmed. Microchip offers two levels of programmers:

1. PICSTART® Plus programmer
2. PROMATE® II programmer

Demonstration boards allow the developer of software code to evaluate the capability and suitability of the device to the application. The demo boards offered are:

- PICDEM-1
- PICDEM-2
- PICDEM-3
- PICDEM-14A

A full description of each of Microchip's development tools is discussed in the **"Development Tools"** section. As new tools are developed, product briefs and user guides may be obtained from the Microchip web site (www.microchip.com) or from your local Microchip Sales Office.

Code development recommendations and techniques are provided in the **"Code Development"** section.

Microchip offers other reference tools to speed the development cycle. These include:

- Application Notes
- Reference Designs
- Microchip web site
- Microchip BBS
- Local Sales Offices with Field Application Support
- Corporate Support Line

Additional avenues of assistance can be found in many Web User Groups including the MIT reflector PIClist. The Microchip web site lists other sites that may be useful references.

1.5 Device Varieties

Once the functional requirements of the device are specified, some other decisions need to be made. These include:

- Memory technology
- Operating voltage
- Operating temperature range
- Operating frequency
- Packaging

Microchip has a large number of options and option combinations, one of which should fulfill your requirements.

1.5.1 Memory Varieties

Memory technology has no effect on the logical operation of a device. Due to the different processing steps required, some electrical characteristics may vary between devices with the same feature set/pinout but with different memory technologies. An example is the electrical characteristic V_{IL} (Input Low Voltage), which may have some difference between a typical EPROM device and a typical ROM device.

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

When discussing the functionality of the device, the memory technology and the voltage range do not matter. Microchip offers three program memory types. The memory type is designated in the part number by the first letter(s) after the family affiliation designators.

1. **C**, as in PIC16**C**XXX. These devices have EPROM type memory.
2. **CR**, as in PIC16**CR**XXX. These devices have ROM type memory.
3. **F**, as in PIC16**F**XXX. These devices have Flash type memory.

1.5.1.1 EPROM

Microchip focuses on Erasable Programmable Read Only Memory (EPROM) technology to give the customers flexibility throughout their entire design cycle. With this technology Microchip offers various packaging options as well as services.

1.5.1.2 Read Only Memory (ROM) Devices

Microchip offers a masked Read Only Memory (ROM) version of several of the highest volume parts, thus giving customers a lower 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 local Microchip sales office.

1.5.1.3 Flash Memory Devices

These devices are electrically erasable, and can therefore be offered in a low cost plastic package. Being electrically erasable, these devices can be both erased and reprogrammed without removal from the circuit. A device will have the same specifications whether it is used for prototype development, pilot programs, or production.

1.5.2 Operating Voltage Range Options

All Mid-Range PICmicro™ MCUs operate over the standard voltage range. Devices are also offered which operate over an extended voltage range (and reduced frequency range). Table 1-1 shows all possible memory types and voltage range designators for the PIC16CXXX MCU family. The designators are in **bold** typeface.

Table 1-1: Device Memory Type and Voltage Range Designators

Memory Type	Voltage Range	
	Standard	Extended
EPROM	PIC16**C**XXX	PIC16**LC**XXX
ROM	PIC16**CR**XXX	PIC16**LCR**XXX
Flash	PIC16**F**XXX	PIC16**LF**XXX

Note:Not all memory types may be available for a particular device.

As you can see in Table 1-2, Microchip specifications its extended range devices at a more conservative voltage range until device characterization has ensured they will be able to meet the goal of their final design specifications.

Table 1-2: Typical Voltage Ranges for Each Device Type

Typical Voltage Range (1)		EPROM		ROM		Flash	
Standard		**C**	4.5 - 6.0V	**CR**	4.5 - 6.0V	**F**	4.5 - 6.0V
Extended	Before device characterization	**LC**	3.0 - 6.0V	**LCR**	3.0 - 6.0V	**LF**	3.0 - 6.0V
	Final specification (2)	**LC**	2.5 - 6.0V	**LCR**	2.5 - 6.0V	**LF**	2.0 - 6.0V

Note 1: Devices fabricated in Microchip's 120K Process Technology will have a maximum limit on VDD of 5.5V. New device data sheets will specify Microchip's technology designation

2: This voltage range depends on the results of device characterization.

1.5.3 Packaging Varieties

Depending on the development phase of your project, one of three package types would be used:

The first is a device with an erasure window. Typically these are found in packages with a ceramic body. These devices are used for the development phase, since the device's program memory can be erased and reprogrammed many times.

The second package type is a low cost plastic package. This package type is used in production where device cost is to be kept to a minimum.

Lastly, there is the DIE option. A DIE is an unpackaged device that has been tested. DIEs are used in low cost designs and designs where board space is at a minimum. Table 1-3 shows a quick summary of this.

Table 1-3: Typical Package Uses

Package Type	Typical Usage
Windowed	Development Mode
Plastic	Production
DIE	Special Applications, such as those which require minimum board space

1.5.3.4 UV Erasable Devices

The UV erasable version of EPROM program memory devices is optimal for prototype development and pilot programs.

These devices can be erased and reprogrammed to any of the configuration modes. Third party programmers are also available; refer to Microchip's *Third Party Guide* (DS00104) for a list of sources.

The amount of time required to completely erase a UV erasable device depends on: the wavelength of the light, its intensity, distance from UV source, the process technology of the device (how small are the memory cells).

Note:	Fluorescent lights and sunlight both emit ultraviolet light at the erasure wavelength. Leaving a UV erasable device's window uncovered could cause, over time, the devices memory cells to become erased. The erasure time for a fluorescent light is about three years, while sunlight requires only about one week. To prevent the memory cells from losing data, an opaque label should be placed over the erasure window.

1.5.3.5 One-Time-Programmable (OTP) Devices

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

OTP devices, packaged in plastic packages, permit the user to program them once. In addition to the program and data EPROM memories, the configuration bits must be programmed.

1.5.3.6 Flash Devices

A Flash device allows its memory to be changed by an electric charge. This means that the system can be designed so that programming may be performed in-circuit. Since no window is required, the lower cost plastic packages can used for these devices.

1.5.3.7 EEPROM Devices

An EEPROM device allows its memory to be erased by an electric charge. This means that the system can be designed so that erasure and reprogramming may be performed in-circuit. Since no window is required, the lower cost plastic packages can used for these devices.

1.5.3.8 ROM Devices

ROM devices have their program memory fixed at the time of the silicon manufacture. Since the program memory cannot be changed, the device can be housed in the lower cost plastic package.

1.5.3.9 DIE

The DIE option allows the board size to become as small as physically possible. The DIE Support document (DS30258) explains general information about using and designing with DIE. There are also individual specification sheets that detail DIE specific information. Manufacturing with DIE requires special knowledge and equipment. This means that the number of manufacturing houses that support DIE will be limited. If you decide to use the DIE option, please research your manufacturing sites to ensure that they will be able to meet the specialized requirements of DIE use.

1.5.3.10 Specialized Services

For OTP customers with established code, Microchip offers two specialized services. These two services, Quick Turn Production Programming and Serialized Quick Turn Production Programming, that allow customers to shorten their manufacturing cycle time.

1.5.3.11 Quick Turn Production (QTP) Programming

Microchip offers this 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 sales office for more details.

1.5.3.12 Serialized Quick Turn Production (SQTP^{SM}) Programming

Microchip offers a this 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.

1.6 Style and Symbol Conventions

Throughout this document, certain style and font format changes are used. Most format changes imply a distinction should be made for the emphasized text. The MCU industry has many symbols and non-conventional word definitions/abbreviations. Table 1-4 provides a description for many of the conventions contained in this document. A glossary is provided in the **"Glossary"** section, which contains more word and abbreviation definitions that are used throughout this manual.

1.6.1 Document Conventions

Table 1-4 defines some of the symbols and terms used throughout this manual.

Table 1-4: Document Conventions

Symbol or Term	Description
set	To force a bit/register to a value of logic '1'.
clear	To force a bit/register to a value of logic '0'.
reset	1) To force a register/bit to its default state. 2) A condition in which the device places itself after a device reset occurs. Some bits will be forced to '0' (such as interrupt enable bits), while others will be forced to '1' (such as the I/O data direction bits).
0xnn or nnh	Designates the number 'nn' in the hexadecimal number system. These conventions are used in the code examples.
B'bbbbbbbb'	Designates the number 'bbbbbbbb' in the binary number system. This convention is used in the text and in figures and tables.
R-M-W	Read - Modify - Write. This is when a register or port is read, then the value is modified, and that value is then written back to the register or port. This action can occur from a single instruction (such as bit set file, BSF) or a sequence of instructions.
: (colon)	Used to specify a range, or the concatenation of registers / bits / pins. An example is TMR1H:TMR1L is the concatenation of two 8-bit registers to form a 16-bit timer value, while SSPM3:SSPM0 are 4-bits used to specify the mode of the SSP module. Concatenation order (left-right) usually specifies a positional relationship (MSb to LSb, higher to lower).
< >	Specifies bit(s) locations in a particular register. An example is SSPCON<SSPM3:SSPM0> (or SSPCON<3:0>) specifies the register and associated bits or bit positions.
`Courier Font`	Used for code examples, binary numbers, and for Instruction Mnemonics in the text.
Times Font	Used for equations and variables.
Times, Bold Font, Italics	Used in explanatory text for items called out from a graphic/equation/example.
Note	Notes present information that we wish to reemphasize, either to help you avoid a common pitfall, or make you aware of operating differences between some device family members. A Note is always in a shaded box (as below), unless used in a table, where it is at the bottom of the table (as in this table). **Note:** This is a note in a note box.
Caution(1)	A caution statement describes a situation that could potentially damage software or equipment.
Warning(1)	A warning statement describes a situation that could potentially cause personnel harm.

Note 1: The information in a caution or a warning is provided for your protection. Please read each caution and warning carefully.

1.6.2 Electrical Specifications

Throughout this manual there will be references to electrical specification parameter numbers. A parameter number represents a unique set of characteristics and conditions that is consistent between every data sheet, though the actual parameter value may vary from device to device.

The **"Electrical Specifications"** section shows all the specifications that are documented for all devices. No one device has all these specifications. This section is intended to let you know the types of parameters that Microchip specifies. The value of each specification is device dependent, though we strongly attempt to keep them consistent across all devices.

Table 1-5: Electrical Specification Parameter Numbering Convention

Parameter Number Format	Comment
Dxxx	DC Specification
Axxx	DC Specification for Analog peripherals
xxx	Timing (AC) Specification
PDxxx	Device Programming DC Specification
Pxxx	Device Programming Timing (AC) Specification

Legend: xxx: represents a number.

1.7 Related Documents

Microchip, as well as other sources, offers additional documentation which can aid in your development with PICmicro MCUs. These lists contain the most common documentation but other documents may also be available. Please check the Microchip web site (www.microchip.com) for the latest published technical documentation.

1.7.1 Microchip Documentation

The following documents are available from Microchip. Many of these documents provide application specific information that give actual examples of using, programming and designing with PICmicro MCUs.

1. **MPASM User's Guide (DS33014)**
   This document explains how to use Microchip's MPASM assembler.
2. **MPLAB™-C Compiler User's Guide (DS51014)**
   This document explains how to use Microchip's MPLAB-C C compiler.
3. **MPLAB User's Guide (DS51025)**
   This document explains how to use Microchip's MPLAB Integrated Development Environment.
4. **MPLAB Editor User's Guide (DS30420)**
   This document explains how to use Microchip's MPLAB built-in editor.
5. **PICMASTER® User's Guide (DS30421)**
   This document explains how to use Microchip's PICMASTER In-Circuit Emulator.
6. **MPSIM User's Guide (DS30027)**
   This document explains how to use Microchip's MPLAB Simulator.
7. **PRO MATE® User's Guide (DS30082)**
   This document explains how to use Microchip's PRO MATE universal programmer.
8. **PICSTART®-Plus User's Guide (DS51028)**
   This document explains how to use Microchip's PICSTART-Plus low-cost universal programmer.
9. ***fuzzy*TECH®-MP User's Guide (DS30389)**
   This document explains how to use the *fuzzy*TECH-MP fuzzy logic code generator.
10. **MP-DriveWay™ User's Guide (DS51027)**
    This document explains how to use the MP-DriveWay code generator.
11. ***fuzzy*TECH-MP Fuzzy Logic Handbook (DS30238)**
    This document explains the basics of *fuzzy*TECH-MP fuzzy.
12. **Embedded Control Handbook Volume I (DS00092)**
    This document contains a plethora of application notes. This is useful for insight on how to use the device (or parts of it) as well as getting started on your particular application due to the availability of extensive code files.
13. **Embedded Control Handbook Volume II (DS00167)**
    This document contains the Math Libraries for PICmicro MCUs.
14. **In-Circuit Serial Programming Guide™ (DS30277)**
    This document discusses implementing In-Circuit Serial Programming.
15. **PICDEM-1 User's Guide (DS351079)**
    This document explains how to use Microchip's PICDEM-1 demo board.
16. **PICDEM-2 User's Guide (DS30374)**
    This document explains how to use Microchip's PICDEM-2 demo board.
17. **PICDEM-3 User's Guide (DS33015)**
    This document explains how to use Microchip's PICDEM-3 demo board.
18. **Third Party Guide (DS00104)**
    This document lists Microchip's third parties, as well as various consultants.
19. **DIE Support (DS30258)**
    This document gives information on using Microchip products in DIE form.

1.7.2 Third Party Documentation

There are several documents available from third party sources around the world. Microchip does not review these documents for technical accuracy, however they may be a helpful source for understanding the operation of Microchip MCU devices. This is not necessarily a complete list, but are the documents that we were aware of at the time of printing. For more information on how to contact some of these sources, as well as any new sources that we become aware of, please visit the Microchip web site.

DOCUMENT	LANGUAGE
The PIC16C5X Microcontroller: A Practical Approach to Embedded Control Bill Rigby/ Terry Dalby, Tecksystems Inc. 0-9654740-0-3	English
Easy PIC'n David Benson, Square 1 Electronics 0-9654162-0-8	English
A Beginners Guide to the Microchip PIC® Nigel Gardner, Bluebird Electronics 1-899013-01-6	English
PIC Microcontroller Operation and Applications DN de Beer, Cape Technikon	English
Digital Systems and Programmable Interface Controllers WP Verburg, Pretoria Technikon	English
Mikroprozessor PIC16C5X Michael Rose, Hüthig 3-7785-2169-1	German
Mikroprozessor PIC17C42 Michael Rose, Hüthig 3-7785-2170-5	German
Les Microcontrolleurs PIC et mise en oeuvre Christian Tavernier, Dunod 2-10-002647-X	French
Micontrolleurs PIC a structure RISC C.F. Urbain, Publitronic 2-86661-058-X	French
New Possibilities with the Microchip PIC RIGA	Russian

DOCUMENT	LANGUAGE
PIC16C5X/71/84 Development and Design, Part 1 United Tech Electronic Co. Ltd 957-21-0807-7	Chinese
PIC16C5X/71/84 Development and Design, Part 2 United Tech Electronic Co. Ltd 957-21-1152-3	Chinese
PIC16C5X/71/84 Development and Design, Part 3 United Tech Electronic Co. Ltd 957-21-1187-6	Chinese
PIC16C5X/71/84 Development and Design, Part 4 United Tech Electronic Co. Ltd 957-21-1251-1	Chinese
PIC16C5X/71/84 Development and Design, Part 5 United Tech Electronic Co. Ltd 957-21-1257-0	Chinese
PIC16C84 MCU Architecture and Software Development ICC Company 957-8716-79-6	Chinese

1.8 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the PIC16CXXX Mid-Range MCU family (that is they may be written for the Base-Line, or the High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to an introduction to Microchip's PICmicro MCUs are:

Title	Application Note #
A Comparison of Low End 8-bit Microcontrollers	AN520
PIC16C54A EMI Results	AN577
Continuous Improvement	AN503
Improving the Susceptibility of an Application to ESD	AN595
Plastic Packaging and the Effects of Surface Mount Soldering Techniques	AN598

1.9 Revision History

Revision A

This is the initial released revision of Microchip's PICmicro MCUs Introduction.

Section 2. Oscillator

HIGHLIGHTS

This section of the manual contains the following major topics:

2.1 Introduction

The internal oscillator circuit is used to generate the device clock. The device clock is required for the device to execute instructions and for the peripherals to function. Four device clock periods generate one internal instruction clock (TCY) cycle.

There are up to eight different modes which the oscillator may have. There are two modes which allow the selection of the internal RC oscillator clock out (CLKOUT) to be driven on an I/O pin, or allow that I/O pin to be used for a general purpose function. The oscillator mode is selected by the device configuration bits. The device configuration bits are nonvolatile memory locations and the operating mode is determined by the value written during device programming. The oscillator modes are:

- LP Low Frequency (Power) Crystal
- XT Crystal/Resonator
- HS High Speed Crystal/Resonator
- RC External Resistor/Capacitor (same as EXTRC with CLKOUT)
- EXTRC External Resistor/Capacitor
- EXTRC External Resistor/Capacitor with CLKOUT
- INTRC Internal 4 MHz Resistor/Capacitor
- INTRC Internal 4 MHz Resistor/Capacitor with CLKOUT

These oscillator options are made available to allow a single device type the flexibility to fit applications with different oscillator requirements. The RC oscillator option saves system cost while the LP crystal option saves power. Configuration bits are used to select the various options. For more details on the device configuration bits, see the **"Device Characteristics"** section.

2.2 Oscillator Configurations

2.2.1 Oscillator Types

Mid-Range devices can have up to eight different oscillator modes. The user can program up to three device configuration bits (FOSC2, FOSC1 and FOSC0) to select one of these eight modes:

- LP Low Frequency (Power) Crystal
- XT Crystal/Resonator
- HS High Speed Crystal/Resonator
- RC External Resistor/Capacitor (same as EXTRC with CLKOUT)
- EXTRC External Resistor/Capacitor
- EXTRC External Resistor/Capacitor with CLKOUT
- INTRC Internal 4 MHz Resistor/Capacitor
- INTRC Internal 4 MHz Resistor/Capacitor with CLKOUT

The main difference between the LP, XT, and HS modes is the gain of the internal inverter of the oscillator circuit which allows the different frequency ranges. Table 2-1 and Table 2-2 give information to aid in selecting an oscillator mode. In general, use the oscillator option with the lowest possible gain which still meet specifications. This will result in lower dynamic currents (IDD). The frequency range of each oscillator mode is the recommended (tested) frequency cutoffs, but the selection of a different gain mode is acceptable as long as a thorough validation is performed (voltage, temperature, component variations (Resistor, Capacitor, and internal microcontroller oscillator circuitry)).

The RC mode and the EXTRC with CLKOUT mode have the same functionality. They are named like this to help describe their operation vs. the other oscillator modes.

Table 2-1: Selecting the Oscillator Mode for Devices with FOSC1:FOSC0

Configuration bits FOSC1:FOSC0	OSC Mode	OSC Feedback Inverter Gain	Comment
1 1	RC	—	Least expensive solution for device oscillation (only an external resistor and capacitor is required). Most variation in time-base. Device's default mode.
1 0	HS	High Gain	High frequency application. Oscillator circuit's mode consumes the most current of the three crystal modes.
0 1	XT	Medium Gain	Standard crystal/resonator frequency. Oscillator circuit's mode consumes the middle current of the three crystal modes.
0 0	LP	Low Gain	Low power/frequency applications. Oscillator circuit's mode consumes the least current of the three crystal modes.

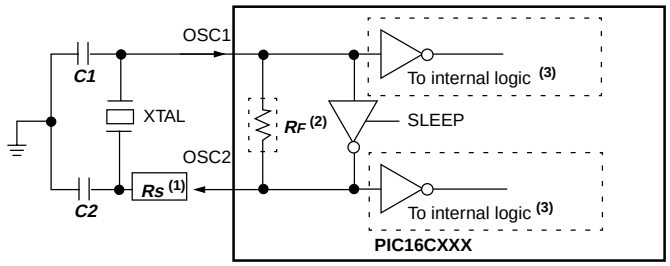
Table 2-2: Selecting the Oscillator Mode for Devices with FOSC2:FOSC0

Configuration bits FOSC2:FOSC0	OSC Mode	OSC Feedback Inverter Gain	Comment
1 1 1	EXTRC with CLKOUT	—	Inexpensive solution for device oscillation. Most variation in timebase. CLKOUT is enabled on pin. Device's default mode.
1 1 0	EXTRC	—	Inexpensive solution for device oscillation. Most variation in timebase. CLKOUT is disabled (use as I/O) on pin.
1 0 1	INTRC with CLKOUT	—	Least expensive solution for device oscillation. 4 MHz oscillator, which can be tuned. CLKOUT is enabled on pin.
1 0 0	INTRC	—	Least expensive solution for device oscillation. 4 MHz oscillator, which can be tuned. CLKOUT is disabled (use as I/O) on pin.
0 1 1	—	—	Reserved
0 1 0	HS	High Gain	High frequency application. Oscillator circuit's mode consumes the most current of the three crystal modes.
0 0 1	XT	Medium Gain	Standard crystal/resonator frequency. Oscillator circuit's mode consumes the middle current of the three crystal modes.
0 0 0	LP	Low Gain	Low power/frequency applications. Oscillator circuit's mode consumes the least current of the three crystal modes.

2.3 Crystal Oscillators / Ceramic Resonators

In XT, LP or HS modes a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation (Figure 2-1). The PICmicro oscillator design requires the use of a parallel cut crystal. Using a series cut crystal may give a frequency out of the crystal manufacturer's specifications. When in XT, LP or HS modes, the device can have an external clock source drive the OSC1 pin (Figure 2-3).

Figure 2-1: Crystal or Ceramic Resonator Operation (HS, XT or LP Oscillator Mode)



Note 1: A series resistor, ***RS***, may be required for AT strip cut crystals.
2: The feedback resistor, ***RF***, is typically in the range of 2 to 10 MΩ.
3: Depending on the device, the buffer to the internal logic may be either before or after the oscillator inverter.

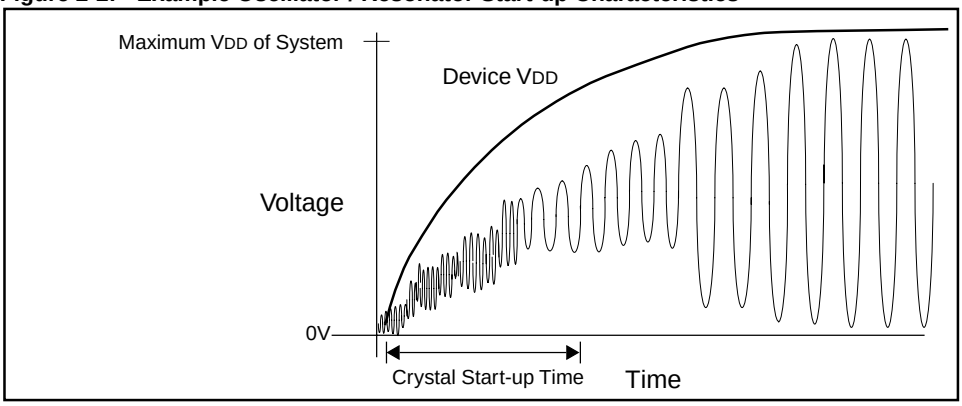
2.3.1 Oscillator / Resonator Start-up

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*** in Figure 2-1)
- Device VDD rise time
- System temperature
- Series resistor value (and type) if used (***Rs*** in Figure 2-1)
- Oscillator mode selection of device (which selects the gain of the internal oscillator inverter)
- Crystal quality
- Oscillator circuit layout
- System noise

Figure 2-2 graphs an example oscillator / resonator start-up. The peak-to-peak voltage of the oscillator waveform can be quite low (less than 50% of device VDD) where the waveform is centered at VDD/2 (refer to parameters D033 and D043 in the **"Electrical Specifications"** section).

Figure 2-2: Example Oscillator / Resonator Start-up Characteristics



2.3.2 Component Selection

Figure 2-1 is a diagram of the devices crystal or ceramic resonator circuitry. The resistance for the feedback resistor, ***RF***, is typically within the 2 to 10 MΩ range. This varies with device voltage, temperature, and process variations. A series resistor, ***Rs***, may be required if an AT strip cut crystal is used. Be sure to include the device's operating voltage and the device's manufacturing process when determining resistor requirements. As you can see in Figure 2-1, the connection to the device's internal logic is device dependent. See the applicable data sheet for device specifics. The typical values of capacitors (***C1***, ***C2***) are given in Table 2-3 and Table 2-4. Each device's data sheet will give the specific values that Microchip tested.

Table 2-3: Typical Capacitor Selection for Ceramic Resonators

Ranges tested:		
Mode	**Frequency**	***C1 / C2*(1)**
XT	455 kHz 2.0 MHz 4.0 MHz	22 - 100 pF 15 - 68 pF 15 - 68 pF
HS	8.0 MHz 16.0 MHz 20.0 MHz	10 - 68 pF 10 - 22 pF TBD

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%
20.0 MHz	TBD	TBD

Note 1: Recommended values of C1 and C2 are identical to the ranges tested above. 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 component or verify oscillator performance.

2: All resonators tested required external capacitors.

Table 2-4: Typical Capacitor Selection for Crystal Oscillator

Mode	Freq	*C1*(1)	*C2*(1)
LP	32 kHz 200 kHz	68 - 100 pF 15 - 30 pF	68 - 100 pF 15 - 30 pF
XT	100 kHz 2 MHz 4 MHz	68 - 150 pF 15 - 30 pF 15 - 30 pF	150 - 200 pF 15 - 30 pF 15 - 30 pF
HS	8 MHz 10 MHz 20 MHz	15 - 30 pF 15 - 30 pF 15 - 30 pF	15 - 30 pF 15 - 30 pF 15 - 30 pF
Crystals used:			
32.768 kHz	Epson C-001R32.768K-A		± 20 PPM
100 kHz	Epson C-2 100.00 KC-P		± 20 PPM
200 kHz	STD XTL 200.000 kHz		± 20 PPM
2.0 MHz	ECS ECS-20-S-2		± 50 PPM
4.0 MHz	ECS ECS-40-S-4		± 50 PPM
10.0 MHz	ECS ECS-100-S-4		± 50 PPM
20.0 MHz	ECS ECS-200-S-4		± 50 PPM

Note 1: Higher capacitance increases the stability of the oscillator but also increases the start-up time. These values are for design guidance only. A series resistor, ***Rs***, may be required in HS mode as well as XT mode to avoid overdriving crystals with low drive level specification. Since each crystal has its own characteristics, the user should consult the crystal manufacturer for appropriate values of external components or verify oscillator performance.

2.3.3 Tuning the Oscillator Circuit

Since Microchip devices have wide operating ranges (frequency, voltage, and temperature; depending on the part and version ordered) and external components (crystals, capacitors,...), of varying quality and manufacture; validation of operation needs to be performed to ensure that the component selection will comply with the requirements of the application.

There are many factors that go into the selection and arrangement of these external components. These factors include:

- amplifier gain
- desired frequency
- resonant frequency(s) of the crystal
- temperature of operation
- supply voltage range
- start-up time
- stability
- crystal life
- power consumption
- simplification of the circuit
- use of standard components
- combination which results in fewest components

2.3.3.1 Determining Best Values for Crystals, Clock Mode, *C1*, *C2*, and *Rs*

The best method for selecting components is to apply a little knowledge and a lot of trial, measurement, and testing.

Crystals are usually selected by their parallel resonant frequency only, however other parameters may be important to your design, such as temperature or frequency tolerance. Application Note AN588 is an excellent reference if you would like to know more about crystal operation and their ordering information.

The PICmicros™ internal oscillator circuit is a parallel oscillator circuit, which requires that a parallel resonant crystal be selected. The load capacitance is usually specified in the 20 pF to 32 pF range. The crystal will oscillate closest to the desired frequency with capacitance in this range. It may be necessary to sometimes juggle these values a bit, as described later, in order to achieve other benefits.

Clock mode is primarily chosen by using the FOSC parameter specification (parameter 1A) in the device's data sheet, based on frequency. Clock modes (except RC) are simply gain selections, lower gain for lower frequencies, higher gain for higher frequencies. It is possible to select a higher or lower gain, if desired, based on the specific needs of the oscillator circuit.

C1* and *C2 should also be initially selected based on the load capacitance as suggested by the crystal manufacturer and the tables supplied in the device data sheet. The values given in the Microchip data sheet can only be used as a starting point, since the crystal manufacturer, supply voltage, and other factors already mentioned may cause your circuit to differ from the one used in the factory characterization process.

Ideally, the capacitance is chosen (within the range of the recommended crystal load preferably) so that it will oscillate at the highest temperature and lowest VDD that the circuit will be expected to perform under. High temperature and low VDD both have a limiting affect on the loop gain, such that if the circuit functions at these extremes, the designer can be more assured of proper operation at other temperatures and supply voltage combinations. The output sine wave should not be clipped in the highest gain environment (highest VDD and lowest temperature) and the sine output amplitude should be great enough in the lowest gain environment (lowest VDD and highest temperature) to cover the logic input requirements of the clock as listed in the device data sheet.

A method for improving start-up is to use a value of ***C2*** greater than ***C1***. This causes a greater phase shift across the crystal at power-up, which speeds oscillator start-up.

Besides loading the crystal for proper frequency response, these capacitors can have the effect of lowering loop gain if their value is increased. ***C2*** can be selected to affect the overall gain of the circuit. A higher ***C2*** can lower the gain if the crystal is being over driven (see also discussion on ***Rs***). Capacitance values that are too high can store and dump too much current through the crystal, so ***C1*** and ***C2*** should not become excessively large. Unfortunately, measuring the wattage through a crystal is tricky business, but if you do not stray too far from the suggested values you should not have to be concerned with this.

A series resistor, ***Rs***, is added to the circuit if, after all other external components are selected to satisfaction, the crystal is still being over driven. This can be determined by looking at the OSC2 pin, which is the driven pin, with an oscilloscope. Connecting the probe to the OSC1 pin will load the pin too much and negatively affect performance. Remember that a scope probe adds its own capacitance to the circuit, so this may have to be accounted for in your design, i.e. if the circuit worked best with a ***C2*** of 20 pF and scope probe was 10 pF, a 30 pF capacitor may actually be called for. The output signal should not be clipping or squashed. Overdriving the crystal can also lead to the circuit jumping to a higher harmonic level or even crystal damage.

The OSC2 signal should be a nice clean sine wave that easily spans the input minimum and maximum of the clock input pin (4V to 5V peak to peak for a 5V VDD is usually good). An easy way to set this is to again test the circuit at the minimum temperature and maximum VDD that the design will be expected to perform in, then look at the output. This should be the maximum amplitude of the clock output. If there is clipping or the sine wave is squashing near VDD and VSS at the top and bottom, and increasing load capacitors will risk too much current through the crystal or push the value too far from the manufacturer's load specification, then add a trimpot between the output pin and ***C2***, and adjust it until the sine wave is clean. Keeping it fairly close to maximum amplitude at the low temperature and high VDD combination will assure this is the maximum amplitude the crystal will see and prevent overdriving. A series resistor, ***Rs***, of the closest standard value, can now be inserted in place of the trimpot. If ***Rs*** is too high, perhaps more than 20k ohms, the input will be too isolated from the output, making the clock more susceptible to noise. If you find a value this high is needed to prevent overdriving the crystal, try increasing ***C2*** to compensate. Try to get a combination where ***Rs*** is around 10k or less, and load capacitance is not too far from the 20 pF or 32 pF manufacturer specification.

2.3.3.1.1 Start-up

The most difficult time for the oscillator to start-up is when waking up from sleep. This is because the load capacitors have both partially charged to some quiescent value, and phase differential at wake-up is minimal. Thus, more time is required to achieve stable oscillation. Remember also that low voltage, high temperatures, and the lower frequency clock modes also impose limitations on loop gain, which in turn affects start-up. Each of the following factors makes thing worse:

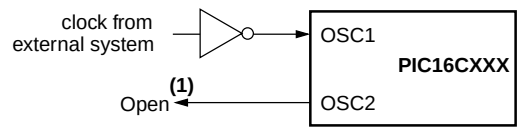
- a low frequency design (with its low gain clock mode)
- a quiet environment (such as a battery operated device)
- operating outside the noisy RF area (such as in a shielded box)
- low voltage
- high temperature
- waking up from sleep.

Noise actually helps a design for oscillator start-up, since it helps kick start the oscillator.

2.3.4 External Clock Input

If the PICmicro's internal oscillator is not being used, and the device will be driven from an external clock, be sure to set the oscillator mode to one of the crystal modes (LP, XT, or HS). That is, something other than one of the RC modes, since RC mode will fight with the injected input. Ideally you would select the mode that corresponds to the frequency injected, but this is of less importance here since the clock is only driving its internal logic, and not a crystal loop circuit. It may be possible to select a clock mode lower than would be needed by an oscillator circuit, and thereby save some of the power that would be used exercising the inverting amplifier. Make sure the OSC2 signal amplitude covers the needed logic thresholds of the device.

Figure 2-3: External Device Clock Input Operation (HS, XT or LP Oscillator Modes)



Note 1: A resistor to ground may be used to reduce system noise. This may increase system current.

2.3.5 External Crystal Oscillator Circuit for Device Clock

Sometimes more than one device needs to be clocked from a single crystal. Since Microchip does not recommend connecting other logic to the PICmicro's internal oscillator circuit, an external crystal oscillator circuit is recommended. Each device will then have an external clock source, and the number of devices that can be driven will depend on the buffer drive capability. This circuit is also useful when more than one device (PICmicro) needs to operate synchronously to each other.

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 2-4 shows implementation of an external 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.

Figure 2-4: External Parallel Resonant Crystal Oscillator Circuit

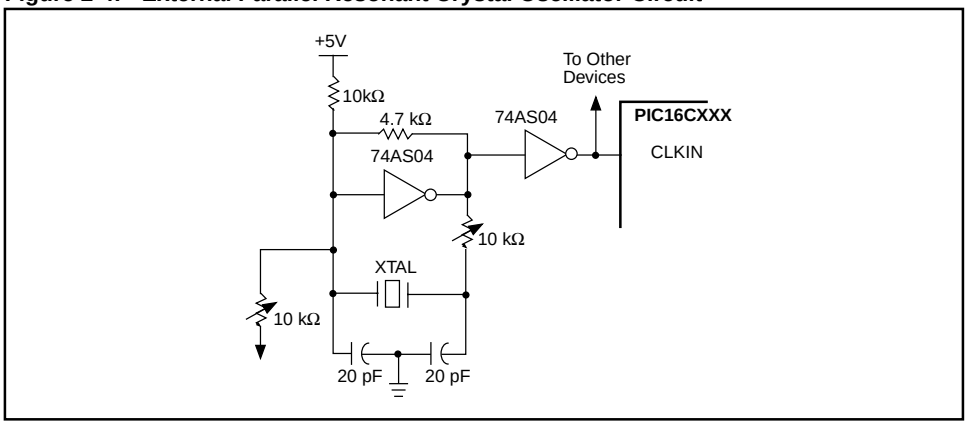
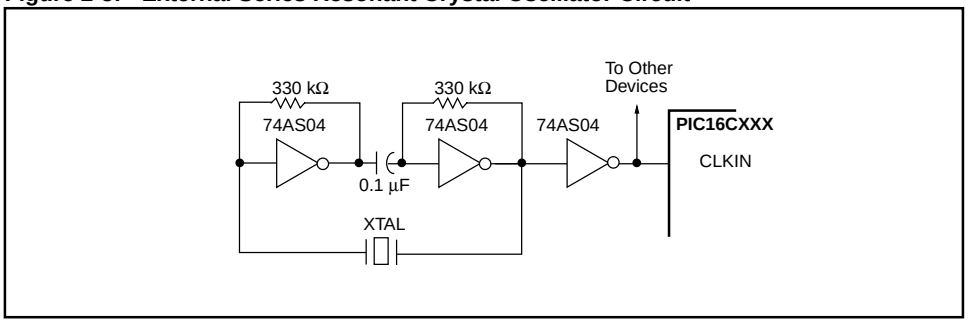


Figure 2-5 shows an external series resonant oscillator circuit. This circuit is also designed to use the fundamental frequency of the crystal. The inverter performs a 180-degree phase shift in a series resonant oscillator circuit. The 330 kΩ resistors provide the negative feedback to bias the inverters in their linear region.

Figure 2-5: External Series Resonant Crystal Oscillator Circuit

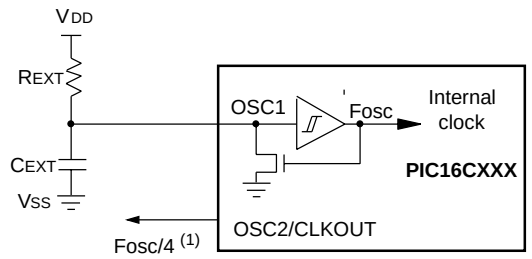


When the device is clocked from an external clock source (as in Figure 2-4 or Figure 2-5) then the microcontroller's oscillator must be configured for LP, XT or HS mode (Figure 2-3).

2.4 External RC Oscillator

For timing insensitive applications the “EXTRC” device option offers additional cost savings. The 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, the oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency, especially for low CEXT values. The user also needs to take into account variation due to tolerance of external REXT and CEXT components used. Figure 2-6 shows how the RC combination is connected to a PIC16CXXX. For REXT values below 2.2 kΩ, 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 keeping REXT between 3 kΩ and 100 kΩ.

Figure 2-6: EXTRC Oscillator Mode



Note 1: This output may also be able to be configured as a general purpose I/O pin.

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 no or small external capacitance, the oscillation frequency can vary dramatically due to changes in external capacitances, such as PCB trace capacitance and package lead frame capacitance.

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

See characterization data for variation of oscillator frequency due to VDD for given REXT/CEXT values as well as frequency variation due to operating temperature for given REXT, CEXT, 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 4-3: "Clock/Instruction Cycle"** in the **“Architecture”** section, for waveform).

2.4.1 RC Start-up

As the device voltage increases, the RC will start its oscillations immediately after the pin voltage levels meet the input threshold specifications (parameters D032 and D042 in the **“Electrical Specifications”** 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

2.5 Internal 4 MHz RC Oscillator

The internal RC oscillator (not on all devices) provides a fixed 4 MHz (nominal) system clock at V_{DD} = 5V and 25°C, see the device data sheet's "Electrical Specifications" section for information on variation over voltage and temperature.

The value in the OSCCAL register is used to tune the frequency of the internal RC oscillator. The calibration value that Microchip programs into the device will "trim" the internal oscillator to remove process variation from the oscillator frequency. The CAL3:CAL0 bits are used for fine calibration within a frequency window. Higher values of CAL3:CAL0 (from 0000 to 1111) yields higher clock speeds.

When a 4 MHz internal RC oscillator frequency cannot be achieved by a CAL3:CAL0 value, the RC oscillator frequency can be increased or decreased by an offset frequency. The CALFST and CALSLW bits are used to enable a positive or negative frequency offset to place the internal RC frequency within the CAL3:CAL0 trim window.

Setting the CALFST bit offsets the internal RC for a higher frequency, while setting the CALSLW bit offsets the internal RC for a lower frequency.

Upon a device reset, the OSCCAL register is forced to the midpoint value (CAL3:CAL0 = 7h, CALFST and CALSLW providing no offset).

Register 2-1: OSCCAL Register

R/W-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0	U-0	U-0
CAL3	CAL2	CAL1	CAL0	CALFST	CALSLW	—	—
bit 7							bit 0

bit 7:4 **CAL3:CAL0:** Internal RC Oscillator Calibration bits
0000 = Lowest clock frequency within the trim range

•
•
•

1111 = Highest clock frequency within the trim range

bit 3 **CALFST:** Oscillator Range Offset bit
1 = Increases the frequency of the internal RC oscillator into the CAL3:CAL0 trim window
0 = No offset provided

bit 2 **CALSLW:** Oscillator Range Offset bit
1 = Decreases the frequency of the internal RC oscillator into the CAL3:CAL0 trim window
0 = No offset provided

Note: When both bits are set, the CALFST bit overrides the CALSLW bit.

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

Note: These bits should be written as '0' when modifying the OSCCAL register, for compatibility with future devices.

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

Note: OSCCAL is used to remove process variation from the internal RC oscillator of the device. The OSCCAL value should not be modified from the Microchip supplied value, and all timing critical functions should be adjusted by the application software.

Figure 2-7 shows the possible device frequencies from the uncalibrated point (at V_{DD} = 5V, 25°C, and OSCCAL = 70h), and the changes achievable by the OSCCAL register.

Figure 2-7: Ideal Internal RC Oscillator Frequency vs. OSCCAL Register Value

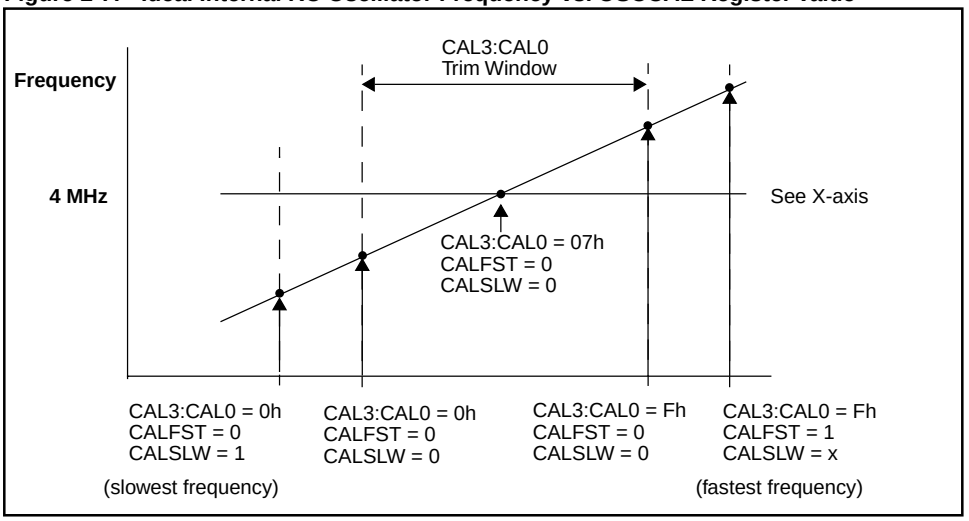


Figure 2-8 shows an example of a device where by selecting one of the CAL3:CAL0 values, the frequency can corrected to 4 MHz. These bits can be considered the fine trimming of the frequency. Sometimes the device frequency at the uncalibrated point cannot be corrected to 4 MHz by the fine trimming of the CAL3:CAL0 bits value. Therefore two additional bits are available which give a large frequency offset (positive and negative) to move the frequency within the range where the fine trimming can work. These bits are the CALSLW and CALFST bits, which offset the internal RC frequency. The action of these bits are shown in Figure 2-9, and Figure 2-10.

Figure 2-8: CAL3:CAL0 Trimming of Internal RC Oscillator Frequency Offset

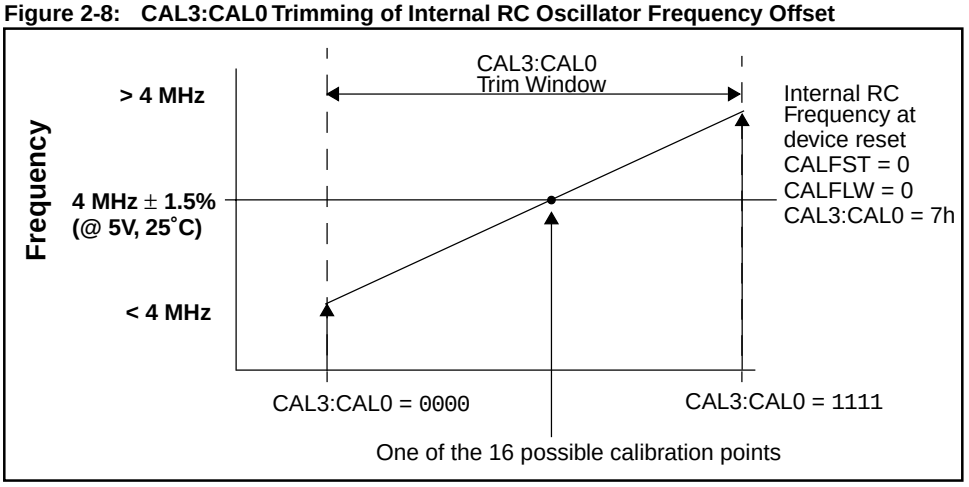


Figure 2-9: CALFST Positive Internal RC Oscillator Frequency Offset

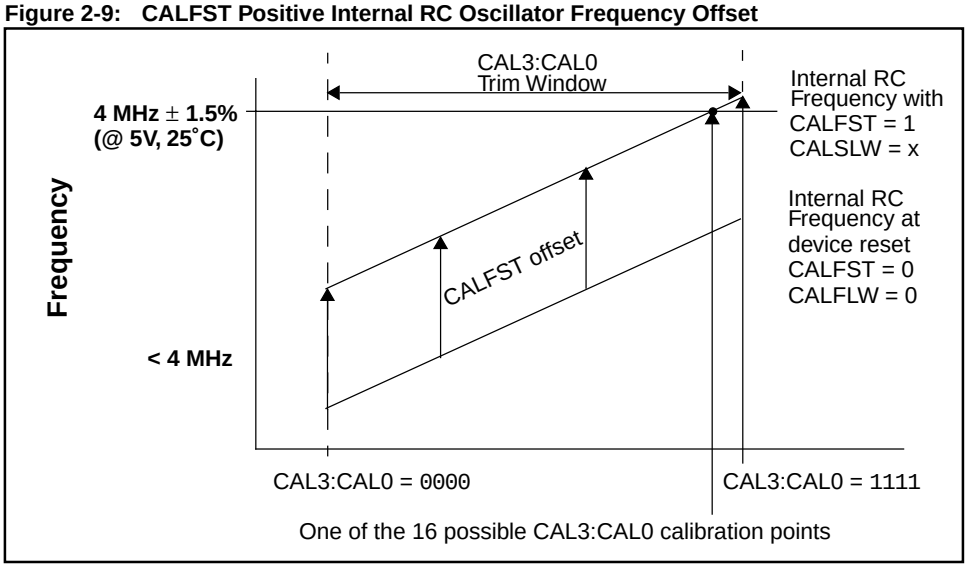
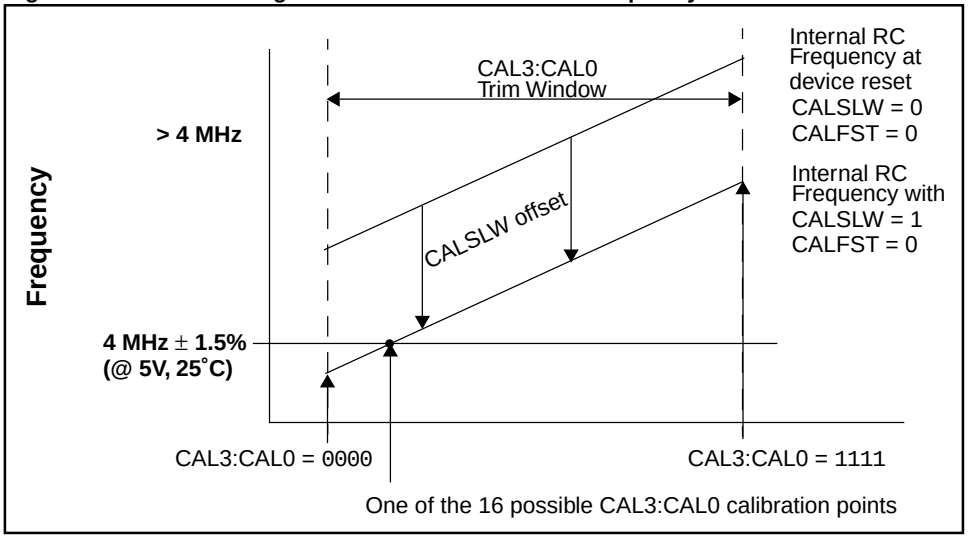


Figure 2-10: CALSLW Negative Internal RC Oscillator Frequency Offset



A calibration instruction is programmed into the last address of the implemented program memory. This instruction contains the calibration value for the internal RC oscillator. This value is programmed as a RETLW XX instruction where XX is the calibration value. In order to retrieve the calibration value, issue a CALL YY instruction where YY is the last location in the device's user accessible program memory. The calibration value is now loaded in the W register. The program should then perform a MOVWF OSCCAL instruction to load the value into the internal RC oscillator calibration register. Table 2-5 shows the location of the calibration value depending on the size of the program memory.

Table 2-5: Calibration Value Location

Program Memory Size (Words)	Calibration Value Location
512	1FFh
1K	3FFh
2K	7FFh
4K	FFFh
8K	1FFFh

Note 1: Erasing the device (windowed devices) will also erase the factory programmed calibration value for the internal oscillator.

Prior to erasing a windowed device, the internal oscillator calibration value must be saved. It is a good idea to write this value on the package of the device to ensure that the calibration value is not accidently lost.
This saved valued must be restored into program memory calibration location before programming the device.

Note 2: OSCCAL<1:0> are unimplemented and should be written as '0'. This will help ensure compatibility with future devices.

2.5.1 Clock Out

The internal RC oscillator can be configured to provide a clock out signal on the CLKOUT pin when the configuration word address (2007h) is programmed with FOSC2, FOSC1, FOSC0 equal to '101' for Internal RC or '111' for External RC. CLKOUT, which is divided by 4, can be used for test purposes or to synchronize other logic.

When the calibration value of the internal RC oscillator is accidently erased, the clock out feature allows the user to determine what the calibration value should be. This is achieved by writing a program which modifies (increments/decrements) the value of the OSCCAL register. When the CLKOUT pin is at 4 MHz (± 1.5%) at 5V and 25˚C, the OSCCAL register has the correct calibration value. This value then needs to be written to a port or shifted out serially, so that the value can be written down and programmed into the calibration location.

2.6 Effects of Sleep Mode on the On-chip Oscillator

When the device executes a SLEEP instruction, the on-chip clocks and oscillator are turned off and the device is held at the beginning of an instruction cycle (Q1 state). With the oscillator off, the OSC1 and OSC2 signals will stop oscillating. Since all the transistor switching currents have been removed, sleep mode achieves the lowest current consumption of the device (only leakage currents). Enabling any on-chip feature that will operate during sleep will increase the current consumed during sleep. The user can wake from SLEEP through external reset, Watchdog Timer Reset or through an interrupt.

Table 2-6: OSC1 and OSC2 Pin States in Sleep Mode

OSC Mode	OSC1 Pin	OSC2 Pin
EXTRC	Floating, external resistor should pull high	At logic low
INTRC	N.A.	N.A.
LP, XT, and HS	Feedback inverter disabled, at quiescent voltage level	Feedback inverter disabled, at quiescent voltage level

See Table 3-1, in the **"Reset"** section, for time-outs due to Sleep and $\overline{\text{MCLR}}$ reset.

2.7 Effects of Device Reset on the On-chip Oscillator

Device resets have no effect on the on-chip crystal oscillator circuitry. The oscillator will continue to operate as it does under normal execution. While in reset, the device logic is held at the Q1 state so that when the device exits reset, it is at the beginning of an instruction cycle.

The OSC2 pin, when used as the external clockout (EXTRC mode), will be held low during reset, and as soon as the $\overline{\text{MCLR}}$ pin is at V_{IH} (input high voltage), the RC will start to oscillate. See Table 3-1, in the **"Reset"** section, for time-outs due to Sleep and $\overline{\text{MCLR}}$ reset.

2.7.1 Power-up Delays

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 72 ms (nominal) on power-up only (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. For additional information on reset operation, see the **"Reset"** section.

2.8 Design Tips

Question 1: ***When looking at the OSC2 pin after power-up with an oscilloscope, there is no clock. What can cause this?***

Answer 1:

1. Executing a SLEEP instruction with no source for wake-up (such as, WDT, $\overline{\text{MCLR}}$, or an Interrupt). Verify that the code does not put device to sleep without providing for wake-up. If it is possible, try waking it up with a low pulse on $\overline{\text{MCLR}}$. Powering up with $\overline{\text{MCLR}}$ held low will also give the crystal oscillator more time to start-up, but the Program Counter will not advance until the $\overline{\text{MCLR}}$ pin is high.
2. The wrong clock mode is selected for the desired frequency. For a blank device, the default oscillator is EXTRC. Most parts come with the clock selected in the default RC mode, which will not start oscillation with a crystal or resonator. Verify that the clock mode has been programmed correctly.
3. The proper power-up sequence has not been followed. If a CMOS part is powered through an I/O pin prior to power-up, bad things can happen (latch up, improper start-up etc.) It is also possible for brown-out conditions, noisy power lines at start-up, and slow V_{DD} rise times to cause problems. Try powering up the device with nothing connected to the I/O, and power-up with a known, good, fast-rise, power supply. It is not as much of a problem as it may sound, but the possibility exists. Refer to the power-up information in the device data sheet for considerations on brown-out and power-up sequences.
4. The ***C1*** and ***C2*** capacitors attached to the crystal have not been connected properly or are not the correct values. Make sure all connections are correct. The device data sheet values for these components will almost always get the oscillator running, they just might not be the optimal values for your design.

Question 2: ***The PICmicro starts, but runs at a frequency much higher than the resonant frequency of the crystal.***

Answer 2:

The gain is too high for this oscillator circuit. Refer to subsection **2.3 "Crystal Oscillators / Ceramic Resonators"** to aid in the selection of ***C2*** (may need to be higher) ***Rs*** (may be needed) and clock mode (wrong mode may be selected). This is especially possible for low frequency crystals, like the common 32.768 kHz.

Question 3: ***The design runs fine, but the frequency is slightly off, what can be done to adjust this?***

Answer 3:

Changing the value of ***C1*** has some affect on the oscillator frequency. If a SERIES resonant crystal is used, it will resonate at a different frequency than a PARALLEL resonant crystal of the same frequency call-out.

Question 4: ***The board works fine, then suddenly quits, or loses time.***

Answer 4:

Other than the obvious software checks that should be done to investigate losing time, it is possible that the amplitude of the oscillator output is not high enough to reliably trigger the oscillator input.

Question 5: ***I'm using a device with the internal RC oscillator and I have accidently erased the calibration value. What can I do?***

Answer 5:

If the frequency of the device does not matter, you can continue to use the device.

If the frequency of the device does matter, you can purchase a new windowed device, or follow the suggestion in subsection **2.5.1 "Clock Out."**

2.9 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the oscillator are:

Title	Application Note #
PIC16/17 Oscillator Design Guide	AN588
Low Power Design using PIC16/17	AN606

2.10 Revision History

Revision A

This is the initial released revision of the PICmicro oscillators description.

Section 3. Reset

HIGHLIGHTS

This section of the manual contains the following major topics:

3.1 Introduction

The reset logic is used to place the device into a known state. The source of the reset can be determined by using the device status bits. The reset logic is designed with features that reduce system cost and increase system reliability.

Devices differentiate between various kinds of reset:

a) Power-on Reset (POR)
b) $\overline{\text{MCLR}}$ reset during normal operation
c) $\overline{\text{MCLR}}$ reset during SLEEP
d) WDT reset during normal operation
e) Brown-out Reset (BOR)
f) Parity Error Reset (PER)

Most registers are unaffected by a reset; their status is unknown on POR and unchanged by all other resets. The other registers are forced to a "reset state" on Power-on Reset, $\overline{\text{MCLR}}$, WDT reset, Brown-out Reset, Parity Error Reset, and on $\overline{\text{MCLR}}$ reset during SLEEP.

The on-chip parity bits that can be used to verify the contents of program memory.

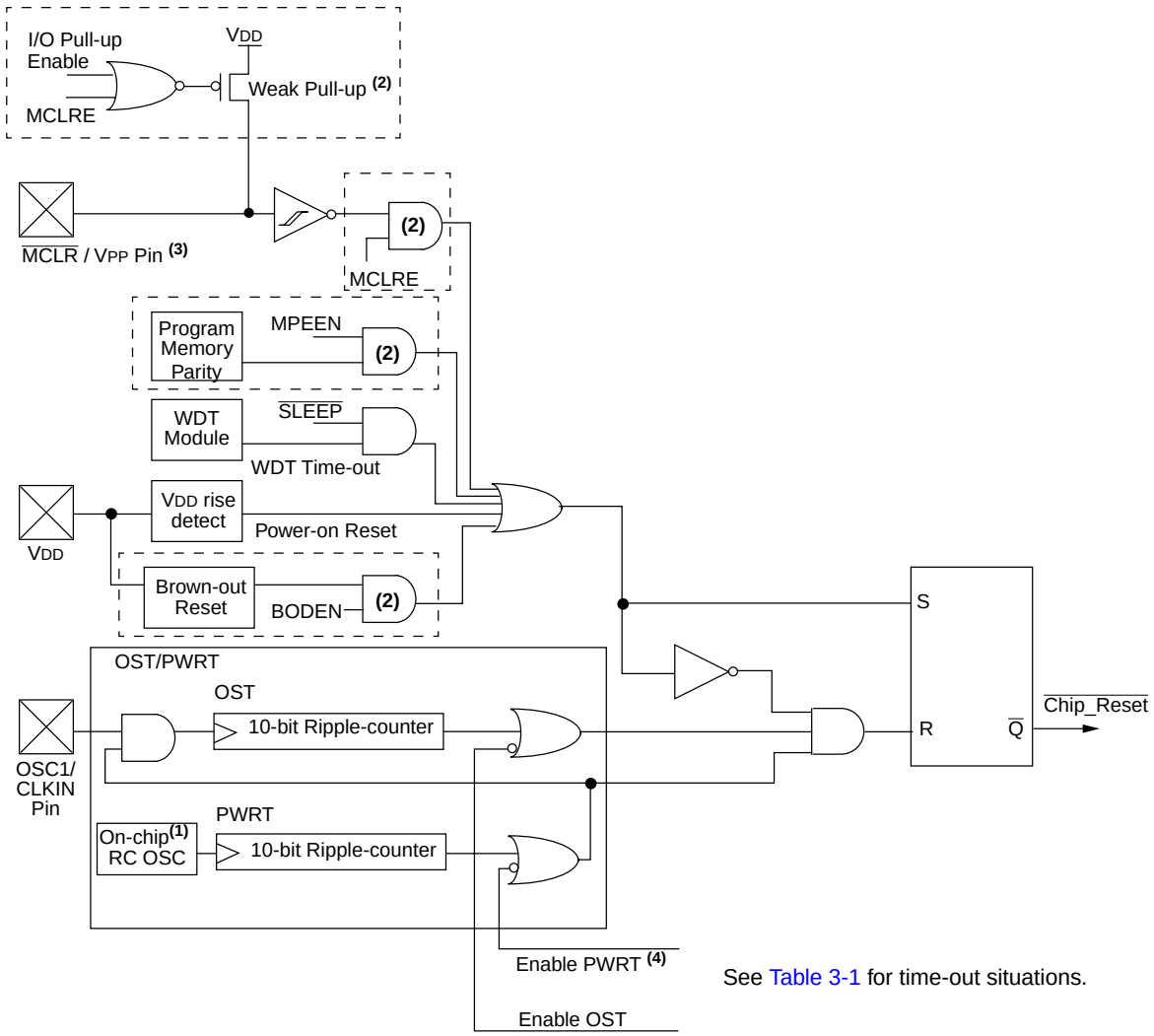
Most registers are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. Status bits $\overline{\text{TO}}$, $\overline{\text{PD}}$, $\overline{\text{POR}}$, $\overline{\text{BOR}}$, and $\overline{\text{PER}}$ are set or cleared differently in different reset situations as indicated in Table 3-2. These bits are used in software to determine the nature of the reset. See Table 3-4 for a full description of the reset states of all registers.

A simplified block diagram of the on-chip reset circuit is shown in Figure 3-1. This block diagram is a superset of reset features. To determine the features that are available on a specific device, please refer to the device's Data Sheet.

Note:	While the PICmicro™ is in a reset state, the internal phase clock is held at Q1 (beginning of an instruction cycle).

All new devices will have a noise filter in the $\overline{\text{MCLR}}$ reset path to detect and ignore small pulses. See parameter 30 in the **"Electrical Specifications"** section for pulse width specification.

Figure 3-1: Simplified Block Diagram of a Super-set On-chip Reset Circuit



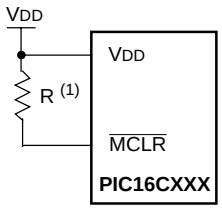
Note 1: This is a separate oscillator from the RC oscillator of the CLKIN pin or the INTRC oscillator.
2: Features in dashed boxes not available on all devices, see device's Data Sheet.
3: In some devices, this pin may be configured as a general purpose Input.
4: The early PICmicro devices have the configuration bit defined as PWRTE = 1 is enabled, while all other devices the configuration bit is defined as $\overline{\text{PWRTE}}$ = 0 is enabled.

3.2 Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST), Brown-out Reset (BOR), and Parity Error Reset (PER)

3.2.1 Power-on Reset (POR)

A Power-on Reset pulse is generated on-chip when VDD rise is detected. To take advantage of the POR, just tie the MCLR pin directly (or through a resistor) to VDD as shown in Figure 3-2. This will eliminate external RC components usually needed to create a Power-on Reset. A minimum rise time for VDD is required. See parameter D003 and parameter D004 in the **"Electrical Specifications"** section for details.

Figure 3-2: Using On-Chip POR

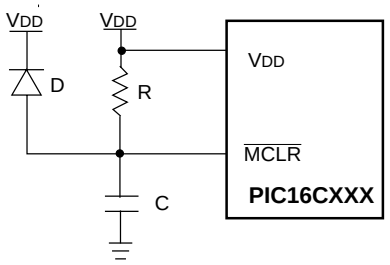


Note: The resistor is optional.

When the device exits the reset condition (begins normal operation), the device operating parameters (voltage, frequency, temperature, etc.) must be within their operating ranges, otherwise the device will not function correctly. Ensure the delay is long enough to get all operating parameters within specification.

Figure 3-3 shows a possible POR circuit for a slow power supply ramp up. The external Power-on Reset circuit is only required if VDD power-up time is too slow. The diode, ***D***, helps discharge the capacitor quickly when VDD powers down.

Figure 3-3: External Power-on Reset Circuit (For Slow VDD Power-up)



Note: R < 40 kΩ is recommended to ensure that the voltage drop across R does not exceed 0.2V. A larger voltage drop will degrade VIH level on the MCLR/VPP pin.

3.2.2 Power-up Timer (PWRT)

The Power-up Timer provides a nominal 72 ms delay on Power-on Reset (POR) or Brown-out Reset (BOR), see parameter 33 in the **"Electrical Specifications"** section. The Power-up Timer operates on a dedicated internal RC oscillator. The device is kept in reset as long as the PWRT is active. The PWRT delay allows VDD to rise to an acceptable level. The power-up timer enable configuration bit can enable/disable the Power-up Timer. The Power-up Timer should always be enabled when Brown-out Reset is enabled. The polarity of the Power-up Timer configuration bit is now $\overline{\text{PWRTE}}$ = 0 for enabled, while the initial definition of the bit was PWRTE = 1 for enabled. Since all new devices will use the $\overline{\text{PWRTE}}$ = 0 for enabled, the text will describe the operation for such devices. Please refer to the individual Data Sheet to ensure the correct polarity for this bit.

The power-up time delay will vary from device to device due to VDD, temperature, and process variations. See DC parameters for details.

3.2.3 Oscillator Start-up Timer (OST)

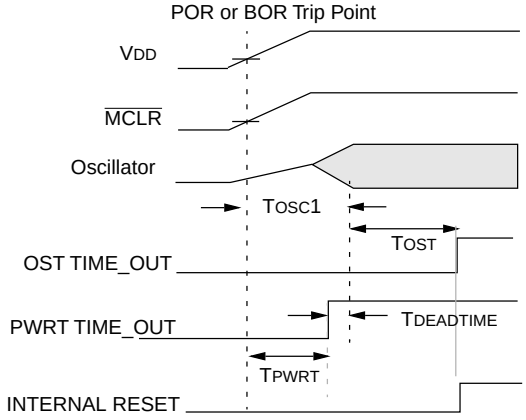
The Oscillator Start-Up Timer (OST) provides a 1024 oscillator cycle delay (from OSC1 input) after the PWRT delay is over. This ensures that the crystal oscillator or resonator has started and is stable.

The OST time-out is invoked only for XT, LP and HS modes and only on Power-on Reset, Brown-out Reset, or wake-up from SLEEP.

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

Figure 3-4 shows the operation of the OST circuit in conjunction with the power-up timer. For low frequency crystals this start-up time can become quite long. That is because the time it takes the low frequency oscillator to start oscillating is longer than the power-up timer's delay. So the time from when the power-up timer times-out, to when the oscillator starts to oscillate is a dead time. There is no minimum or maximum time for this dead time (*TDEADTIME*).

Figure 3-4: Oscillator Start-up Time



Tosc1 = time for the crystal oscillator to react to an oscillation level detectable by the Oscillator Start-up Timer (OST).

TOST = 1024TOSC.

3.2.4 Power-up Sequence

On power-up, the time-out sequence is as follows: First the internal POR is detected, then, if enabled, the PWRT time-out is invoked. After the PWRT time-out is over, the OST is activated. The total time-out will vary based on oscillator configuration and $\overline{PWRTE}$ bit status. For example, in RC mode with the $\overline{PWRTE}$ bit set (PWRT disabled), there will be no time-out at all. Figure 3-5, Figure 3-6 and Figure 3-7 depict time-out sequences.

Since the time-outs occur from the internal POR pulse, if $\overline{MCLR}$ is kept low long enough, the time-outs will expire. Then bringing $\overline{MCLR}$ high will begin execution immediately (Figure 3-7). This is useful for testing purposes or to synchronize more than one device operating in parallel.

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

Table 3-1 shows the time-outs that occur in various situations, while Figure 3-5 through Figure 3-8 show four different cases that can happen on powering up the device.

Table 3-1: Time-out in Various Situations

Oscillator Configuration	Power-up Timer		Brown-out Reset	Wake-up from SLEEP
	Enabled	Disabled		
XT, HS, LP	72 ms + 1024TOSC	1024TOSC	72 ms + 1024TOSC	1024TOSC
RC	72 ms	— (1)	72 ms	— (1)

Note 1: Devices with the Internal/External RC option have a nominal 250 µs delay.

Figure 3-5: Time-out Sequence on Power-up ($\overline{MCLR}$ Tied to VDD)

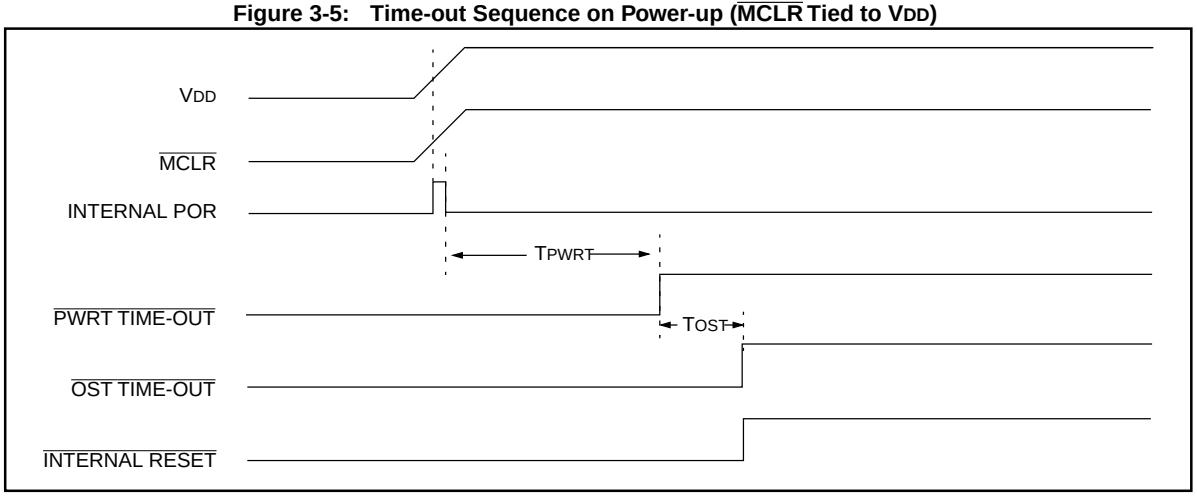


Figure 3-6: Time-out Sequence on Power-up (MCLR not Tied to VDD): Case 1

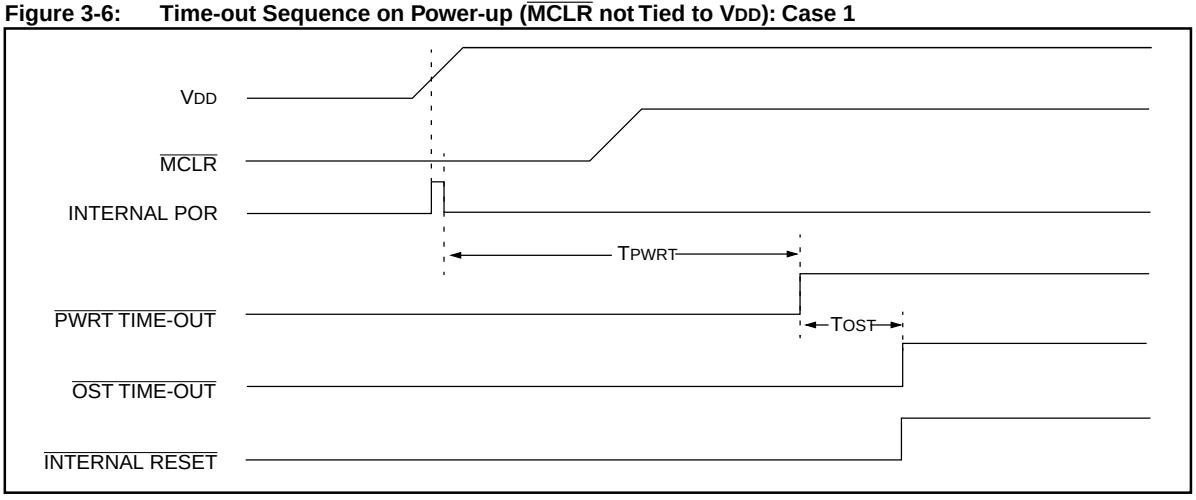


Figure 3-7: Time-out Sequence on Power-up (MCLR not Tied to VDD): Case 2

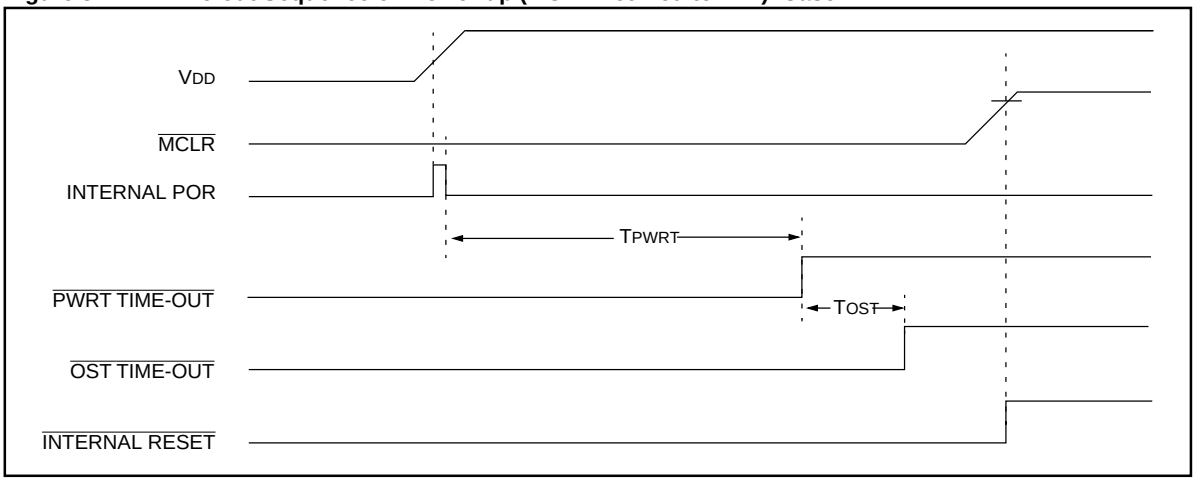
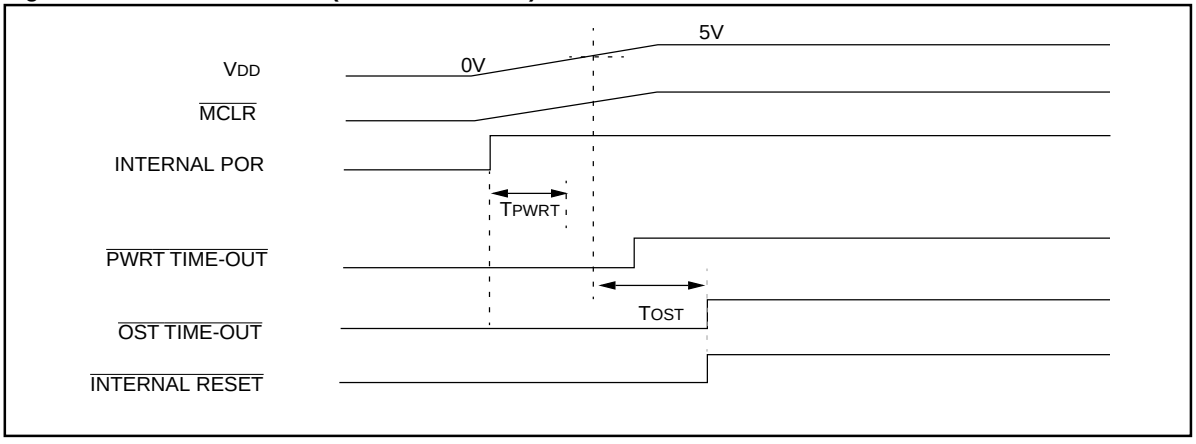


Figure 3-8: Slow Rise Time (MCLR Tied to VDD)



3.2.5 Brown-out Reset (BOR)

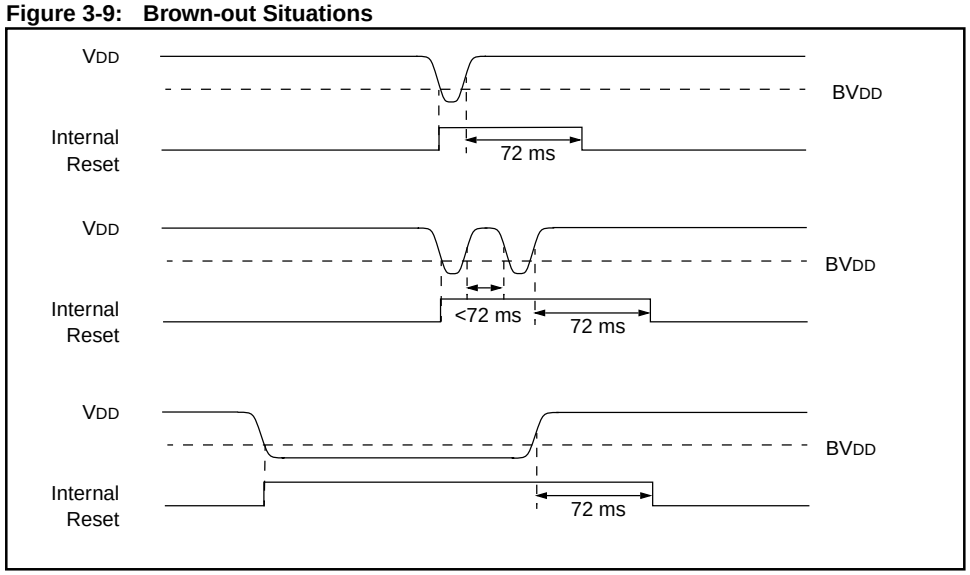
On-chip Brown-out Reset circuitry places the device into reset when the device voltage falls below a trip point (BVDD). This ensures that the device does not continue program execution outside the valid operation range of the device. Brown-out resets are typically used in AC line applications or large battery applications where large loads may be switched in (such as automotive), and cause the device voltage to temporarily fall below the specified operating minimum.

Note:	Before using the on-chip brown-out for a voltage supervisory function (monitor battery decay), 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 4.0V, parameter D005 in the **"Electrical Specifications"** 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 parameter 35. The chip will remain in Brown-out Reset until VDD rises above BVDD. The Power-up Timer will now be invoked and will keep the chip in reset an additional 72 ms. If VDD drops below BVDD while the Power-up Timer is running, the chip will go back into Reset and the Power-up Timer will be re-initialized. Once VDD rises above BVDD, the Power-up Timer will again start a 72 ms time delay. Figure 3-9 shows typical Brown-out situations.

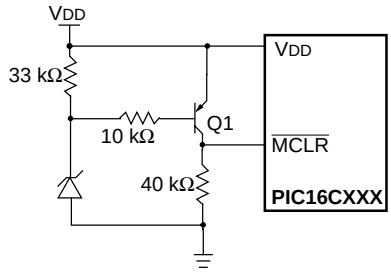
With the BODEN bit set, all voltages below BVDD will hold the device in the reset state. This includes during the power-up sequence.

Figure 3-9: Brown-out Situations



Some devices do not have the on-chip brown-out circuit, and in other cases there are some applications where the Brown-out Reset trip point of the device may not be at the desired level. Figure 3-10 and Figure 3-11 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.

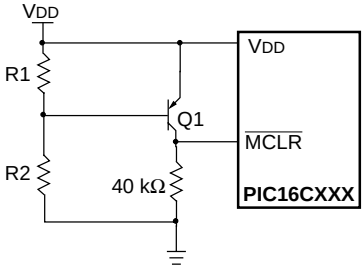
Figure 3-10: External Brown-out Protection Circuit 1



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

Note 1: Internal Brown-out Reset circuitry should be disabled when using this circuit.
2: Resistors should be adjusted for the characteristics of the transistor.

Figure 3-11: External Brown-out Protection Circuit 2



Note 1: This brown-out circuit is less expensive, albeit less accurate. Transistor Q1 turns off when VDD is below a certain level such that:

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

2: Internal Brown-out Reset circuitry should be disabled when using this circuit.
3: Resistors should be adjusted for the characteristics of the transistor.

3.3 Registers and Status Bit Values

Table 3-2: Status Bits and Their Significance

$\overline{POR}$	$\overline{BOR}^{(1)}$	$\overline{TO}$	$\overline{PD}$	Condition
0	x	1	1	Power-on Reset
0	x	0	x	Illegal, $\overline{TO}$ is set on $\overline{POR}$
0	x	x	0	Illegal, $\overline{PD}$ is set on $\overline{POR}$
1(2)	0	1	1	Brown-out Reset
1(2)	1(2)	0	1	WDT Reset
1(2)	1(2)	0	0	WDT Wake-up
1(2)	1(2)	u	u	$\overline{MCLR}$ reset during normal operation
1(2)	1(2)	1	0	$\overline{MCLR}$ reset during SLEEP

Legend: u = unchanged, x = unknown, - = unimplemented bit, reads as '0'.
Note 1: Not all devices have BOR circuitry.
2: These bits are unchanged for the given conditions, and when initialized (set) after a POR or a BOR will read as a '1'.

Table 3-3: Initialization Condition for Special Registers

Condition	Program Counter	STATUS Register	PCON Register
Power-on Reset	000h	0001 1xxx	u--- -10x
$\overline{MCLR}$ reset during normal operation	000h	000u uuuu	u--- -uuu
$\overline{MCLR}$ reset during SLEEP	000h	0001 0uuu	u--- -uuu
WDT reset	000h	0000 1uuu	u--- -uuu
WDT Wake-up	PC + 1	uuu0 0uuu	u--- -uuu
Brown-out Reset	000h	0001 1uuu	u--- -uu0
Interrupt Wake-up from SLEEP	PC + 1(1)	uuu1 0uuu	u--- -uuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, reads as '0'.
Note 1: When the wake-up is due to an interrupt and global enable bit, GIE, is set the PC is loaded with the interrupt vector (0004h) after execution of PC+1.
2: If a status bit is not implemented, that bit will be read as '0'.

Table 3-4: Initialization Conditions for Special Function Registers

Register	Power-on Reset Brown-out Reset	$\overline{\text{MCLR}}$ Reset during: - normal operation - SLEEP or WDT Reset	Wake-up from SLEEP through: - interrupt - WDT time-out
ADCAPL	0000 0000	0000 0000	uuuu uuuu
ADCAPH	0000 0000	0000 0000	uuuu uuuu
ADCON0	0000 00-0	0000 00-0	uuuu uu-u
ADCON1	---- -000	---- -000	---- -uuu
ADRES	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADTMRL	0000 0000	0000 0000	uuuu uuuu
ADMRH	0000 0000	0000 0000	uuuu uuuu
CCP1CON	--00 0000	--00 0000	--uu uuuu
CCP2CON	0000 0000	0000 0000	uuuu uuuu
CCPR1L	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1H	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2L	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2H	xxxx xxxx	uuuu uuuu	uuuu uuuu
CMCON	00-- 0000	00-- 0000	uu-- uuuu
EEADR	xxxx xxxx	uuuu uuuu	uuuu uuuu
EECON1	---0 x000	---0 q000	---0 uuuu
EECON2	-	-	-
EEDATA	xxxx xxxx	uuuu uuuu	uuuu uuuu
FSR	xxxx xxxx	uuuu uuuu	uuuu uuuu
GPIO	--xx xxxx	--uu uuuu	--uu uuuu
I2CADD	0000 0000	0000 0000	uuuu uuuu
I2CBUF	xxxx xxxx	uuuu uuuu	uuuu uuuu
I2CCON	0000 0000	0000 0000	uuuu uuuu
I2CSTAT	--00 0000	--00 0000	--uu uuuu
INDF	-	-	-
INTCON	0000 000x	0000 000u	uuuu uuuu(1)
LCDCON	00-0 0000	00-0 0000	uu-u uuuu
LCDD00 to LCDD15	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDPS	---- 0000	---- 0000	---- uuuu
LCDSE	1111 1111	1111 1111	uuuu uuuu
OPTION_REG	1111 1111	1111 1111	uuuu uuuu
OSCCAL	0111 00--	uuuu uu--	uuuu uu--
PCL	0000 0000	0000 0000	PC + 1(2)
PCLATH	---0 0000	---0 0000	---u uuuu
PCON	---- --0u	---- --uu	---- --uu
PIE1	0000 0000	0000 0000	uuuu uuuu
PIE2	---- ---0	---- ---0	---- ---u
PIR1	0000 0000	0000 0000	uuuu uuuu

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

Note 1: One or more bits in INTCON and/or PIR1 will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

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

Table 3-4: Initialization Conditions for Special Function Registers (Cont.'d)

Register	Power-on Reset Brown-out Reset	$\overline{\text{MCLR}}$ Reset during: - normal operation - SLEEP or WDT Reset	Wake-up from SLEEP through: - interrupt - WDT time-out
PIR2	---- ---0	---- ---0	---- ---u
PORTA	--xx xxxx	--uu uuuu	--uu uuuu
PORTB	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTD	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTE	---- -xxx	---- -uuu	---- -uuu
PORTF	0000 0000	0000 0000	uuuu uuuu
PORTG	0000 0000	0000 0000	uuuu uuuu
PR2	1111 1111	1111 1111	1111 1111
PREFA	0000 0000	0000 0000	uuuu uuuu
PREFB	0000 0000	0000 0000	uuuu uuuu
RCSTA	0000 -00x	0000 -00x	uuuu -uuu
RCREG	0000 0000	0000 0000	uuuu uuuu
SLPCON	0011 1111	0011 1111	uuuu uuuu
SPBRG	0000 0000	0000 0000	uuuu uuuu
SSPBUF	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPCON	0000 0000	0000 0000	uuuu uuuu
SSPADD	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	0000 0000	0000 0000	uuuu uuuu
STATUS	0001 1xxx	000q quuu(3)	uuuq quuu(3)
T1CON	--00 0000	--uu uuuu	--uu uuuu
T2CON	-000 0000	-000 0000	-uuu uuuu
TMR0	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1H	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR2	0000 0000	0000 0000	uuuu uuuu
TRIS	--11 1111	--11 1111	--uu uuuu
TRISA	--11 1111	--11 1111	--uu uuuu
TRISB	1111 1111	1111 1111	uuuu uuuu
TRISC	1111 1111	1111 1111	uuuu uuuu
TRISD	1111 1111	1111 1111	uuuu uuuu
TRISE	0000 -111	0000 -111	uuuu -uuu
TRISF	1111 1111	1111 1111	uuuu uuuu
TRISG	1111 1111	1111 1111	uuuu uuuu
TXREG	0000 0000	0000 0000	uuuu uuuu
TXSTA	0000 -010	0000 -010	uuuu -uuu
VRCON	000- 0000	000- 0000	uuu- uuuu
W	xxxx xxxx	uuuu uuuu	uuuu uuuu

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

Note 1: One or more bits in INTCON and/or PIR1 will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIE bit is set, the PC is loaded with the interrupt vector (0004h).

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

3.3.1 Power Control (PCON) and STATUS Registers

The Power Control (PCON) register contains a status bit to allow differentiation between a Power-on Reset (POR) to an external $\overline{MCLR}$ Reset or WDT Reset. It also contains a status bit to determine if a Brown-out Reset ($\overline{BOR}$) occurred. The power control/status register, PCON has up to four bits.

The $\overline{BOR}$ (Brown-out Reset) bit, is unknown on a Power-on-reset. It must initially be set by the user and checked on subsequent resets to see if $\overline{BOR}$ = '0' indicating that a Brown-out Reset has occurred. The $\overline{BOR}$ status bit is a “don’t care” bit and is not necessarily predictable if the brown-out circuit is disabled (by clearing the BODEN bit in the Configuration word).

The $\overline{POR}$ (Power-on Reset) bit, is cleared on a Power-on Reset and is unaffected otherwise. The user sets this bit following a Power-on Reset. On subsequent resets if $\overline{POR}$ is ‘0’, it will indicate that a Power-on Reset must have occurred.

The $\overline{PER}$ (Parity Error Reset) bit, is cleared on a Parity Error Reset and must be set by user software. It will also be set on a Power-on Reset.

The MPEEN (Memory Parity Error Enable) bit, reflects the status of the MPEEN bit in configuration word. It is unaffected by any reset or interrupt.

Note:	$\overline{BOR}$ is unknown on Power-on Reset. It must then be set by the user and checked on subsequent resets to see if $\overline{BOR}$ is clear, indicating a brown-out has occurred. The $\overline{BOR}$ status bit is a don't care and is not necessarily predictable if the brown-out circuit is disabled (by clearing the BODEN bit in the Configuration word).

Register 3-1: PCON Register

R-u	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
MPEEN	—	—	—	—	$\overline{PER}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

bit 7 **MPEEN**: Memory Parity Error Circuitry Status bit
This bit reflects the value of the MPEEN configuration bit.

bit 6:3 **Unimplemented:** Read as '0'

bit 2 **$\overline{PER}$**: Memory Parity Error Reset Status bit
1 = No parity error reset occurred
0 = A program memory fetch parity error occurred
(must be set in software after a Power-on Reset or Parity Error Reset occurs)

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

bit 0 **$\overline{BOR}$**: Brown-out Reset Status bit
1 = No Brown-out Reset occurred
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset or Power-on Reset occurs)

Legend		
R = Readable bit	W = Writable bit	u = unchanged bit
U = Unimplemented bit, read as ‘0’		- n = Value at POR reset

Note:	Not all bits may be implemented.

The STATUS register contains two bits ($\overline{TO}$ and $\overline{PD}$), which when used in conjunction with the PCON register bits provide the user with enough information to determine the cause of the reset.

Register 3-2: STATUS Register

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7 **IRP:** Register Bank Select bit (used for indirect addressing)
1 = Bank 2, 3 (100h - 1FFh)
0 = Bank 0, 1 (00h - FFh)

For devices with only Bank0 and Bank1 the IRP bit is reserved, always maintain this bit clear.

bit 6:5 **RP1:RP0**: Register Bank Select bits (used for direct addressing)
11 = Bank 3 (180h - 1FFh)
10 = Bank 2 (100h - 17Fh)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)

Each bank is 128 bytes. For devices with only Bank0 and Bank1 the IRP bit is reserved, always maintain this bit clear.

bit 4 **$\overline{TO}$**: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3 **$\overline{PD}$**: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit2 **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/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{borrow}$ the polarity is reversed)
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

bit 0 **C**: Carry/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)
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 occurred

Note: For $\overline{borrow}$ the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

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

3.4 Design Tips

Question 1: ***When my system is subjected to an environment with ESD and EMI, it operates erratically.***

Answer 1:

If the device you are using does not have filtering to the on-chip master clear circuit (Appendix C), ensure that proper external filtering is placed on the $\overline{\text{MCLR}}$ pin to remove narrow pulses. Electrical Specification parameter 35 specifies the pulse width required to cause a reset.

Question 2: ***With JW (windowed) devices my system resets and operates properly. With an OTP device, my system does not operate properly.***

Answer 2:

The most common reason for this is that the windowed device (JW) has not had its window covered. The background light causes the device to power-up in a different state than would typically be seen in a device where no light is present. In most cases all the General Purpose RAM and Special Function Registers were not initialized properly.

3.5 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to Resets are:

Title	Application Note #
Power-up Trouble Shooting	AN607
Power-up Considerations	AN522

3.6 Revision History

Revision A

This is the initial released revision of the Reset description.

Section 4. Architecture

HIGHLIGHTS

This section of the manual contains the following major topics:

4.1 Introduction

The high performance of the PICmicro™ devices can be attributed to a number of architectural features commonly found in RISC microprocessors. These include:

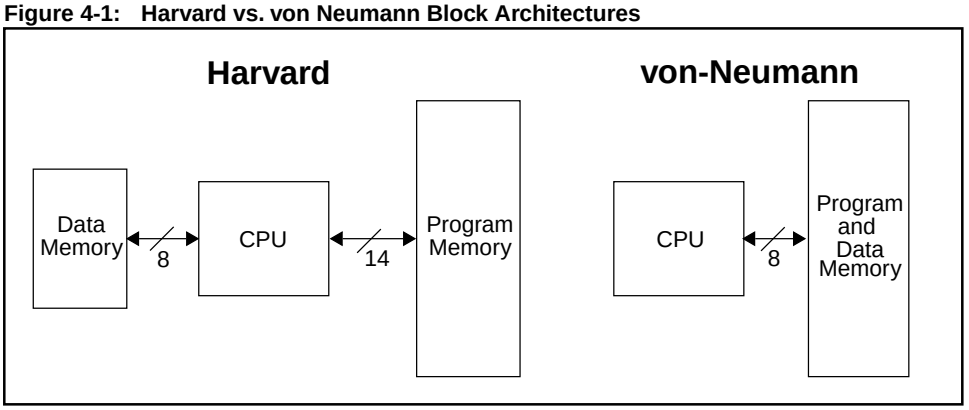
- Harvard architecture
- Long Word Instructions
- Single Word Instructions
- Single Cycle Instructions
- Instruction Pipelining
- Reduced Instruction Set
- Register File Architecture
- Orthogonal (Symmetric) Instructions

Figure 4-2 shows a simple core memory bus arrangement for Mid-Range MCU devices.

Harvard Architecture:

Harvard architecture has the program memory and data memory as separate memories and are accessed from separate buses. This improves bandwidth over traditional von Neumann architecture in which program and data are fetched from the same memory using the same bus. To execute an instruction, a von Neumann machine must make one or more (generally more) accesses across the 8-bit bus to fetch the instruction. Then data may need to be fetched, operated on, and possibly written. As can be seen from this description, that bus can be extremely conjested. While with a Harvard architecture, the instruction is fetched in a single instruction cycle (all 14-bits). While the program memory is being accessed, the data memory is on an independent bus and can be read and written. These separated buses allow one instruction to execute while the next instruction is fetched. A comparison of Harvard vs. von-Neumann architectures is shown in Figure 4-1.

Figure 4-1: Harvard vs. von Neumann Block Architectures



Long Word Instructions:

Long word instructions have a wider (more bits) instruction bus than the 8-bit Data Memory Bus. This is possible because the two buses are separate. This further allows instructions to be sized differently than the 8-bit wide data word which allows a more efficient use of the program memory, since the program memory width is optimized to the architectural requirements.

Single Word Instructions:

Single Word instruction opcodes are 14-bits wide making it possible to have all single word instructions. A 14-bit wide program memory access bus fetches a 14-bit instruction in a single cycle. With single word instructions, the number of words of program memory locations equals the number of instructions for the device. This means that all locations are valid instructions.

Typically in the von Neumann architecture, most instructions are multi-byte. In general, a device with 4-KBytes of program memory would allow approximately 2K of instructions. This 2:1 ratio is generalized and dependent on the application code. Since each instruction may take multiple bytes, there is no assurance that each location is a valid instruction.

Instruction Pipeline:

The instruction pipeline is a two-stage pipeline which overlaps the fetch and execution of instructions. The fetch of the instruction takes one TCY, while the execution takes another TCY. However, due to the overlap of the fetch of current instruction and execution of previous instruction, an instruction is fetched and another instruction is executed every single TCY.

Single Cycle Instructions:

With the Program Memory bus being 14-bits wide, the entire instruction is fetched in a single machine cycle (TCY). The instruction contains all the information required and is executed in a single cycle. There may be a one cycle delay in execution if the result of the instruction modified the contents of the Program Counter. This requires the pipeline to be flushed and a new instruction to be fetched.

Reduced Instruction Set:

When an instruction set is well designed and highly orthogonal (symmetric), fewer instructions are required to perform all needed tasks. With fewer instructions, the whole set can be more rapidly learned.

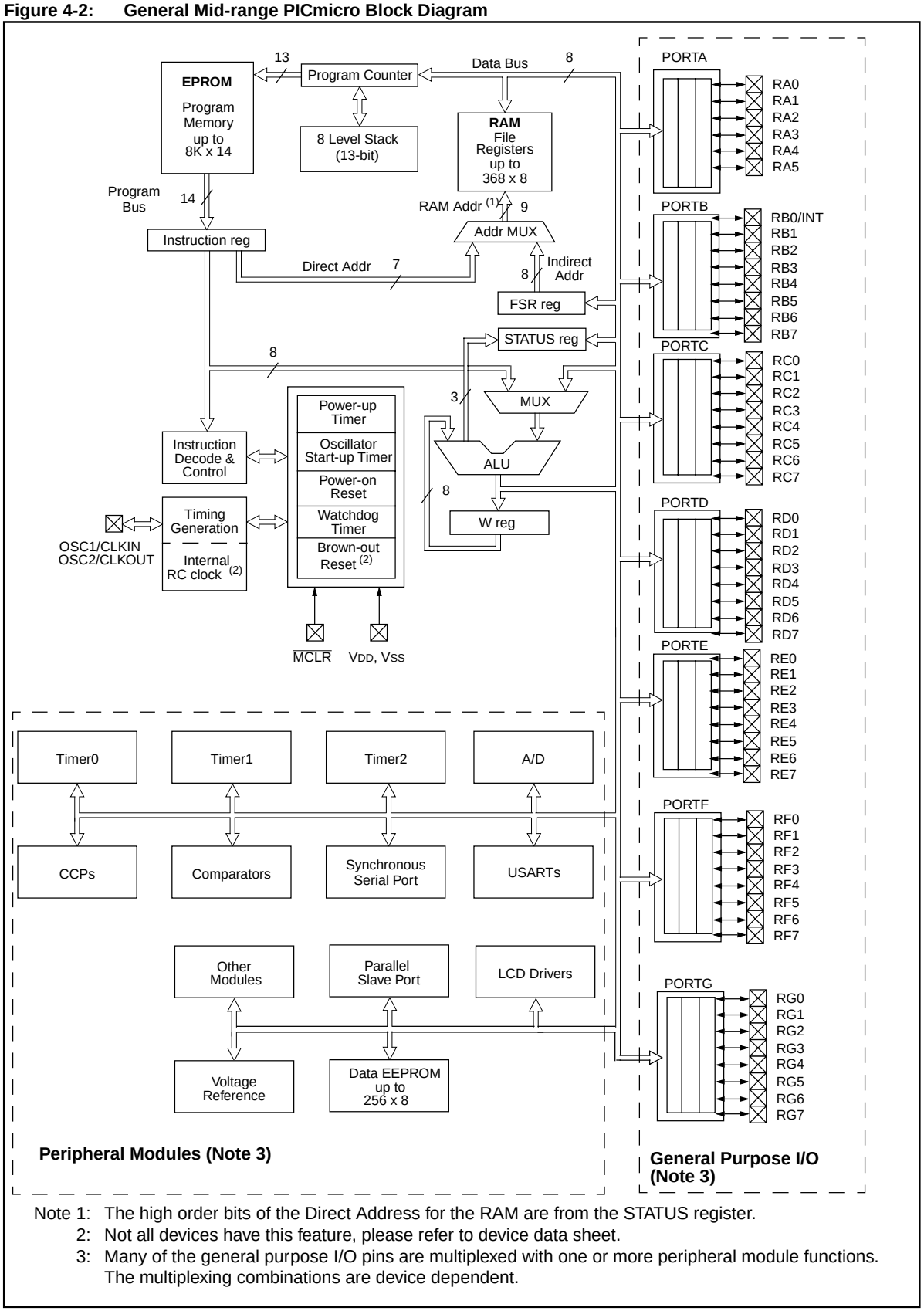
Register File Architecture:

The register files/data memory can be directly or indirectly addressed. All special function registers, including the program counter, are mapped in the data memory.

Orthogonal (Symmetric) Instructions:

Orthogonal instructions make it possible to carry out any operation on any register using any addressing mode. This symmetrical nature and lack of “special instructions” make programming simple yet efficient. In addition, the learning curve is reduced significantly. The mid-range instruction set uses only two non-register oriented instructions, which are used for two of the cores features. One is the SLEEP instruction which places the device into the lowest power use mode. The other is the CLRWDT instruction which verifies the chip is operating properly by preventing the on-chip Watchdog Timer (WDT) from overflowing and resetting the device.

Figure 4-2: General Mid-range PICmicro Block Diagram

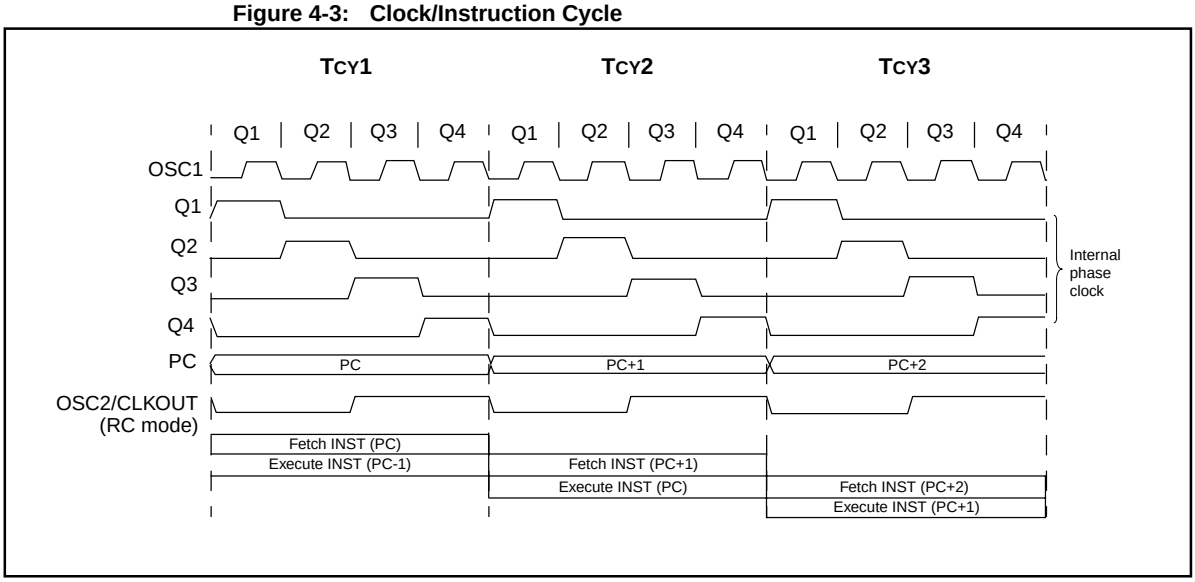


Note 1: The high order bits of the Direct Address for the RAM are from the STATUS register.
2: Not all devices have this feature, please refer to device data sheet.
3: Many of the general purpose I/O pins are multiplexed with one or more peripheral module functions. The multiplexing combinations are device dependent.

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 illustrated in Figure 4-3, and Example 4-1.

Figure 4-3: Clock/Instruction Cycle



4.3 Instruction Flow/Pipelining

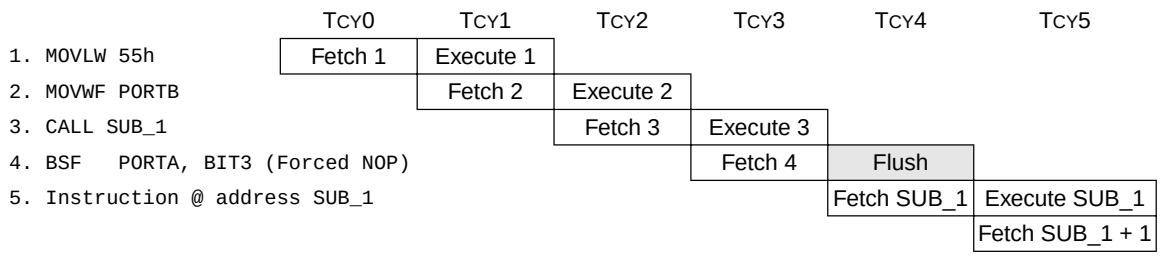
An "Instruction Cycle" consists of four Q cycles (Q1, Q2, Q3, and Q4). Fetch takes one instruction cycle while decode and execute takes another instruction cycle. However, due to Pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g. GOTO) then an extra cycle is required to complete the instruction (Example 4-1).

The instruction **fetch** 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).

Example 4-1 shows the operation of the two stage pipeline for the instruction sequence shown. At time TCY0, the first instruction is fetched from program memory. During TCY1, the first instruction executes while the second instruction is fetched. During TCY2, the second instruction executes while the third instruction is fetched. During TCY3, the fourth instruction is fetched while the third instruction (CALL SUB_1) is executed. When the third instruction completes execution, the CPU forces the address of instruction four onto the Stack and then changes the Program Counter (PC) to the address of SUB_1. This means that the instruction that was fetched during TCY3 needs to be "flushed" from the pipeline. During TCY4, instruction four is flushed (executed as a NOP) and the instruction at address SUB_1 is fetched. Finally during TCY5, instruction five is executed and the instruction at address SUB_1 + 1 is fetched.

Example 4-1: Instruction Pipeline Flow



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

4.4 I/O Descriptions

Table 4-1 gives a brief description of the functions that may be multiplexed to a port pin. Multiple functions may exist on one port pin. When multiplexing occurs, the peripheral module's functional requirements may force an override of the data direction (TRIS bit) of the port pin (such as in the A/D and LCD modules).

Table 4-1: I/O Descriptions

Pin Name	Pin Type	Buffer Type	Description
			Analog Input Channels
AN0	I	Analog	
AN1	I	Analog	
AN2	I	Analog	
AN3	I	Analog	
AN4	I	Analog	
AN5	I	Analog	
AN6	I	Analog	
AN7	I	Analog	
AN8	I	Analog	
AN9	I	Analog	
AN10	I	Analog	
AN11	I	Analog	
AN12	I	Analog	
AN13	I	Analog	
AN14	I	Analog	
AN15	I	Analog	
AVDD	P	P	Analog Power
AVSS	P	P	Analog Ground
C1	I	Analog	LCD Voltage Generation
C2	I	Analog	LCD Voltage Generation
CCP1	I/O	ST	Capture1 input/Compare1 output/PWM1 output
CCP2	I/O	ST	Capture2 input/Compare2 output/PWM2 output.
CDAC	O	Analog	A/D ramp current source output. Normally connected to external capacitor to generate a linear voltage ramp.
CK	I/O	ST	USART Synchronous Clock, always associated with TX pin function (See related TX, RX, DT)
CLKIN	I	ST/CMOS	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKIN, OSC2/CLKOUT pins)
CLKOUT	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate. Always associated with OSC2 pin function. (See related OSC2, OSC1)
CMPA	O	—	Comparator A output
CMPB	O	—	Comparator B output

Legend: TTL = TTL-compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
SM = SMBus compatible input. An external resistor is required if this pin is used as an output
NPU = N-channel pull-up
PU = Weak internal pull-up
No-P diode = No P-diode to VDD
AN = Analog input or output
I = input
O = output
P = Power
L = LCD Driver

Table 4-1: I/O Descriptions (Cont.'d)

Pin Name	Pin Type	Buffer Type	Description
COM0	L	—	LCD Common Driver0
COM1	L	—	LCD Common Driver1
COM2	L	—	LCD Common Driver2
COM3	L	—	LCD Common Driver3
$\overline{CS}$	I	TTL	chip select control for parallel slave port (See related $\overline{RD}$ and $\overline{WR}$)
DT	I/O	ST	USART Synchronous Data. Always associated RX pin function. (See related RX, TX, CK)
			GP is a bi-directional I/O port. Some pins of port GP can be software programmed for internal weak pull-ups on the inputs.
GP0	I/O	TTL/ST	TTL input buffer as general purpose I/O, Schmitt Trigger input buffer when used as the serial programming mode.
GP1	I/O	TTL/ST	TTL input buffer as general purpose I/O, Schmitt Trigger input buffer when used as the serial programming mode.
GP2	I/O	ST	
GP3	I	TTL	
GP4	I/O	TTL	
GP5	I/O	TTL	
INT	I	ST	External Interrupt
$\overline{MCLR}$/VPP	I/P	ST	Master clear (reset) input or programming voltage input. This pin is an active low reset to the device.
NC	—	—	These pins should be left unconnected.
OSC1	I	ST/CMOS	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode. CMOS otherwise.
OSC2	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
PBTN	I	ST	Input with weak pull-up resistor, can be used to generate an interrupt.
PSP0	I/O	TTL	Parallel Slave Port for interfacing to a microprocessor port. These pins have TTL input buffers when PSP module is enabled.
PSP1	I/O	TTL	
PSP2	I/O	TTL	
PSP3	I/O	TTL	
PSP4	I/O	TTL	
PSP5	I/O	TTL	
PSP6	I/O	TTL	
PSP7	I/O	TTL	
			PORTA is a bi-directional I/O port.
RA0	I/O	TTL	
RA1	I/O	TTL	
RA2	I/O	TTL	
RA3	I/O	TTL	
RA4	I/O	ST	RA4 is an open drain when configured as output.
RA5	I/O	TTL	

Legend: TTL = TTL-compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
SM = SMBus compatible input. An external resistor is required if this pin is used as an output
NPU = N-channel pull-up
PU = Weak internal pull-up
No-P diode = No P-diode to VDD
AN = Analog input or output
I = input
O = output
P = Power
L = LCD Driver

Table 4-1: I/O Descriptions (Cont.'d)

Pin Name	Pin Type	Buffer Type	Description
			PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0	I/O	TTL	
RB1	I/O	TTL	
RB2	I/O	TTL	
RB3	I/O	TTL	
RB4	I/O	TTL	Interrupt on change pin.
RB5	I/O	TTL	Interrupt on change pin.
RB6	I/O	TTL/ST	Interrupt on change pin. Serial programming clock. TTL input buffer as general purpose I/O, Schmitt Trigger input buffer when used as the serial programming clock.
RB7	I/O	TTL/ST	Interrupt on change pin. Serial programming data. TTL input buffer as general purpose I/O, Schmitt Trigger input buffer when used as the serial programming data.
			PORTC is a bi-directional I/O port.
RC0	I/O	ST	
RC1	I/O	ST	
RC2	I/O	ST	
RC3	I/O	ST	
RC4	I/O	ST	
RC5	I/O	ST	
RC6	I/O	ST	
RC7	I/O	ST	
$\overline{RD}$	I	TTL	Read control for parallel slave port (See also $\overline{WR}$ and $\overline{CS}$ pins)
			PORTD is a bi-directional I/O port.
RD0	I/O	ST	
RD1	I/O	ST	
RD2	I/O	ST	
RD3	I/O	ST	
RD4	I/O	ST	
RD5	I/O	ST	
RD6	I/O	ST	
RD7	I/O	ST	
			PORTE is a bi-directional I/O port.
RE0	I/O	ST	
RE1	I/O	ST	
RE2	I/O	ST	
RE3	I/O	ST	
RE4	I/O	ST	
RE5	I/O	ST	
RE6	I/O	ST	
RE7	I/O	ST	

Legend: TTL = TTL-compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
SM = SMBus compatible input. An external resistor is required if this pin is used as an output
NPU = N-channel pull-up
PU = Weak internal pull-up
No-P diode = No P-diode to V_{DD}
AN = Analog input or output
I = input
O = output
P = Power
L = LCD Driver

Table 4-1: I/O Descriptions (Cont.'d)

Pin Name	Pin Type	Buffer Type	Description
REFA	O	CMOS	Programmable reference A output.
REFB	O	CMOS	Programmable reference B output.
			PORTF is a digital input or LCD Segment Driver Port
RF0	I/O	ST	
RF1	I/O	ST	
RF2	I/O	ST	
RF3	I/O	ST	
RF4	I/O	ST	
RF5	I/O	ST	
RF6	I/O	ST	
RF7	I/O	ST	
			PORTG is a digital input or LCD Segment Driver Port
RG0	I/O	ST	
RG1	I/O	ST	
RG2	I/O	ST	
RG3	I/O	ST	
RG4	I/O	ST	
RG5	I/O	ST	
RG6	I/O	ST	
RG7	I/O	ST	
RX	I	ST	USART Asynchronous Receive
SCL	I/O	ST	Synchronous serial clock input/output for I^2C mode.
SCLA	I/O	ST	Synchronous serial clock for I^2C interface.
SCLB	I/O	ST	Synchronous serial clock for I^2C interface.
SDA	I/O	ST	I^2C™ Data I/O
SDAA	I/O	ST	Synchronous serial data I/O for I^2C interface
SDAB	I/O	ST	Synchronous serial data I/O for I^2C interface
SCK	I/O	ST	Synchronous serial clock input/output for SPI mode.
SDI	I	ST	SPI Data In
SDO	O	—	SPI Data Out (SPI mode)
$\overline{SS}$	I	ST	SPI Slave Select input
SEG00 to SEG31	I/L	ST	LCD Segment Driver00 through Driver31.
SUM	O	AN	AN1 summing junction output. This pin can be connected to an external capacitor for averaging small duration pulses.
T0CKI	I	ST	Timer0 external clock input
T1CKI	I	ST	Timer1 external clock input
T1OSO	O	CMOS	Timer1 oscillator output
T1OSI	I	CMOS	Timer1 oscillator input
TX	O	—	USART Asynchronous Transmit (See related RX)

Legend: TTL = TTL-compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
SM = SMBus compatible input. An external resistor is required if this pin is used as an output
NPU = N-channel pull-up
PU = Weak internal pull-up
No-P diode = No P-diode to V_{DD}
AN = Analog input or output
I = input
O = output
P = Power
L = LCD Driver

I^2C is a trademark of Philips Corporation.

Table 4-1: I/O Descriptions (Cont.'d)

Pin Name	Pin Type	Buffer Type	Description
VLCD1	P	—	LCD Voltage
VLCD2	P	—	LCD Voltage
VLCD3	P	—	LCD Voltage
VLCDADJ	I	Analog	LCD Voltage Generation
VREF	I	Analog	Analog High Voltage Reference input. DR reference voltage output on devices with comparators.
VREF+	I	Analog	Analog High Voltage Reference input. Usually multiplexed onto an analog pin.
VREF-	I	Analog	Analog Low Voltage Reference input. Usually multiplexed onto an analog pin.
VREG	O	—	This pin is an output to control the gate of an external N-FET for voltage regulation.
VSS	P	—	Ground reference for logic and I/O pins.
VDD	P	—	Positive supply for logic and I/O pins.
$\overline{WR}$	I	TTL	Write control for parallel slave port (See $\overline{CS}$ and $\overline{RD}$ pins also).

Legend: TTL = TTL-compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
SM = SMBus compatible input. An external resistor is required if this pin is used as an output
NPU = N-channel pull-up
PU = Weak internal pull-up
No-P diode = No P-diode to VDD
AN = Analog input or output
I = input
O = output
P = Power
L = LCD Driver

4.5 Design Tips

No related design tips at this time.

4.6 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to Architecture are:

Title	Application Note #
No related application notes at this time.	

4.7 Revision History

Revision A

This is the initial released revision of the PICmicro's Architecture description.

Section 5. CPU and ALU

HIGHLIGHTS

This section of the manual contains the following major topics:

5.1 Introduction

The Central Processing Unit (CPU) is responsible for using the information in the program memory (instructions) to control the operation of the device. Many of these instructions operate on data memory. To operate on data memory, the Arithmetic Logical Unit (ALU) is required. In addition to performing arithmetical and logical operations, the ALU controls status bits (which are found in the STATUS register). The result of some instructions force status bits to a value depending on the state of the result.

The machine codes that the CPU recognizes are show in Table 5-1 (as well as the instruction mnemonics that the MPASM uses to generate these codes).

Table 5-1: Mid-Range MCU Instruction Set

Mnemonic, Operands		Description	Cycles	14-Bit Instruction Word				Status Bits Affected	Notes
				MSb			LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff-	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

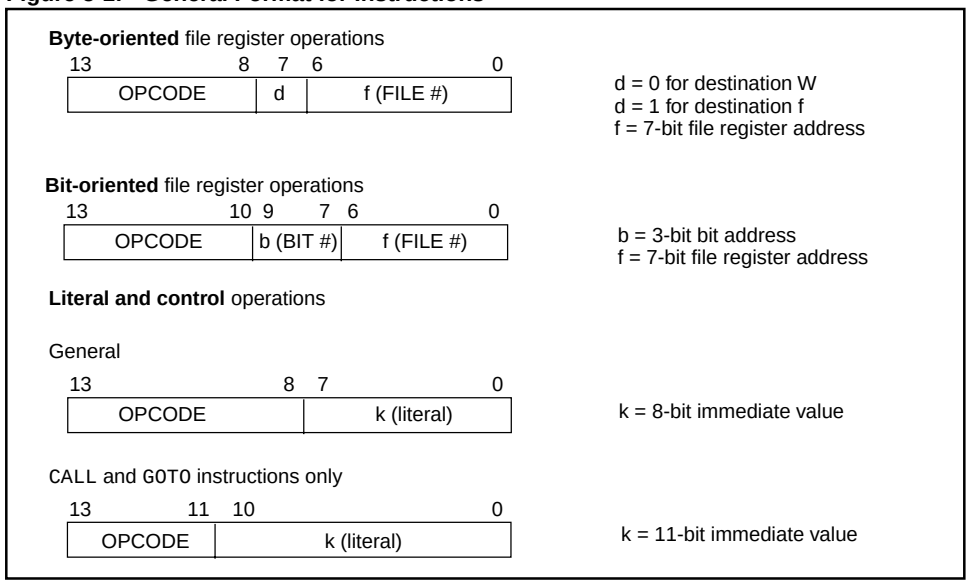
2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

5.2 General Instruction Format

The Mid-Range MCU instructions can be broken down into four general formats as shown in Figure 5-1. As can be seen the opcode for the instruction varies from 3-bits to 6-bits. This variable opcode size is what allows 35 instructions to be implemented.

Figure 5-1: General Format for Instructions



5.3 Central Processing Unit (CPU)

The CPU can be thought of as the “brains” of the device. It is responsible for fetching the correct instruction for execution, decoding that instruction, and then executing that instruction.

The CPU sometimes works in conjunction with the ALU to complete the execution of the instruction (in arithmetic and logical operations).

The CPU controls the program memory address bus, the data memory address bus, and accesses to the stack.

5.4 Instruction Clock

Each instruction cycle (TCY) is comprised of four Q cycles (Q1-Q4). The Q cycle time is the same as the device oscillator cycle time (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

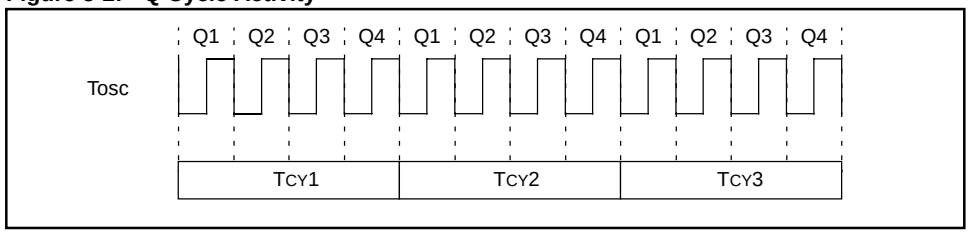
Q2: Instruction Read Data Cycle or No operation

Q3: Process the Data

Q4: Instruction Write Data Cycle or No operation

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

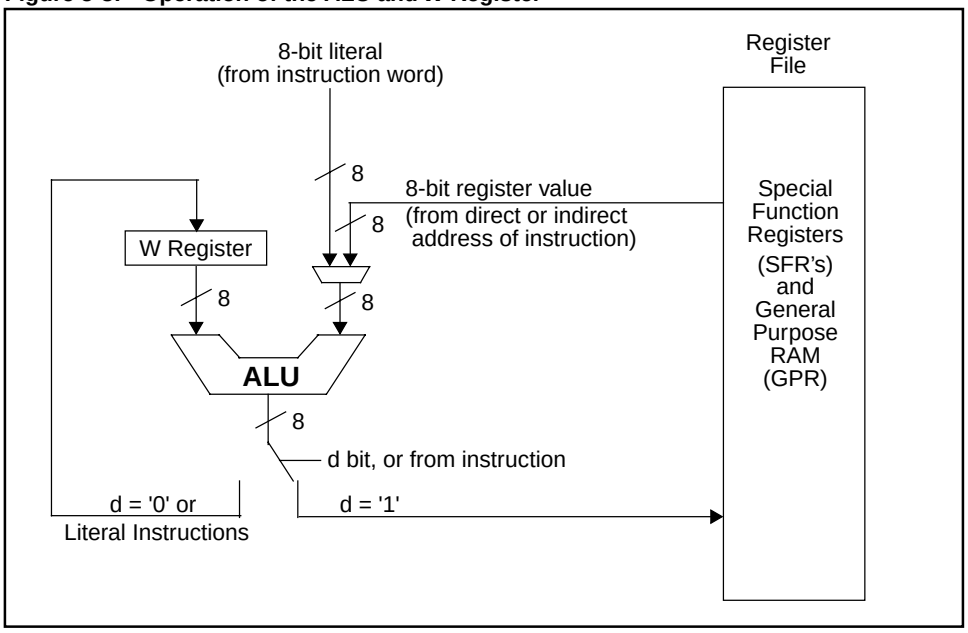
Figure 5-2: Q Cycle Activity



5.5 Arithmetic Logical Unit (ALU)

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

Figure 5-3: Operation of the ALU and W Register



The ALU is 8-bits wide and is capable of addition, subtraction, shift and logical operations. Unless otherwise mentioned, arithmetic operations are two's complement in nature. In two-operand instructions, typically one operand is the working register (W register). The other operand is a file register or an immediate constant. In single operand instructions, the operand is either the W register or a file register.

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

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

5.6 STATUS Register

The STATUS register, shown in Figure 5-1, contains the arithmetic status of the ALU, the RESET status and the bank select bits for data memory. Since the selection of the Data Memory banks is controlled by this register, it is required to be present in every bank. Also, this register is in the same relative position (offset) in each bank (see **Figure 6-5: "Register File Map"** in the **"Memory Organization"** section).

The STATUS register can be the destination for any instruction, as with any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper-three bits and set the Z bit. This leaves the STATUS register as `000u u1uu` (where u = unchanged).

It is recommended, therefore, that only `BCF, BSF, SWAPF` and `MOVWF` instructions are used to alter the STATUS register because these instructions do not affect the Z, C or DC bits from the STATUS register. For other instructions, not affecting any status bits, see Table 5-1.

Note 1: Some devices do not require the IRP and RP1 (STATUS<7:6>) bits. These bits are not used by the Section 5. CPU and ALU and should be maintained clear. Use of these bits as general purpose R/W bits is NOT recommended, since this may affect upward code compatibility with future products.

Note 2: The C and DC bits operate as a $\overline{\text{borrow}}$ and $\overline{\text{digit borrow}}$ bit, respectively, in subtraction.

Register 5-1: STATUS Register

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7 **IRP:** Register Bank Select bit (used for indirect addressing)
1 = Bank 2, 3 (100h - 1FFh)
0 = Bank 0, 1 (00h - FFh)

For devices with only Bank0 and Bank1 the IRP bit is reserved, always maintain this bit clear.

bit 6:5 **RP1:RP0**: Register Bank Select bits (used for direct addressing)
11 = Bank 3 (180h - 1FFh)
10 = Bank 2 (100h - 17Fh)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)

Each bank is 128 bytes. For devices with only Bank0 and Bank1 the IRP bit is reserved, always maintain this bit clear.

bit 4 **$\overline{TO}$**: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3 **$\overline{PD}$**: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit2 **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/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{borrow}$ the polarity is reversed)
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

bit 0 **C**: Carry/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)
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 occurred

Note: For $\overline{borrow}$ the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

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

5.7 OPTION_REG Register

The OPTION_REG register is a readable and writable register which contains various control bits to configure the TMR0/WDT prescaler, the external INT Interrupt, TMR0, and the weak pull-ups on PORTB.

Register 5-2: OPTION_REG Register

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of INT pin
0 = Interrupt on falling edge of INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

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

Note: To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

5.8 PCON Register

The Power Control (PCON) register contains flag bit(s), that together with the TO and PD bits, allows the user to differentiate between the device resets.

Note 1:	$\overline{BOR}$ is unknown on Power-on Reset. It must then be set by the user and checked on subsequent resets to see if $\overline{BOR}$ is clear, indicating a brown-out has occurred. The $\overline{BOR}$ status bit is a don't care and is not necessarily predictable if the brown-out circuit is disabled (by clearing the BODEN bit in the Configuration word).
Note 2:	It is recommended that the $\overline{POR}$ bit be cleared after a power-on reset has been detected, so that subsequent power-on resets may be detected.

Register 5-3: PCON Register

R-u	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
MPEEN	—	—	—	—	$\overline{PER}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

bit 7 **MPEEN**: Memory Parity Error Circuitry Status bit
This bit reflects the value of the MPEEN configuration bit.

bit 6:3 **Unimplemented:** Read as '0'

bit 2 **$\overline{PER}$**: Memory Parity Error Reset Status bit
1 = No error occurred
0 = A program memory fetch parity error occurred
(must be set in software after a Power-on Reset occurs)

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

bit 0 **$\overline{BOR}$**: Brown-out Reset Status bit
1 = No Brown-out Reset occurred
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

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

5.9 Design Tips

Question 1: ***My program algorithm does not seem to function correctly.***

Answer 1:

1. The destination of the instruction may be specifying the W register (d = 0) instead of the file register (d = 1).
2. The register bank select bits (RP1:RP0 or IRP) may not be properly selected. Also if interrupts are used, the register bank select bits may not be properly restored when exiting the interrupt handler.

Question 2: ***I cannot seem to modify the STATUS register flags.***

Answer 2:

if the STATUS register is the destination for an instruction that affects the Z, DC, or C bits, the write to these bits is disabled. These bits are set or cleared based on device logic. Therefore, to modify bits in the STATUS register it is recommended to use the BCF and BSF instructions.

5.10 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the CPU or the ALU are:

Title	Application Note #
Fixed Point Routines	AN617
IEEE 754 Compliant Floating Point Routines	AN575
Digital Signal Processing with the PIC16C74	AN616
Math Utility Routines	AN544
Implementing IIR Digital Filters	AN540
Implementation of Fast Fourier Transforms	AN542
Tone Generation	AN543
Servo Control of a DC Brushless Motor	AN532
Implementation of the Data Encryption Standard using the PIC17C42	AN583
PIC16C5X / PIC16CXX Utility Math Routines	AN526
Real Time Operating System for PIC16/17	AN585

5.11 Revision History

Revision A

This is the initial released revision of the CPU and ALU description.

Section 6. Memory Organization

HIGHLIGHTS

This section of the manual contains the following major topics:

6.1 Introduction

There are two memory blocks in the Section 6. Memory Organization; program memory and data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle.

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

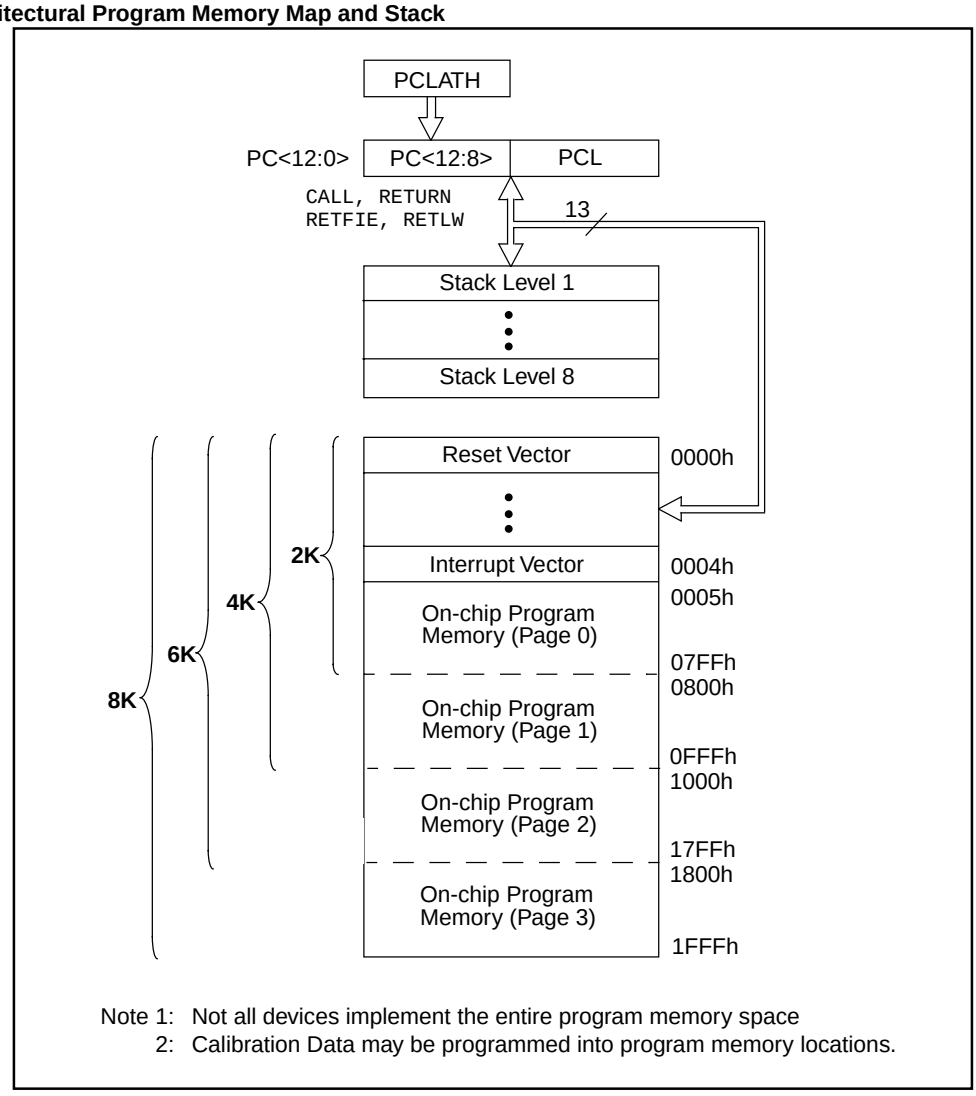
6.2 Program Memory Organization

Mid-Range MCU devices have a 13-bit program counter capable of addressing an 8K x 14 program memory space. The width of the program memory bus (instruction word) is 14-bits. Since all instructions are a single word, a device with an 8K x 14 program memory has space for 8K of instructions. This makes it much easier to determine if a device has sufficient program memory for a desired application.

This program memory space is divided into four pages of 2K words each (0h - 7FFh, 800h - FFFh, 1000h - 17FFh, and 1800h - 1FFFh). Figure 6-1 shows the program memory map as well as the 8 level deep hardware stack. Depending on the device, only a portion of this memory may be implemented. Please refer to the device data sheet for the available memory.

To jump between the program memory pages, the high bits of the Program Counter (PC) must be modified. This is done by writing the desired value into a SFR called PCLATH (**P**rogram **C**ounter **Lat**ch **H**igh). If sequential instructions are executed, the program counter will cross the page boundaries without any user intervention. For devices that have less than 8K words, accessing a location above the physically implemented address will cause a wraparound. That is, in a 4K-word device accessing 17FFh actually addresses 7FFh. 2K-word devices (or less) do not require paging.

Figure 6-1: Architectural Program Memory Map and Stack



Note 1: Not all devices implement the entire program memory space
2: Calibration Data may be programmed into program memory locations.

6.2.1 Reset Vector

On any device, a reset forces the Program Counter (PC) to address 0h. We call this address the "Reset Vector Address" since this is the address that program execution will branch to when a device reset occurs.

Any reset will also clear the contents of the PCLATH register. This means that any branch at the Reset Vector Address (0h) will jump to that location in PAGE0 of the program memory.

6.2.2 Interrupt Vector

When an interrupt is acknowledged the PC is forced to address 0004h. We call this the "Interrupt Vector Address". When the PC is forced to the interrupt vector, the PCLATH register is not modified. Once in the service interrupt routine (ISR), this means that before any write to the PC, the PCLATH register should be written with the value that will specify the desired location in program memory. Before the PCLATH register is modified by the Interrupt Service Routine (ISR) the contents of the PCLATH may need to be saved, so it can be restored before returning from the ISR.

6.2.3 Calibration Information

Some devices have calibration information stored in their program memory. This information is programmed by Microchip when the device is under final test. The use of these values allows the application to achieve better results. The calibration information is typically at the end of program memory, and is implemented as a `RETLW` instruction with the literal value being the specified calibration information.

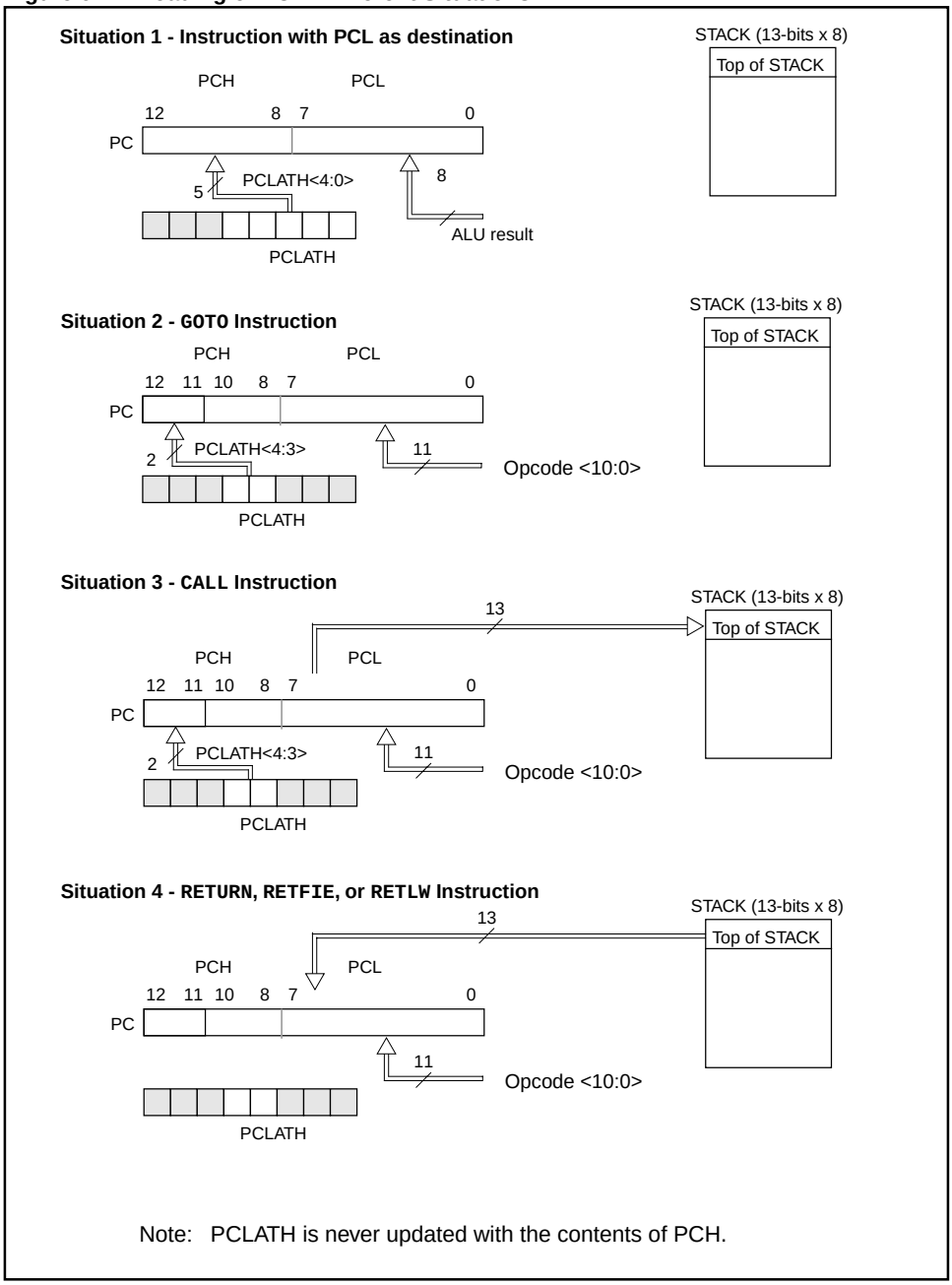
Note:	For windowed devices, write down all calibration values **BEFORE** erasing. This allows the device's calibration values to be restored when the device is re-programmed. When possible writing the values on the package is recommended.

6.2.4 Program Counter (PC)

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 13-bits wide. The low byte is called the PCL register. This register is readable and writable. The high byte is called the PCH register. This register contains the PC<12:8> bits and is not directly readable or writable. All updates to the PCH register go through the PCLATH register.

Figure 6-2 shows the four situations for the loading of the PC. Situation 1 shows how the PC is loaded on a write to PCL (PCLATH<4:0> → PCH). Situation 2 shows how the PC is loaded during a GOTO instruction (PCLATH<4:3> → PCH). Situation 3 shows how the PC is loaded during a CALL instruction (PCLATH<4:3> → PCH), with the PC loaded (PUSHed) onto the Top of Stack. Situation 4 shows how the PC is loaded during one of the return instructions where the PC loaded (POPed) from the Top of Stack.

Figure 6-2: Loading of PC In Different Situations



Note: PCLATH is never updated with the contents of PCH.

6.2.4.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). When doing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256 byte block).

Note:	Any write to the Program Counter (PCL), will cause the lower five bits of the PCLATH to be loaded into PCH.

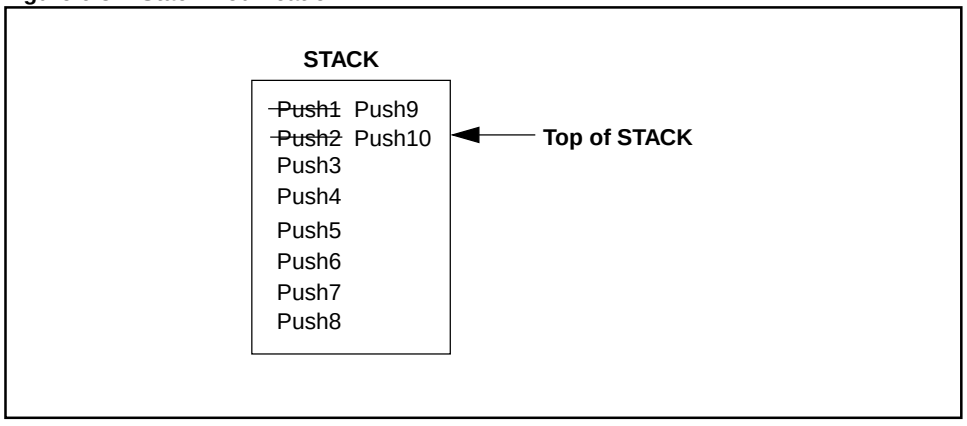
6.2.5 Stack

The stack allows a combination of up to 8 program calls and interrupts to occur. The stack contains the return address from this branch in program execution.

Mid-Range MCU devices have an 8-level deep x 13-bit wide hardware stack. The stack space is not part of either program or data space and the stack pointer is not readable or writable. The PC is PUSHed onto the stack when a CALL instruction is executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not modified when the stack is PUSHed or POPed.

After the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on). An example of the overwriting of the stack is shown in Figure 6-3.

Figure 6-3: Stack Modification



Note 1:	There are no status bits to indicate stack overflow or stack underflow conditions.
Note 2:	There are no instructions/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 address.

6.2.6 Program Memory Paging

Some devices have program memory sizes greater then 2K words, but the CALL and GOTO instructions only have a 11-bit address range. This 11-bit address range allows a branch within a 2K program memory page size. To allow CALL and GOTO instructions to address the entire 1K program memory address range, there must be another two bits to specify the program memory page. These paging bits come from the PCLATH<4:3> bits (Figure 6-2). When doing a CALL or GOTO instruction, the user must ensure that page bits (PCLATH<4:3>) are programmed so that the desired program memory page is addressed (Figure 6-2). When one of the return instructions is executed, the entire 13-bit PC is POPed from the stack. Therefore, manipulation of the PCLATH<4:3> is not required for the return instructions.

Note:	Devices with program memory sizes 2K words and less, ignore both paging bits (PCLATH<4:3>), which are used to access program memory when more than one page is available. The use of PCLATH<4:3> as general purpose read/write bits (for these devices) is not recommended since this may affect upward compatibility with future products. Devices with program memory sizes between 2K words and 4K words, ignore the paging bit (PCLATH<4>), which is used to access program memory pages 2 and 3 (1000h - 1FFFh). The use of PCLATH<4> as a general purpose read/write bit (for these devices) is not recommended since this may affect upward compatibility with future products.

Example 6-1 shows the calling of a subroutine in page 1 of the program memory. This example assumes that PCLATH is saved and restored by the interrupt service routine (if interrupts are used).

Example 6-1: Call of a Subroutine in Page1 from Page0

```
     ORG 0x500
     BSF    PCLATH,3  ; Select Page1 (800h-FFFh)
     CALL   SUB1_P1   ; Call subroutine in Page1 (800h-FFFh)
       :              ;
       :              ;
     ORG 0x900        ;
SUB1_P1:              ; called subroutine Page1 (800h-FFFh)
       :              ;
     RETURN           ; return to Call subroutine in Page0 (000h-7FFh)
                      ;
```

6.3 Data Memory Organization

Data memory is made up of the Special Function Registers (SFR) area, and the General Purpose Registers (GPR) area. The SFRs control the operation of the device, while GPRs are the general area for data storage and scratch pad operations.

The data memory is banked for both the GPR and SFR areas. The GPR area is banked to allow greater than 96 bytes of general purpose RAM to be addressed. SFRs are for the registers that control the peripheral and core functions. Banking requires the use of control bits for bank selection. These control bits are located in the STATUS Register (STATUS<7:5>). Figure 6-5 shows one of the data memory map organizations, this organization is device dependent.

To move values from one register to another register, the value must pass through the W register. This means that for all register-to-register moves, two instruction cycles are required.

The entire data memory can be accessed either directly or indirectly. Direct addressing may require the use of the RP1:RP0 bits. Indirect addressing requires the use of the File Select Register (FSR). Indirect addressing uses the Indirect Register Pointer (IRP) bit of the STATUS register for accesses into the Bank0 / Bank1 or the Bank2 / Bank3 areas of data memory.

6.3.1 General Purpose Registers (GPR)

Some Mid-Range MCU devices have banked memory in the GPR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other resets.

The register file can be accessed either directly, or using the File Select Register FSR, indirectly. Some devices have areas that are shared across the data memory banks, so a read / write to that area will appear as the same location (value) regardless of the current bank. We refer to this area as the Common RAM.

6.3.2 Special Function Registers (SFR)

The SFRs are used by the CPU and Peripheral Modules for controlling the desired operation of the device. These registers are implemented as 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 in this section, while those related to the operation of the peripheral features are described in the section of that peripheral feature.

All Mid-Range MCU devices have banked memory in the SFR area. Switching between these banks requires the RP0 and RP1 bits in the STATUS register to be configured for the desired bank. Some SFRs are initialized by a Power-on Reset and other resets, while other SFRs are unaffected.

Note:	The Special Function Register (SFR) Area may have General Purpose Registers (GPRs) mapped in these locations.

The register file can be accessed either directly, or using the File Select Register FSR, indirectly.

6.3.3 Banking

The data memory is partitioned into four banks. Each bank contains General Purpose Registers and Special Function Registers. Switching between these banks requires the RP0 and RP1 bits in the STATUS register to be configured for the desired bank when using direct addressing. The IRP bit in the STATUS register is used for indirect addressing.

Table 6-1: Direct and Indirect Addressing of Banks

Accessed Bank	Direct (RP1:RP0)	Indirect (IRP)
0	0 0	0
1	0 1	
2	1 0	1
3	1 1	

Each Bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers. All data memory is implemented as static RAM. All Banks may contain special function registers. Some “high use” special function registers from Bank0 are mirrored in the other banks for code reduction and quicker access.

Through the evolution of the products, there are a few variations in the layout of the Data Memory. The data memory organization that will be the standard for all new devices is shown in Figure 6-5. This Memory map has the last 16-bytes mapped across all memory banks. This is to reduce the software overhead for context switching. The registers in **bold** will be in every device. The other registers are peripheral dependent. Not every peripheral’s registers are shown, because some file addresses have a different registers from those shown. As with all the figures, tables, and specifications presented in this reference guide, verify the details with the device specific data sheet.

Figure 6-4: Direct Addressing

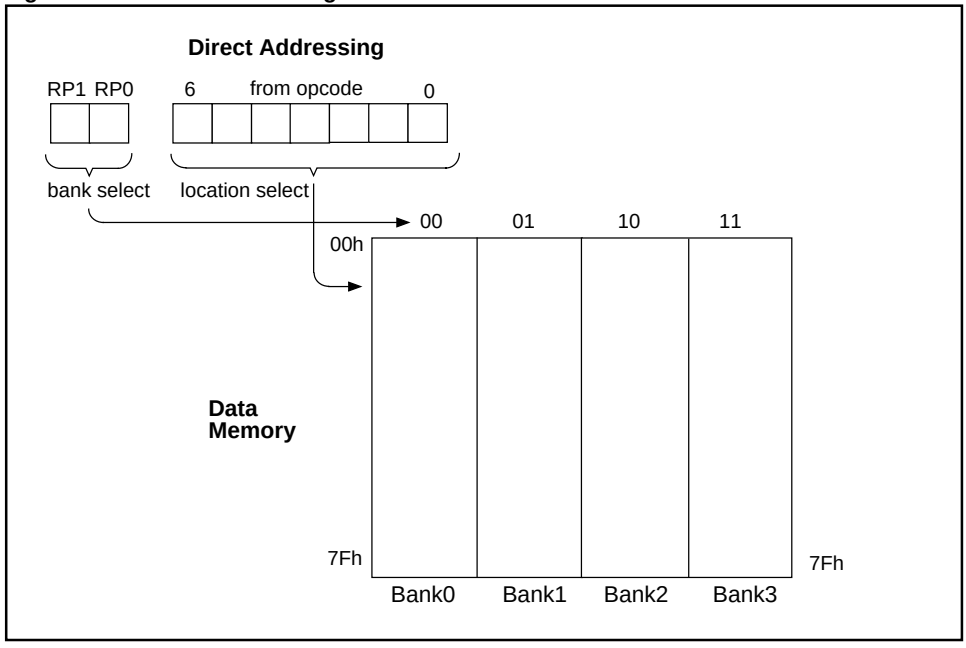


Figure 6-5: Register File Map

	File Address		File Address		File Address		File Address
INDF	00h	**INDF**	80h	**INDF**	100h	**INDF**	180h
TMR0	01h	**OPTION_REG**	81h	**TMR0**	101h	**OPTION_REG**	181h
PCL	02h	**PCL**	82h	**PCL**	102h	**PCL**	182h
STATUS	03h	**STATUS**	83h	**STATUS**	103h	**STATUS**	183h
FSR	04h	**FSR**	84h	**FSR**	104h	**FSR**	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h	PORTF	107h	TRISF	187h
PORTD	08h	TRISD	88h	PORTG	108h	TRISG	188h
PORTE	09h	TRISE	89h		109h		189h
PCLATH	0Ah	**PCLATH**	8Ah	**PCLATH**	10Ah	**PCLATH**	18Ah
INTCON	0Bh	**INTCON**	8Bh	**INTCON**	10Bh	**INTCON**	18Bh
PIR1	0Ch	**PIE1**	8Ch		10Ch		18Ch
PIR2	0Dh	PIE2	8Dh		10Dh		18Dh
TMR1L	0Eh	**PCON**	8Eh		10Eh		18Eh
TMR1H	0Fh	OSCCAL	8Fh		10Fh		18Fh
T1CON	10h		90h		110h		190h
TMR2	11h		91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPATAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch		9Ch		11Ch		19Ch
CCP2CON	1Dh		9Dh		11Dh		19Dh
ADRES	1Eh		9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Registers (2)	20h – 7Fh	General Purpose Registers (3)	A0h – EFh	General Purpose Registers (3)	120h – 16Fh	General Purpose Registers (3)	1A0h – 1EFh
		Mapped in Bank0 70h - 7Fh (4)	F0h – FFh	Mapped in Bank0 70h - 7Fh (4)	170h – 17Fh	Mapped in Bank0 70h - 7Fh (4)	1F0h – 1FFh
Bank0		Bank1		Bank2 (5)		Bank3 (5)	

Note 1: Registers in **BOLD** will be present in every device.
2: Not all locations may be implemented. Unimplemented locations will read as '0'.
3: These locations may not be implemented. Depending on the device, accesses to the unimplemented locations operate differently. Please refer to the specific device data sheet for details.
4: Some device do not map these registers into Bank0. In devices where these registers are mapped into Bank0, these registers are referred to as common RAM
5: Some devices may not implement these banks. Locations in unimplemented banks will read as '0'.
6: General Purpose Registers (GPRs) may be located in the Special Function Register (SFR) area.

The map in Figure 6-6 shows the register file memory map of some 18-pin devices. Unimplemented registers will read as '0'.

Figure 6-6: Register File Map

	File Address		File Address
INDF	00h	**INDF**	80h
TMR0	01h	**OPTION_REG**	81h
PCL	02h	**PCL**	82h
STATUS	03h	**STATUS**	83h
FSR	04h	**FSR**	84h
PORTA	05h	**TRISA**	85h
PORTB	06h	**TRISB**	86h
	07h	**PCON**	87h
ADCON0 / EEDATA (2)	08h	ADCON1 / EECON1 (2)	88h
ADRES / EEADR (2)	09h	ADRES / EECON2 (2)	89h
PCLATH	0Ah	**PCLATH**	8Ah
INTCON	0Bh	**INTCON**	8Bh
General Purpose Registers (3)	0Ch – 7Fh	General Purpose Registers (4)	8Ch – FFh
Bank0		Bank1	

Note 1: Registers in **BOLD** will be present in every device.
2: These registers may not be implemented, or are implemented as other registers in some devices.
3: Not all locations may be implemented. Unimplemented locations will read as '0'.
4: These locations are unimplemented in Bank1. Access to these unimplemented locations will access the corresponding Bank0 register.

6.3.4 Indirect Addressing, INDF, and FSR Registers

Indirect addressing is a mode of addressing data memory where the data memory address in the instruction is not fixed. An SFR register is used as a pointer to the data memory location that is to be read or written. Since this pointer is in RAM, the contents can be modified by the program. This can be useful for data tables in the data memory. Figure 6-7 shows the operation of indirect addressing. This shows the moving of the value to the data memory address specified by the value of the FSR register.

Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses the register pointed to by the File Select Register, FSR. Reading the INDF register itself indirectly (FSR = '0') will read 00h. Writing to the INDF register indirectly results in a no-operation (although status bits may be affected). An effective 9-bit address is generated by the concatenation of the IRP bit (STATUS<7>) with the 8-bit FSR register, as shown in Figure 6-8.

Figure 6-7: Indirect Addressing

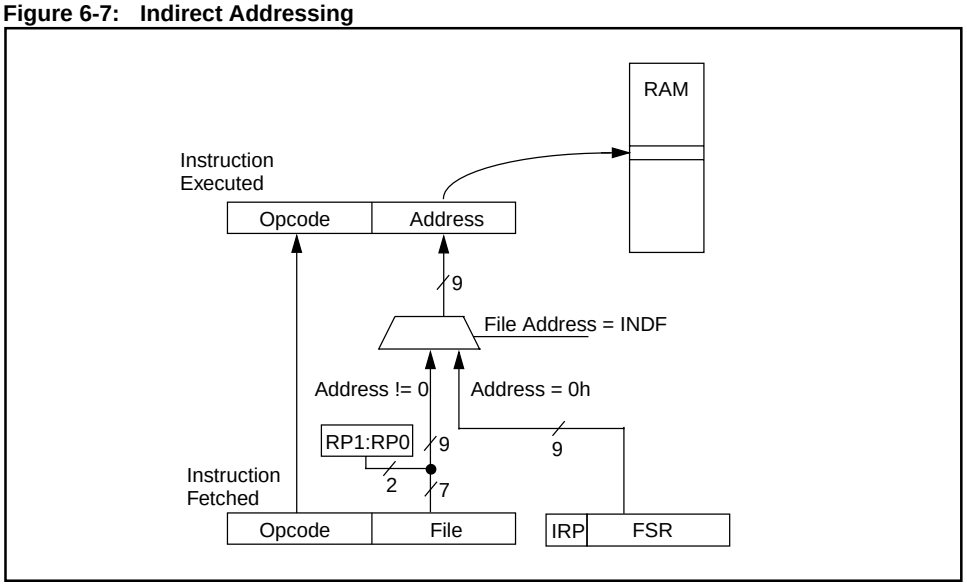
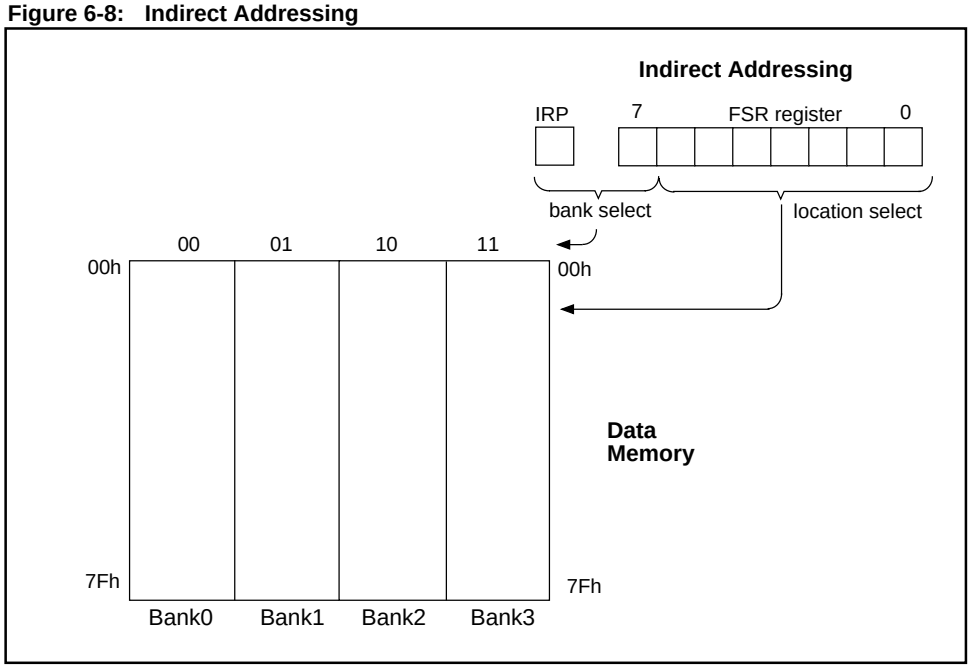


Figure 6-8: Indirect Addressing



Example 6-2 shows a simple use of indirect addressing to clear RAM (locations 20h-2Fh) 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.

Example 6-2: Indirect Addressing

```
         BCF      STATUS, IRP    ; Indirect addressing Bank0/1
         MOVLW    0x20           ; Initialize pointer to RAM
         MOVWF    FSR            ;
NEXT     CLRF     INDF           ; Clear INDF register
         INCF     FSR,F          ; Inc pointer
         BTFSS    FSR,4          ; All done?
         GOTO     NEXT           ; NO, clear next
CONTINUE                         ;
         :                       ; YES, continue
```

6.4 Initialization

Example 6-3 shows how the bank switching occurs for Direct addressing, while Example 6-4 shows some code to do initialization (clearing) of General Purpose RAM.

Example 6-3: Bank Switching

```
    CLRF    STATUS          ; Clear STATUS register (Bank0)
    :                       ;
    BSF     STATUS, RP0     ; Bank1
    :                       ;
    BCF     STATUS, RP0     ; Bank0
    :                       ;
    MOVLW   0x60            ; Set RP0 and RP1 in STATUS register, other
    XORWF   STATUS, F       ;   bits unchanged (Bank3)
    :                       ;
    BCF     STATUS, RP0     ; Bank2
    :                       ;
    BCF     STATUS, RP1     ; Bank0
```

Example 6-4: RAM Initialization

```
       CLRF   STATUS       ; Clear STATUS register (Bank0)
       MOVLW  0x20         ; 1st address (in bank) of GPR area
       MOVWF  FSR          ; Move it to Indirect address register
Bank0_LP
       CLRF   INDF0        ; Clear GPR at address pointed to by FSR
       INCF   FSR          ; Next GPR (RAM) address
       BTFSS  FSR, 7       ; End of current bank ? (FSR = 80h, C = 0)
       GOTO   Bank0_LP     ; NO, clear next location
;
; Next Bank (Bank1)
; (** ONLY REQUIRED IF DEVICE HAS A BANK1 **)
;
       MOVLW  0xA0         ; 1st address (in bank) of GPR area
       MOVWF  FSR          ; Move it to Indirect address register
Bank1_LP
       CLRF   INDF0        ; Clear GPR at address pointed to by FSR
       INCF   FSR          ; Next GPR (RAM) address
       BTFSS  STATUS, C    ; End of current bank? (FSR = 00h, C = 1)
       GOTO   Bank1_LP     ; NO, clear next location
;
; Next Bank (Bank2)
; (** ONLY REQUIRED IF DEVICE HAS A BANK2 **)
;
       BSF    STATUS, IRP  ; Select Bank2 and Bank3
                           ;    for Indirect addressing
       MOVLW  0x20         ; 1st address (in bank) of GPR area
       MOVWF  FSR          ; Move it to Indirect address register
Bank2_LP
       CLRF   INDF0        ; Clear GPR at address pointed to by FSR
       INCF   FSR          ; Next GPR (RAM) address
       BTFSS  FSR, 7       ; End of current bank? (FSR = 80h, C = 0)
       GOTO   Bank2_LP     ; NO, clear next location
;
; Next Bank (Bank3)
; (** ONLY REQUIRED IF DEVICE HAS A BANK3 **)
;
       MOVLW  0xA0         ; 1st address (in bank) of GPR area
       MOVWF  FSR          ; Move it to Indirect address register
Bank3_LP
       CLRF   INDF0        ; Clear GPR at address pointed to by FSR
       INCF   FSR          ; Next GPR (RAM) address
       BTFSS  STATUS, C    ; End of current bank? (FSR = 00h, C = 1)
       GOTO   Bank3_LP     ; NO, clear next location
     :                     ; YES, All GPRs (RAM) is cleared
```

6.5 Design Tips

Question 1: ***Program execution seems to get lost.***

Answer 1:

When a device with more then 2K words of program memory is used, the calling of subroutines may require that the PCLATH register be loaded prior to the CALL (or GOTO) instruction to specify the correct program memory page that the routine is located on. The following instructions will correctly load PCLATH register, regardless of the program memory location of the label SUB_1.

```
          MOVLW    HIGH (SUB_1)    ; Select Program Memory Page of
          MOVWF    PCLATH          ;     Routine.
          CALL     SUB_1           ; Call the desired routine
          :
          :
SUB_1     :                        ; Start of routine
          :
          RETURN                   ; Return from routine
```

Question 2: ***I need to initialize RAM to '0's. What is an easy way to do that?***

Answer 2:

Example 6-4 shows this. If the device you are using does not use all 4 data memory banks, some of the code may be removed.

6.6 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to memory are:

Title	Application Note #
Implementing a Table Read	AN556

6.7 Revision History

Revision A

This is the initial released revision of the Memory Organization description.

Section 7. Data EEPROM

HIGHLIGHTS

This section of the manual contains the following major topics:

7.1 Introduction

The EEPROM data memory is readable and writable during normal operation (full VDD range). This memory is not directly mapped in the register file space. Instead it is indirectly addressed through the Special Function Registers. There are four SFRs used to read and write this memory. These registers are:

- EECON1
- EECON2 (not a physically implemented register)
- EEDATA
- EEADR

EEDATA holds the 8-bit data for read/write, and EEADR holds the address of the EEPROM location being accessed. The 8-bit EEADR register can access up to 256 locations of Data EEPROM. The EEADR register can be thought of as the indirect addressing register of the Data EEPROM. EECON1 contains the control bits, while EECON2 is the register used to initiate the read/write.

Some devices will implement less then the entire memory map. The address range always starts at 0h, and goes throughout the memory available. Table 7-1 shows some of the possible common device memory sizes and the address range for those sizes.

Table 7-1: Possible Data EEPROM Memory Sizes

Data EEPROM Size (1)	Address Range
64	0h - 3Fh
128	0h - 7Fh
256	0h - FFh

Note 1: Presently, devices are only offered with 64 bytes of Data EEPROM.

The EEPROM data memory allows byte read and write. A byte write automatically erases the location and writes the new data (erase before write). The EEPROM data memory is rated for high erase/write cycles. The write time is controlled by an on-chip timer. The write-time will vary with voltage and temperature as well as from chip to chip. Please refer to the AC specifications for exact limits.

When the device is code protected, the CPU may continue to read and write the data EEPROM memory. The device programmer can no longer access this memory.

7.2 Control Register

Register 7-1: EECON1 Register

U-0	U-0	U-0	R/W-1	R/W-1	R/W-x	R/S-0	R/S-x
—	—	—	EEIF (1)	WRERR	WREN	WR	RD
bit 7							bit 0

bit 7:5 **Unimplemented:** Read as '0'

bit 4 **EEIF**: EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed (must be cleared in software)
0 = The write operation is not complete or has not been started

bit 3 **WRERR**: EEPROM Error Flag bit
1 = A write operation is prematurely terminated
(any $\overline{\text{MCLR}}$ reset or any WDT reset during normal operation)
0 = The write operation completed

bit 2 **WREN**: EEPROM Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the data EEPROM

bit 1 **WR**: Write Control bit
1 = initiates a write cycle. The bit is cleared by hardware once write is complete.
The WR bit can only be set (not cleared) in software.
0 = Write cycle to the data EEPROM is complete

bit 0 **RD**: Read Control bit
1 = Initiates an EEPROM read. Read takes one cycle. RD is cleared in hardware.
The RD bit can only be set (not cleared) in software.
0 = Does not initiate an EEPROM read

Legend		
R = Readable bit	W = Writable bit	S = Settable bit
U = Unimplemented bit, read as '0'		- n = Value at POR reset

Note 1: Future devices will have this bit in the PIR register.

7.3 EEADR

The EEADR register can address up to a maximum of 256 bytes of data EEPROM.

The unused address bits are decoded. This means that these bits must always be '0' to ensure that the address is in the Data EEPROM memory space.

7.4 EECON1 and EECON2 Registers

EECON1 is the control register with five low order bits physically implemented. The upper-three bits are unimplemented and read as '0's.

Control bits RD and WR initiate read and write, respectively. These bits cannot be cleared, only set, in software. They are cleared in hardware at completion of the read or write operation. The inability to clear the WR bit in software prevents the accidental, premature termination of a write operation.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a $\overline{\text{MCLR}}$ reset or a WDT time-out reset during normal operation. In these situations, following reset, the user can check the WRERR bit and rewrite the location. The data and address will be unchanged in the EEDATA and EEADR registers.

Interrupt flag bit EEIF is set when write is complete. It must be cleared in software.

EECON2 is not a physical register. Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the Data EEPROM write sequence.

7.5 Reading the EEPROM Data Memory

To read a data memory location, the user must write the address to the EEADR register and then set control bit RD (EECON1<0>). The data is available, in the very next instruction cycle, in the EEDATA register; therefore it can be read by the next instruction. EEDATA will hold this value until another read or until it is written to by the user (during a write operation).

Example 7-1: Data EEPROM Read

```
BCF     STATUS, RP0  ; Bank0
MOVLW   CONFIG_ADDR  ; Any location in Data EEPROM memory space
MOVWF   EEADR        ; Address to read
BSF     STATUS, RP0  ; Bank1
BSF     EECON1, RD   ; EE Read
BCF     STATUS, RP0  ; Bank0
MOVF    EEDATA, W    ; W = EEDATA
```

7.6 Writing to the EEPROM Data Memory

To write an EEPROM data location, the user must first write the address to the EEADR register and the data to the EEDATA register. Then the user must follow a specific sequence to initiate the write for each byte.

Example 7-2: Data EEPROM Write

```
                BSF     STATUS, RP0  ; Bank1
                BCF     INTCON, GIE  ; Disable INTs.
                BSF     EECON1, WREN ; Enable Write
                MOVLW   55h          ;
Required        MOVWF   EECON2       ; 55h must be written to EECON2
Sequence        MOVLW   AAh          ;   to start write sequence
                MOVWF   EECON2       ; Write AAh
                BSF     EECON1,WR    ; Set WR bit begin write
                BSF     INTCON, GIE  ; Enable INTs.
```

The write will not initiate if the above sequence is not exactly followed (write 55h to EECON2, write AAh to EECON2, then set WR bit) for each byte. We strongly recommend that interrupts be disabled during this code segment.

Additionally, the WREN bit in EECON1 must be set to enable write. This mechanism prevents accidental writes to data EEPROM due to errant (unexpected) code execution (i.e., lost programs). The user should keep the WREN bit clear at all times, except when updating EEPROM. The WREN bit is not cleared by hardware

After a write sequence has been initiated, clearing the WREN bit will not affect this write cycle. The WR bit will be inhibited from being set unless the WREN bit is set.

At the completion of the write cycle, the WR bit is cleared in hardware and the EE Write Complete Interrupt Flag bit (EEIF) is set. The user can either enable this interrupt or poll this bit. EEIF must be cleared by software.

7.7 Write Verify

Depending on the application, good programming practice may dictate that the value written to the Data EEPROM be verified (Example 7-3) as the value that was intended to be written. This should be used in applications where an EEPROM bit will be stressed near the specification limit. The Total Endurance disk will help determine your comfort level.

Example 7-3: Write Verify

```
       BCF   STATUS, RP0 ; Bank0
       :                 ; Any code can go here
       :                 ;
       MOVF  EEDATA, W   ; Must be in Bank0
       BSF   STATUS, RP0 ; Bank1
READ
       BSF   EECON1, RD  ; YES, Read the value written
       BCF   STATUS, RP0 ; Bank0
;
; Is the value written (in W reg) and read (in EEDATA) the same?
;
       SUBWF EEDATA, W   ;
       BTFSS STATUS, Z   ; Is difference 0?
       GOTO  WRITE_ERR   ; NO, Write error
       :                 ; YES, Good write
       :                 ; Continue program
```

7.8 Protection Against Spurious Writes

There are conditions when the device may not want to write to the data EEPROM memory. To protect against spurious EEPROM writes, various mechanisms have been built-in. On power-up, WREN is cleared. Also, the Power-up Timer (72 ms duration) prevents EEPROM write.

The write initiate sequence and the WREN bit together help prevent an accidental write during brown-outs, power glitches, and software malfunction.

7.9 Data EEPROM Operation During Code Protected Configuration

When the device is code protected, the CPU is able to read and write data to the Data EEPROM.

For ROM devices, there are two code protection bits. One for the ROM program memory and one for the Data EEPROM memory. See the Device Programming Specification for more information about these bits.

7.10 Initialization

The Data EEPROM module does not have an initialization sequence such as other modules. To do a read of the Data EEPROM refer to Example 7-1. To do a write to the Data EEPROM refer to Example 7-2, and to verify that the write completed successfully refer to Example 7-3.

As for the General Purpose RAM, it is a good idea to initialize all Data EEPROM locations to a known state. This initialization may take place at the time of device programming or an application diagnostic mode, since on reset you may not want the Data EEPROM to be cleared.

An Application Diagnostic mode may be a condition on the I/O pins that the device tests for after the device power-ups. Then depending on this mode, the device would do some diagnostic function. The state for the I/O pins would need to be something that would not be possible without the injected levels to force this diagnostic mode.

7.11 Design Tips

Question 1: ***Why do the data EEPROM locations not contain the data that I wrote?***

Answer 1:

There are a few possibilities, but the most likely is that you did not exactly follow the write sequence as shown in Example 7-2. If you are using this code segment ensure that all interrupts are disabled during this sequence.

Question 2: ***Why is the data in the data EEPROM is getting corrupted?***

Answer 2:

The data will only change when a Data EEPROM write occurs. Inadvertent writes may occur when the device is in a brown-out condition (out of operating specification) and the device is not being forced to the reset state. During a brown-out, either the internal brown-out circuitry should be enabled (when available) or external circuitry should be used to reset the PICmicro MCU to ensure that no data EEPROM writes occur when the device is out of the valid operating range.

7.12 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to data EEPROM are:

Title	Application Note #
EEPROM Endurance Tutorial	AN601
How to get 10 Million Cycles out of your Microchip Serial EEPROM	AN602
Basic Serial EEPROM Operation	AN536
Everything a System Engineer needs to know about Serial EEPROM Endurance	AN537
Using the Microchip Endurance Predictive Software	AN562

7.13 Revision History

Revision A

This is the initial released revision of the Data EEPROM description.

Section 8. Interrupts

HIGHLIGHTS

This section of the manual contains the following major topics:

8.1 Introduction

PICmicro MCUs can have many sources of interrupt. These sources generally include one interrupt source for each peripheral module, though some modules may generate multiple interrupts (such as the USART module). The current interrupts are:

- INT Pin Interrupt (external interrupt)
- TMR0 Overflow Interrupt
- PORTB Change Interrupt (pins RB7:RB4)
- Comparator Change Interrupt
- Parallel Slave Port Interrupt
- USART Interrupts
- Receive Interrupt
- Transmit Interrupt
- A/D Conversion Complete Interrupt
- LCD Interrupt.
- Data EEPROM Write Complete Interrupt
- Timer1 Overflow Interrupt
- Timer2 Overflow Interrupt
- CCP Interrupt
- SSP Interrupt

There is a minimum of one register used in the control and status of the interrupts. This register is:

- INTCON

Additionally, if the device has peripheral interrupts, then it will have registers to enable the peripheral interrupts and registers to hold the interrupt flag bits. Depending on the device, the registers are:

- PIE1
- PIR1
- PIE2
- PIR2

We will generically refer to these registers as PIR and PIE. If future devices provide more interrupt sources, they will be supported by additional register pairs, such as PIR3 and PIE3.

The Interrupt Control Register, INTCON, records individual flag bits for core interrupt requests. It also has various individual enable bits and the global interrupt enable bit.

The Global Interrupt Enable bit, GIE (INTCON<7>), enables (if set) all un-masked interrupts or disables (if cleared) all interrupts. Individual interrupts can be disabled through their corresponding enable bits in the INTCON register. The GIE bit is cleared on reset.

The "return from interrupt" instruction, RETFIE, exits the interrupt routine as well as sets the GIE bit, which allows any pending interrupt to execute.

The INTCON register contains these interrupts: INT Pin Interrupt, the RB Port Change Interrupt, and the TMR0 Overflow Interrupt. The INTCON register also contains the Peripheral Interrupt Enable bit, PEIE. The PEIE bit will enable/disable the peripheral interrupts from vectoring when the PEIE bit is set/cleared.

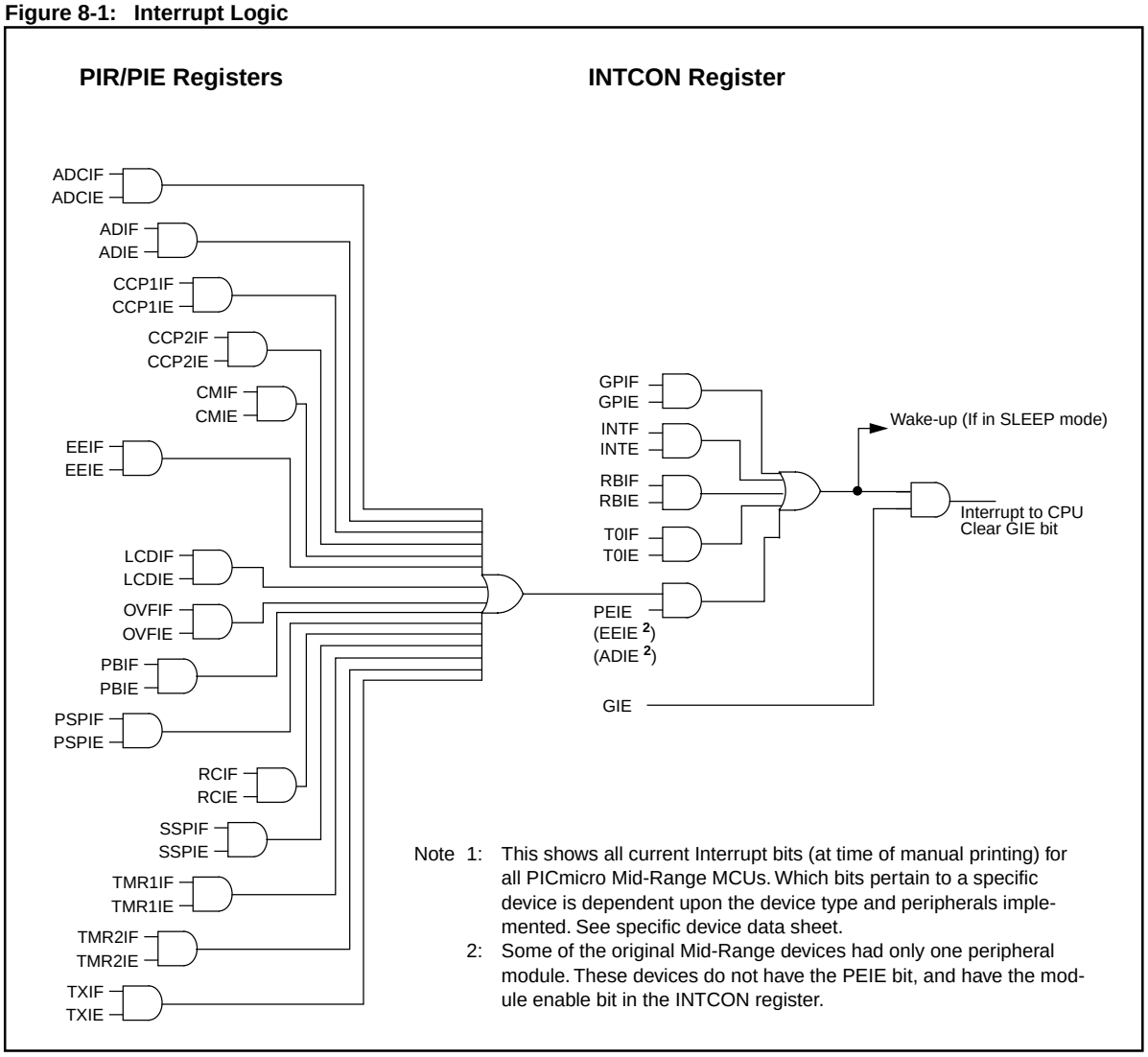
When an interrupt is responded to, the GIE bit is cleared to disable any further interrupt, the return address is pushed into the stack and the PC is loaded with 0004h. Once in the interrupt service routine the source(s) of the interrupt can be determined by polling the interrupt flag bits. Generally the interrupt flag bit(s) must be cleared in software before re-enabling the global interrupt to avoid recursive interrupts.

Once in the interrupt service routine the source(s) of the interrupt can be determined by polling the interrupt flag bits. Individual interrupt flag bits are set regardless of the status of their corresponding mask bit or the GIE bit.

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

Note 2: When an instruction that clears the GIE bit is executed, any interrupts that were pending for execution in the next cycle are ignored. The CPU will execute a NOP in the cycle immediately following the instruction which clears the GIE bit. The interrupts which were ignored are still pending to be serviced when the GIE bit is set again.

Figure 8-1: Interrupt Logic



Note 1: This shows all current Interrupt bits (at time of manual printing) for all PICmicro Mid-Range MCUs. Which bits pertain to a specific device is dependent upon the device type and peripherals implemented. See specific device data sheet.

2: Some of the original Mid-Range devices had only one peripheral module. These devices do not have the PEIE bit, and have the module enable bit in the INTCON register.

8.2 Control Registers

Generally devices have a minimum of three registers associated with interrupts. The INTCON register which contains Global Interrupt Enable bit, GIE, as well as the Peripheral Interrupt Enable bit, PEIE, and the PIE / PIR register pair which enable the peripheral interrupts and display the interrupt flag status.

8.2.1 INTCON Register

The INTCON Register is a readable and writable register which contains various enable and flag bits.

Note:	Interrupt flag bits get set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>).This feature allows for software polling.

Register 8-1: INTCON Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GIE	PEIE (3)	T0IE	INTE (2)	RBIE (1, 2)	T0IF	INTF (2)	RBIF (1, 2)
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all un-masked interrupts
0 = Disables all interrupts

bit 6 **PEIE:** Peripheral Interrupt Enable bit
1 = Enables all un-masked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **T0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INTE:** INT External Interrupt Enable bit
1 = Enables the INT external interrupt
0 = Disables the INT external interrupt

bit 3 **RBIE (1):** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **T0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF:** INT External Interrupt Flag bit
1 = The INT external interrupt occurred (must be cleared in software)
0 = The INT external interrupt did not occur

bit 0 **RBIF (1):** RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

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

Note 1:	In some devices, the RBIE bit may also be known as GPIE and the RBIF bit may be know as GPIF.
Note 2:	Some devices may not have this feature. For those devices this bit is reserved.
Note 3:	In devices with only one peripheral interrupt, this bit may be EEIE or ADIE.

8.2.2 PIE Register(s)

Depending on the number of peripheral interrupt sources, there may be multiple Peripheral Interrupt Enable registers (PIE1, PIE2). These registers contain the individual enable bits for the Peripheral interrupts. These registers will be generically referred to as PIE. If the device has a PIE register, The PEIE bit must be set to enable any of these peripheral interrupts.

Note:	Bit PEIE (INTCON<6>) must be set to enable any of the peripheral interrupts.

Although, the PIE register bits have a general bit location with each register, future devices may not have consistent placement. Bit location inconsistencies will not be a problem if you use the supplied Microchip Include files for the symbolic use of these bits. This will allow the Assembler/Compiler to automatically take care of the placement of these bits by specifying the correct register and bit name.

Register 8-2: PIE Register

R/W-0
(Note 1)
bit 7 ... bit 0

bit **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

bit **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit **CCP2IE**: CCP2 Interrupt Enable bit
1 = Enables the CCP2 interrupt
0 = Disables the CCP2 interrupt

bit **SSPIE**: Synchronous Serial Port Interrupt Enable bit
1 = Enables the SSP interrupt
0 = Disables the SSP interrupt

bit **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit **ADCIE**: Slope A/D Converter comparator Trip Interrupt Enable bit
1 = Enables the Slope A/D interrupt
0 = Disables the Slope A/D interrupt

bit **OVFIE**: Slope A/D TMR Overflow Interrupt Enable bit
1 = Enables the Slope A/D TMR overflow interrupt
0 = Disables the Slope A/D TMR overflow interrupt

bit **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

bit **EEIE**: EE Write Complete Interrupt Enable bit
1 = Enables the EE write complete interrupt
0 = Disables the EE write complete interrupt

bit **LCDIE**: LCD Interrupt Enable bit
1 = Enables the LCD interrupt
0 = Disables the LCD interrupt

bit **CMIE**: Comparator Interrupt Enable bit
1 = Enables the Comparator interrupt
0 = Disables the Comparator interrupt

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

Note 1: The bit position of the enable bits is device dependent. Please refer to the device data sheet for bit placement.

8.2.3 PIR Register(s)

Depending on the number of peripheral interrupt sources, there may be multiple Peripheral Interrupt Flag registers (PIR1, PIR2). These registers contain the individual flag bits for the peripheral interrupts. These registers will be generically referred to as PIR.

Note 1:	Interrupt flag bits get set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>).
Note 2:	User software should ensure the appropriate interrupt flag bits are cleared (by software) prior to enabling an interrupt, and after servicing that interrupt.

Although, the PIR bits have a general bit location within each register, future devices may not be able to be consistent with that. It is recommended that you use the supplied Microchip Include files for the symbolic use of these bits. This will allow the Assembler/Compiler to automatically take care of the placement of these bits within the specified register.

Register 8-3: PIR Register

R/W-0
(Note 1)
bit 7 ... bit 0

bit **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

bit **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit **CCP1IF**: CCP1 Interrupt Flag bit

Capture Mode
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred

Compare Mode
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred

PWM Mode
Unused in this mode

bit **CCP2IF**: CCP2 Interrupt Flag bit

Capture Mode
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred

Compare Mode
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred

PWM Mode
Unused in this mode

bit **SSPIF**: Synchronous Serial Port Interrupt Flag bit
1 = The transmission/reception is complete
0 = Waiting to transmit/receive

bit **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer, RCREG, is full (cleared when RCREG is read)
0 = The USART receive buffer is empty

bit **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer, TXREG, is empty (cleared when TXREG is written)
0 = The USART transmit buffer is full

bit **ADIF**: A/D Converter Interrupt Flag bit
1 = An A/D conversion completed (must be cleared in software)
0 = The A/D conversion is not complete

Register 8-3: PIR Register (Cont'd)

bit **ADCIF**: Slope A/D Converter Comparator Trip Interrupt Flag bit
1 = An A/D conversion completed (must be cleared in software)
0 = The A/D conversion is not complete

bit **OVFIF**: Slope A/D TMR Overflow Interrupt Flag bit
1 = Slope A/D TMR overflowed (must be cleared in software)
0 = Slope A/D TMR did not overflow

bit **PSPIF:** Parallel Slave Port Read/Write Interrupt Flag bit
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

bit **EEIF**: EE Write Complete Interrupt Flag bit
1 = The data EEPROM write operation is complete (must be cleared in software)
0 = The data EEPROM write operation is not complete

bit **LCDIF**: LCD Interrupt Flag bit
1 = LCD interrupt has occurred (must be cleared in software)
0 = LCD interrupt has not occurred

bit **CMIF**: Comparator Interrupt Flag bit
1 = Comparator input has changed (must be cleared in software)
0 = Comparator input has not changed

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

Note 1: The bit position of the flag bits is device dependent. Please refer to the device data sheet for bit placement.

8.3 Interrupt Latency

Interrupt latency is defined as the time from the interrupt event (the interrupt flag bit gets set) to the time that the instruction at address 0004h starts execution (when that interrupt is enabled).

For synchronous interrupts (typically internal), the latency is 3TCY.

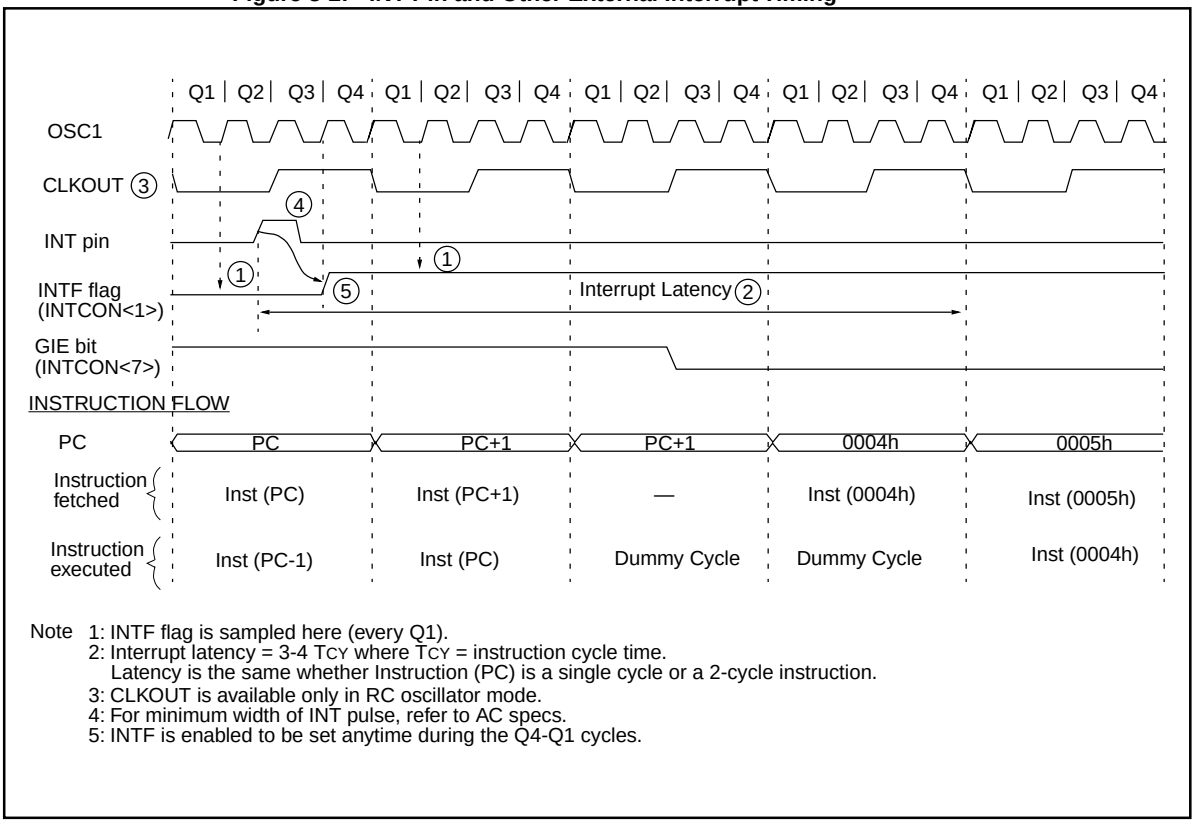
For asynchronous interrupts (typically external), such as the INT or Port RB Change Interrupt, the interrupt latency will be 3 - 3.75TCY (instruction cycles). The exact latency depends upon when the interrupt event occurs (Figure 8-2) in relation to the instruction cycle.

The latency is the same for both one and two cycle instructions.

8.4 INT and External Interrupts

The external interrupt on the INT pin is edge triggered: either rising if the INTEDG bit (OPTION<6>) is set, or falling, if the INTEDG bit is clear. When a valid edge appears on the INT pin, the INTF flag bit (INTCON<1>) is set. This interrupt can be enabled/disabled by setting/clearing the INTE enable bit (INTCON<4>). The INTF bit must be cleared in software in the interrupt service routine before re-enabling this interrupt. The INT interrupt can wake-up the processor from SLEEP, if the INTE bit was set prior to going into SLEEP. The status of the GIE bit decides whether or not the processor branches to the interrupt vector following wake-up. See the **"Watchdog Timer and Sleep Mode"** section for details on SLEEP and for timing of wake-up from SLEEP through INT interrupt.

Figure 8-2: INT Pin and Other External Interrupt Timing



Note 1: INTF flag is sampled here (every Q1).
2: Interrupt latency = 3-4 TCY where TCY = instruction cycle time.
Latency is the same whether Instruction (PC) is a single cycle or a 2-cycle instruction.
3: CLKOUT is available only in RC oscillator mode.
4: For minimum width of INT pulse, refer to AC specs.
5: INTF is enabled to be set anytime during the Q4-Q1 cycles.

Note: Any interrupts caused by external signals (such as timers, capture, change on port) will have similar timing.

8.5 Context Saving During Interrupts

During an interrupt, only the return PC value is saved on the stack. Typically, users may wish to save key registers during an interrupt e.g. W register and STATUS register. This has to be implemented in software.

The action of saving information is commonly referred to as “PUSHing,” while the action of restoring the information before the return is commonly referred to as “POPing.” These (PUSH, POP) are not instruction mnemonics, but are conceptual actions. This action can be implemented by a sequence of instructions. For ease of code transportability, these code segments can be made into MACROs (see MPASM Assembler User’s Guide for details on creating macros).

Example 8-1 stores and restores the STATUS and W registers for devices with common RAM (such as the PIC16C77). The user register, W_TEMP, must be defined across all banks and must be defined at the same offset from the bank base address (i.e., W_TEMP is defined at 0x70 - 0x7F in Bank0). The user register, STATUS_TEMP, must be defined in Bank0, in this example STATUS_TEMP is also in Bank0.

The steps of Example 8-1:

1. Stores the W register regardless of current bank.
2. Stores the STATUS register in Bank0.
3. Executes the Interrupt Service Routine (ISR) code.
4. Restores the STATUS (and bank select bit register).
5. Restores the W register.

If additional locations need to be saved before executing the Interrupt Service Routine (ISR) code, they should be saved after the STATUS register is saved (step 2), and restored before the STATUS register is restored (step 4).

Example 8-1: Saving the STATUS and W Registers in RAM (for Devices with Common RAM)

```
MOVWF   W_TEMP          ; Copy W to a Temporary Register
                        ;    regardless of current bank
SWAPF   STATUS,W        ; Swap STATUS nibbles and place
                        ;    into W register
MOVWF   STATUS_TEMP     ; Save STATUS to a Temporary register
                        ;    in Bank0
:
: (Interrupt Service Routine (ISR) )
:
SWAPF   STATUS_TEMP,W   ; Swap original STATUS register value
                        ;    into W (restores original bank)
MOVWF   STATUS          ; Restore STATUS register from
                        ;    W register
SWAPF   W_TEMP,F        ; Swap W_Temp nibbles and return
                        ;    value to W_Temp
SWAPF   W_TEMP,W        ; Swap W_Temp to W to restore original
                        ;    W value without affecting STATUS
```

Example 8-2 stores and restores the STATUS and W registers for devices without common RAM (such as the PIC16C74A). The user register, W_TEMP, must be defined across all banks and must be defined at the same offset from the bank base address (i.e., W_TEMP is defined at 0x70 - 0x7F in Bank0). The user register, STATUS_TEMP, must be defined in Bank0.

Within the 70h - 7Fh range (Bank0), wherever W_TEMP is expected the corresponding locations in the other banks should be dedicated for the possible saving of the W register.

The steps of Example 8-2:

1. Stores the W register regardless of current bank.
2. Stores the STATUS register in Bank0.
3. Executes the Interrupt Service Routine (ISR) code.
4. Restores the STATUS (and bank select bit register).
5. Restores the W register.

If additional locations need to be saved before executing the Interrupt Service Routine (ISR) code, they should be saved after the STATUS register is saved (step 2), and restored before the STATUS register is restored (step 4).

Example 8-2: Saving the STATUS and W Registers in RAM (for Devices without Common RAM)

```
     MOVWF   W_TEMP          ; Copy W to a Temporary Register
                             ;    regardless of current bank
     SWAPF   STATUS,W        ; Swap STATUS nibbles and place
                             ;    into W register
     BCF     STATUS,RP0      ; Change to Bank0 regardless of
                             ;    current bank
     MOVWF   STATUS_TEMP     ; Save STATUS to a Temporary register
                             ;    in Bank0
     :
     : (Interrupt Service Routine (ISR) )
     :
     SWAPF   STATUS_TEMP,W   ; Swap original STATUS register value
                             ;    into W (restores original bank)
     MOVWF   STATUS          ; Restore STATUS register from
                             ;    W register
     SWAPF   W_TEMP,F        ; Swap W_Temp nibbles and return
                             ;    value to W_Temp
     SWAPF   W_TEMP,W        ; Swap W_Temp to W to restore original
                             ;    W value without affecting STATUS
```

Example 8-3 stores and restores the STATUS and W registers for devices with general purpose RAM only in Bank0 (such as the PIC16C620). The Bank must be tested before saving any of the user registers. , W_TEMP, must be defined across all banks and must be defined at the same offset from the bank base address. The user register, STATUS_TEMP, must be defined in Bank0.

The steps of Example 8-3:

1. Test current bank.
2. Stores the W register regardless of current bank.
3. Stores the STATUS register in Bank0.
4. Executes the Interrupt Service Routine (ISR) code.
5. Restores the STATUS (and bank select bit register).
6. Restores the W register.

If additional locations need to be saved before executing the Interrupt Service Routine (ISR) code, they should be saved after the STATUS register is saved (step 2), and restored before the STATUS register is restored (step 4).

Example 8-3: Saving the STATUS and W Registers in RAM (for Devices with General Purpose RAM Only in Bank0)

```
Push
      BTFSS   STATUS, RP0         ; In Bank 0?
      GOTO    RP0CLEAR            ; YES,
      BCF     STATUS, RP0         ; NO, Force to Bank 0
      MOVWF   W_TEMP              ; Store W register
      SWAPF   STATUS, W           ; Swap STATUS register and
      MOVWF   STATUS_TEMP         ;   store in STATUS_TEMP
      BSF     STATUS_TEMP, 1      ; Set the bit that corresponds to RP0
      GOTO    ISR_Code            ; Push completed
RP0CLEAR
      MOVWF   W_TEMP              ; Store W register
      SWAPF   STATUS, W           ; Swap STATUS register and
      MOVWF   STATUS_TEMP         ;   store in STATUS_TEMP
;
ISR_Code
      :
      : (Interrupt Service Routine (ISR) )
      :
;
Pop
      SWAPF   STATUS_TEMP, W      ; Restore Status register
      MOVWF   STATUS              ;
      BTFSS   STATUS, RP0         ; In Bank 1?
      GOTO    Restore_WREG        ; NO,
      BCF     STATUS, RP0         ; YES, Force Bank 0
      SWAPF   W_TEMP, F           ; Restore W register
      SWAPF   W_TEMP, W           ;
      BSF     STATUS, RP0         ; Back to Bank 1
      RETFIE                      ; POP completed
Restore_WREG
      SWAPF   W_TEMP, F           ; Restore W register
      SWAPF   W_TEMP, W           ;
      RETFIE                      ; POP completed
```

8.6 Initialization

Example 8-4 shows the initialization and enabling of device interrupts, where PIE1_MASK1 value is the value to write into the interrupt enable register.

Example 8-5 shows how to create macro definitions for functions. Macros **must** be defined before they are used. For debugging ease, it may help if macros are placed in other files that are included at assembly time. This allows the source to be viewed without all the clutter of the required macros. These files must be included before the macro is used, but it simplifies debugging, if all include files are done at the top of the source file. Example 8-6 shows this structure.

Example 8-7 shows a typical Interrupt Service Routine structure. This ISR uses macros for the saving and restoring of registers before the execution of the interrupt code.

Example 8-4: Initialization and Enabling of Interrupts

```
PIE1_MASK1   EQU B'01101010'   ; This is the Interrupt Enable
      :                        ;    Register mask value
      :
      CLRF   STATUS         ; Bank0
      CLRF   INTCON         ; Disable interrupts and clear some flags
      CLRF   PIR1           ; Clear all flag bits
      BSF    STATUS, RP0    ; Bank1
      MOVLW  PIE1_MASK1     ; This is the initial masking for PIE1
      MOVWF  PIE1           ;
      BCF    STATUS, RP0    ; Bank0
      BSF    INTCON, GIE    ; Enable Interrupts
```

Example 8-5: Register Saving / Restoring as Macros

```
PUSH_MACRO    MACRO           ; This Macro Saves register contents
     MOVWF    W_TEMP          ; Copy W to a Temporary Register
                              ;    regardless of current bank
     SWAPF    STATUS,W        ; Swap STATUS nibbles and place
                              ;    into W register
     MOVWF    STATUS_TEMP     ; Save STATUS to a Temporary register
                              ;    in Bank0
     ENDM                     ; End this Macro
;
POP_MACRO     MACRO           ; This Macro Restores register contents
     SWAPF    STATUS_TEMP,W   ; Swap original STATUS register value
                              ;    into W (restores original bank)
     MOVWF    STATUS          ; Restore STATUS register from
                              ;    W register
     SWAPF    W_TEMP,F        ; Swap W_Temp nibbles and return
                              ;    value to W_Temp
     SWAPF    W_TEMP,W        ; Swap W_Temp to W to restore original
                              ;    W value without affecting STATUS
     ENDM                     ; End this Macro
```

Example 8-6: Source File Template

```
     LIST   p = p16C77      ; List Directive,
;    Revision History
;
     #INCLUDE    <P16C77.INC>    ; Microchip Device Header File
;
     #INCLUDE    <MY_STD.MAC>    ; Include my standard macros
     #INCLUDE    <APP.MAC>       ; File which includes macros specific
                                 ;   to this application
; Specify Device Configuration Bits
     __CONFIG    _XT_OSC & _PWRTE_ON & _BODEN_OFF & _CP_OFF & _WDT_ON
;
     org   0x00            ; Start of Program Memory
RESET_ADDR   :             ; First instruction to execute after a reset


     end
```

Example 8-7: Typical Interrupt Service Routine (ISR)

```
    org  ISR_ADDR           ;
      PUSH_MACRO            ; MACRO that saves required context registers,
                            ;   or in-line code
      CLRF   STATUS         ; Bank0
      BTFSC  PIR1, TMR1IF   ; Timer1 overflow interrupt?
      GOTO   T1_INT         ; YES
      BTFSC  PIR1, ADIF     ; NO, A/D interrupt?
      GOTO   AD_INT         ; YES, do A/D thing
      :                     ; NO, do this for all sources
      :                     ;
      BTFSC  PIR1, LCDIF    ; NO, LCD interrupt
      GOTO   LCD_INT        ; YES, do LCD thing
      BTFSC  INTCON, RBIF   ; NO, Change on PORTB interrupt?
      GOTO   PORTB_INT      ; YES, Do PortB Change thing
INT_ERROR_LP1               ; NO, do error recovery
      GOTO   INT_ERROR_LP1  ; This is the trap if you enter the ISR
                            ;    but there were no expected
                            ;    interrupts
T1_INT                      ; Routine when the Timer1 overflows
      :                     ;
      BCF     PIR1, TMR1IF  ; Clear the Timer1 overflow interrupt flag
      GOTO    END_ISR       ; Ready to leave ISR (for this request)
AD_INT                      ; Routine when the A/D completes
      :                     ;
      BCF     PIR1, ADIF    ; Clear the A/D interrupt flag
      GOTO    END_ISR       ; Ready to leave ISR (for this request)
LCD_INT                     ; Routine when the LCD Frame begins
      :                     ;
      BCF     PIR1, LCDIF   ; Clear the LCD interrupt flag
      GOTO    END_ISR       ; Ready to leave ISR (for this request)
PORTB_INT                   ; Routine when PortB has a change
      :                     ;
END_ISR                     ;
        POP_MACRO           ; MACRO that restores required registers,
                            ;   or in-line code
        RETFIE              ; Return and enable interrupts
```

8.7 Design Tips

Question 1: ***An algorithm does not give the correct results.***

Answer 1:

Assuming that the algorithm is correct and that interrupts are enabled during the algorithm, ensure that are registers that are used by the algorithm and by the interrupt service routine are saved and restored. If not some registers may be corrupted by the execution of the ISR.

Question 2: ***My system seems to lock up.***

Answer 2:

If interrupts are being used, ensure that the interrupt flag is cleared after servicing that interrupt (but before executing the `RETFIE` instruction). If the interrupt flag remains set when the `RETFIE` instruction is executed, program execution immediately returns to the interrupt vector, since there is an outstanding enabled interrupt.

8.8 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to this section are:

Title	Application Note #
Using the PortB Interrupt On Change as an External Interrupt	AN566

8.9 Revision History

Revision A

This is the initial released revision of the interrupt description.

Section 9. I/O Ports

HIGHLIGHTS

This section of the manual contains the following major topics:

9.1 Introduction

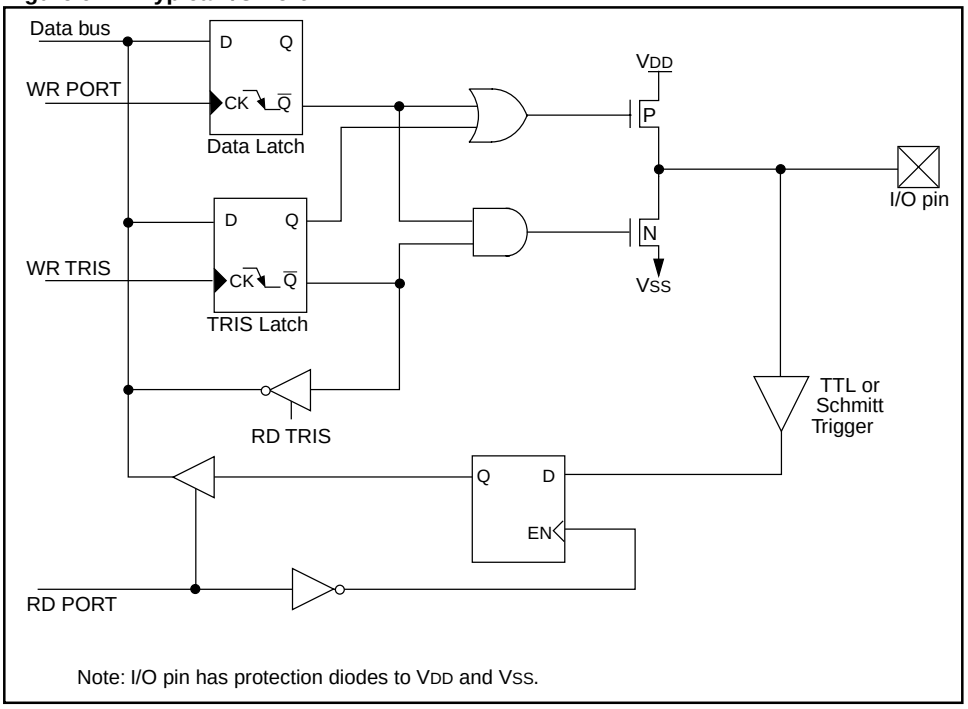
General purpose I/O pins can be considered the simplest of peripherals. They allow the PICmicro™ to monitor and control other devices. To add flexibility and functionality to a device, some pins are multiplexed with an alternate function(s). These functions depend on which peripheral features are on the device. In general, when a peripheral is functioning, that pin may not be used as a general purpose I/O pin.

For most ports, the I/O pin's direction (input or output) is controlled by the data direction register, called the TRIS register. TRIS<x> controls the direction of PORT<x>. A '1' in the TRIS bit corresponds to that pin being an input, while a '0' corresponds to that pin being an output. An easy way to remember is that a '1' looks like an I (input) and a '0' looks like an O (output).

The PORT register is the latch for the data to be output. When the PORT is read, the device reads the levels present on the I/O pins (not the latch). This means that care should be taken with read-modify-write commands on the ports and changing the direction of a pin from an input to an output.

Figure 9-1 shows a typical I/O port. This does not take into account peripheral functions that may be multiplexed onto the I/O pin. Reading the PORT register reads the status of the pins whereas writing to it will write to the port latch. All write operations (such as BSF and BCF instructions) are read-modify-write operations. Therefore a write to a port implies that the port pins are read, this value is modified, and then written to the port data latch.

Figure 9-1: Typical I/O Port



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

When peripheral functions are multiplexed onto general I/O pins, the functionality of the I/O pins may change to accommodate the requirements of the peripheral module. Examples of this are the Analog-to-Digital (A/D) converter and LCD driver modules, which force the I/O pin to the peripheral function when the device is reset. In the case of the A/D, this prevents the device from consuming excess current if any analog levels were on the A/D pins after a reset occurred.

With some peripherals, the TRIS bit is overridden while the peripheral is enabled. Therefore, read-modify-write instructions (`BSF, BCF, XORWF`) with TRIS as destination should be avoided. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

PORT pins may be multiplexed with analog inputs and analog VREF input. The operation of each of these pins is selected, to be an analog input or digital I/O, by clearing/setting the control bits in the ADCON1 register (A/D Control Register1). When selected as an analog input, these pins will read as '0's.

The TRIS registers control the direction of the port pins, even when they are being used as analog inputs. The user must ensure the TRIS bits are maintained set when using the pins as analog inputs.

Note 1: If pins are multiplexed with Analog inputs, then on a Power-on Reset these pins are configured as analog inputs, as controlled by the ADCON1 register. Reading port pins configured as analog inputs read a '0'.

Note 2: If pins are multiplexed with comparator inputs, then on a Power-on Reset these pins are configured as analog inputs, as controlled by the CMCON register. Reading port pins configured as analog inputs read a '0'.

Note 3: If pins are multiplexed with LCD driver segments, then on a Power-on Reset these pins are configured as LCD driver segments, as controlled by the LCDSE register. To configure the pins as a digital port, the corresponding bits in the LCDSE register must be cleared. Any bit set in the LCDSE register overrides any bit settings in the corresponding TRIS register.

Note 4: Pins may be multiplexed with the Parallel Slave Port (PSP). For the PSP to function, the I/O pins must be configured as digital inputs and the PSPMODE bit must be set.

Note 5: At present the Parallel Slave Port (PSP) is only multiplexed onto PORTD and PORTE. The microprocessor port becomes enabled when the PSPMODE bit is set. In this mode, the user must make sure that the TRISE bits are set (pins are configured as digital inputs) and that PORTE is configured for digital I/O. PORTD will override the values in the TRISD register. In this mode the PORTD and PORTE input buffers are TTL. The control bits for the PSP operation are located in TRISE.

9.2 PORTA and the TRISA Register

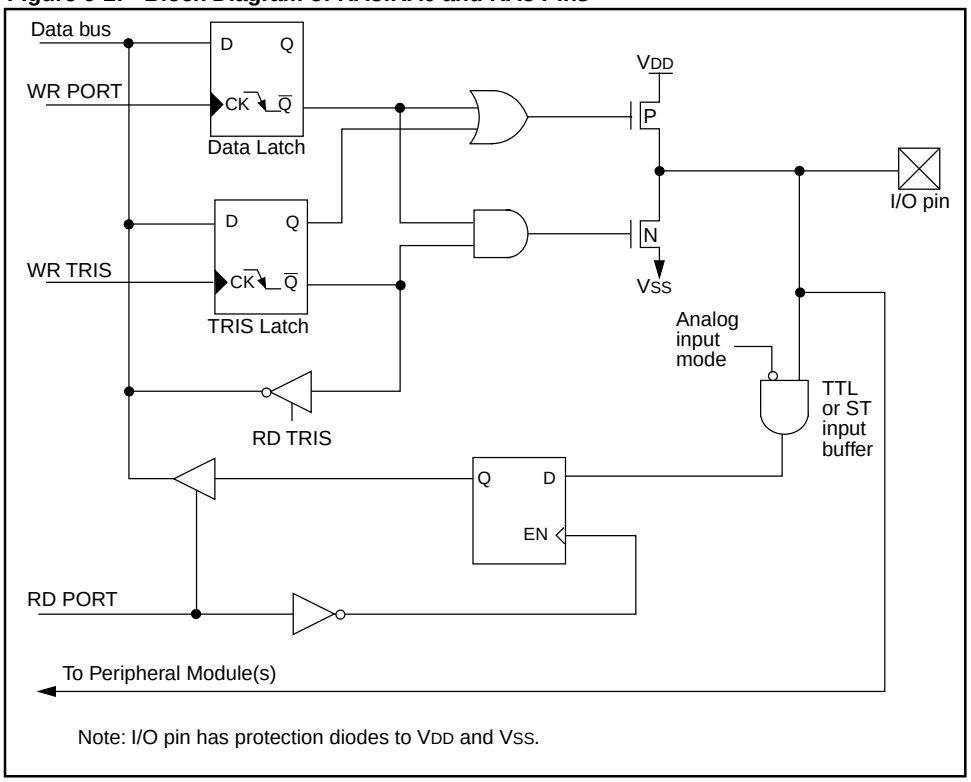
The RA4 pin is a Schmitt Trigger input and an open drain output. All other RA port pins have TTL input levels and full CMOS output drivers. All pins have data direction bits (TRIS registers) which can configure these pins as output or input.

Setting a TRISA register bit puts the corresponding output driver in a hi-impedance mode. Clearing a bit in the TRISA register puts the contents of the output latch on the selected pin(s).

Example 9-1: Initializing PORTA

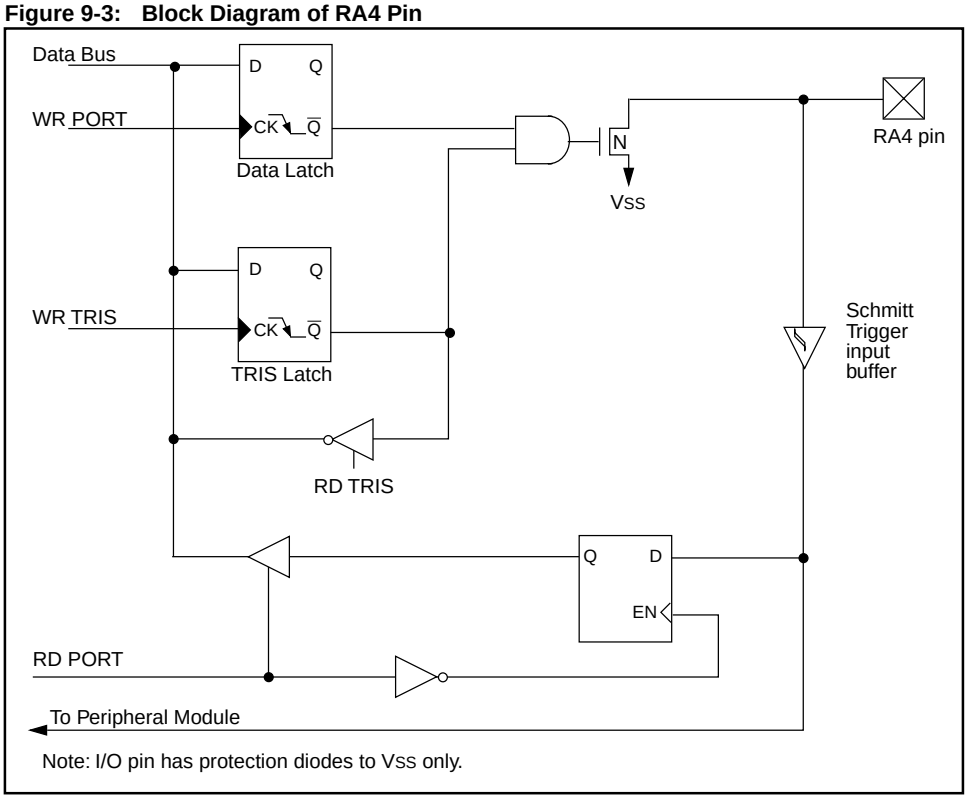
```
    CLRF    STATUS        ; Bank0
    CLRF    PORTA         ; Initialize PORTA by clearing output
                          ;    data latches
    BSF     STATUS, RP0   ; Select Bank1
    MOVLW   0xCF          ; Value used to initialize data direction
    MOVWF   TRISA         ; PORTA<3:0> = inputs PORTA<5:4> = outputs
                          ;    TRISA<7:6> always read as '0'
```

Figure 9-2: Block Diagram of RA3:RA0 and RA5 Pins



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

Figure 9-3: Block Diagram of RA4 Pin



9.3 PORTB and the TRISB Register

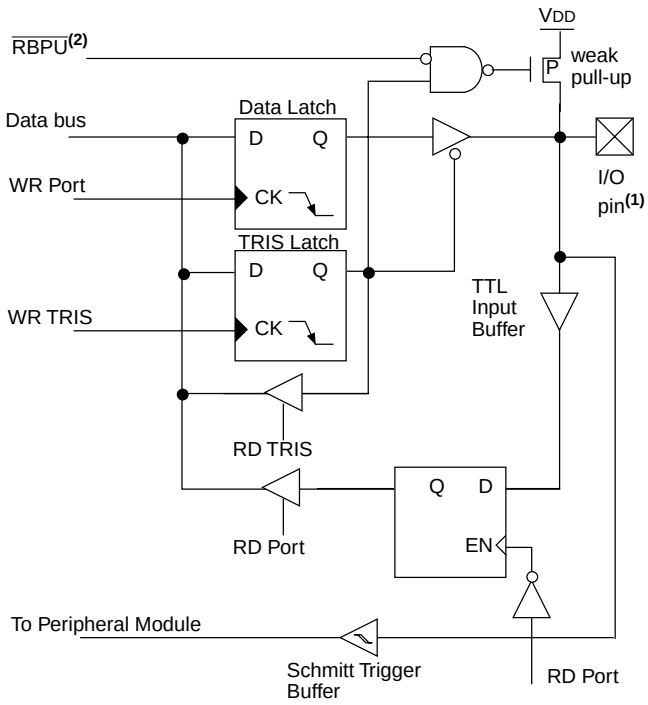
PORTB is an 8-bit wide bi-directional port. The corresponding data direction register is TRISB. Setting a bit in the TRISB register puts the corresponding output driver in a high-impedance input mode. Clearing a bit in the TRISB register puts the contents of the output latch on the selected pin(s).

Example 9-2: Initializing PORTB

```
    CLRF    STATUS       ; Bank0
    CLRF    PORTB        ; Initialize PORTB by clearing output
                         ;    data latches
    BSF     STATUS, RP0  ; Select Bank1
    MOVLW   0xCF         ; Value used to initialize data direction
    MOVWF   TRISB        ; PORTB<3:0> = inputs, PORTB<5:4> = outputs
                         ;    PORTB<7:6> = inputs
```

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

Figure 9-4: Block Diagram of RB3:RB0 Pins



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (OPTION<7>).

Four of PORTB's pins, RB7:RB4, have an interrupt on change feature. Only pins configured as inputs can cause this interrupt to occur (i.e. any RB7:RB4 pin configured as an output is excluded from the interrupt on change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB Port Change Interrupt with flag bit RBIF (INTCON<0>).

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

a) Any read or write of PORTB. This will end the mismatch condition.

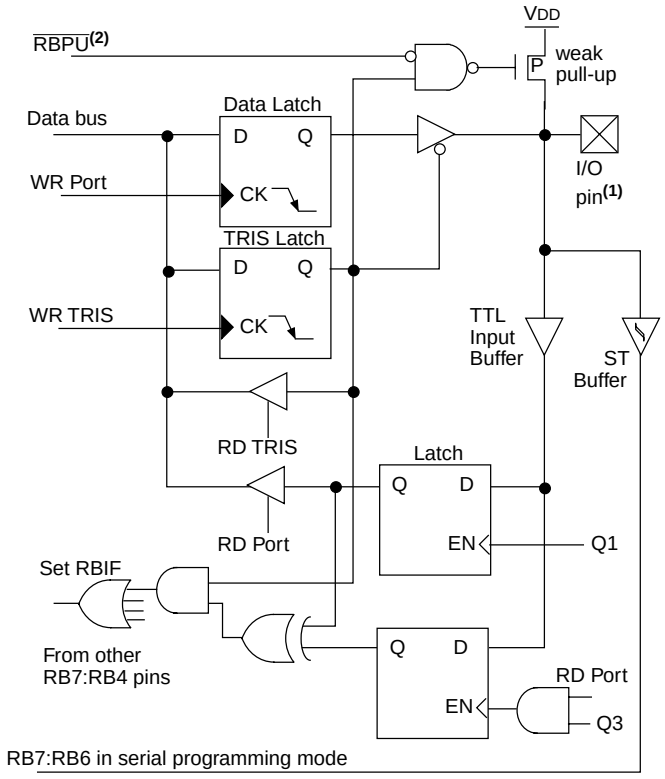
b) Clear flag bit RBIF.

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

This interrupt on mismatch feature, together with software configurable pull-ups on these four pins allow easy interface to a keypad and make it possible for wake-up on key-depression.

The interrupt on change feature is recommended for wake-up on key depression and operations where PORTB is only used for the interrupt on change feature. Polling of PORTB is not recommended while using the interrupt on change feature.

Figure 9-5: Block Diagram of RB7:RB4 Pins



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (OPTION<7>).
3: In sleep mode the device is in Q1 state.

9.4 PORTC and the TRISC Register

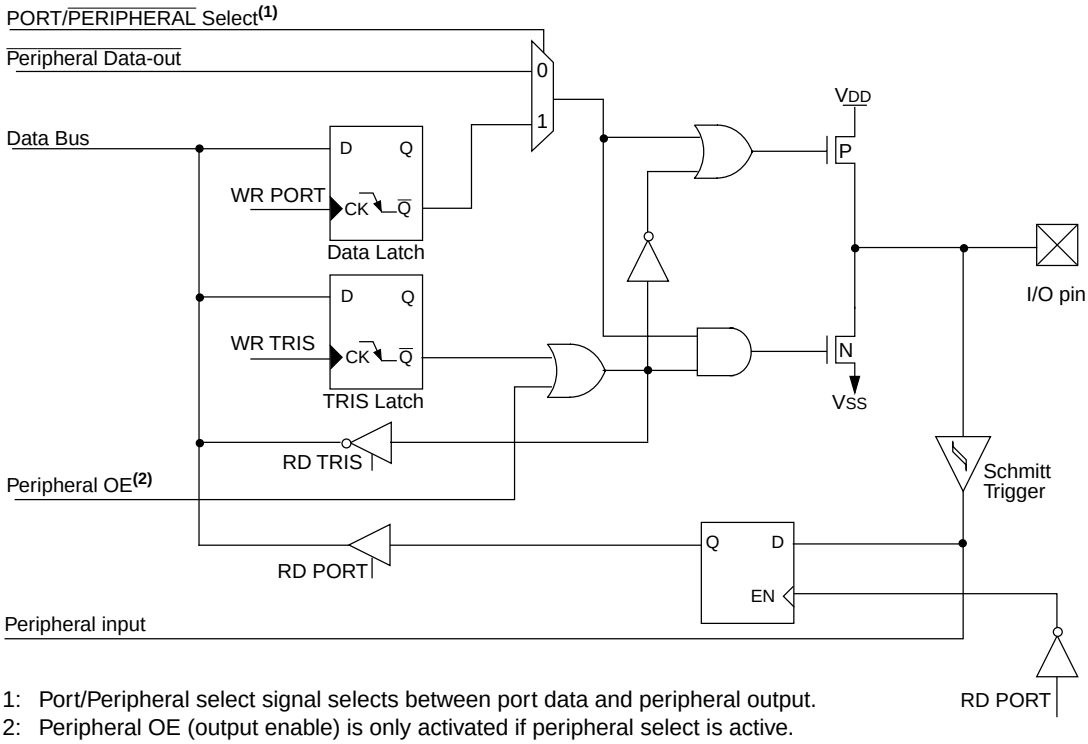
PORTC is an 8-bit bi-directional port. Each pin is individually configurable as an input or output through the TRISC register. PORTC pins have Schmitt Trigger input buffers.

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input.

Example 9-3: Initializing PORTC

```
    CLRF    STATUS       ; Bank0
    CLRF    PORTC        ; Initialize PORTC by clearing output
                         ;    data latches
    BSF     STATUS, RP0  ; Select Bank1
    MOVLW   0xCF         ; Value used to initialize data direction
    MOVWF   TRISC        ; PORTC<3:0> = inputs, PORTC<5:4> = outputs
                         ;    PORTC<7:6> = inputs
```

Figure 9-6: PORTC Block Diagram (Peripheral Output Override)



Note 1: Port/Peripheral select signal selects between port data and peripheral output.
2: Peripheral OE (output enable) is only activated if peripheral select is active.
3: I/O pins have diode protection to VDD and VSS.

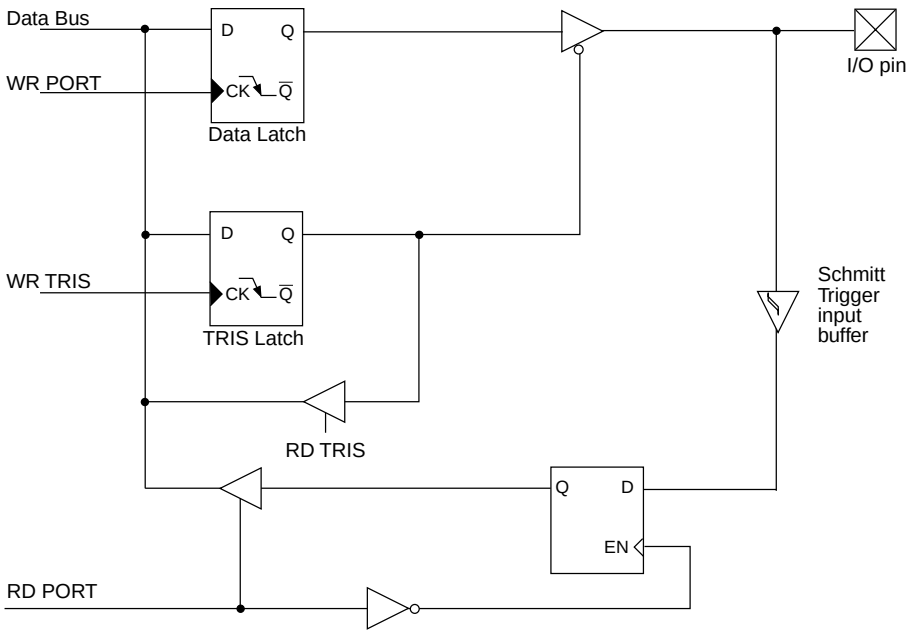
9.5 PORTD and the TRISD Register

PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

Example 9-4: Initializing PORTD

```
    CLRF    STATUS        ; Bank0
    CLRF    PORTD         ; Initialize PORTD by clearing output
                          ;    data latches
    BSF     STATUS, RP0   ; Select Bank1
    MOVLW   0xCF          ; Value used to initialize data direction
    MOVWF   TRISD         ; PORTD<3:0> = inputs, PORTD<5:4> = outputs
                          ;    PORTD<7:6> = inputs
```

Figure 9-7: Typical PORTD Block Diagram (in I/O Port Mode)



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

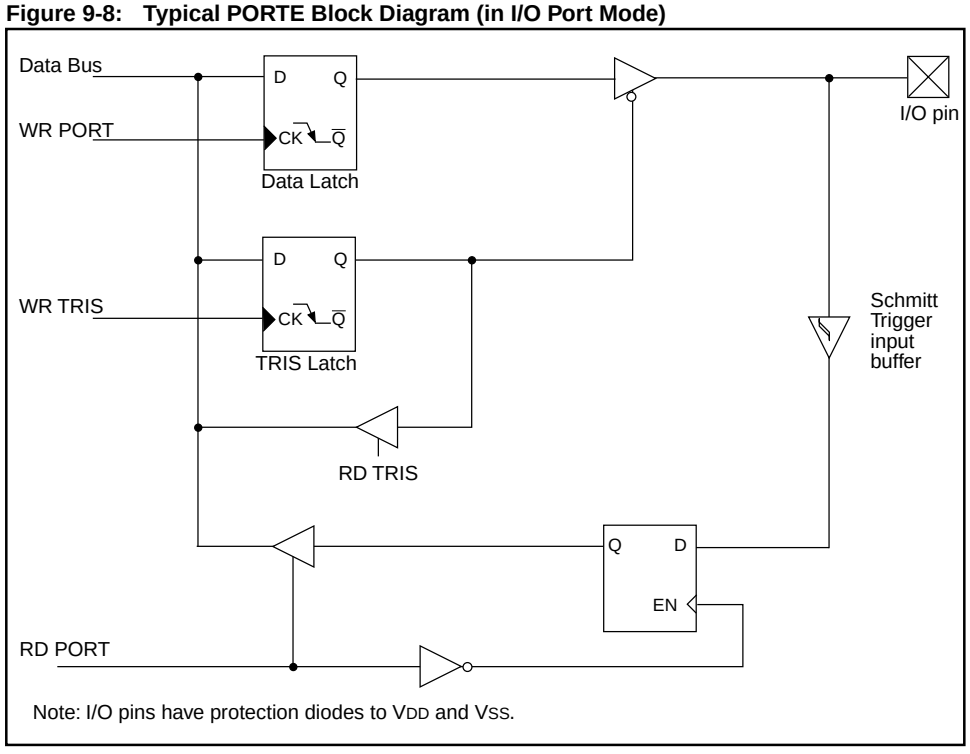
9.6 PORTE and the TRISE Register

PORTE can be up to an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

Example 9-5: Initializing PORTE

```
    CLRF    STATUS       ; Bank0
    CLRF    PORTE        ; Initialize PORTE by clearing output
                         ;    data latches
    BSF     STATUS, RP0  ; Select Bank1
    MOVLW   0x03         ; Value used to initialize data direction
    MOVWF   TRISE        ; PORTE<1:0> = inputs, PORTE<7:2> = outputs
```

Figure 9-8: Typical PORTE Block Diagram (in I/O Port Mode)



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

Note: On some devices with PORTE, the upper bits of the TRISE register are used for the Parallel Slave Port control and status bits.

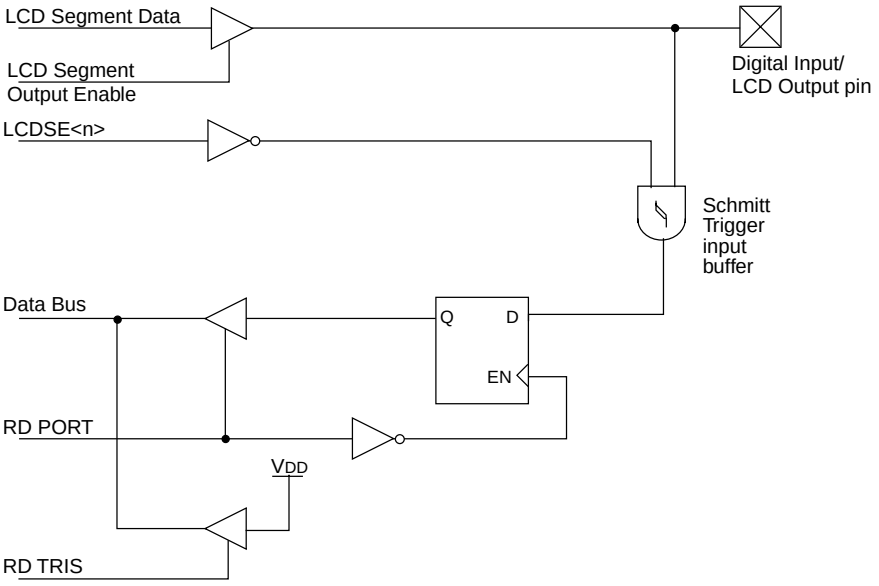
9.7 PORTF and the TRISF Register

PORTF is a digital input only port. Each pin is multiplexed with an LCD segment driver. These pins have Schmitt Trigger input buffers.

Example 9-6: Initializing PORTF

```
    BCF   STATUS, RP0    ; Select Bank2
    BSF   STATUS, RP1    ;
    BCF   LCDSE, SE16    ; Make all PORTF
    BCF   LCDSE, SE12    ;    digital inputs
```

Figure 9-9: PORTF LCD Block Diagram



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

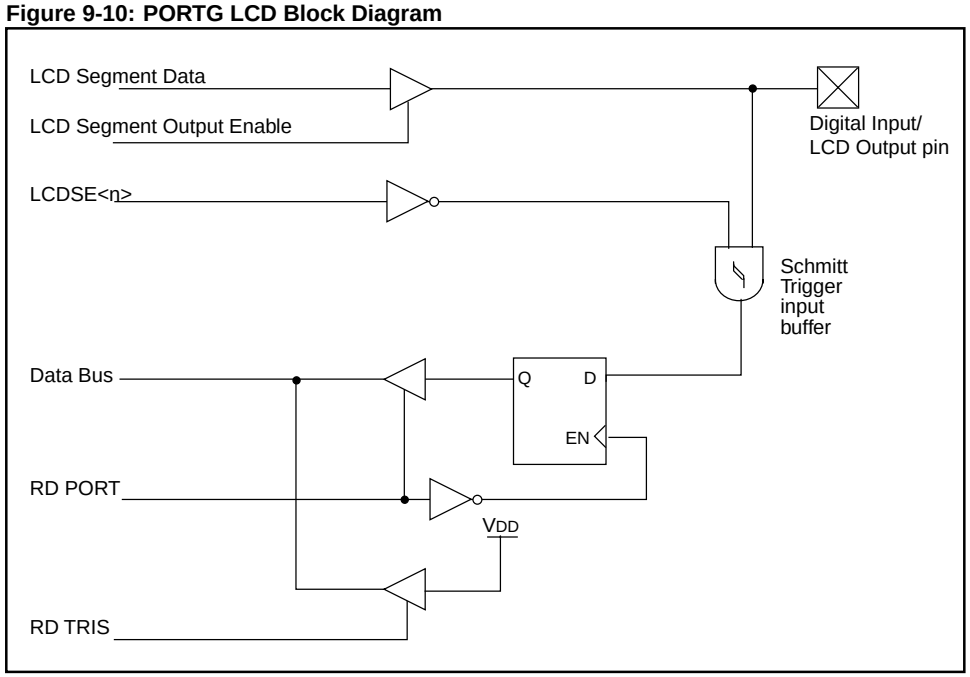
9.8 PORTG and the TRISG Register

PORTG is a digital input only port. Each pin is multiplexed with an LCD segment driver. These pins have Schmitt Trigger input buffers.

Example 9-7: Initializing PORTG

```
    BCF   STATUS, RP0    ; Select Bank2
    BSF   STATUS, RP1    ;
    BCF   LCDSE, SE27    ; Make all PORTG
    BCF   LCDSE, SE20    ;    and PORTE<7> digital inputs
```

Figure 9-10: PORTG LCD Block Diagram



9.9 GPIO and the TRISGP Register

GPIO is an 8-bit I/O register. Only the low order six bits are implemented (GP5:GP0). Bits 7 and 6 are unimplemented and read as '0's. Any GPIO pin (except GP3) can be programmed individually as input or output. The GP3 pin is an input only pin.

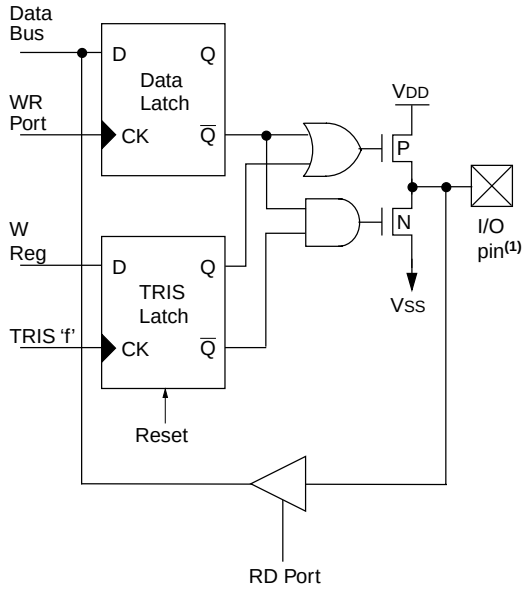
The TRISGP register controls the data direction for GPIO pins. A '1' in a TRISGP register bit puts the corresponding output driver in a hi-impedance mode. A '0' puts the contents of the output data latch on the selected pins, enabling the output buffer. The exceptions are GP3 which is input only and its TRIS bit will always read as '1'. Upon reset, the TRISGP register is all '1's, making all pins inputs.

A read of the GPIO port, reads the pins not the output data latches. Any input must be present until read by an input instruction (e.g., `MOVF GPIO,W`). The outputs are latched and remain unchanged until the output latch is rewritten.

Example 9-8: Initializing GPIO

```
    CLRF   STATUS       ; Bank0
    CLRF   GPIO         ; Initialize GPIO by clearing output
                        ;   data latches
    BSF    STATUS, RP0  ; Select Bank1
    MOVLW  0xCF         ; Value used to initialize data direction
    MOVWF  TRISGP       ; GP<3:0> = inputs GP<5:4> = outputs
                        ;   TRISGP<7:6> always read as '0'
```

Figure 9-11: Block Diagram of GP5:GP0 Pins



Note 1: I/O pins have protection diodes to VDD and VSS.
GP3 is input only with no data latch and no output drivers.

The configuration word can set several I/O's to alternate functions. When acting as alternate functions the pins will read as '0' during port read. The GP0, GP1, and GP3 pins can be configured with weak pull-ups and also with interrupt on change. The interrupt on change and weak pull-up functions are not pin selectable. Interrupt on change is enabled by setting INTCON<3>. If the device configuration bits select one of the external oscillator modes, the GP4 and GP5 pin's GPIO functions are overridden and these pins are used for the oscillator.

9.10 I/O Programming Considerations

When using the ports (and GPIO) as I/O, design considerations need to be taken into account to ensure that the operation is as intended.

9.10.1 Bi-directional I/O Ports

Any instruction which performs a write operation actually does a read followed by a write operation. The BCF and BSF instructions, for example, 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 to an output, the content of the data latch may now be unknown.

Reading the port register, 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 (ex. BCF, BSF, etc.) on a port, the value of the port pins is read, the desired operation is performed on this value, and the value is then written to the port latch.

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

Example 9-9: Read-Modify-Write Instructions on an I/O Port

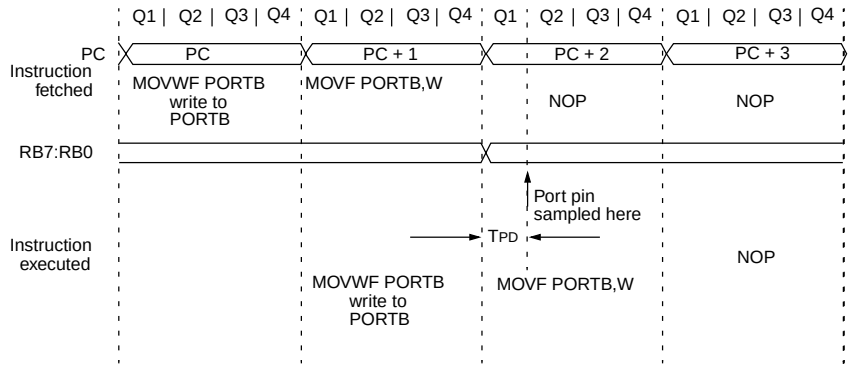
```
; Initial PORT settings: PORTB<7:4> Inputs
;                        PORTB<3:0> Outputs
; PORTB<7:6> have external pull-ups and are not connected to other circuitry
;
;                       PORT latch  PORT pins
;                       ----------  ---------
    BCF  PORTB, 7     ; 01pp pppp    11pp pppp
    BCF  PORTB, 6     ; 10pp pppp    11pp pppp
    BSF  STATUS, RP0  ;
    BCF  TRISB, 7     ; 10pp pppp    11pp pppp
    BCF  TRISB, 6     ; 10pp pppp    10pp pppp
;
; Note that the user may have expected the pin values to be 00pp ppp.
; The 2nd BCF caused RB7 to be latched as the pin value (high).
```

A pin configured as an output, actively driving a Low or High should not be driven from external devices at the same time in order to change the level on this pin (“wired-or,” “wired-and”). The resulting high output currents may damage the chip.

9.10.2 Successive Operations on an I/O Port

The actual write to an I/O port happens at the end of an instruction cycle, whereas for reading, the data must be valid at the beginning of the instruction cycle (Figure 9-12). 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 the next instruction which causes that file to be read into the CPU is executed. Otherwise, the previous state of that pin may be read into the CPU rather than the new state. When in doubt, it is better to separate these instructions with a NOP or another instruction not accessing this I/O port.

Figure 9-12: Successive I/O Operation



This example shows a write to PORTB followed by a read from PORTB.

Note: Data setup time = (0.25TCY - TPD)
whereTCY = instruction cycle
TPD = propagation delay

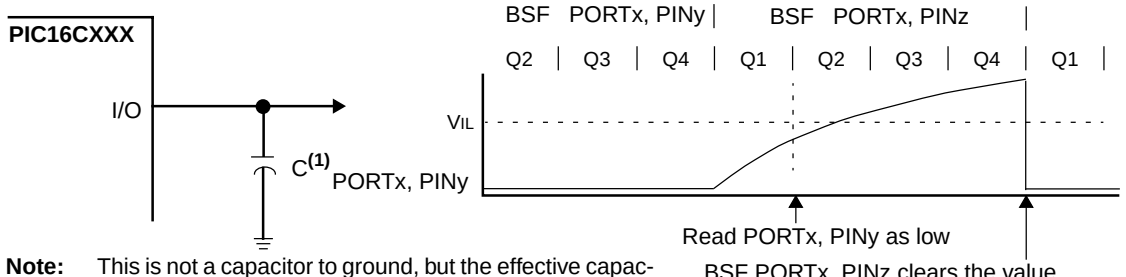
Therefore, at higher clock frequencies, a write followed by a read may be problematic due to external capacitance.

Figure 9-13 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 an 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 9-13: I/O Connection Issues



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

9.11 Initialization

See the section describing each port for examples of initialization of the ports.

Note:	It is recommended that when initializing the port, the data latch (PORT register) should be initialized first, and then the data direction (TRIS register). This will eliminate a possible pin glitch, since the PORT data latch values power up in a random state.

9.12 Design Tips

Question 1: ***Code will not toggle any I/O ports, but the oscillator is running. What can I be doing wrong?***

Answer 1:

1. Have the TRIS registers been initialized properly? These registers can be written to directly in the second bank (Bank1). In most cases the user is not switching to Bank1 (`BSF STATUS,RP0`) before writing zeros to the TRIS register.
2. If you are setting up the TRIS registers properly in Bank1 (RP0 = 1), you may not be returning to Bank0 before writing to the ports (`BCF STATUS,RP0`).
3. Is there a peripheral multiplexed onto those pins that are enabled?
4. Is the Watchdog Timer enabled (done at programming)? If it is enabled, is it being cleared properly with a `CLRWDT` instruction at least every 9 ms (or more if prescaled)?
5. Are you using the correct instructions to write to the port? More than one person has used the `MOVF` command when they should have used `MOVWF`.
6. For parts with interrupts, are the interrupts disabled? If not, try disabling them to verify they are not interfering.

Question 2: ***When my program reads a port, I get a different value than what I put in the port register. What can cause this?***

Answer 2:

1. When a port is read, it is always the pin that is read, regardless of its being set to input or output. So if a pin is set to an input, you will read the value on the pin regardless of the register value.
2. If a pin is set to output, say it has a one in the data latch; if it is shorted to ground you will still read a zero on the pin. This is very useful for building fault tolerant systems, or handling I^2C™ bus conflicts. (The I^2C bus is only driven low, and the pin is tristated for a one. If the pin is low and you are not driving it, some other device is trying to take the bus).
3. Mid-Range MCU devices all have at least one open drain (or open collector) pin. These pins can only drive a zero or tristate. For most Mid-Range devices this is pin RA4. Open drain pins must have a pull-up resistor to have a high state. This pin is useful for driving odd voltage loads. The pull-up can be connected to a voltage (typically less than V_{DD}) which becomes the high state.

Question 3: ***I have a PIC16CXXX with pin RB0 configured as an interrupt input, but am not getting interrupted. When I change my routine to poll the pin, it reads the high input and operates fine. What is the problem?***

Answer 3:

PORTB accepts TTL input levels (on most parts), so when you have an input of say 3V (with V_{DD} = 5V), you will read a one. However the buffer to the interrupt structure from pin RB0 is Schmitt Trigger, which requires a higher voltage (than TTL input) before the high input is registered. So it is possible to read a one, but not get the interrupt. The interrupt was given a Schmitt Trigger input with hysteresis to minimize noise problems. It is one thing to have short noise spikes on a pin that is a data input that can potentially cause bad data, but quite another to permit noise to cause an interrupt, hence the difference.

I^2C is a trademark of Philips Corporation.

Question 4: ***When I perform a BCF instruction, other pins get cleared in the port. Why?***

Answer 4:

1. Another case where a read-modify-write instruction may seem to change other pin values unexpectedly can be illustrated as follows: Suppose you make PORTC all outputs and drive the pins low. On each of the port pins is an LED connected to ground, such that a high output lights it. Across each LED is a 100 µF capacitor. Let's also suppose that the processor is running very fast, say 20 MHz. Now if you go down the port setting each pin in order; `BSF PORTC,0` then `BSF PORTC,1` then `BSF PORTC,2` and so on, you may see that only the last pin was set, and only the last LED actually turns on. This is because the capacitors take a while to charge. As each pin was set, the pin before it was not charged yet and so was read as a zero. This zero is written back out to the port latch (r-m-w, remember) which clears the bit you just tried to set the instruction before. This is usually only a concern at high speeds and for successive port operations, but it can happen, so take it into consideration.
2. If this is on a PIC16C7XX device, you have not configured the I/O pins properly in the ADCON1 register. If a pin is configured for analog input, any read of that pin will read a zero, regardless of the voltage on the pin. This is an exception to the normal rule that the pin state is always read. You can still configure an analog pin as an output in the TRIS register, and drive the pin high or low by writing to it, but you will always read a zero. Therefore if you execute a Read-Modify-Write instruction (see previous question) all analog pins are read as zero, and those not directly modified by the instruction will be written back to the port latch as zero. A pin configured as analog is expected to have values that may be neither high nor low to a digital pin, or floating. Floating inputs on digital pins are a no-no, and can lead to high current draw in the input buffer, so the input buffer is disabled.

9.13 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to I/O ports are:

Title	Application Note #
Improving the Susceptibility of an Application to ESD	AN595
Clock Design using Low Power/Cost Techniques	AN615
Implementing Wake-up on Keystroke	AN528
Interfacing to AC Power Lines	AN521
Multiplexing LED Drive and a 4 x 4 Keypad Sampling	AN529
Using PIC16C5X as an LCD Drivers	AN563
Serial Port Routines Without Using TMR0	AN593
Implementation of an Asynchronous Serial I/O	AN510
Using the PORTB Interrupt on Change Feature as an External Interrupt	AN566
Implementing Wake-up on Keystroke	AN522
Apple Desktop Bus	AN591
Software Implementation of Asynchronous Serial I/O	AN555
Communicating with the I^2C Bus using the PIC16C5X	AN515
Interfacing 93CX6 Serial EEPROMs to the PIC16C5X Microcontrollers	AN530
Logic Powered Serial EEPROMs	AN535
Interfacing 24LCXXB Serial EEPROMs to the PIC16C54	AN567
Using the 24XX65 and 24XX32 with Stand-alone PIC16C54 Code	AN558

9.14 Revision History

Revision A

This is the initial released revision of the I/O Ports description.

Section 10. Parallel Slave Port

HIGHLIGHTS

This section of the manual contains the following major topics:

10.1 Introduction

Some devices have an 8-bit wide Parallel Slave Port (PSP). This port is multiplexed onto one of the devices I/O ports. The PORT operates as an 8-bit wide Parallel Slave Port, or microprocessor port, when the PSPMODE control bit is set. In this mode, the input buffers are TTL.

In slave mode the module is asynchronously readable and writable by the external world through $\overline{RD}$ control input pin and the $\overline{WR}$ control input pin.

It can directly interface to an 8-bit microprocessor data bus. The external microprocessor can read or write the PORT latch as an 8-bit latch. Setting the PSPMODE bit enables port pins to be the $\overline{RD}$ input, the $\overline{WR}$ input, and the $\overline{CS}$ (chip select) input.

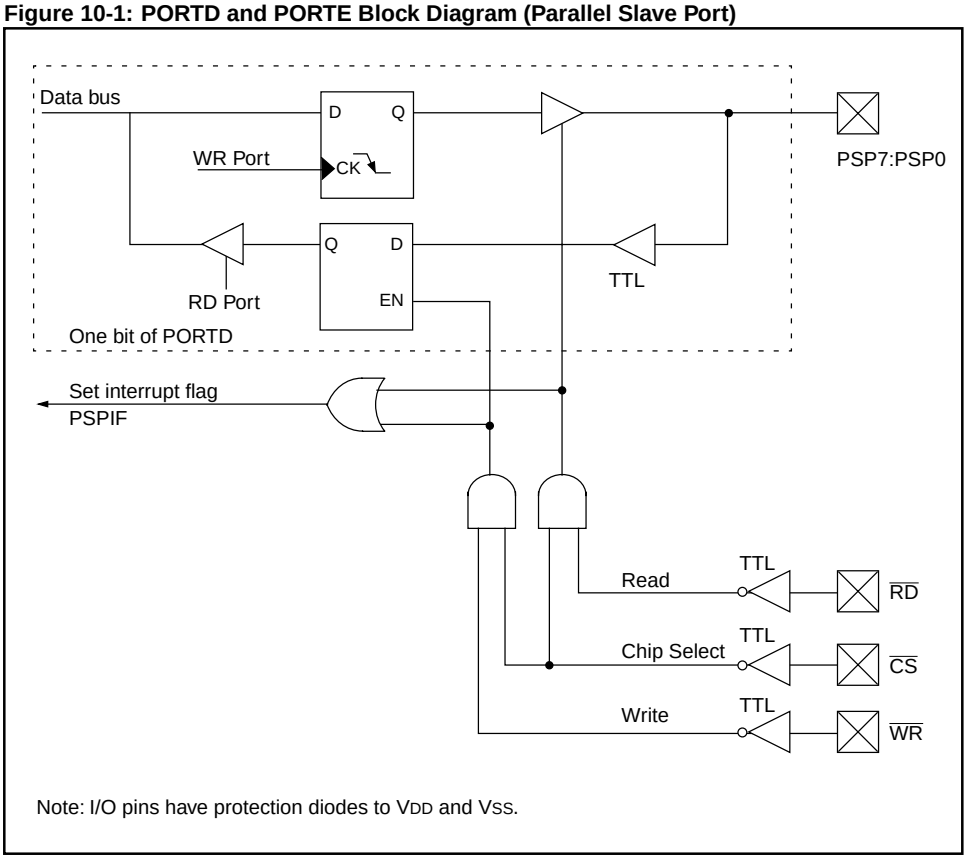
Note 1: At present the Parallel Slave Port (PSP) is only multiplexed onto PORTD and PORTE. The microprocessor port becomes enabled when the PSPMODE bit is set. In this mode, the user must make sure that PORTD and PORTE are configured as digital I/O. That is, peripheral modules multiplexed onto the PSP functions are disabled (such as the A/D).
When PORTE is configured for digital I/O. PORTD will override the values in the TRISD register.

Note 2: In this mode the PORTD and PORTE input buffers are TTL. The control bits for the PSP operation are located in TRISE.

There are actually two 8-bit latches, one for data-out (from the PICmicro) and one for data input. The user writes 8-bit data to PORT data latch and reads data from the port pin latch (note that they have the same address). In this mode, the TRIS register is ignored, since the microprocessor is controlling the direction of data flow.

Figure 10-1 shows the block diagram for the PSP.

Figure 10-1: PORTD and PORTE Block Diagram (Parallel Slave Port)



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

10.2 Control Register

Register 10-1: TRISE Register

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0
bit 7							bit 0

bit 7 **IBF:** Input Buffer Full Status bit

1 = A word has been received and waiting to be read by the CPU
0 = No word has been received

bit 6 **OBF**: Output Buffer Full Status bit

1 = The output buffer still holds a previously written word
0 = The output buffer has been read

bit 5 **IBOV**: Input Buffer Overflow Detect bit (in microprocessor mode)

1 = A write occurred when a previously input word has not been read (must be cleared in software)

0 = No overflow occurred

bit 4 **PSPMODE**: Parallel Slave Port Mode Select bit

1 = Parallel slave port mode
0 = General purpose I/O mode

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

bit 2 **TRISE2**: RE2 direction control bit

1 = Input
0 = Output

bit 1 **TRISE1**: RE1 direction control bit

1 = Input
0 = Output

bit 0 **TRISE0**: RE0 direction control bit

1 = Input
0 = Output

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

10.3 Operation

A write to the PSP from the external system, occurs when both the $\overline{CS}$ and $\overline{WR}$ lines are first detected low. When either the $\overline{CS}$ or $\overline{WR}$ lines become high (edge triggered), the Input Buffer Full status flag bit IBF (TRISE<7>) is set on the Q4 clock cycle, following the next Q2 cycle, to signal the write is complete. The interrupt flag bit, PSPIF, is also set on the same Q4 clock cycle. The IBF flag bit is inhibited from being cleared for additional TCY cycles (see parameter 66). If the IBF flag bit is cleared by reading the PORTD input latch, and this has to be a read-only instruction (i.e., MOVF) and not a read-modify-write instruction. The input Buffer Overflow status flag bit IBOV (TRISE<5>) is set if a second write to the Parallel Slave Port is attempted when the previous byte has not been read out of the buffer.

A read from the PSP from the external system, occurs when both the $\overline{CS}$ and $\overline{RD}$ lines are first detected low. The Output Buffer Full status flag bit OBF (TRISE<6>) is cleared immediately indicating that the PORTD latch was read by the external bus. When either the $\overline{CS}$ or $\overline{RD}$ pin becomes high (edge triggered), the interrupt flag bit PSPIF is set on the Q4 clock cycle, following the next Q2 cycle, indicating that the read is complete. OBF remains low until data is written to PORTD by the user firmware.

Input Buffer Full Status Flag bit IBF, is set if a received word is waiting to be read by the CPU. Once the PORT input latch is read, the IBF bit is cleared. The IBF bit is a read only status bit. Output Buffer Full Status Flag bit OBF, is set if a word written to PORT latch is waiting to be read by the external bus. Once the PORTD output latch is read by the microprocessor, OBF is cleared. Input Buffer Overflow Status Flag bit IBOV is set if a second write to the microprocessor port is attempted when the previous word has not been read by the CPU (the first word is retained in the buffer).

When not in Parallel Slave Port mode, the IBF and OBF bits are held clear. However, if flag bit IBOV was previously set, it must be cleared in the software.

An interrupt is generated and latched into flag bit PSPIF when a read or a write operation is completed. Interrupt flag bit PSPIF must be cleared by user software and the interrupt can be disabled by clearing interrupt enable bit PSPIE.

Table 10-1: PORTE Functions

Name	Function
$\overline{RD}$	Read Control Input in parallel slave port mode: $\overline{RD}$ 1 = Not a read operation 0 = Read operation. Reads PORTD register (if chip selected)
$\overline{WR}$	Write Control Input in parallel slave port mode: $\overline{WR}$ 1 = Not a write operation 0 = Write operation. Writes PORTD register (if chip selected)
$\overline{CS}$	Chip Select Control Input in parallel slave port mode: $\overline{CS}$ 1 = Device is not selected 0 = Device is selected

Note: The PSP may have other functions multiplexed onto the same pins. For the PSP to operate, the pins must be configured as digital I/O.

10.4 Operation in Sleep Mode

When in sleep mode the microprocessor may still read and write the Parallel Slave Port. These actions will set the PSPIF bit. If the PSP interrupts are enabled, this will wake the processor from sleep mode so that the PSP data latch may be either read, or written with the next value for the microprocessor.

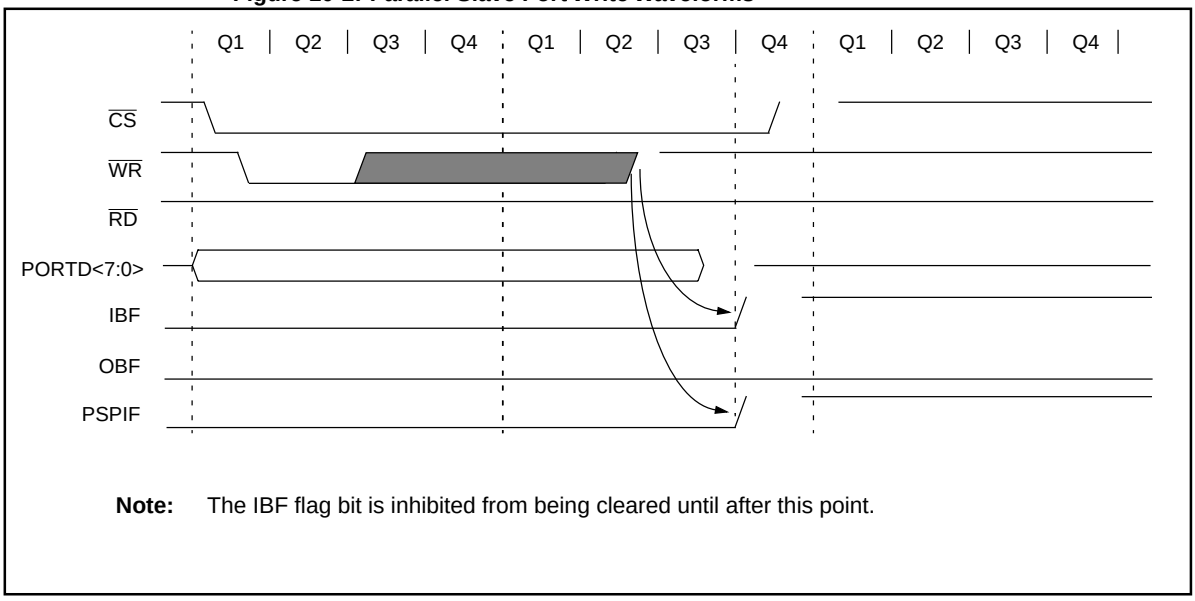
10.5 Effect of a Reset

After any reset the PSP is disabled and PORTD and PORTE are forced to their default mode.

10.6 PSP Waveforms

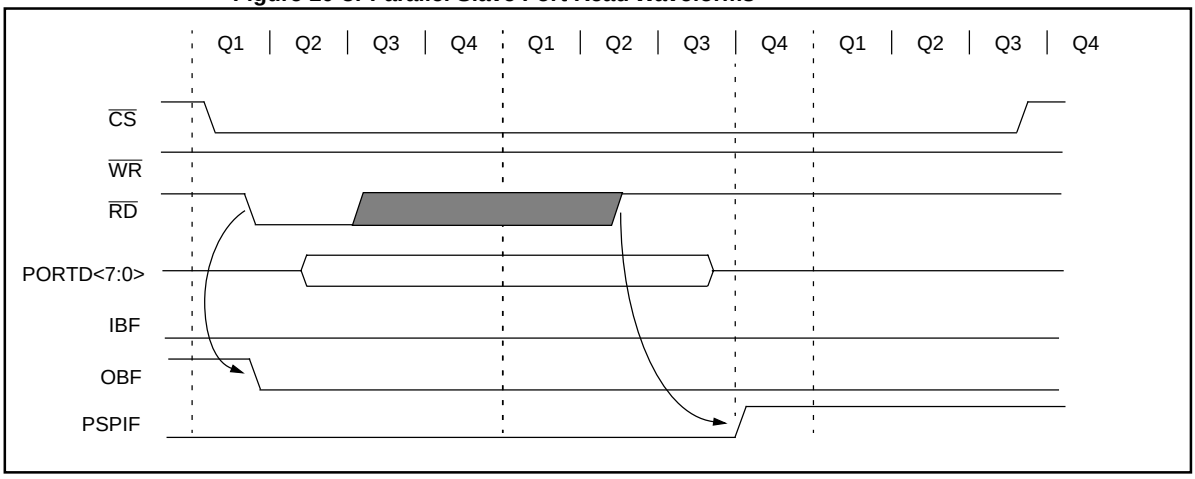
Figure 10-2 shows the waveform for a write from the microprocessor to the PSP, while Figure 10-3 shows the waveform for a read of the PSP by the microprocessor.

Figure 10-2: Parallel Slave Port Write Waveforms



Note: The IBF flag bit is inhibited from being cleared until after this point.

Figure 10-3: Parallel Slave Port Read Waveforms



10.7 Design Tips

Question 1: ***Migrating from the PIC16C74 to the PIC16C74A, the operation of the PSP seems to have changed.***

Answer 1:

Yes, a design change was made so the PIC16C74A is edge sensitive (while the PIC16C74 was level sensitive). See Appendix C.9 for more information.

10.8 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the Parallel Slave Port are:

Title	Application Note #
Using the 8-bit Parallel Slave Port	AN579

10.9 Revision History

Revision A

This is the initial released revision of the Parallel Slave Port description.

Section 11. Timer0

HIGHLIGHTS

This section of the manual contains the following major topics:

11.1 Introduction

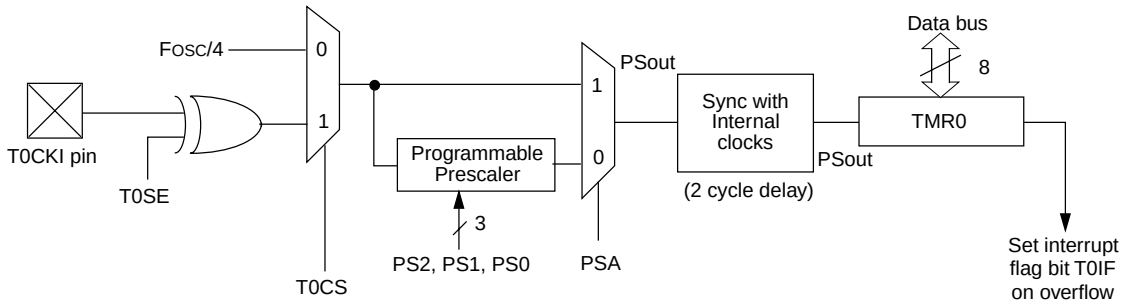
The Timer0 module has the following features:

- 8-bit timer/counter
- Readable and writable
- 8-bit software programmable prescaler
- Clock source selectable to be external or internal
- Interrupt on overflow from FFh to 00h
- Edge select for external clock

Note:	To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

Figure 11-1 shows a simplified block diagram of the Timer0 module.

Figure 11-1: Timer0 Block Diagram



Note 1: T0CS, T0SE, PSA, PS2:PS0 (OPTION_REG<5:0>).
2: The prescaler is shared with Watchdog Timer (refer to Figure 11-6 for detailed block diagram).

11.2 Control Register

The OPTION_REG register is a readable and writable register which contains various control bits to configure the TMR0/WDT prescaler, the External INT Interrupt, TMR0, and the weak pull-ups on PORTB.

Note:	To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

Register 11-1: OPTION_REG Register

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$ (1)	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$ (1):** Weak Pull-up Enable bit
1 = Weak pull-ups are disabled
0 = Weak pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of INT pin
0 = Interrupt on falling edge of INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2:0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

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

Note 1:	Some devices call this bit $\overline{\text{GPPU}}$. Devices that have the $\overline{\text{RBPU}}$ bit, have the weak pull-ups on PORTB, while devices that have the $\overline{\text{GPPU}}$ have the weak pull-ups on the GPIO Port.

11.3 Operation

Timer mode is selected by clearing the T0CS bit (OPTION<5>). In timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0 register is written, the increment is inhibited for the following two instruction cycles (Figure 11-2 and Figure 11-3). The user can work around this by writing an adjusted value to the TMR0 register.

Counter mode is selected by setting the T0CS bit (OPTION<5>). In counter mode, Timer0 will increment either on every rising or falling edge of the T0CKI pin. The incrementing edge is determined by the Timer0 Source Edge Select the T0SE bit (OPTION<4>). Clearing the T0SE bit selects the rising edge. Restrictions on the external clock input are discussed in detail in Subsection **11.5 "Using Timer0 with an External Clock"** .

The prescaler is mutually exclusively shared between the Timer0 module and the Watchdog Timer. The prescaler assignment is controlled in software by the PSA control bit (OPTION<3>). Clearing the PSA bit will assign the prescaler to the Timer0 module. The prescaler is not readable or writable. When the prescaler is assigned to the Timer0 module, prescale values of 1:2, 1:4,..., 1:256 are selectable. Subsection **11.6 "TMR0 Prescaler"** details the operation of the prescaler.

Any write to the TMR0 register will cause a 2 instruction cycle (2TCY) inhibit. That is, after the TMR0 register has been written with the new value, TMR0 will not be incremented until the third instruction cycle later (Figure 11-2). When the prescaler is assigned to the Timer0 module, any write to the TMR0 register will immediately update the TMR0 register and clear the prescaler. The incrementing of Timer0 (TMR0 and Prescaler) will also be inhibited 2 instruction cycles (TCY). So if the prescaler is configured as 2, then after a write to the TMR0 register TMR0 will not increment for 4 Timer0 clocks (Figure 11-3). After that, TMR0 will increment every prescaler number of clocks later.

Figure 11-2: Timer0 Timing: Internal Clock/No Prescale

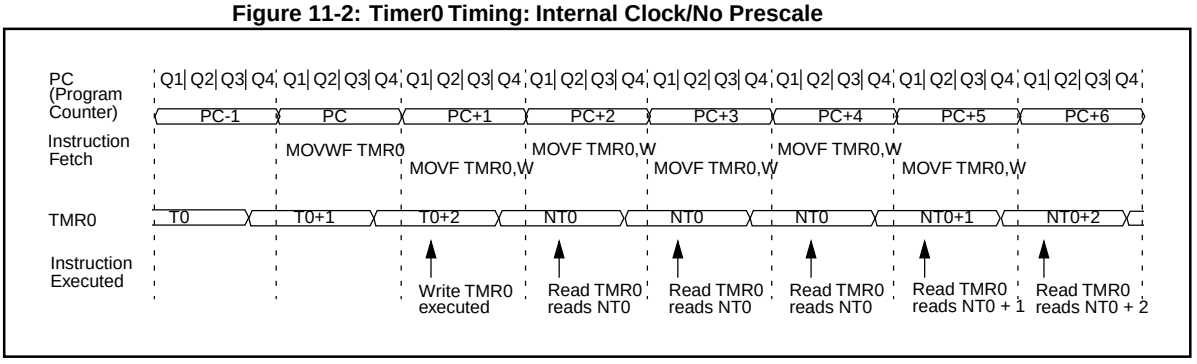
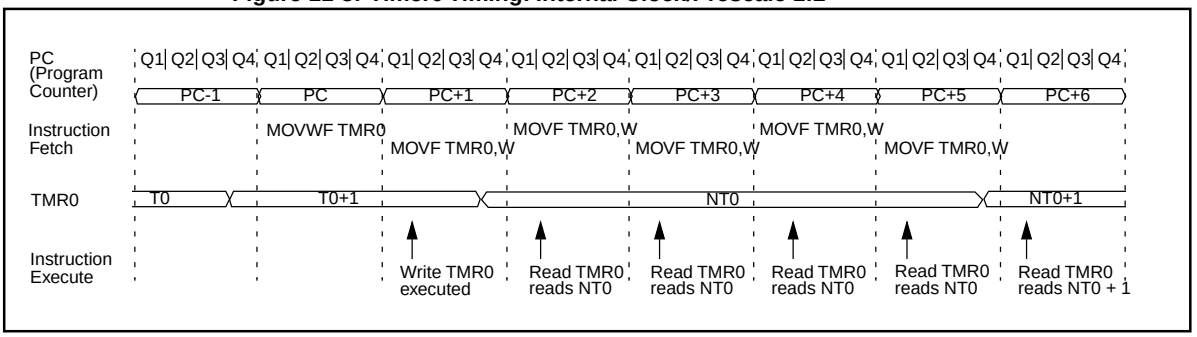


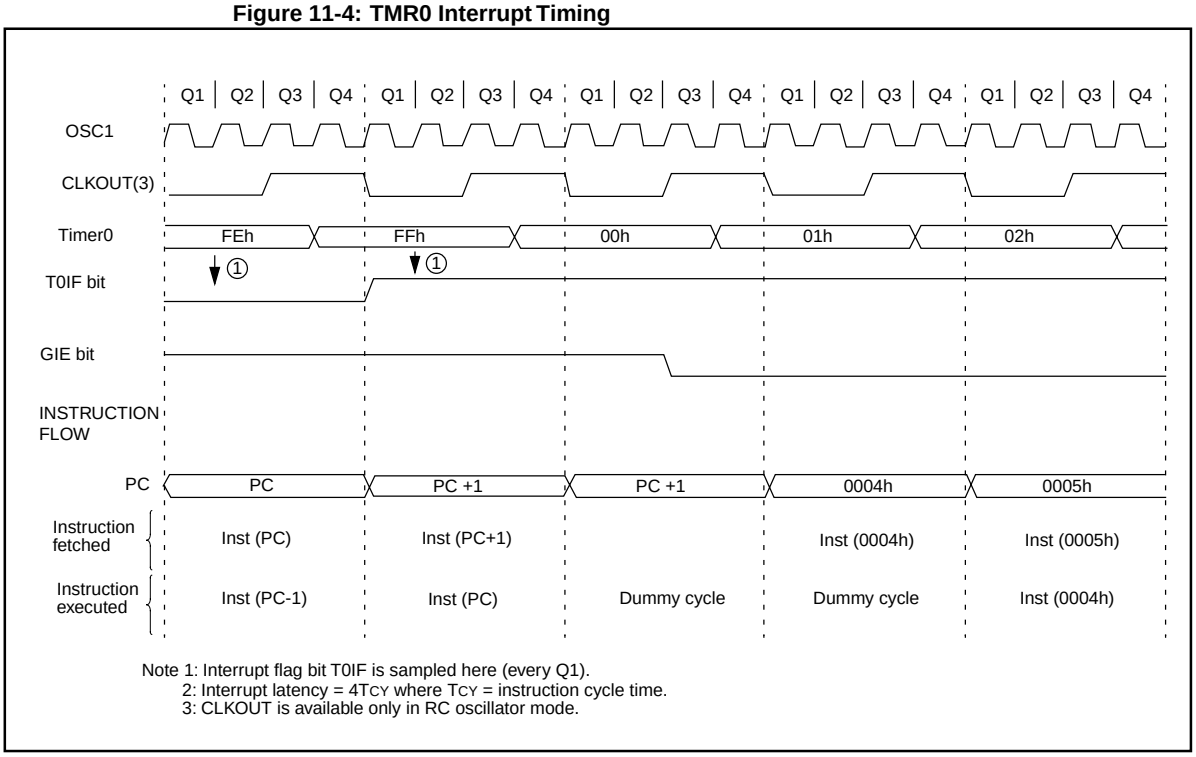
Figure 11-3: Timer0 Timing: Internal Clock/Prescale 1:2



11.4 TMR0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h. This overflow sets bit T0IF (INTCON<2>). The interrupt can be masked by clearing bit T0IE (INTCON<5>). Bit T0IF must be cleared in software by the Timer0 module interrupt service routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from SLEEP since the timer is shut-off during SLEEP. See Figure 11-4 for Timer0 interrupt timing.

Figure 11-4: TMR0 Interrupt Timing



Note 1: Interrupt flag bit T0IF is sampled here (every Q1).
2: Interrupt latency = 4TCY where TCY = instruction cycle time.
3: CLKOUT is available only in RC oscillator mode.

11.5 Using Timer0 with an External Clock

When an external clock input is used for Timer0, it must meet certain requirements as detailed in **11.5.1 "External Clock Synchronization."** These requirements ensure the external clock can be synchronized with the internal phase clock (TOSC). Also, there is a delay in the actual incrementing of Timer0 after synchronization.

11.5.1 External Clock Synchronization

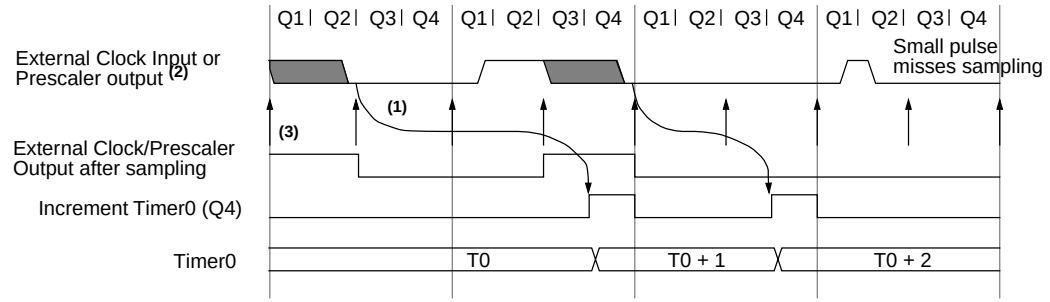
When no prescaler is used, the external clock input is the same as the prescaler output. The synchronization of T0CKI with the internal phase clocks is accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the internal phase clocks (Figure 11-5). Therefore, it is necessary for T0CKI to be high for at least 2Tosc (and a small RC delay of 20 ns) and low for at least 2Tosc (and a small RC delay of 20 ns). Refer to parameters 40, 41 and 42 in the electrical specification of the desired device.

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

11.5.2 TMR0 Increment Delay

Since the prescaler output is synchronized with the internal clocks, there is a small delay from the time the external clock edge occurs to the time the Timer0 module is actually incremented. Figure 11-5 shows the delay from the external clock edge to the timer incrementing.

Figure 11-5: Timer0 Timing with External Clock



Note 1: Delay from clock input change to Timer0 increment is 3Tosc to 7Tosc. (Duration of Q = Tosc). Therefore, the error in measuring the interval between two edges on Timer0 input = ±4Tosc max.
2: External clock if no prescaler selected, Prescaler output otherwise.
3: The arrows indicate the points in time where sampling occurs.

11.6 TMR0 Prescaler

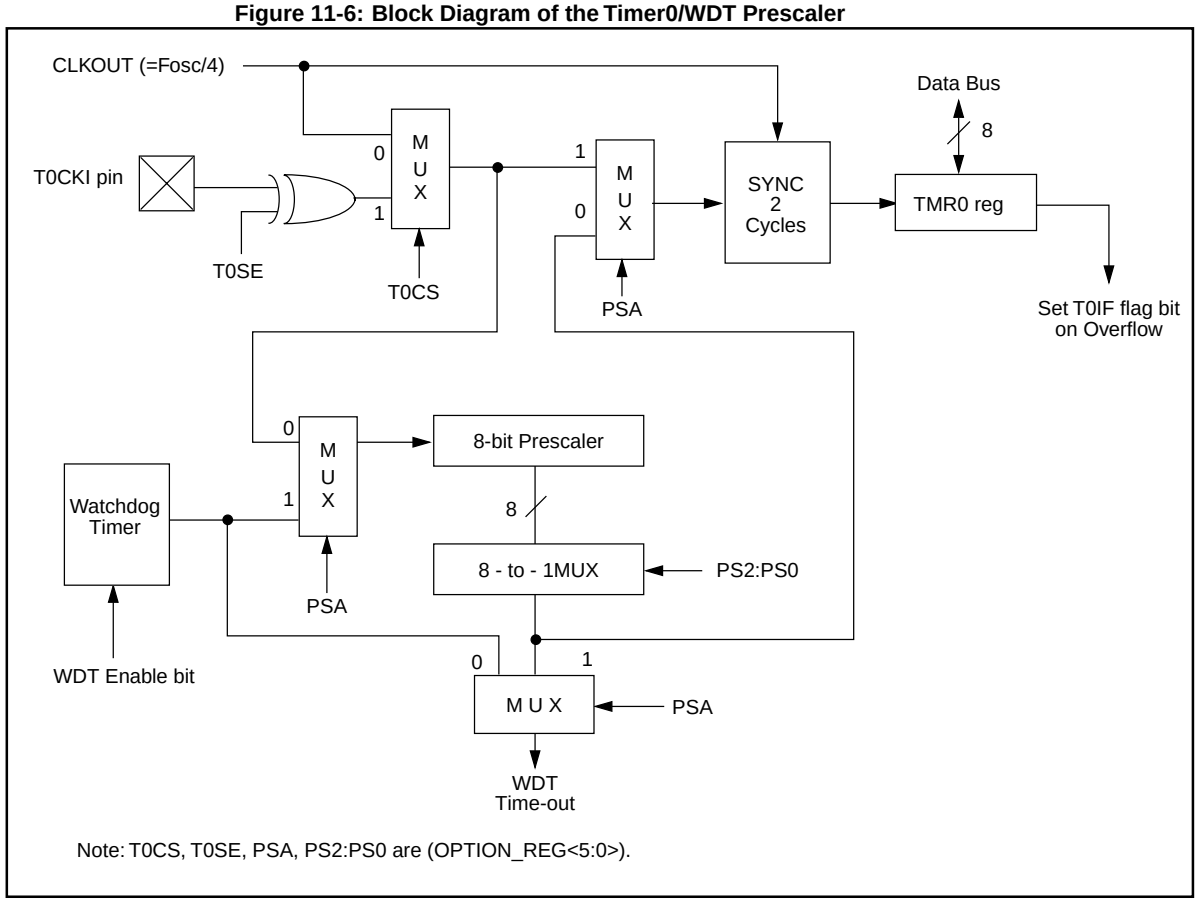
An 8-bit counter is available as a prescaler for the Timer0 module, or as a postscaler for the Watchdog Timer (Figure 11-6). For simplicity, this counter is being referred to as “prescaler” in the Timer0 description. Thus, a prescaler assignment for the Timer0 module means that there is no postscaler for the Watchdog Timer, and vice-versa.

Note:	There is only one prescaler available which is mutually exclusively shared between the Timer0 module and the Watchdog Timer.

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

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., `CLRF TMR0, MOVWF TMR0, BSF TMR0,`x....etc.) will clear the prescaler. When assigned to WDT, a `CLRWDT` instruction will clear the prescaler along with the Watchdog Timer. The prescaler is not readable or writable.

Figure 11-6: Block Diagram of the Timer0/WDT Prescaler



Note: T0CS, T0SE, PSA, PS2:PS0 are (OPTION_REG<5:0>).

11.6.1 Switching Prescaler Assignment

The prescaler assignment is fully under software control, i.e., it can be changed "on the fly" during program execution.

Note:	To avoid an unintended device RESET, the following instruction sequence (shown in Example 11-1) must be executed when changing the prescaler assignment from Timer0 to the WDT. This sequence must be followed even if the WDT is disabled.

In Example 11-1, the first modification of the OPTION_REG does not need to be included if the final desired prescaler is other then 1:1. If the final prescaler value is to be 1:1, then a temporary prescale value is set (other than 1:!), and the final prescale value is set in the last modification of OPTION_REG.

Example 11-1: Changing Prescaler (Timer0→WDT)

Lines 2 and 3 do NOT have to be included if the final desired prescale value is other than 1:1. If 1:1 is final desired value, then a temporary prescale value is set in lines 2 and 3 and the final prescale value will be set in lines 10 and 11.

```
1)  BSF    STATUS, RP0   ;Bank1
2)  MOVLW  b'xx0x0xxx'   ;Select clock source and prescale value of
3)  MOVWF  OPTION_REG    ;other than 1:1
4)  BCF    STATUS, RP0   ;Bank0
5)  CLRF   TMR0          ;Clear TMR0 and prescaler
6)  BSF    STATUS, RP1   ;Bank1
7)  MOVLW  b'xxxx1xxx'   ;Select WDT, do not change prescale value
8)  MOVWF  OPTION_REG    ;
9)  CLRWDT               ;Clears WDT and prescaler
10) MOVLW  b'xxxx1xxx'   ;Select new prescale value and WDT
11) MOVWF  OPTION_REG    ;
12) BCF    STATUS, RP0   ;Bank0
```

To change prescaler from the WDT to the Timer0 module use the sequence shown in Example 11-2.

Example 11-2: Changing Prescaler (WDT→Timer0)

```
    CLRWDT                  ; Clear WDT and prescaler
    BSF      STATUS, RP0    ; Bank1
    MOVLW    b'xxxx0xxx'    ; Select TMR0, new prescale
    MOVWF    OPTION_REG     ; value and clock source
    BCF      STATUS, RP0    ; Bank0
```

11.6.2 Initialization

Since Timer0 has a software programmable clock source, there are two examples to show the initialization of Timer0 with each source. Example 11-3 shows the initialization for the internal clock source (timer mode), while Example 11-4 shows the initialization for the external clock source (counter mode).

Example 11-3: Timer0 Initialization (Internal Clock Source)

```
        CLRF    TMR0            ; Clear Timer0 register
        CLRF    INTCON          ; Disable interrupts and clear T0IF
        BSF     STATUS, RP0     ; Bank1
        MOVLW   0xC3            ; PortB pull-ups are disabled,
        MOVWF   OPTION_REG      ;   Interrupt on rising edge of RB0
                                ;   Timer0 increment from internal clock
                                ;   with a prescaler of 1:16.
        BCF     STATUS, RP0     ; Bank0
;**     BSF     INTCON, T0IE    ; Enable TMR0 interrupt
;**     BSF     INTCON, GIE     ; Enable all interrupts
;
; The TMR0 interrupt is disabled, do polling on the overflow bit
;
T0_OVFL_WAIT
        BTFSS   INTCON, T0IF
        GOTO    T0_OVFL_WAIT
; Timer has overflowed
```

Example 11-4: Timer0 Initialization (External Clock Source)

```
    CLRF    TMR0            ; Clear Timer0 register
    CLRF    INTCON          ; Disable interrupts and clear T0IF
    BSF     STATUS, RP0     ; Bank1
    MOVLW   0x37            ; PortB pull-ups are enabled,
    MOVWF   OPTION_REG      ;   Interrupt on falling edge of RB0
                            ;   Timer0 increment from external clock
                            ;   on the high-to-low transition of T0CKI
                            ;   with a prescaler of 1:256.
    BCF     STATUS, RP0     ; Bank0
;** BSF     INTCON, T0IE    ; Enable TMR0 interrupt
;** BSF     INTCON, GIE     ; Enable all interrupts
;
; The TMR0 interrupt is disabled, do polling on the overflow bit
;
T0_OVFL_WAIT
    BTFSS   INTCON, T0IF
    GOTO    T0_OVFL_WAIT
; Timer has overflowed
```

11.7 Design Tips

Question 1: ***I am implementing a counter/clock, but the clock loses time or is inaccurate.***

Answer 1:

If you are polling TMR0 to see if it has rolled over to zero. You could do this by executing:

```
wait    MOVF    TMR0,W      ; read the timer into W
        BTFSS   STATUS,Z    ; see if it was zero, if so,
                            ;    break from loop
        GOTO     wait       ; if not zero yet, keep waiting
```

Two possible scenarios to lose clock cycles are:

1. If you are incrementing TMR0 from the internal instruction clock, or an external source that is about as fast, the overflow could occur during the two cycle GOTO, so you could miss it. In this case the TMR0 source should be prescaled.
   Or you could do a test to see if it has rolled over by checking for less than a nominal value:

   ```
   Wait    movlw  3
           subwf  TMR0,W
           btfsc  STATUS,C
           goto   Wait
   ```

2. When writing to TMR0, two instruction clock cycles are lost. Often you have a specific time period you want to count, say 100 decimal. In that case you might put 156 into TMR0 (256 - 100 = 156). However, since two instruction cycles are lost when you write to TMR0 (for internal logic synchronization), you should actually write 158 to the timer.

11.8 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to Timer0 are:

Title	Application Note #
Frequency Counter Using PIC16C5X	AN592
A Clock Design using the PIC16C54 for LED Display and Switch Inputs	AN590

11.9 Revision History

Revision A

This is the initial released revision of the Timer0 Module description.

Section 12. Timer1

HIGHLIGHTS

This section of the manual contains the following major topics:

12.1 Introduction

The Timer1 module is a 16-bit timer/counter consisting of two 8-bit registers (TMR1H and TMR1L) which are readable and writable. The TMR1 Register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The Timer1 Interrupt, if enabled, is generated on overflow which is latched in the TMR1IF interrupt flag bit. This interrupt can be enabled/disabled by setting/clearing the TMR1IE interrupt enable bit.

Timer1 can operate in one of three modes:

- As a synchronous timer
- As a synchronous counter
- As an asynchronous counter

The operating mode is determined by clock select bit, TMR1CS (T1CON<1>), and the synchronization bit, $\overline{\text{T1SYNC}}$ (Figure 12-1).

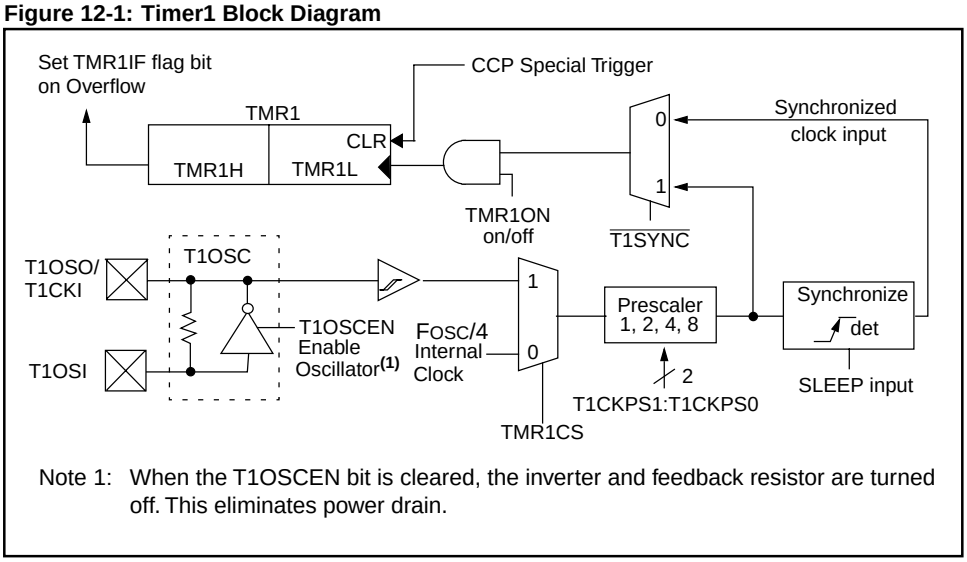
In timer mode, Timer1 increments every instruction cycle. In counter mode, it increments on every rising edge of the external clock input on pin T1CKI.

Timer1 can be turned on and off using theTMR1ON control bit (T1CON<0>).

Timer1 also has an internal “reset input”, which can be generated by a CCP module.

Timer1 has the capability to operate off an external crystal. When the Timer1 oscillator is enabled (T1OSCEN is set), the T1OSI and T1OSO pins become inputs. That is, their corresponding TRIS values are ignored.

Figure 12-1: Timer1 Block Diagram



Note 1: When the T1OSCEN bit is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

12.2 Control Register

Register 12-1 shows the Timer1 control register.

Register 12-1: T1CON: Timer1 Control Register

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

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

bit 5:4 **T1CKPS1:T1CKPS0**: Timer1 Input Clock Prescale Select bits

11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN**: Timer1 Oscillator Enable bit

1 = Oscillator is enabled
0 = Oscillator is shut off. The oscillator inverter and feedback resistor are turned off to eliminate power drain

bit 2 **$\overline{\text{T1SYNC}}$**: Timer1 External Clock Input Synchronization Select bit

When TMR1CS = 1:

1 = Do not synchronize external clock input
0 = Synchronize external clock input

When TMR1CS = 0:

This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS**: Timer1 Clock Source Select bit

1 = External clock from pin T1OSO/T1CKI (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR1ON**: Timer1 On bit

1 = Enables Timer1
0 = Stops Timer1

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

12.3 Timer1 Operation in Timer Mode

Timer mode is selected by clearing the TMR1CS (T1CON<1>) bit. In this mode, the input clock to the timer is FOSC/4. The synchronize control bit, $\overline{\text{T1SYNC}}$ (T1CON<2>), has no effect since the internal clock is always synchronized.

12.4 Timer1 Operation in Synchronized Counter Mode

Counter mode is selected by setting the TMR1CS bit. In this mode the timer increments on every rising edge of clock input on the T1OSI pin when the oscillator enable bit (T1OSCEN) is set, or the T1OSO/T1CKI pin when the T1OSCEN bit is cleared.

If the $\overline{\text{T1SYNC}}$ bit is cleared, then the external clock input is synchronized with internal phase clocks. The synchronization is done after the prescaler stage. The prescaler is an asynchronous ripple-counter.

In this configuration, during SLEEP mode, Timer1 will not increment even if the external clock is present, since the synchronization circuit is shut off. The prescaler however will continue to increment.

12.4.1 External Clock Input Timing for Synchronized Counter Mode

When an external clock input is used for Timer1 in synchronized counter mode, it must meet certain requirements. The external clock requirement is due to internal phase clock (Tosc) synchronization. Also, there is a delay in the actual incrementing of TMR1 after synchronization.

When the prescaler is 1:1, the external clock input is the same as the prescaler output. The synchronization of T1CKI with the internal phase clocks is accomplished by sampling the prescaler output on alternating Tosc clocks of the internal phase clocks. Therefore, it is necessary for the T1CKI pin to be high for at least 2Tosc (and a small RC delay) and low for at least 2Tosc (and a small RC delay). Refer to parameters 45, 46, and 47 in the **"Electrical Specifications"** section.

When a prescaler other than 1:1 is used, the external clock input is divided by the asynchronous ripple-counter prescaler so that the prescaler output is symmetrical. In order for the external clock to meet the sampling requirement, the ripple-counter must be taken into account. Therefore, it is necessary for the T1CKI pin to have a period of at least 4Tosc (and a small RC delay) divided by the prescaler value. Another requirement on the T1CKI pin high and low time is that they do not violate the minimum pulse width requirements). Refer to parameters 40, 42, 45, 46, and 47 in the **"Electrical Specifications"** section.

12.5 Timer1 Operation in Asynchronous Counter Mode

If $\overline{\text{T1SYNC}}$ (T1CON<2>) is set, the external clock input is not synchronized. The timer continues to increment asynchronously to the internal phase clocks. The timer will continue to run during SLEEP and can generate an interrupt on overflow which will wake-up the processor. However, special precautions in software are needed to read/write the timer (Subsection **12.5.2 "Reading and Writing Timer1 in Asynchronous Counter Mode"**). Since the counter can operate in sleep, Timer1 can be used to implement a true real-time clock.

In asynchronous counter mode, Timer1 cannot be used as a time-base for capture or compare operations.

12.5.1 External Clock Input Timing with Unsynchronized Clock

If the $\overline{\text{T1SYNC}}$ control bit is set, the timer will increment completely asynchronously. The input clock must meet certain minimum high time and low time requirements. Refer to the Device Data Sheet Electrical Specifications Section, timing parameters 45, 46, and 47.

12.5.2 Reading and Writing Timer1 in Asynchronous Counter Mode

Reading TMR1H or TMR1L while the timer is running from an external asynchronous clock, will guarantee a valid read (taken care of in hardware). However, the user should keep in mind that reading the 16-bit timer in two 8-bit values itself poses certain problems since the timer may overflow between the reads.

For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers while the register is incrementing. This may produce an unpredictable value in the timer register.

Reading the 16-bit value requires some care, since two separate reads are required to read the entire 16-bits. Example 12-1 shows why this may not be a straight forward read of the 16-bit register.

Example 12-1: Reading 16-bit Register Issues

TMR1	Sequence 1		Sequence 2	
	Action	TMPH:TMPL	Action	TMPH:TMPL
04FFh	READ TMR1L	xxxxh	READ TMR1H	xxxxh
0500h	Store in TMPL	xxFFh	Store in TMPH	04xxh
0501h	READ TMR1H	xxFFh	READ TMR1L	04xxh
0502h	Store in TMPH	05FFh	Store in TMPL	0401h

Example 12-2 shows a routine to read the 16-bit timer value with experiencing the issues shown in Example 12-1. This is useful if the timer cannot be stopped.

Example 12-2: Reading a 16-bit Free-Running Timer

```
; All interrupts are disabled
       MOVF   TMR1H, W    ; Read high byte
       MOVWF  TMPH        ;
       MOVF   TMR1L, W    ; Read low byte
       MOVWF  TMPL        ;
       MOVF   TMR1H, W    ; Read high byte
       SUBWF  TMPH,  W    ; Sub 1st read with 2nd read
       BTFSC  STATUS,Z    ; Is result = 0
       GOTO   CONTINUE    ; Good 16-bit read
;
; TMR1L may have rolled over between the read of the high and low bytes.
; Reading the high and low bytes now will read a good value.
;
       MOVF   TMR1H, W    ; Read high byte
       MOVWF  TMPH        ;
       MOVF   TMR1L, W    ; Read low byte
       MOVWF  TMPL        ;
; Re-enable the Interrupt (if required)
CONTINUE                  ; Continue with your code
```

Writing a 16-bit value to the 16-bit TMR1 register is straight forward. First the TMR1L register is cleared to ensure that there are many Timer1 clock/oscillator cycles before there is a rollover into the TMR1H register. The TMR1H register is then loaded, and finally the TMR1L register is loaded. Example 12-3 shows this:

Example 12-3: Writing a 16-bit Free Running Timer

```
; All interrupts are disabled
       CLRF   TMR1L       ; Clear Low byte, Ensures no
                          ;   rollover into TMR1H
       MOVLW  HI_BYTE     ; Value to load into TMR1H
       MOVWF  TMR1H, F    ; Write High byte
       MOVLW  LO_BYTE     ; Value to load into TMR1L
       MOVWF  TMR1H, F    ; Write Low byte
; Re-enable the Interrupt (if required)
CONTINUE                  ; Continue with your code
```

12.6 Timer1 Oscillator

A crystal oscillator circuit is built in between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting the T1OSCEN control bit (T1CON<3>). The oscillator is a low power oscillator, rated up to 200 kHz operation. It will continue to run during SLEEP. It is primarily intended for a 32 kHz crystal, which is an ideal frequency for real-time keeping. Table 12-1 shows the capacitor selection for the Timer1 oscillator.

The Timer1 oscillator is identical to the LP oscillator. The user must provide a software time delay to ensure proper oscillator start-up.

Note:	This allows the counter to operate (increment) when the device is in sleep mode, which allows Timer1 to be used as a real-time clock.

Table 12-1: Capacitor Selection for the Timer1 Oscillator

Osc Type	Freq	C1	C2
LP	32 kHz	33 pF	33 pF
	100 kHz	15 pF	15 pF
	200 kHz	15 pF	15 pF
Crystals Tested:			
32.768 kHz	Epson C-001R32.768K-A		± 20 PPM
100 kHz	Epson C-2 100.00 KC-P		± 20 PPM
200 kHz	STD XTL 200.000 kHz		± 20 PPM

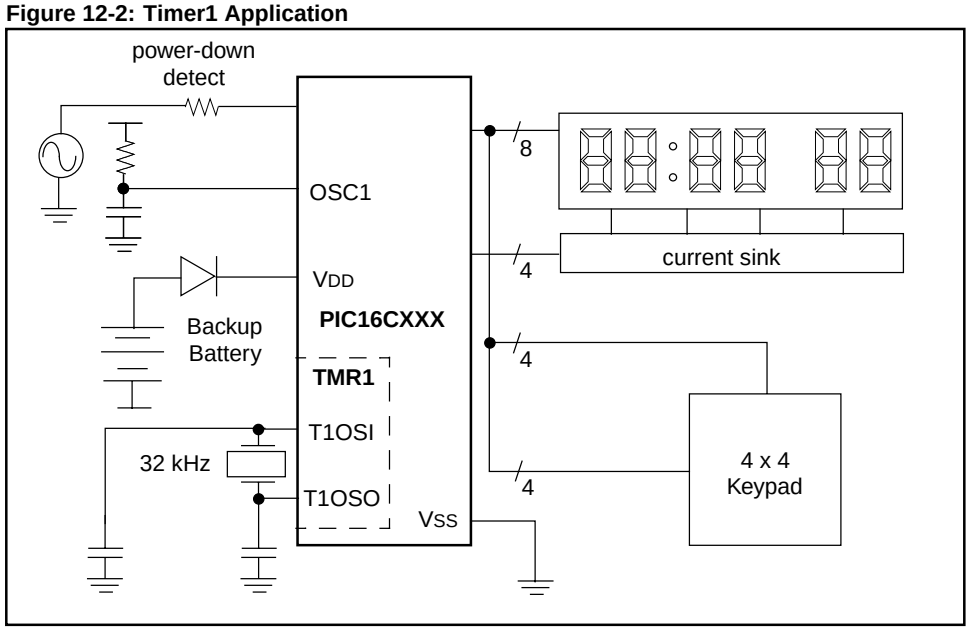
Note 1: Higher capacitance increases the stability of oscillator but also increases the start-up time.

2: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

12.6.1 Typical Application

This feature is typically used in applications where real-time needs to be kept, but it is also desirable to have the lowest possible power consumption. The Timer1 oscillator allows the device to be placed in sleep, while the timer continues to increment. When Timer1 overflows the interrupt could wake-up the device so that the appropriate registers could be updated.

Figure 12-2: Timer1 Application



12.7 Sleep Operation

When Timer1 is configured for asynchronous operation, the TMR1 registers will continue to increment for each timer clock (or prescale multiple of clocks). When the TMR1 register overflows, the TMR1IF bit will get set, and if enabled generate an interrupt that will wake the processor from sleep mode.

The Timer1 oscillator will add a delta current, due to the operation of this circuitry. That is, the power-down current will no longer only be the leakage current of the device, but also the active current of the Timer1 oscillator and other Timer1 circuitry.

12.8 Resetting Timer1 Using a CCP Trigger Output

If a CCP module is configured in compare mode to generate a "special event trigger" (CCP1M3:CCP1M0 = 1011), this signal resets Timer1.

Note: The special event trigger from the CCP module does not set interrupt flag bit TMR1IF.

Timer1 must be configured for either timer or synchronized counter mode to take advantage of the special event trigger feature. If Timer1 is running in asynchronous counter mode, this reset operation may not work, and should not be used.

In the event that a write to Timer1 coincides with a special event trigger from the CCP module, the write will take precedence.

In this mode of operation, the CCPRxH:CCPRxL register pair effectively becomes the period register for Timer1.

12.9 Resetting of Timer1 Register Pair (TMR1H:TMR1L)

TMR1H and TMR1L registers are not reset on a POR or any other reset, only by the CCP special event triggers.

T1CON register is reset to 00h on a Power-on Reset or a Brown-out Reset. In any other reset, the register is unaffected.

12.10 Timer1 Prescaler

The prescaler counter is cleared on writes to the TMR1H or TMR1L registers.

Table 12-2: Registers Associated with Timer1 as a Timer/Counter

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
INTCON	GIE	PEIE	T0IE	INTE	RBIE(2)	T0IF	INTF	RBIF(2)	0000 000x	0000 000u
PIR	TMR1IF (1)								0	0
PIE	TMR1IE (1)								0	0
TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 register								xxxx xxxx	uuuu uuuu
T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'.
Shaded cells are not used by the Timer1 module.

Note 1: The placement of this bit is device dependent.
2: These bits may also be named GPIE and GPIF.

12.11 Initialization

Since Timer1 has a software programmable clock source, there are three examples to show the initialization of each mode. Example 12-4 shows the initialization for the internal clock source, Example 12-5 shows the initialization for the external clock source, and Example 12-6 shows the initialization of the external oscillator mode.

Example 12-4: Timer1 Initialization (Internal Clock Source)

```
    CLRF   T1CON          ; Stop Timer1, Internal Clock Source,
                          ;   T1 oscillator disabled, prescaler = 1:1
    CLRF   TMR1H          ; Clear Timer1 High byte register
    CLRF   TMR1L          ; Clear Timer1 Low byte register
    CLRF   INTCON         ; Disable interrupts
    BSF    STATUS, RP0    ; Bank1
    CLRF   PIE1           ; Disable peripheral interrupts
    BCF    STATUS, RP0    ; Bank0
    CLRF   PIR1           ; Clear peripheral interrupts Flags
    MOVLW  0x30           ; Internal Clock source with 1:8 prescaler
    MOVWF  T1CON          ;   Timer1 is stopped and T1 osc is disabled
    BSF    T1CON, TMR1ON  ; Timer1 starts to increment
;
; The Timer1 interrupt is disabled, do polling on the overflow bit
;
T1_OVFL_WAIT
    BTFSS  PIR1, TMR1IF
    GOTO   T1_OVFL_WAIT
;
; Timer has overflowed
;
    BCF    PIR1, TMR1IF
```

Example 12-5: Timer1 Initialization (External Clock Source)

```
    CLRF   T1CON          ; Stop Timer1, Internal Clock Source,
                          ;   T1 oscillator disabled, prescaler = 1:1
    CLRF   TMR1H          ; Clear Timer1 High byte register
    CLRF   TMR1L          ; Clear Timer1 Low byte register
    CLRF   INTCON         ; Disable interrupts
    BSF    STATUS, RP0    ; Bank1
    CLRF   PIE1           ; Disable peripheral interrupts
    BCF    STATUS, RP0    ; Bank0
    CLRF   PIR1           ; Clear peripheral interrupts Flags
    MOVLW  0x32           ; External Clock source with 1:8 prescaler
    MOVWF  T1CON          ;   Clock source is synchronized to device
                          ;   Timer1 is stopped and T1 osc is disabled
    BSF    T1CON, TMR1ON  ; Timer1 starts to increment
;
; The Timer1 interrupt is disabled, do polling on the overflow bit
;
T1_OVFL_WAIT
    BTFSS  PIR1, TMR1IF
    GOTO   T1_OVFL_WAIT
;
; Timer has overflowed
;
    BCF    PIR1, TMR1IF
```

Example 12-6: Timer1 Initialization (External Oscillator Clock Source)

```
    CLRF   T1CON         ; Stop Timer1, Internal Clock Source,
                         ;   T1 oscillator disabled, prescaler = 1:1
    CLRF   TMR1H         ; Clear Timer1 High byte register
    CLRF   TMR1L         ; Clear Timer1 Low byte register
    CLRF   INTCON        ; Disable interrupts
    BSF    STATUS, RP0   ; Bank1
    CLRF   PIE1          ; Disable peripheral interrupts
    BCF    STATUS, RP0   ; Bank0
    CLRF   PIR1          ; Clear peripheral interrupts Flags
    MOVLW  0x3E          ; External Clock source with oscillator
    MOVWF  T1CON         ;   circuitry, 1:8 prescaler, Clock source
                         ;   is asynchronous to device
                         ;   Timer1 is stopped
    BSF    T1CON, TMR1ON ; Timer1 starts to increment
;
; The Timer1 interrupt is disabled, do polling on the overflow bit
;
T1_OVFL_WAIT
    BTFSS  PIR1, TMR1IF
    GOTO   T1_OVFL_WAIT
;
; Timer has overflowed
;
    BCF    PIR1, TMR1IF
```

12.12 Design Tips

Question 1: ***Timer1 does not seem to be keeping accurate time.***

Answer 1:

There are a few reasons that this could occur

1. You should never write to Timer1, where that could cause the loss of time. In most cases that means you should not write to the TMR1L register, but if the conditions are ok, you may write to the TMR1H register. Normally you write to the TMR1H register if you want the Timer1 overflow interrupt to be sooner then the full 16-bit time-out.
2. You should ensure the your layout uses good PCB layout techniques so that noise does not couple onto the Timer1 oscillator lines.

12.13 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to Timer1 are:

Title	Application Note #
Using Timer1 in Asynchronous Clock Mode	AN580
Low Power Real Time Clock	AN582
Yet another Clock using the PIC16C92X	AN649

12.14 Revision History

Revision A

This is the initial released revision of the Timer1 module description.

Section 13. Timer2

HIGHLIGHTS

This section of the manual contains the following major topics:

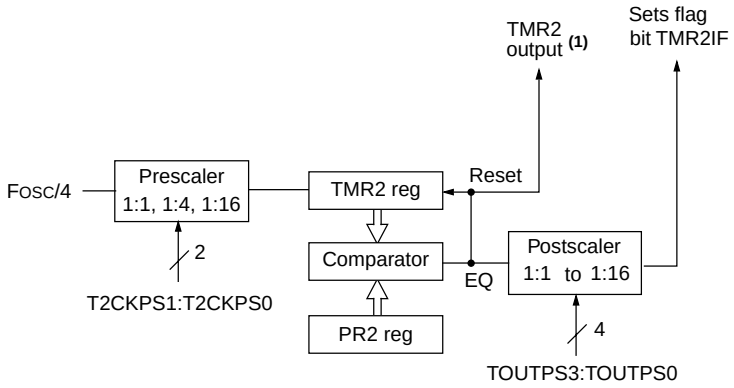
13.1 Introduction

Timer2 is an 8-bit timer with a prescaler, a postscaler, and a period register. Using the prescaler and postscaler at their maximum settings, the overflow time is the same as a 16-bit timer.

Timer2 is the PWM time-base when the CCP module(s) is used in the PWM mode.

Figure 13-1 shows a block diagram of Timer2. The postscaler counts the number of times that the TMR2 register matched the PR2 register. This can be useful in reducing the overhead of the interrupt service routine on the CPU performance.

Figure 13-1: Timer2 Block Diagram



Note: TMR2 register output can be software selected by the SSP Module as a baud clock.

13.2 Control Register

Register 13-1 shows the Timer2 control register.

Register 13-1: T2CON: Timer2 Control Register

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

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

bit 6:3 **TOUTPS3:TOUTPS0**: Timer2 Output Postscale Select bits

0000 = 1:1 Postscale
0001 = 1:2 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit

1 = Timer2 is on
0 = Timer2 is off

bit 1:0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits

00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

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

13.3 Timer Clock Source

The Timer2 module has one source of input clock, the device clock (FOSC/4). A prescale option of 1:1, 1:4 or 1:16 is software selected by control bits T2CKPS1:T2CKPS0 (T2CON<1:0>).

13.4 Timer (TMR2) and Period (PR2) Registers

The TMR2 register is readable and writable, and is cleared on all device resets. Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register.

TMR2 is cleared when a WDT, POR, $\overline{\text{MCLR}}$, or a BOR reset occurs, while the PR2 register is set.

Timer2 can be shut off (disabled from incrementing) by clearing the TMR2ON control bit (T2CON<2>). This minimizes the power consumption of the module.

13.5 TMR2 Match Output

The match output of TMR2 goes to two sources:

1. Timer2 Postscaler
2. SSP Clock Input

There are four bits which select the postscaler. This allows the postscaler a 1:1 to 1:16 scaling (inclusive). After the postscaler overflows, the TMR2 interrupt flag bit (TMR2IF) is set to indicate the Timer2 overflow. This is useful in reducing the software overhead of the Timer2 interrupt service routine, since it will only execute once every postscaler # of matches.

The match output of TMR2 is also routed to the Synchronous Serial Port module, which may software select this as the clock source for the shift clock.

13.6 Clearing the Timer2 Prescaler and Postscaler

The prescaler and postscaler counters are cleared when any of the following occurs:

- a write to the TMR2 register
- a write to the T2CON register

  Note: When T2CON is written TMR2 does not clear.
- any device reset (Power-on Reset, $\overline{\text{MCLR}}$ reset, Watchdog Timer Reset, Brown-out Reset, or Parity Error Reset)

13.7 Sleep Operation

During sleep, TMR2 will not increment. The prescaler will retain the last prescale count, ready for operation to resume after the device wakes from sleep.

Table 13-1: Registers Associated with Timer2

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR, PER	Value on all other resets
INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
PIR	TMR2IF (1)								0	0
PIE	TMR2IE (1)								0	0
TMR2	Timer2 module's register								0000 0000	0000 0000
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
PR2	Timer2 Period Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'.
Shaded cells are not used by the Timer2 module.

Note 1: The position of this bit is device dependent.

13.8 Initialization

Example 13-1 shows how to initialize the Timer2 module, including specifying the Timer2 prescaler and postscaler.

Example 13-1: Timer2 Initialization

```
       CLRF   T2CON         ; Stop Timer2, Prescaler = 1:1,
                            ;   Postscaler = 1:1
       CLRF   TMR2          ; Clear Timer2 register
       CLRF   INTCON        ; Disable interrupts
       BSF    STATUS, RP0   ; Bank1
       CLRF   PIE1          ; Disable peripheral interrupts
       BCF    STATUS, RP0   ; Bank0
       CLRF   PIR1          ; Clear peripheral interrupts Flags
       MOVLW  0x72          ; Postscaler = 1:15, Prescaler = 1:16
       MOVWF  T2CON         ;   Timer2 is off
       BSF    T2CON, TMR2ON ; Timer2 starts to increment
;
; The Timer2 interrupt is disabled, do polling on the overflow bit
;
T2_OVFL_WAIT
       BTFSS  PIR1, TMR2IF   ; Has TMR2 interrupt occurred?
       GOTO   T2_OVFL_WAIT   ; NO, continue loop
;
; Timer has overflowed
;
       BCF    PIR1, TMR2IF   ; YES, clear flag and continue.
```

13.9 Design Tips

No related Design Tips at this time.

13.10 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the Timer2 Module are:

Title	Application Note #
Using the CCP Module	AN594
Air Flow Control using Fuzzy Logic	AN600
Adaptive Differential Pulse Code Modulation using PICmicros	AN643

13.11 Revision History

Revision A

This is the initial released revision of the TImer2 module description.

Section 14. Compare/Capture/PWM (CCP)

HIGHLIGHTS

This section of the manual contains the following major topics:

14.1 Introduction

Each CCP (Capture/Compare/PWM) module contains a 16-bit register which can operate as a 16-bit capture register, as a 16-bit compare register or as a 10-bit PWM master/slave Duty Cycle register. The CCP modules are identical in operation, with the exception of the operation of the special event trigger.

Each CCP module has 3 registers. Multiple CCP modules may exist on a single device. Throughout this section we use generic names for the CCP registers. These generic names are shown in Table 14-1.

Table 14-1: Specific to Generic CCP Nomenclature

Generic Name	CCP1	CCP2	Comment
CCPxCON	CCP1CON	CCP2CON	CCP control register
CCPRxH	CCPR1H	CCPR2H	CCP High byte
CCPRxL	CCPR1L	CCPR2L	CCP Low byte
CCPx	CCP1	CCP2	CCP pin

Table 14-2 shows the resources of the CCP modules, in each of its modes. While Table 14-3 shows the interactions between the CCP modules, where CCPx is one CCP module and CCPy is another CCP module.

Table 14-2: CCP Mode - Timer Resource

CCP Mode	Timer Resource
Capture	Timer1
Compare	Timer1
PWM	Timer2

Table 14-3: Interaction of Two CCP Modules

CCPx Mode	CCPy Mode	Interaction
Capture	Capture	Same TMR1 time-base.
Capture	Compare	The compare should be configured for the special event trigger, which clears TMR1.
Compare	Compare	The compare(s) should be configured for the special event trigger, which clears TMR1.
PWM	PWM	The PWMs will have the same frequency, and update rate (TMR2 interrupt).
PWM	Capture	None
PWM	Compare	None

14.2 Control Register

Register 14-1: CCPxCON Register

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

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

bit 5:4 **DCxB1:DCxB0**: PWM Duty Cycle bit1 and bit0

Capture Mode:

Unused

Compare Mode:

Unused

PWM Mode:

These bits are the two LSbs (bit1 and bit0) of the 10-bit PWM duty cycle. The upper eight bits (DCx9:DCx2) of the duty cycle are found in CCPRxL.

bit 3:0 **CCPxM3:CCPxM0**: CCPx Mode Select bits

0000 = Capture/Compare/PWM off (resets CCPx module)
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode,
Initialize CCP pin Low, on compare match force CCP pin High (CCPIF bit is set)
1001 = Compare mode,
Initialize CCP pin High, on compare match force CCP pin Low (CCPIF bit is set)
1010 = Compare mode,
Generate software interrupt on compare match
(CCPIF bit is set, CCP pin is unaffected)
1011 = Compare mode,
Trigger special event (CCPIF bit is set)
11xx = PWM mode

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

14.3 Capture Mode

In Capture mode, CCPRxH:CCPRxL captures the 16-bit value of the TMR1 register when an event occurs on pin CCPx. An event is defined as:

- Every falling edge
- Every rising edge
- Every 4th rising edge
- Every 16th rising edge

An event is selected by control bits CCPxM3:CCPxM0 (CCPxCON<3:0>). When a capture is made, the interrupt request flag bit, CCPxIF, is set. The CCPxIF bit must be cleared in software. If another capture occurs before the value in register CCPRx is read, the previous captured value will be lost.

Note:	Timer1 must be running in timer mode or synchronized counter mode for the CCP module to use the capture feature. In asynchronous counter mode, the capture operation may not work.

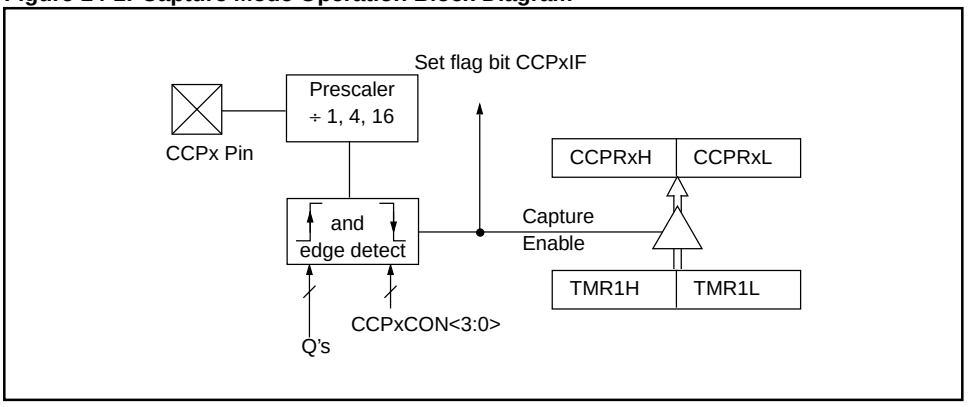
As can be seen in Figure 14-1, a capture does not reset the 16-bit TMR1 register. This is so Timer1 can also be used as the timebase for other operations. The time between two captures can easily be computed as the difference between the value of the second capture that of the first capture. When Timer1 overflows, the TMR1IF bit will be set and if enabled an interrupt will occur, allowing the time base to be extended to greater than 16-bits.

14.3.1 CCP Pin Configuration

In Capture mode, the CCPx pin should be configured as an input by setting its corresponding TRIS bit.

Note:	If the CCPx pin is configured as an output, a write to the port can cause a capture condition.

Figure 14-1: Capture Mode Operation Block Diagram



The prescaler can be used to get a very fine average resolution on a constant input frequency. For example, if we have a stable input frequency and we set the prescaler to 1:16, then the total error for those 16 periods is 1 TCY. This gives an effective resolution of TCY/16, which at 20 MHz is 12.5 ns. This technique is only valid where the input frequency is "stable" over the 16 samples. Without using the prescaler (1:1), each sample would have a resolution of TCY.

14.3.2 Changing Between Capture Modes

When the Capture mode is changed, a capture interrupt may be generated. The user should keep the CCPxIE bit clear to disable these interrupts and should clear the CCPxIF flag bit following any such change in operating mode.

14.3.2.1 CCP Prescaler

There are four prescaler settings, specified by bits CCPxM3:CCPxM0. Whenever the CCP module is turned off, or the CCP module is not in capture mode, the prescaler counter is cleared. This means that any reset will clear the prescaler counter.

Switching from one capture prescale setting to another may generate an interrupt. Also, the prescaler counter will not be cleared, therefore the first capture may be from a nonzero prescaler. Example 14-1 shows the recommended method for switching between capture prescale settings. This example also clears the prescaler counter and will not generate the interrupt.

Example 14-1: Changing Between Capture Prescalers

```
CLRF    CCP1CON         ; Turn CCP module off
MOVLW   NEW_CAPT_PS     ; Load the W reg with the new prescaler
                        ;   mode value and CCP ON
MOVWF   CCP1CON         ; Load CCP1CON with this value
```

To clear the Capture prescaler count, the CCP module must be configured into any non-capture CCP mode (Compare, PWM, or CCP off modes).

14.3.3 Sleep Operation

When the device is placed in sleep, Timer1 will not increment (since it is in synchronous mode), but the prescaler will continue to count events (not synchronized). When a specified capture event occurs, the CCPxIF bit will be set, but the capture register will not be updated. If the CCP interrupt is enabled, the device will wake-up from sleep. The value in the 16-bit TMR1 register is **not** transferred to the 16-bit capture register, but since the timer was not incrementing, this value should not have any meaning. Effectively, this allows the CCP pin to be used as another external interrupt.

14.3.4 Effects of a Reset

The CCP module is off, and the value in the capture prescaler is forced to 0.

14.4 Compare Mode

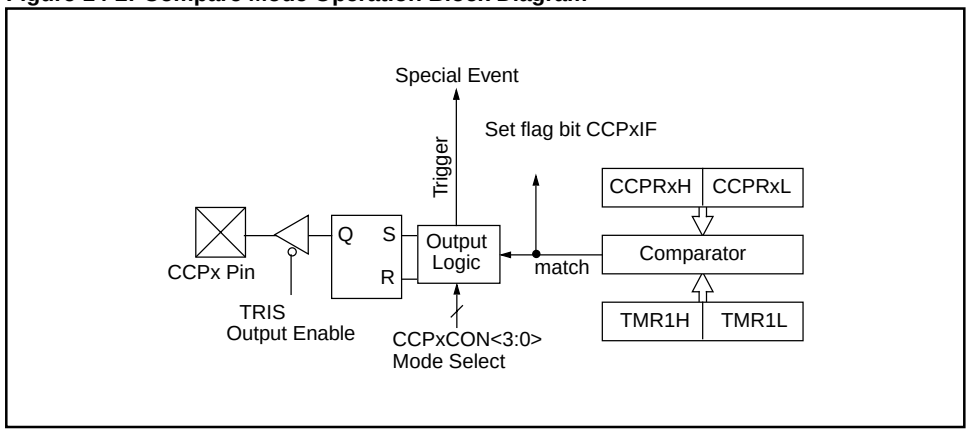
In Compare mode, the 16-bit CCPRx register value is constantly compared against the TMR1 register pair value. When a match occurs, the CCPx pin is:

- Driven High
- Driven Low
- Remains Unchanged

The action on the pin is based on the value of control bits CCPxM3:CCPxM0 (CCPxCON<3:0>). At the same time, a compare interrupt is also generated.

Note:	Timer1 must be running in Timer mode or Synchronized Counter mode if the CCP module is using the compare feature. In Asynchronous Counter mode, the compare operation may not work.

Figure 14-2: Compare Mode Operation Block Diagram



14.4.1 CCP Pin Operation in Compare Mode

The user must configure the CCPx pin as an output by clearing the appropriate TRIS bit.

Note:	Clearing the CCPxCON register will force the CCPx compare output latch to the default low level. This is not the Port I/O data latch.

Selecting the compare output mode, forces the state of the CCP pin to the state that is opposite of the match state. So if the Compare mode is selected to force the output pin low on match, then the output will be forced high until the match occurs (or the mode is changed).

14.4.2 Software Interrupt Mode

When generate Software Interrupt mode is chosen, the CCPx pin is not affected. Only a CCP interrupt is generated (if enabled).

14.4.3 Special Event Trigger

In this mode, an internal hardware trigger is generated which may be used to initiate an action.

The special event trigger output of CCPx resets the TMR1 register pair. This allows the CCPRx register to effectively be a 16-bit programmable period register for Timer1.

For some devices, the special trigger output of the CCP module resets the TMR1 register pair, and starts an A/D conversion (if the A/D module is enabled).

Note:	The special event trigger will not set the Timer1 interrupt flag bit, TMR1IF.

14.4.4 Sleep Operation

When the device is placed in sleep, Timer1 will not increment (since in synchronous mode), and the state of the module will not change. If the CCP pin is driving a value, it will continue to drive that value. When the device wakes-up, it will continue form this state.

14.4.5 Effects of a Reset

The CCP module is off.

14.5 PWM Mode

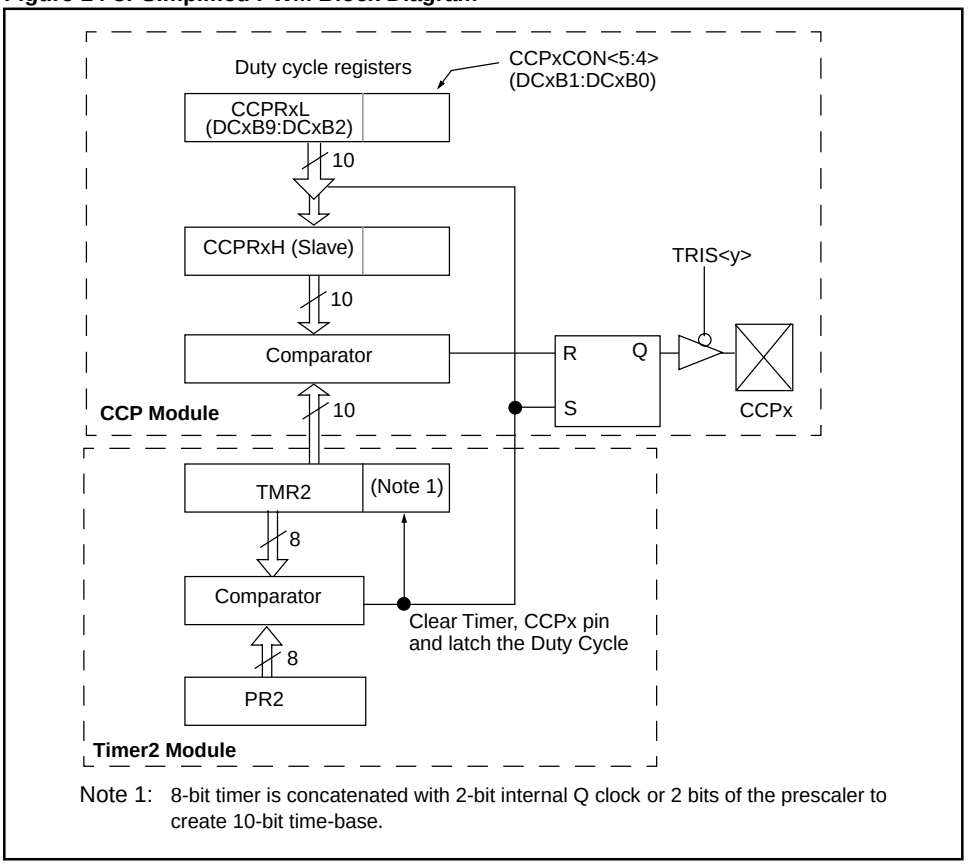
In Pulse Width Modulation (PWM) mode, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCPx pin is multiplexed with the PORT data latch, the corresponding TRIS bit must be cleared to make the CCPx pin an output.

Note:	Clearing the CCPxCON register will force the CCPx PWM output latch to the default low level. This is not the port I/O data latch.

Figure 14-3 shows a simplified block diagram of the CCP module in PWM mode.

For a step by step procedure on how to set up the CCP module for PWM operation, see Subsection **14.5.3 "Set-up for PWM Operation."**

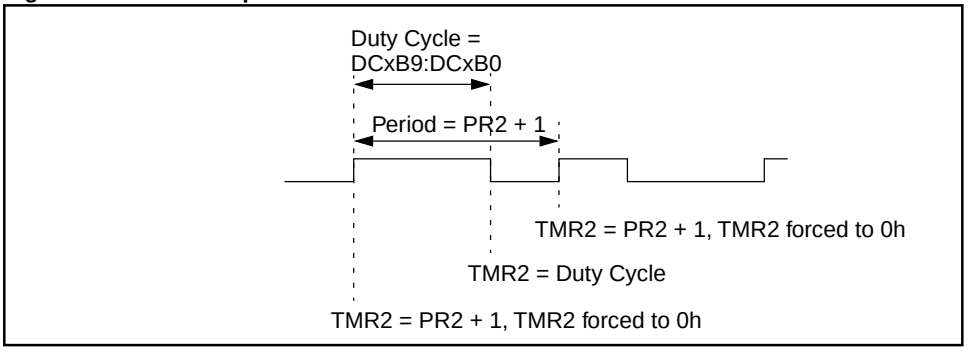
Figure 14-3: Simplified PWM Block Diagram



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.

A PWM output (Figure 14-4) has a time-base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

Figure 14-4: PWM Output



14.5.1 PWM Period

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

PWM period = [(PR2) + 1] • 4 • TOSC • (TMR2 prescale value), specified in units of time

PWM frequency (FPWM) is defined as 1 / [PWM period].

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCPx pin is set (exception: if PWM duty cycle = 0%, the CCPx pin will not be set)
- The PWM duty cycle is latched from CCPRxL into CCPRxH

Note:	The Timer2 postscaler is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

14.5.2 PWM Duty Cycle

The PWM duty cycle is specified by writing to the CCPRxL register and to the DCxB1:DCxB0 (CCPxCON<5:4>) bits. Up to 10-bit resolution is available: the CCPRxL contains the eight MSbs and CCPxCON<5:4> contains the two LSbs. This 10-bit value is represented by DCxB9:DCxB0. The following equation is used to calculate the PWM duty cycle:

PWM duty cycle = (DCxB9:DCxB0 bits value) • Tosc • (TMR2 prescale value), in units of time

The DCxB9:DCxB0 bits can be written to at any time, but the duty cycle value is not latched into CCPRxH until after a match between PR2 and TMR2 occurs (which is the end of the current period). In PWM mode, CCPRxH is a read-only register.

The CCPRxH register and a 2-bit internal latch are used to double buffer the PWM duty cycle. This double buffering is essential for glitchless PWM operation.

When CCPRxH and 2-bit latch match the value of TMR2 concatenated with the internal 2-bit Q clock (or two bits of the TMR2 prescaler), the CCPx pin is cleared. This is the end of the duty cycle.

Maximum PWM resolution (bits) for a given PWM frequency:

$$= \frac{\log\left(\frac{FOSC}{FPWM}\right)}{\log(2)} \text{ bits}$$

Note:	If the PWM duty cycle value is longer than the PWM period, the CCPx pin will not be cleared. This allows a duty cycle of 100%.

14.5.2.2 Minimum Resolution

The minimum resolution (in time) of each bit of the PWM duty cycle depends on the prescaler of Timer2.

Table 14-4: Minimum Duty Cycle Bit Time

Prescaler Value	T2CKPS1:T2CKPS0	Minimum Resolution (Time)
1	0 0	TOSC
4	0 1	TCY
16	1 x	4 TCY

Example 14-2: PWM Period and Duty Cycle Calculation

Desired PWM frequency is 78.125 kHz,
Fosc = 20 MHz
TMR2 prescale = 1

$1/78.125\ kHz = [(PR2) + 1] \cdot 4 \cdot 1/20\ MHz \cdot 1$

$12.8\ \mu s = [(PR2) + 1] \cdot 4 \cdot 50\ ns \cdot 1$

$PR2 = 63$

Find the maximum resolution of the duty cycle that can be used with a 78.125 kHz frequency and 20 MHz oscillator:

$1/78.125\ kHz = 2^{PWM\ RESOLUTION} \cdot 1/20\ MHz \cdot 1$

$12.8\ \mu s = 2^{PWM\ RESOLUTION} \cdot 50\ ns \cdot 1$

$256 = 2^{PWM\ RESOLUTION}$

$\log(256) = (PWM\ Resolution) \cdot \log(2)$

$8.0 = PWM\ Resolution$

At most, an 8-bit resolution duty cycle can be obtained from a 78.125 kHz frequency and a 20 MHz oscillator, i.e., $0 \le DCxB9:DCxB0 \le 255$. Any value greater than 255 will result in a 100% duty cycle.

In order to achieve higher resolution, the PWM frequency must be decreased. In order to achieve higher PWM frequency, the resolution must be decreased.

Table 14-5 lists example PWM frequencies and resolutions for Fosc = 20 MHz. The TMR2 prescaler and PR2 values are also shown.

Table 14-5: Example PWM Frequencies and Bit Resolutions at 20 MHz

PWM Frequency	1.22 kHz	4.88 kHz	19.53 kHz	78.12 kHz	156.3 kHz	208.3 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Maximum Resolution (bits)	10	10	10	8	7	5.5

14.5.3 Set-up for PWM Operation

The following steps configure the CCP module for PWM operation:

1. Establish the PWM period by writing to the PR2 register.
2. Establish the PWM duty cycle by writing to the DCxB9:DCxB0 bits.
3. Make the CCPx pin an output by clearing the appropriate TRIS bit.
4. Establish the TMR2 prescale value and enable Timer2 by writing to T2CON.
5. Configure the CCP module for PWM operation.

14.5.4 Sleep Operation

When the device is placed in sleep, Timer2 will not increment, and the state of the module will not change. If the CCP pin is driving a value, it will continue to drive that value. When the device wakes-up, it will continue from this state.

14.5.5 Effects of a Reset

The CCP module is off.

14.6 Initialization

The CCP module has three modes of operation. Example 14-3 shows the initialization of capture mode, Example 14-4 shows the initialization of compare mode, and Example 14-5 shows the initialization of PWM mode.

Example 14-3: Capture Initialization

```
    CLRF   CCP1CON       ; CCP Module is off
    CLRF   TMR1H         ; Clear Timer1 High byte
    CLRF   TMR1L         ; Clear Timer1 Low byte
    CLRF   INTCON        ; Disable interrupts and clear T0IF
    BSF    STATUS, RP0   ; Bank1
    BSF    TRISC, CCP1   ; Make CCP pin input
    CLRF   PIE1          ; Disable peripheral interrupts
    BCF    STATUS, RP0   ; Bank0
    CLRF   PIR1          ; Clear peripheral interrupts Flags
    MOVLW  0x06          ; Capture mode, every 4th rising edge
    MOVWF  CCP1CON       ;
    BSF    T1CON, TMR1ON ; Timer1 starts to increment
;
; The CCP1 interrupt is disabled,
; do polling on the CCP Interrupt flag bit
;
Capture_Event
    BTFSS  PIR1, CCP1IF
    GOTO   Capture_Event
;
; Capture has occurred
;
    BCF    PIR1, CCP1IF  ; This needs to be done before next compare
```

Example 14-4: Compare Initialization

```
    CLRF   CCP1CON       ; CCP Module is off
    CLRF   TMR1H         ; Clear Timer1 High byte
    CLRF   TMR1L         ; Clear Timer1 Low byte
    CLRF   INTCON        ; Disable interrupts and clear T0IF
    BSF    STATUS, RP0   ; Bank1
    BCF    TRISC, CCP1   ; Make CCP pin output if controlling state of pin
    CLRF   PIE1          ; Disable peripheral interrupts
    BCF    STATUS, RP0   ; Bank0
    CLRF   PIR1          ; Clear peripheral interrupts Flags
    MOVLW  0x08          ; Compare mode, set CCP1 pin on match
    MOVWF  CCP1CON       ;
    BSF    T1CON, TMR1ON ; Timer1 starts to increment
;
; The CCP1 interrupt is disabled,
; do polling on the CCP Interrupt flag bit
;
Compare_Event
    BTFSS  PIR1, CCP1IF
    GOTO   Compare_Event
;
; Compare has occurred
;
    BCF    PIR1, CCP1IF   ; This needs to be done before next compare
```

Example 14-5: PWM Initialization

```
    CLRF   CCP1CON       ; CCP Module is off
    CLRF   TMR2          ; Clear Timer2
    MOVLW  0x7F          ;
    MOVWF  PR2           ;
    MOVLW  0x1F          ;
    MOVWF  CCPR1L        ; Duty Cycle is 25% of PWM Period
    CLRF   INTCON        ; Disable interrupts and clear T0IF
    BSF    STATUS, RP0   ; Bank1
    BCF    TRISC, PWM1   ; Make pin output
    CLRF   PIE1          ; Disable peripheral interrupts
    BCF    STATUS, RP0   ; Bank0
    CLRF   PIR1          ; Clear peripheral interrupts Flags
    MOVLW  0x2C          ; PWM mode, 2 LSbs of Duty cycle = 10
    MOVWF  CCP1CON       ;
    BSF    T2CON, TMR2ON ; Timer2 starts to increment
;
; The CCP1 interrupt is disabled,
; do polling on the TMR2 Interrupt flag bit
;
PWM_Period_Match
    BTFSS  PIR1, TMR2IF
    GOTO   PWM_Period_Match
;
; Update this PWM period and the following PWM Duty cycle
;
    BCF    PIR1, TMR2IF
```

14.7 Design Tips

Question 1: ***What timers can I use for the capture and compare modes?***

Answer 1:

The capture and compare modes are designed around Timer1, so no other timer can be used for these functions. This also means that if multiple CCP modules (in parts with more than one) are being used for a capture or compare function, they will share the same timer.

Question 2: ***What timers can I use with the PWM mode?***

Answer 2:

The PWM mode is designed around Timer2, so no other timer can be used for this function. (It is the only timer with a period register associated with it.) If multiple CCP modules (in parts with more than one) are doing PWM they will share the same timer, meaning they will have the same PWM period and frequency.

Question 3: ***Can I use one CCP module to do capture (or compare) AND PWM at the same time, since they use different timers as their reference?***

Answer 3:

The timers may be different, but other logic functions are shared. However you can switch from one mode to the other. For a device with two CCP modules, you can also have CCP1 set up for PWM and CCP2 set up for capture or compare (or vice versa) since they are two independent modules.

Question 4: ***How does a reset affect the CCP module?***

Answer 4:

Any reset will turn the CCP module off. See the section on resets to see reset values.

Question 5: ***I am setting up the CCP1CON module for "Compare Mode, trigger special event" (1011) which resets TMR1. When a compare match occurs, will I have both the TMR1 and the CCP1 interrupts pending (TMR1IF is set, CCP1IF is set)?***

Answer 5:

The CCP1IF flag will be set on the match condition. TMR1IF is set when Timer1 overflows, and the special trigger reset of Timer1 is not considered an overflow. However, if both the CCPR1L and CCPR1H registers are set at FFh, then an overflow occurs at the same time as the match, which will then set both CCP1IF and TMR1IF.

Question 6: ***How do I use Timer2 as a general purpose timer, with an interrupt flag on rollover?***

Answer 6:

Timer2 always resets to zero when it equals PR2 and flag bit TMR2IF always gets set at this time. By putting FFh into PR2, you will get an interrupt on overflow at FFh, as you would with Timer0, for instance. Quite often it is desirable to have an event occur at a periodic rate, perhaps an interrupt driven event. Normally an initial value would be placed into the timer so that the overflow will occur at the desired time. This value would have to be placed back into the timer every time it overflowed to make the interrupts occur at the same desired rate. The benefit of Timer2 is that a value can be written to PR2 that will cause it to reset at your desired time interval. This means you do not have the housekeeping chore of reloading the timer every time it overflows, since PR2 maintains its value.

Question 7: ***I am using a CCP module in PWM mode. The duty cycle being output is almost always 100%, even when my program writes a value like 7Fh to the duty cycle register, which should be 50%. What am I doing wrong?***

Answer 7:

1. The value in CCPRxL is higher than PR2. This happens quite often when a user desires a fast PWM output frequency and will write a small value in the PR2. In this case, if a value of 7Eh were written to PR2, then a value 7Fh in CCPRxL will result in 100% duty cycle.
2. If the TRIS bit corresponding to the CCP output pin you are using is configured as an input, the PWM output cannot drive the pin. In this case the pin would float and duty cycle may appear to be 0%, 100% or some other floating value.

Question 8: ***I want to determine a signal frequency using the CCP module in capture mode to find the period. I am currently resetting Timer1 on the first edge, then using the value in the capture register on the second edge as the time period. The problem is that my code to clear the timer does not occur until almost twelve instructions after the first capture edge (interrupt latency plus saving of registers in interrupt) so I cannot measure very fast frequencies. Is there a better way to do this?***

Answer 8:

You do not need to zero the counter to find the difference between two pulse edges. Just take the first captured value and put it into another set of registers. Then when the second capture event occurs, just subtract the first event from the second. Assuming that your pulse edges are not so far apart that the counter can wrap around past the last capture value, the answer will always be correct. This is illustrated by the following example:

1. First captured value is FFFEh. Store this value in two registers.
2. The second capture value is 0001h (the counter has incremented three times).
3. 0001h - FFFEh = 0003, which is the same as if you had cleared Timer1 to zero and let it count to 3. (Theoretically, except that there was a delay getting to the code that clears Timer1, so actual values would differ).

The interrupt overhead is now less important because the values are captured automatically. For even faster inputs do not enable interrupts and just test the flag bit in a loop. If you must also capture very long time periods, such that the timer can wrap around past the previous capture value, then consider using an auto-scaling technique that starts with a large prescale and shorten the prescale as you converge on the exact frequency.

14.8 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the CCP modules are:

Title	Application Note #
Using the CCP Modules	AN594
Implementing Ultrasonic Ranging	AN597
Air Flow Control Using Fuzzy Logic	AN600
Adaptive Differential Pulse Code Modulation	AN643

14.9 Revision History

Revision A

This is the initial released revision of the CCP module description.

Section 15. Synchronous Serial Port (SSP)

HIGHLIGHTS

This section of the manual contains the following major topics:

Note:	Please refer to Appendix C.2 or the device data sheet to determine which devices use this module.

15

SSP

I^2C is a trademark of Philips Corporation.

15.1 Introduction

The Synchronous Serial Port (SSP) module is a serial interface useful for communicating with other peripherals or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D converters, etc. The SSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI™)
- Inter-Integrated Circuit (I^2C™)
  - Slave mode
  - I/O slope control, and Start and Stop bit detection to ease software implementation of Master and Multi-master modes

SPI is a registered trademark of Motorola Corporation.
I^2C is a trademark of Philips Corporation.

15.2 Control Registers

Register 15-1: SSPSTAT: Synchronous Serial Port Status Register

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:** SPI data input sample phase

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

bit 6 **CKE**: SPI Clock Edge Select (Figure 15-3, Figure 15-4, and Figure 15-5)

CKP = 0 (SSPCON<4>)

1 = Data transmitted on rising edge of SCK
0 = Data transmitted on falling edge of SCK

CKP = 1 (SSPCON<4>)

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 SSP module is disabled)

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 SSP module is disabled)

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/$\overline{Write}$ 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 $\overline{ACK}$ bit.

1 = Read
0 = Write

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 = Transmit in progress, SSPBUF is full
0 = Transmit complete, SSPBUF is empty

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

Register 15-2: SSPCON: Synchronous Serial Port Control Register

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	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL**: Write Collision Detect bit

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 and the SSPBUF is no longer updated. Overflow can only occur 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.

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. SSPOV must be cleared in software in either mode.
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 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

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 mode:
SCK release control
1 = Enable clock
0 = Holds clock low (clock stretch) (Used to ensure data setup time)

Register 15-2: SSPCON: Synchronous Serial Port Control Register (Cont'd)

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. $\overline{SS}$ pin control enabled.
0101 = SPI slave mode, clock = SCK pin. $\overline{SS}$ pin control disabled. $\overline{SS}$ can be used as I/O pin
0110 = I^2C slave mode, 7-bit address
0111 = I^2C slave mode, 10-bit address
1000 = Reserved
1001 = Reserved
1010 = Reserved
1011 = I^2C firmware controlled master mode (slave idle)
1100 = Reserved
1101 = Reserved
1110 = I^2C slave mode, 7-bit address with start and stop bit interrupts enabled
1111 = I^2C slave mode, 10-bit address with start and stop bit interrupts enabled

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

15.3 SPI Mode

The SPI mode allows 8-bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported, as well as Microwire™ (sample edge) when the SPI is in the master mode.

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 ($\overline{SS}$)

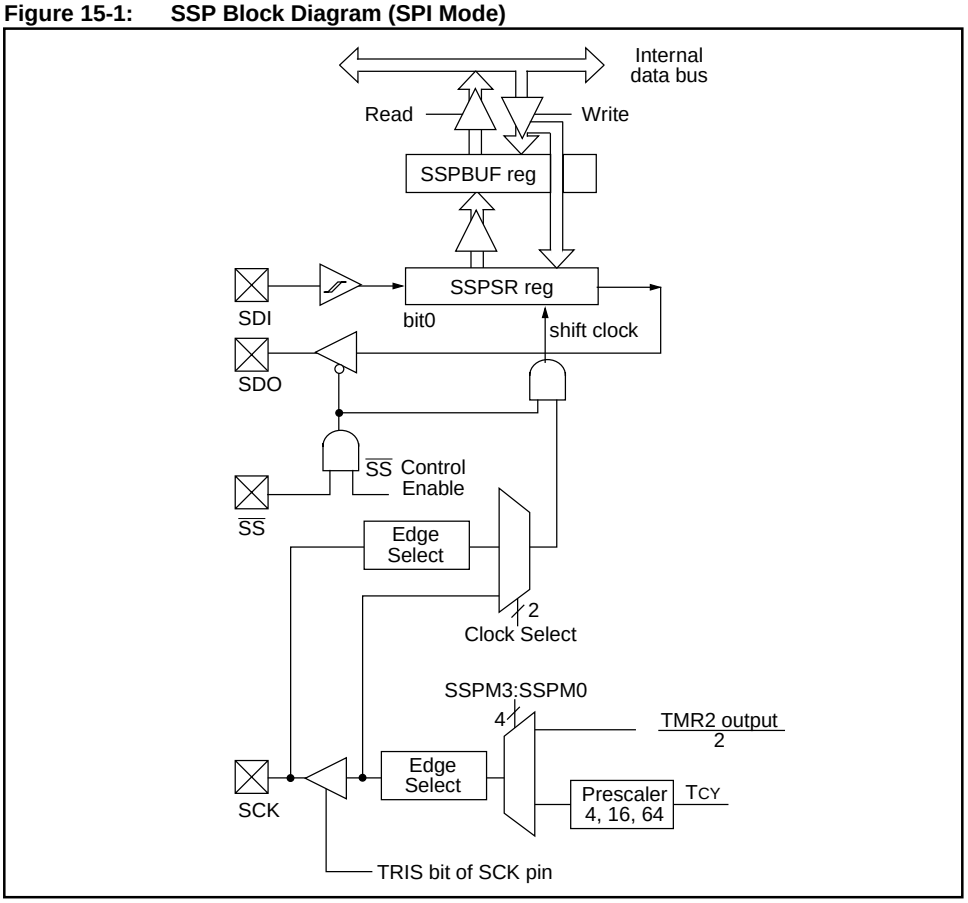
15.3.1 Operation

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits in the SSPCON register (SSPCON<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)
- Clock edge (output data on rising/falling edge of SCK)
- Data Input Sample Phase
- Clock Rate (Master mode only)
- Slave Select Mode (Slave mode only)

Figure 15-1 shows the block diagram of the SSP module, when in SPI mode.

Figure 15-1: SSP Block Diagram (SPI Mode)



The SSP 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 interrupt flag bit, SSPIF, 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 (SSPCON<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, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally the SSP 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. The shaded instruction is only required if the received data is meaningful (some SPI applications are transmit only).

Example 15-1: Loading the SSPBUF (SSPSR) Register

```
     BCF   STATUS, RP1    ;Specify Bank1
     BSF   STATUS, RP0    ;
LOOP BTFSS SSPSTAT, BF    ;Has data been received (transmit complete)?
     GOTO  LOOP           ;No
     BCF   STATUS, RP0    ;Specify Bank0
     MOVF  SSPBUF, W      ;W reg = contents of SSPBUF
     MOVWF RXDATA         ;Save in user RAM, if data is meaningful
     MOVF  TXDATA, W      ;W reg = contents of TXDATA
     MOVWF SSPBUF         ;New data to xmit
```

The SSPSR is not directly readable or writable, and can only be accessed from addressing the SSPBUF register. Additionally, the SSP status register (SSPSTAT) indicates the various status conditions.

Microwire is a trademark of National Semiconductor.

15.3.2 Enabling SPI I/O

To enable the serial port the SSP Enable bit, SSPEN (SSPCON<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit which re-initializes the SSPCON register, and then set the SSPEN bit. This configures the SDI, SDO, SCK, and $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, they must have their data direction bits (in the TRIS register) appropriately programmed. That is:

- SDI must have the TRIS bit set
- SDO must have the TRIS bit cleared
- SCK (Master mode) must have the TRIS bit cleared
- SCK (Slave mode) must have the TRIS bit set
- $\overline{SS}$ must have the TRIS bit set

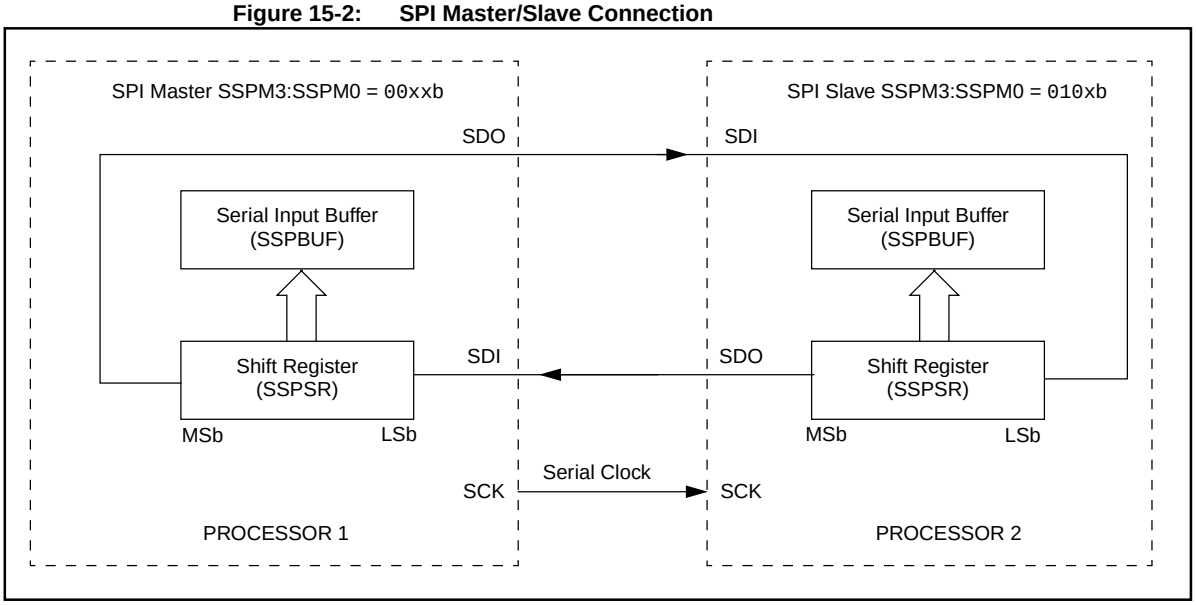
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value. An example would be in master mode where you are only sending data (to a display driver), then both SDI and $\overline{SS}$ could be used as general purpose outputs by clearing their corresponding TRIS register bits.

15.3.3 Typical Connection

Figure 15-2 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 edge of the clock specified by the SMP bit. 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 15-2: SPI Master/Slave Connection



15.3.4 Master Operation

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave (Processor 2) is to broadcast data by the software protocol.

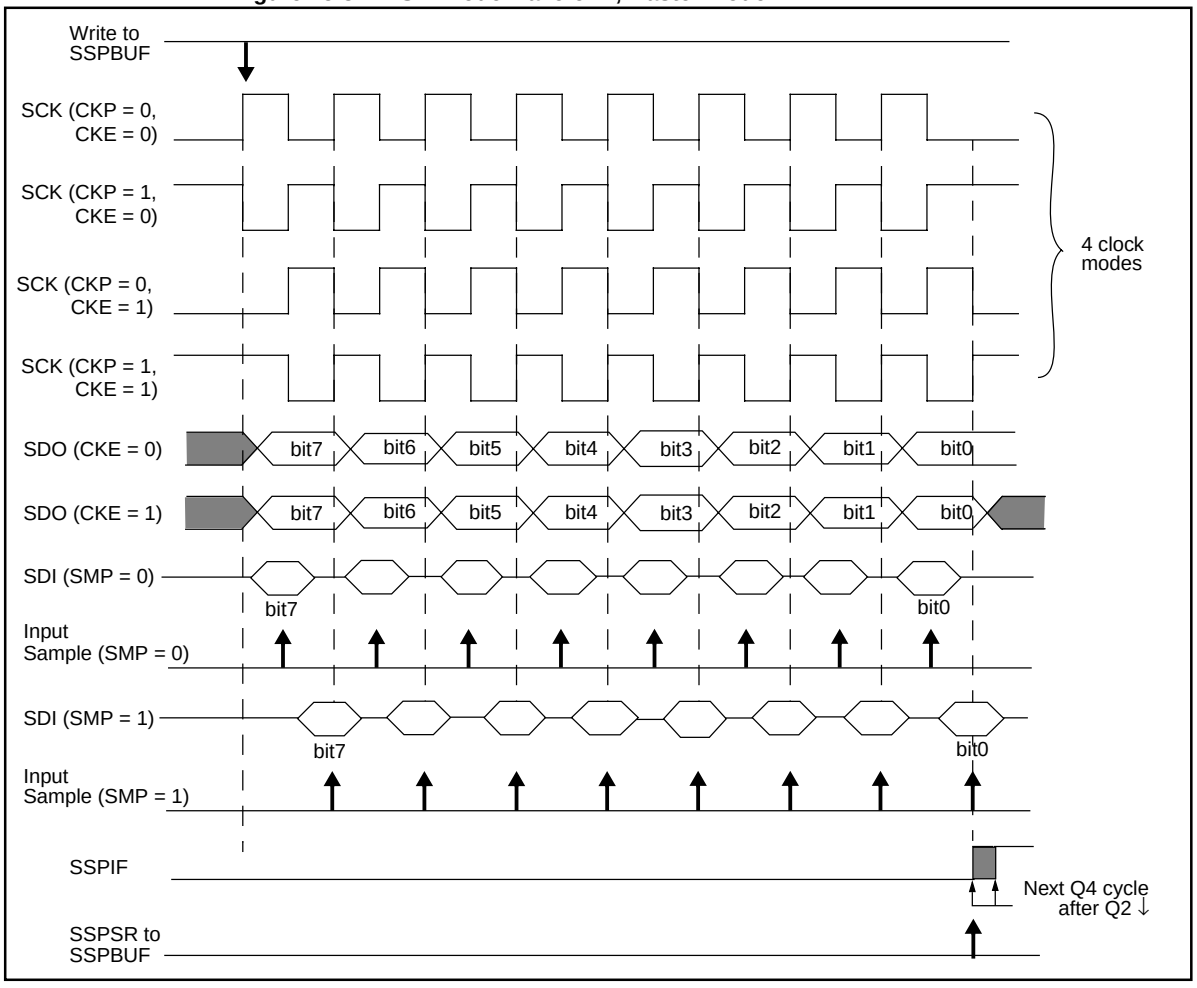
In master mode the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As 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 (SSPCON<4>). This then would give waveforms for SPI communication as shown in Figure 15-3, Figure 15-4, and Figure 15-5 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 data rate of 5 Mbps (at 20 MHz).

Figure 15-3: SPI Mode Waveform, Master Mode



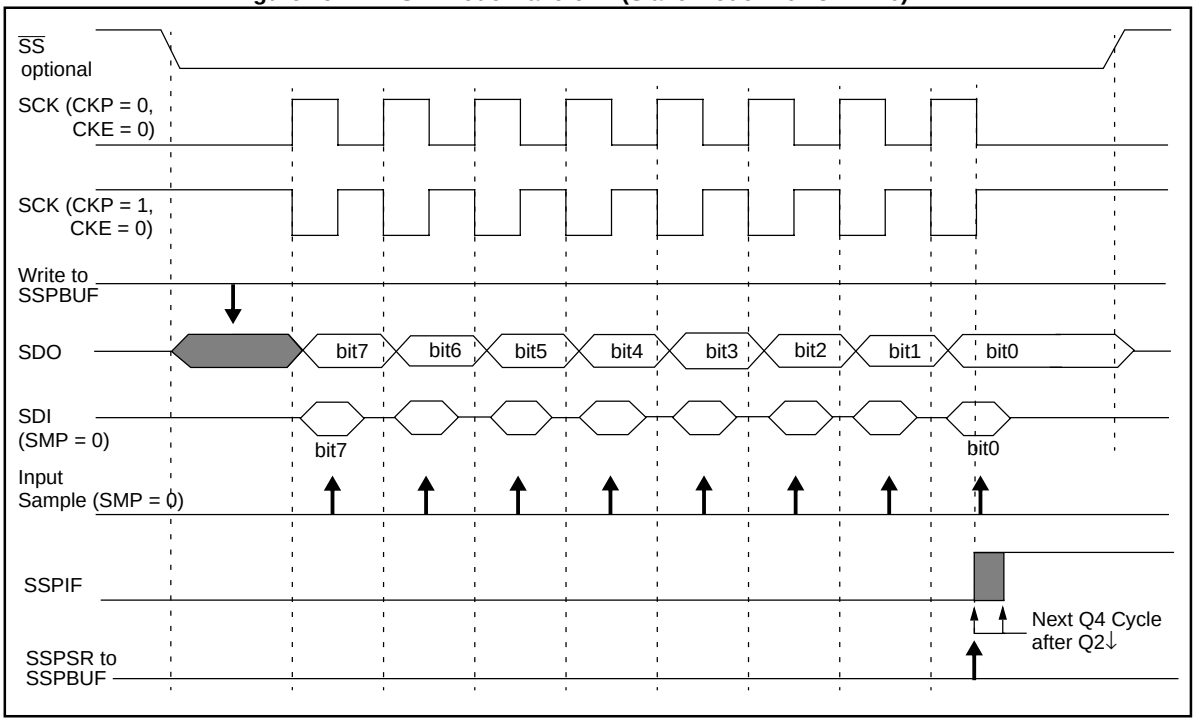
15.3.5 Slave Operation

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 is set.

The clock polarity is selected by appropriately programming bit CKP (SSPCON<4>). This then would give waveforms for SPI communication as shown in Figure 15-3, Figure 15-4, and Figure 15-5 where the MSb is transmitted first. When in slave mode the external clock must meet the minimum high and low times.

In sleep mode, the slave can transmit and receive data. When a byte is received, the device will wake-up from sleep, if the interrupt is enabled.

Figure 15-4: SPI Mode Waveform (Slave Mode With CKE = 0)



15.3.6 Slave Select Mode

When in slave select mode, the $\overline{SS}$ pin allows multi-drop for multiple slaves with a single master. The SPI must be in slave mode (SSPCON<3:0> = 04h) and the TRIS bit, for the $\overline{SS}$ pin, must be set for the slave select mode to be enabled. 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.

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 VDD. If the SPI is used in Slave Mode with the CKE bit is set, 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 by forcing the $\overline{SS}$ pin to a high level or clearing the SSPEN bit (Figure 15-6).

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 15-5: SPI Mode Waveform (Slave Select Mode With CKE = 1)

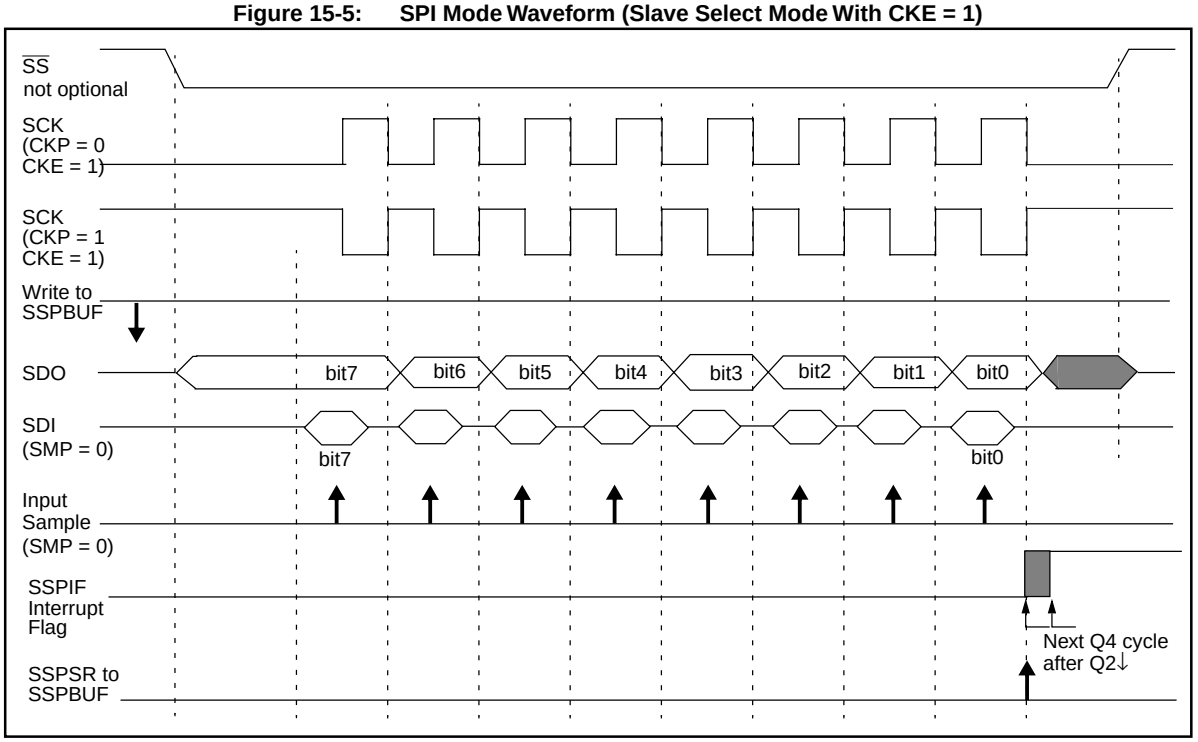
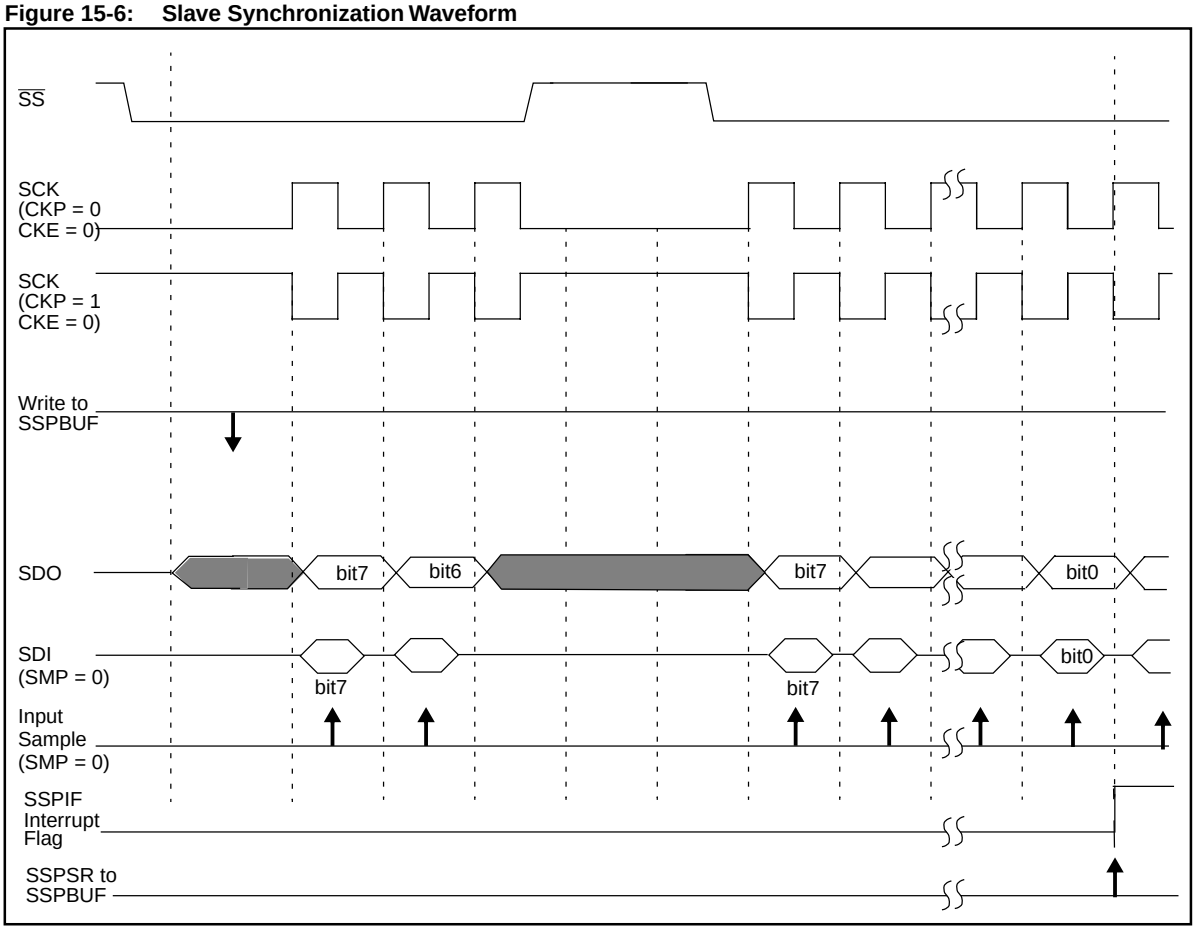


Figure 15-6: Slave Synchronization Waveform



15.3.7 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 SSP interrupt flag bit will be set and if enabled will wake the device from sleep.

15.3.8 Effects of a Reset

A reset disables the SSP module and terminates the current transfer.

Table 15-1: Registers Associated with SPI Operation

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other resets
INTCON	GIE	PEIE	T0IE	INTE	RBIE(2)	T0IF	INTF	RBIF(2)	0000 000x	0000 000u
PIR	SSPIF (1)								0	0
PIE	SSPIE (1)								0	0
SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
TRISC	PORTC Data Direction Control Register								1111 1111	1111 1111
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 SPI mode.

Note 1: The position of this bit is device dependent.

2: These bits may also be named GPIE and GPIF.

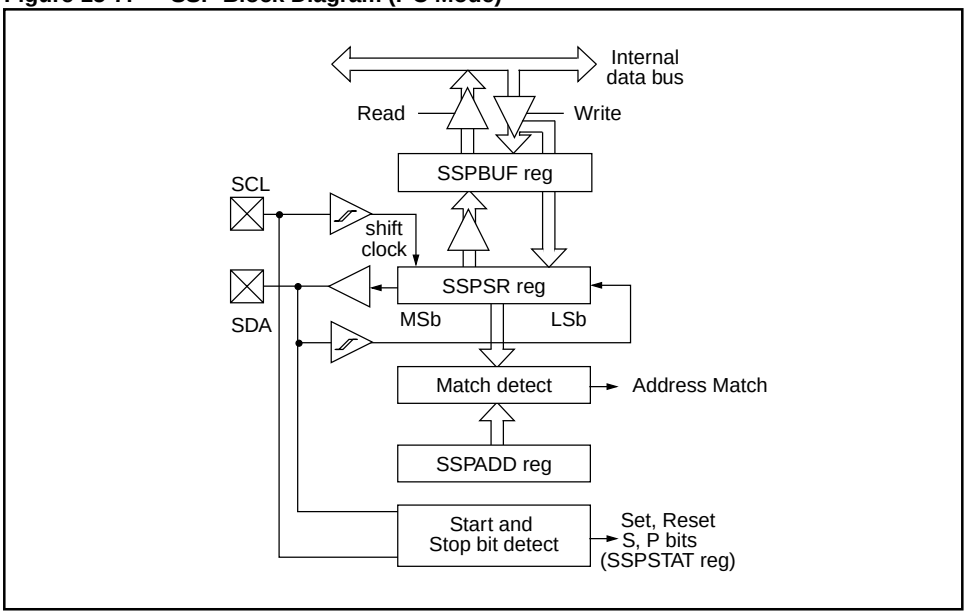
15.4 SSP I²C Operation

The SSP module in I²C mode fully implements all slave functions, except general call support, and provides interrupts on start and stop bits in hardware to facilitate software implementations of the master functions. The SSP module implements the standard mode specifications as well as 7-bit and 10-bit addressing. Appendix A gives an overview of the I²C bus specification.

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 user must configure these pins as inputs through the TRIS bits. The SSP module functions are enabled by setting SSP Enable bit, SSPEN (SSPCON<5>).

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 15-7: SSP Block Diagram (I²C Mode)



The SSP module has five registers for I^2C operation. They are:

- SSP Control Register (SSPCON)
- SSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer (SSPBUF)
- SSP Shift Register (SSPSR) - Not directly accessible
- SSP Address Register (SSPADD)

The SSPCON register allows control of the I^2C operation. Four mode selection bits (SSPCON<3:0>) allow one of the following I^2C modes to be selected:

- I^2C Slave mode (7-bit address)
- I^2C Slave mode (10-bit address)
- I^2C Firmware controlled Multi-Master mode (start and stop bit interrupts enabled)
- I^2C Firmware controlled Multi-Master mode (start and stop bit interrupts enabled)
- I^2C Firmware controlled Master mode, slave is idle

Before selecting any I^2C mode, the SCL and SDA pins must be programmed to inputs by setting the appropriate TRIS bits. Selecting an I^2C mode, by setting the SSPEN bit, enables the SCL and SDA pins to be used as the clock and data lines in I^2C 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 the SSPOV bit (SSPCON<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).

15.4.1 Slave Mode

In slave mode, the SCL and SDA pins must be configured as inputs (TRIS set). The SSP 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 SSP 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 message completed.

b) The overflow bit, SSPOV (SSPCON<6>), was set before the message completed.

In this case, the SSPSR register value is not loaded into the SSPBUF, but the SSPIF and SSPOV bits 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 SSP module is shown in Device Data Sheet electrical specifications parameters 100 and 101.

15.4.1.1 Addressing

Once the SSP 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 eight SCL pulse.

b) The buffer full bit, BF, is set on the falling edge of the eigth SCL pulse.

c) An $\overline{ACK}$ pulse is generated.

d) SSP interrupt flag bit, SSPIF, is set (interrupt is generated if enabled) - on the falling edge of the ninth 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. The R/$\overline{W}$ bit (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 (the SSPIF, BF, and UA (SSPSTAT<1>) bits are set).
2. Update the SSPADD register with second (low) byte of Address (clears the UA bit and releases the SCL line).
3. Read the SSPBUF register (clears the BF bit) and clear the SSPIF flag bit.
4. Receive second (low) byte of Address (the SSPIF, BF, and UA bits are set).
5. Update the SSPADD register with the high byte of Address. This will clear the UA bit and releases SCL line.
6. Read the SSPBUF register (clears the BF bit) and clear the SSPIF flag bit.
7. Receive repeated START condition.
8. Receive first (high) byte of Address (the SSPIF and BF bits are set).
9. Read the SSPBUF register (clears the BF bit) and clear the SSPIF flag bit.

Note:	Following the RESTART 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.

Table 15-2: Data Transfer Received Byte Actions

Status Bits as Data Transfer is Received				
BF	**SSPOV**	**SSPSR → SSPBUF**	**Generate $\overline{ACK}$ Pulse**	**Set bit SSPIF (SSP Interrupt occurs if enabled)**
0	0	Yes	Yes	Yes
1	0	No	No	Yes
1	1	No	No	Yes
0	1	Yes	No	Yes

Note: Shaded cells show the conditions where the user software did not properly clear the overflow condition.

15.4.1.2 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 the BF bit (SSPSTAT<0>) is set or the SSPOV bit (SSPCON<6>) is set. So when a byte is received, with these conditions, and attempts to move from the SSPSR register to the SSPBUF register, no acknowledge pulse is given.

An SSP interrupt is generated for each data transfer byte. The SSPIF flag bit must be cleared in software. The SSPSTAT register is used to determine the status of the receive byte.

Figure 15-8: I²C Waveforms for Reception (7-bit Address)

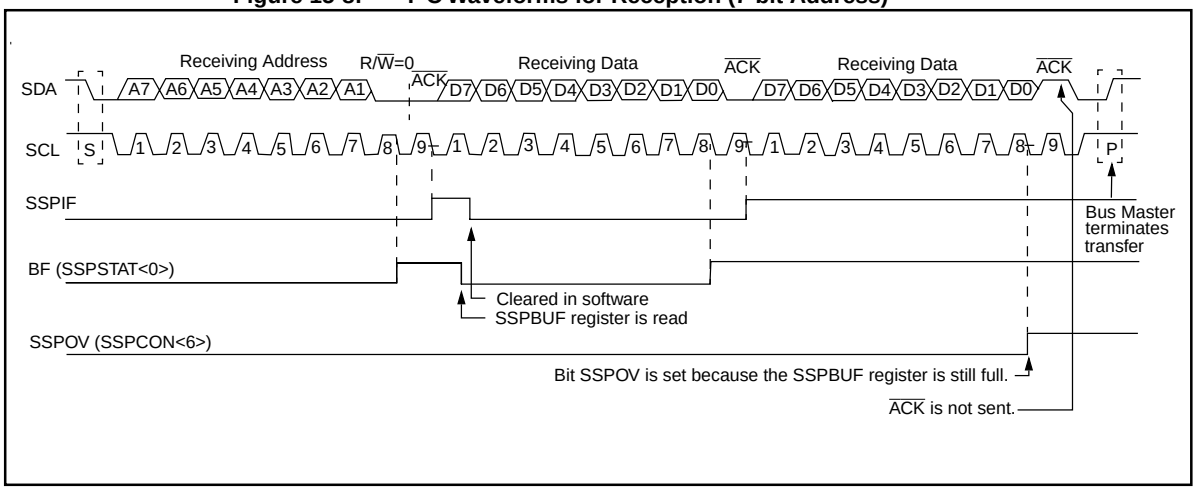
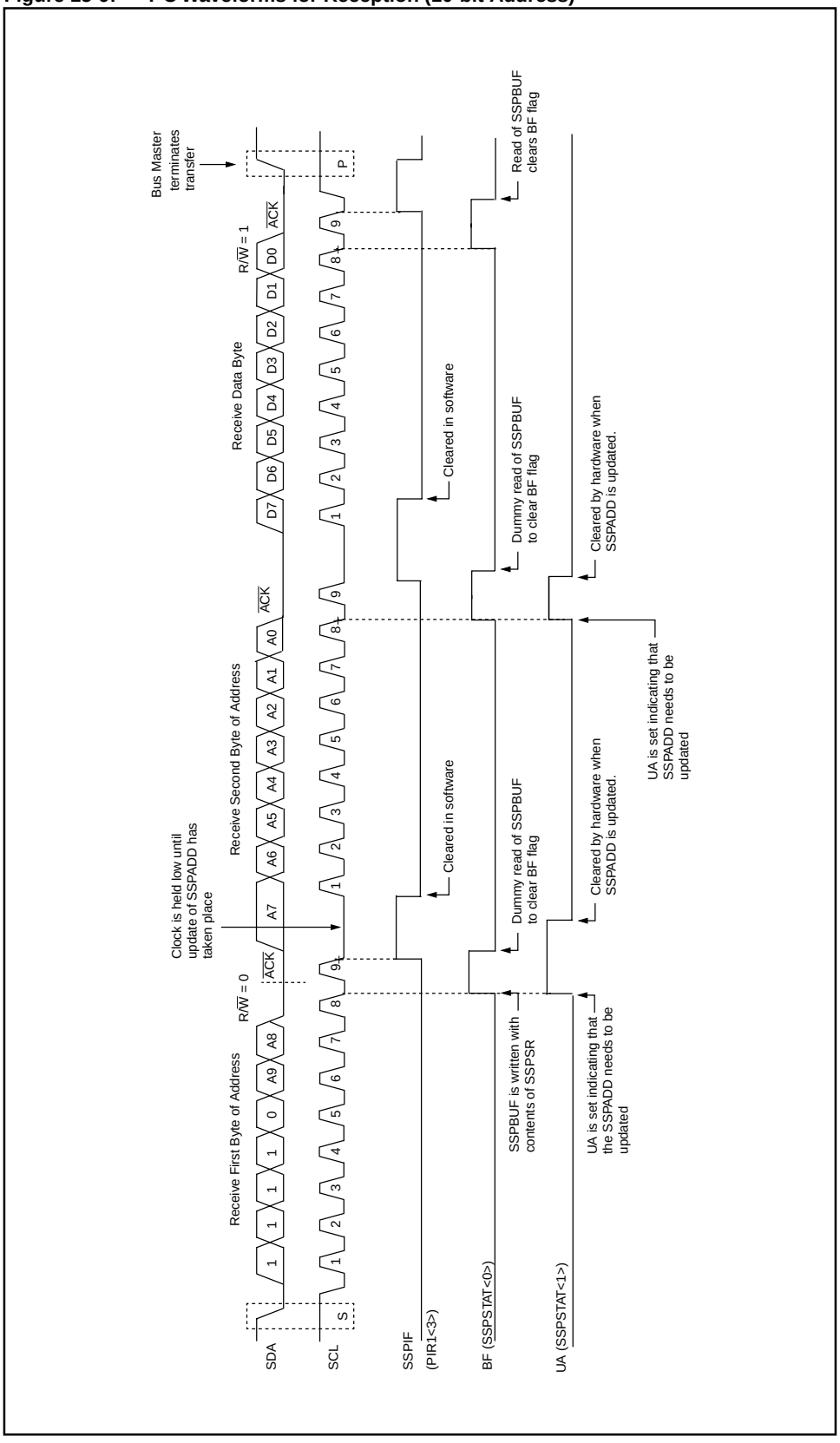


Figure 15-9: I²C Waveforms for Reception (10-bit Address)



15.4.1.3 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 the SCL pin should be enabled by setting the CKP bit (SSPCON<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 15-10).

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 15-10: I^2C Waveforms for Transmission (7-bit Address)

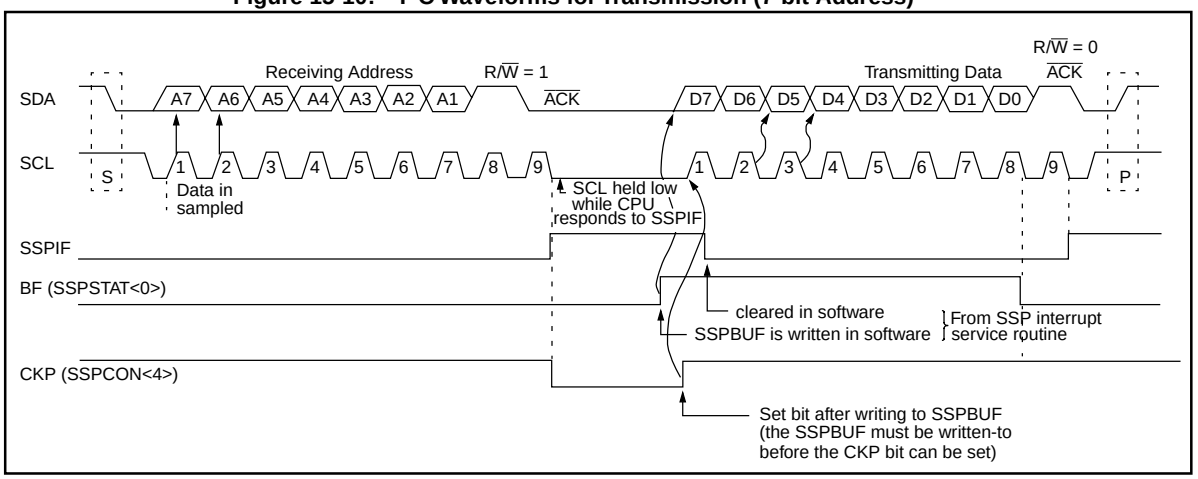
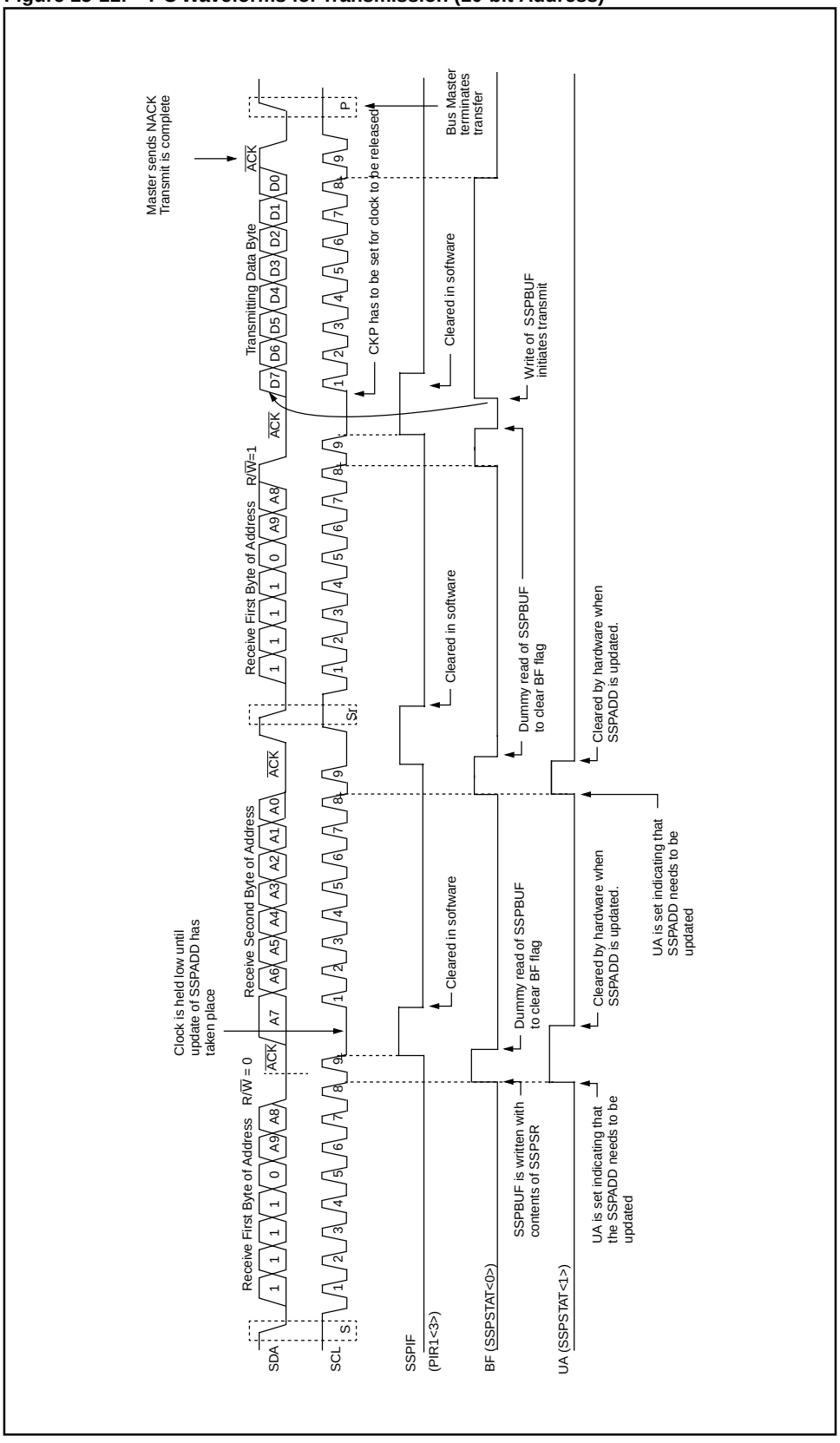


Figure 15-11: I²C Waveforms for Transmission (10-bit Address)



15.4.1.4 Clock Arbitration

Clock arbitration has the SCL pin to inhibit the master device from sending the next clock pulse. The SSP module in I^2C slave mode will hold the SCL pin low when the CPU needs to respond to the SSP interrupt (SSPIF bit is set and the CKP bit is cleared). The data that needs to be transmitted will need to be written to the SSPBUF register, and then the CKP bit will need to be set to allow the master to generate the required clocks.

15.4.2 Master Mode (Firmware)

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 SSP 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 clearing the corresponding TRIS bit(s). The output level is always low, irrespective of the value(s) in the PORT register. So when transmitting data, a '1' data bit must have it's TRIS bit set (input) and a '0' data bit must have it's TRIS bit cleared (output). The same scenario is true for the SCL line with the TRIS bit.

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

Master mode of operation can be done with either the slave mode idle (SSPM3:SSPM0 = 1011) or with the slave active (SSPM3:SSP0 = 1110 or 1111). When the slave modes are enabled, the software needs to differentiate the source(s) of the interrupt.

15.4.3 Multi-Master Mode (Firmware)

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 SSP module is disabled. Control of the I^2C bus may be taken when the P bit (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 to see if the signal level is the expected output level. This check only needs to be done when a high level is output. If a high level is expected and a low level is present, the device needs to release the SDA and SCL lines (set the TRIS bits). There are two stages where this arbitration can be lost, they are:

- Address transfer
- Data transfer

When the slave logic is enabled, the slave continues to receive. If arbitration was lost during the address transfer stage, communication to the device may be in progress. If addressed an $\overline{ACK}$ pulse will be generated. If arbitration was lost during the data transfer stage, the device will need to retransfer the data at a later time.

15.4.4 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).

15.4.5 Effect of a Reset

A reset disables the SSP module and terminates the current transfer.

Table 15-3: Registers Associated with I^2C Operation

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other resets
INTCON	GIE	PEIE	T0IE	INTE	RBIE(2)	T0IF	INTF	RBIF(2)	0000 000x	0000 000u
PIR	SSPIF (1)								0	0
PIE	SSPIE (1)								0	0
SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	0000 0000
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
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 SSP in I^2C mode.
Note 1: The positions of these bits are device dependent.
2: These bits may also be named GPIE and GPIF.

15.5 Initialization

Example 15-2: SPI Master Mode Initialization

```
    CLRF   STATUS       ; Bank 0
    CLRF   SSPSTAT      ; SMP = 0, CKE = 0, and clear status bits
    BSF    SSPSTAT, CKE ; CKE = 1
    MOVLW  0x31         ; Set up SPI port, Master mode, CLK/16,
    MOVWF  SSPCON       ;   Data xmit on falling edge (CKE=1 & CKP=1)
                        ;   Data sampled in middle (SMP=0 & Master mode)
    BSF    STATUS, RP0  ; Bank 1
    BSF    PIE, SSPIE   ; Enable SSP interrupt
    BCF    STATUS, RP0  ; Bank 0
    BSF    INTCON, GIE  ; Enable, enabled interrupts
    MOVLW  DataByte     ; Data to be Transmitted
                        ;   Could move data from RAM location
    MOVWF  SSPBUF       ; Start Transmission
```

15.5.1 SSP Module / Basic SSP Module Compatibility

When upgrading from the Basic SSP module, the SSPSTAT register contains two additional control bits. These bits are only used in SPI mode and are:

- SMP, SPI data input sample phase
- CKE, SPI Clock Edge Select

To be compatible with the SPI of the Basic SSP module, these bits must be appropriately configured. If these bits are not at the states shown in Table 15-4, improper SPI communication may occur.

Table 15-4: New Bit States for Compatibility

Basic SSP Module	SSP Module		
CKP	CKP	CKE	SMP
1	1	0	0
0	0	0	0

15.6 Design Tips

Question 1: ***Using SPI mode, I do not seem able to talk to an SPI device.***

Answer 1:

Ensure that you are using the correct SPI mode for that device. This SPI supports all four SPI modes so you should be able to get it to function. Check the clock polarity and the clock phase.

Question 2: ***Using I^2C mode, I do not seem able to make the master mode work.***

Answer 2:

This SSP module does not have master mode fully automated in hardware, see Application Note AN578 for software which uses the SSP module to implement master mode. If you require a fully automated hardware implementation of I^2C Master Mode, please refer to the Microchip Line Card for devices that have the Master SSP module.

Note:	At the time of printing only the High-end family of devices (PIC17CXXX) have devices with the Master SSP module implemented.

Question 3: ***Using I^2C mode, I write data to the SSPBUF register, but the data did not transmit.***

Answer 3:

Ensure that you set the CKP bit to release the I^2C clock.

15.7 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the SSP Module are:

Title	Application Note #
Use of the SSP Module in the I ^{2}C Multi-Master Environment.	AN578
Using Microchip 93 Series Serial EEPROMs with Microcontroller SPI Ports	AN613
Software Implementation of I^{2}C Bus Master	AN554
Use of the SSP module in the Multi-master Environment	AN578
Interfacing PIC16C64/74 to Microchip SPI Serial EEPROM	AN647
Interfacing a Microchip PIC16C92x to Microchip SPI Serial EEPROM	AN668

15.8 Revision History

Revision A

This is the initial released revision of the SSP module description.

Section 16. Basic Sychronous Serial Port (BSSP)

HIGHLIGHTS

This section of the manual contains the following major topics:

Note:	Please refer to Appendix C.2 or the device data sheet to determine which devices use this module.

SPI is a trademark of Motorola Corporation.
I^2C is a trademark of Philips Corporation.

16.1 Introduction

The Basic Synchronous Serial Port (BSSP) 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 BSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI™)
- Inter-Integrated Circuit (I^2C™)
  - Slave mode
  - I/O slope control, Start and Stop bits to ease software implementation of Master and Multi-master modes

I^2C is a trademark of Philips Corporation.

16.2 Control Registers

Register 16-1: SSPSTAT: Synchronous Serial Port Status Register

U-0	U-0	R-0	R-0	R-0	R-0	R-0	R-0
—	—	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

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

bit 5 **D/$\overline{A}$**: Data/$\overline{\text{Address}}$ 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 SSP module is disabled)

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 SSP module is disabled)

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 $\overline{ACK}$ bit.

1 = Read
0 = Write

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 = Transmit in progress, SSPBUF is full
0 = Transmit complete, SSPBUF is empty

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

Register 16-2: SSPCON: Synchronous Serial Port Control Register

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	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL**: Write Collision Detect bit

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. 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.

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. SSPOV must be cleared in software in either mode.
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 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

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 mode:
SCK release control
1 = Enable clock
0 = Holds clock low (clock stretch) (Used to ensure data setup time)

Register 16-2: SSPCON: Synchronous Serial Port Control Register (Cont'd)

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. $\overline{SS}$ pin control enabled.
0101 = SPI slave mode, clock = SCK pin. $\overline{SS}$ pin control disabled. $\overline{SS}$ can be used as I/O pin
0110 = I^2C slave mode, 7-bit address
0111 = I^2C slave mode, 10-bit address
1000 = Reserved
1001 = Reserved
1010 = Reserved
1011 = I^2C Firmware controlled Master mode (slave idle)
1100 = Reserved
1101 = Reserved
1110 = I^2C Firmware controlled Multi-Master mode,
7-bit address with start and stop bit interrupts enabled
1111 = I^2C Firmware controlled Master mode,
10-bit address with start and stop bit interrupts enabled

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

16.3 SPI™ Mode

The SPI mode allows 8-bits of data to be synchronously transmitted and received simultaneously. 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 ($\overline{SS}$)

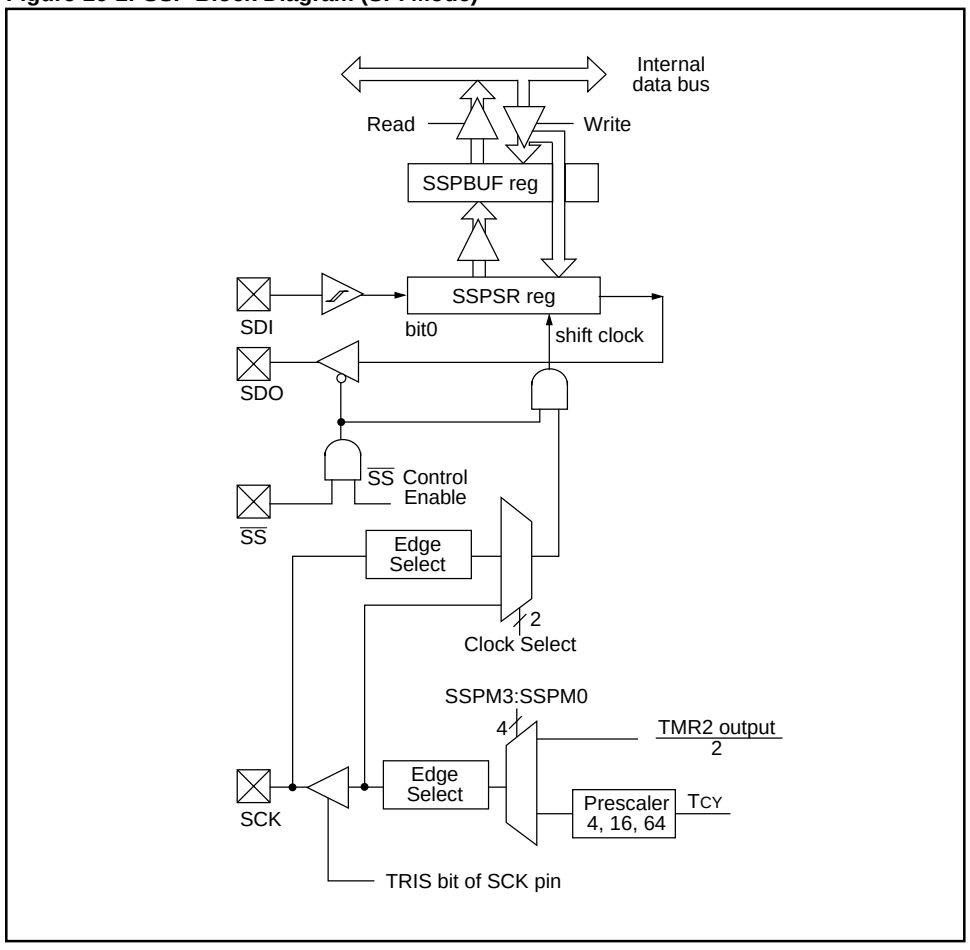
16.3.1 Operation

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits in the SSPCON register (SSPCON<5:0>). 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 (Output/Input data on the Rising/Falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select Mode (Slave mode only)

Figure 16-1 shows the block diagram of the SSP module, when in SPI mode.

Figure 16-1: SSP Block Diagram (SPI Mode)



SPI is a trademark of Motorola Corporations.

The SSP 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 previously written to the SSPSR, until the received data is ready. Once the 8-bits of data have been received, that information is moved to the SSPBUF register. Then the buffer full detect bit, BF (SSPSTAT <0>), and interrupt flag bit, SSPIF, are set. This double buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was received. Any write to the SSPBUF register during transmission/reception of data will be ignored, and the write collision detect bit, WCOL (SSPCON<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, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally the SSP Interrupt is used to determine when the transmission/reception has completed. The SSPBUF can then be read (if data is meaningful) and/or the SSPBUF (SSPSR) can be 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 16-1 shows the loading of the SSPBUF (SSPSR) for data transmission. The shaded instruction is only required if the received data is meaningful (some SPI applications are transmit only).

Example 16-1: Loading the SSPBUF (SSPSR) Register

```
     BCF   STATUS, RP1    ;Specify Bank1
     BSF   STATUS, RP0    ;
LOOP BTFSS SSPSTAT, BF    ;Has data been received (transmit complete)?
     GOTO  LOOP           ;No
     BCF   STATUS, RP0    ;Specify Bank0
     MOVF  SSPBUF, W      ;W reg = contents of SSPBUF
     MOVWF RXDATA         ;Save in user RAM, if data is meaningful
     MOVF  TXDATA, W      ;W reg = contents of TXDATA
     MOVWF SSPBUF         ;New data to xmit
```

The SSPSR is not directly readable or writable, and can only be accessed from addressing the SSPBUF register. Additionally, the SSP status register (SSPSTAT) indicates the various status conditions.

16.3.2 Enabling SPI I/O

To enable the serial port, SSP enable bit, SSPEN (SSPCON<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit which re-initializes the SSPCON register, and then set the SSPEN bit. This configures the SDI, SDO, SCK, and $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, they must have their data direction bits (in the TRIS register) appropriately programmed. That is:

- SDI must have the TRIS bit set
- SDO must have the TRIS bit cleared
- SCK (Master mode) must have the TRIS bit cleared
- SCK (Slave mode) must have the TRIS bit set
- $\overline{SS}$ must have the TRIS bit set

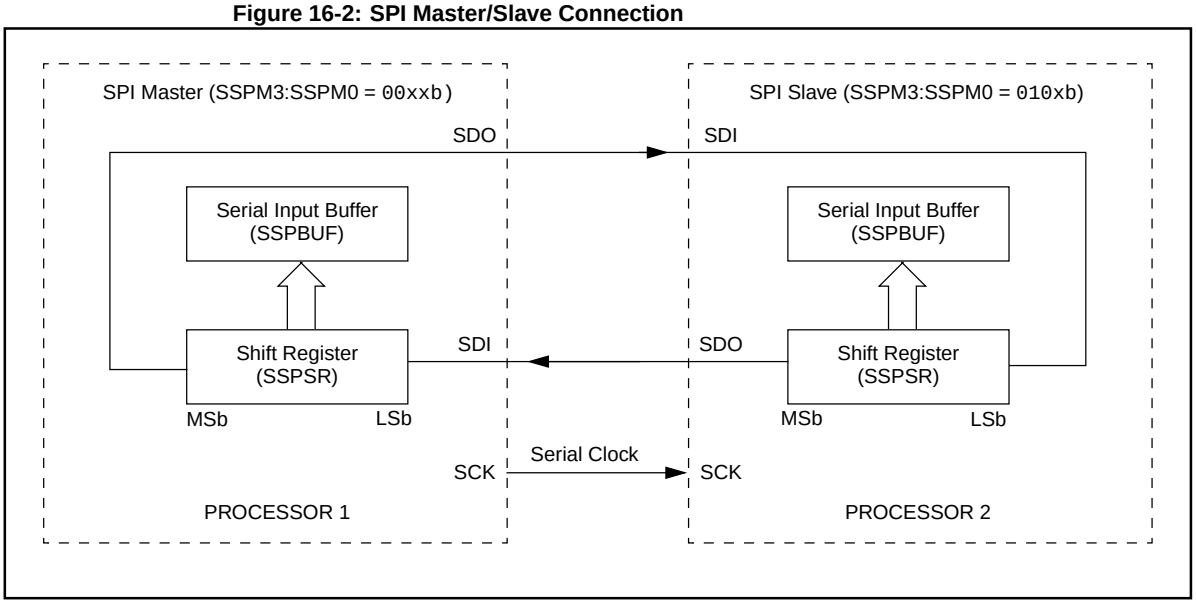
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value. An example would be in master mode where you are only sending data (to a display driver), then both SDI and $\overline{SS}$ could be used as general purpose outputs by clearing their corresponding TRIS register bits.

16.3.3 Typical Connection

Figure 16-2 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 16-2: SPI Master/Slave Connection



16.3.4 Master Operation

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave (Processor 2) wishes to broadcast data by the software protocol.

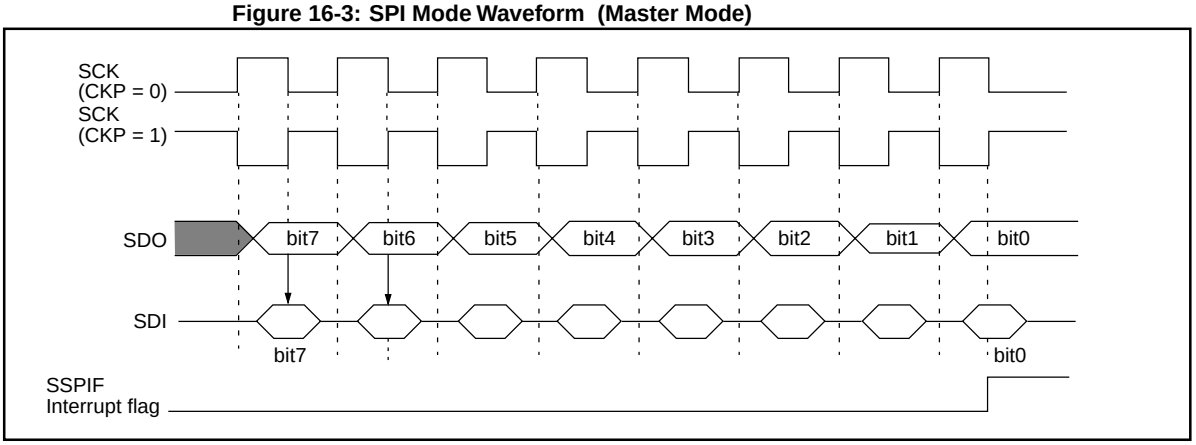
In master mode the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As 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 the CKP bit (SSPCON<4>). This then would give waveforms for SPI communication as shown in Figure 16-5 and Figure 16-5 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 data rate of 5 Mbps (at 20 MHz).

Figure 16-3: SPI Mode Waveform (Master Mode)



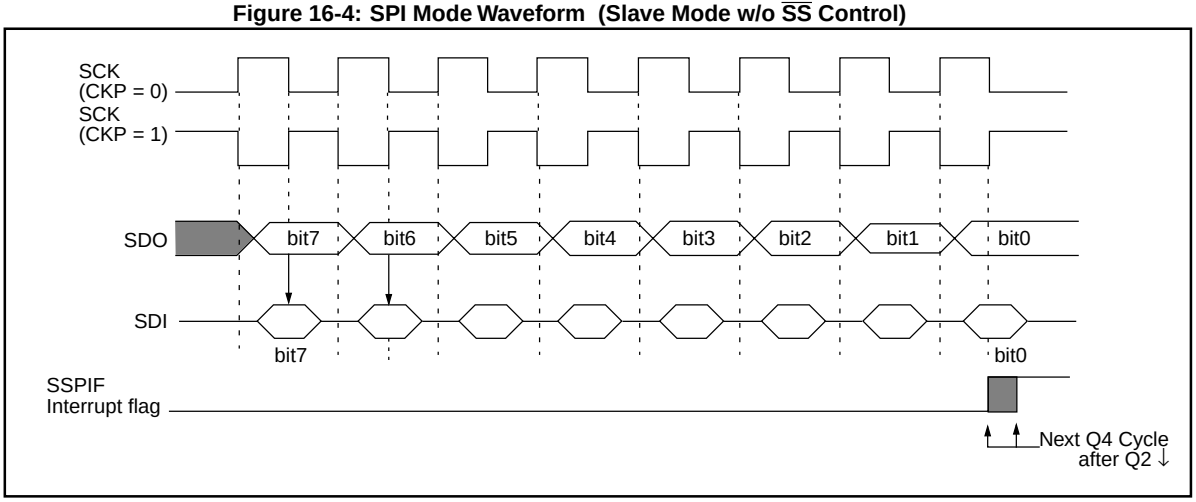
16.3.5 Slave Operation

In slave mode, the data is transmitted and received as the external clock pulses appear on SCK. When the last bit is latched the SSPIF interrupt flag bit is set.

The clock polarity is selected by appropriately programming the CKP bit (SSPCON<4>). This then would give waveforms for SPI communication as shown in Figure 16-5 and Figure 16-5 where the MSb is transmitted first. When in slave mode the external clock must meet the minimum high and low times.

In sleep mode, the slave can transmit and receive data and wake the device from sleep if the interrupt is enabled.

Figure 16-4: SPI Mode Waveform (Slave Mode w/o $\overline{SS}$ Control)



16.3.6 Slave Select Mode

The $\overline{SS}$ pin allows a synchronous slave mode. The SPI must be in slave mode (SSPCON<3:0> = 04h) and the TRIS bit must be set the for the synchronous slave mode to be enabled. 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. If the $\overline{SS}$ pin is taken low without resetting SPI mode, the transmission will continue from the point at which it was taken high. To clear the bit counter the Basic SSP module must be disabled and then re-enabled. External pull-up/pull-down resistors may be desirable, depending on the application.

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 16-5: SPI Mode Waveform (Slave Mode with $\overline{ss}$ Control)

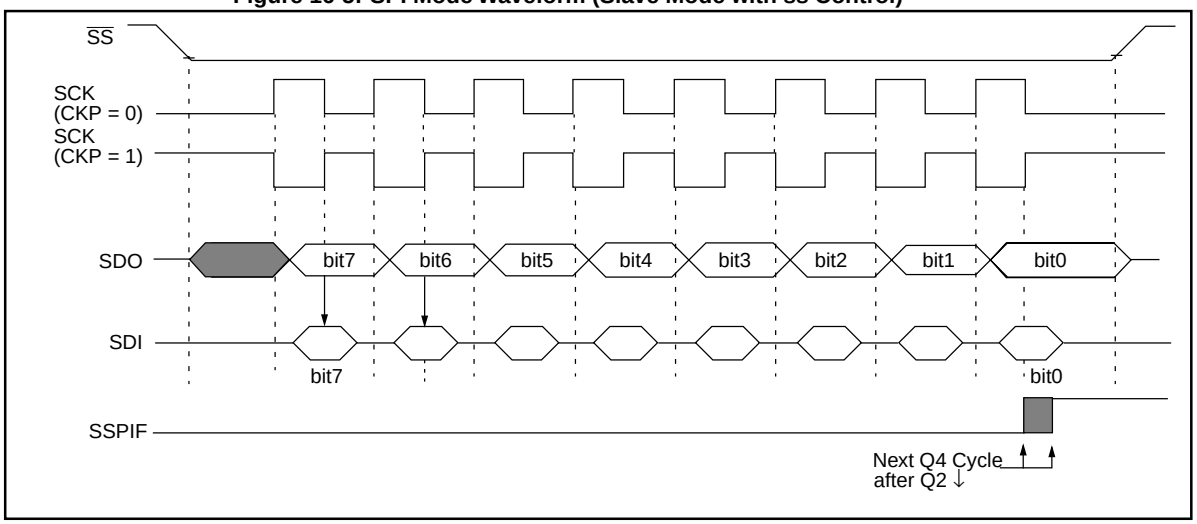
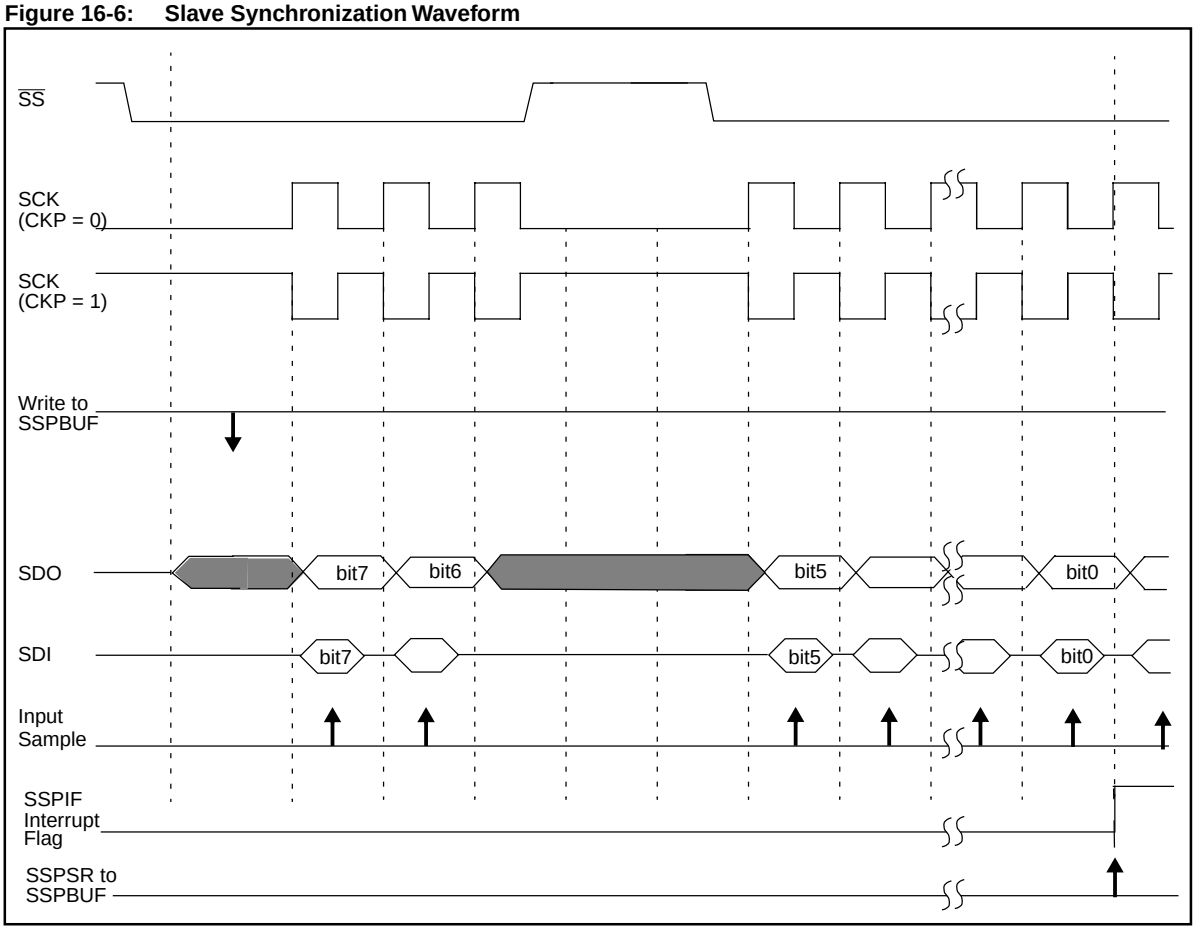


Figure 16-6: Slave Synchronization Waveform



16.3.7 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 SSP interrupt flag bit will be set and if enabled will wake the device from sleep.

16.3.8 Effects of a Reset

A reset disables the SSP module and terminates the current transfer.

Table 16-1: Registers Associated with SPI Operation

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
INTCON	GIE	PEIE	T0IE	INTE	RBIE(2)	T0IF	INTF	RBIF(2)	0000 000x	0000 000u
PIR	SSPIF (1)								0	0
PIE	SSPIE (1)								0	0
SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
SSPSTAT	—	—	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'.
Shaded cells are not used by the SSP in SPI mode.

Note 1: The position of this bit is device dependent.
2: These bits can also be named GPIE and GPIF.

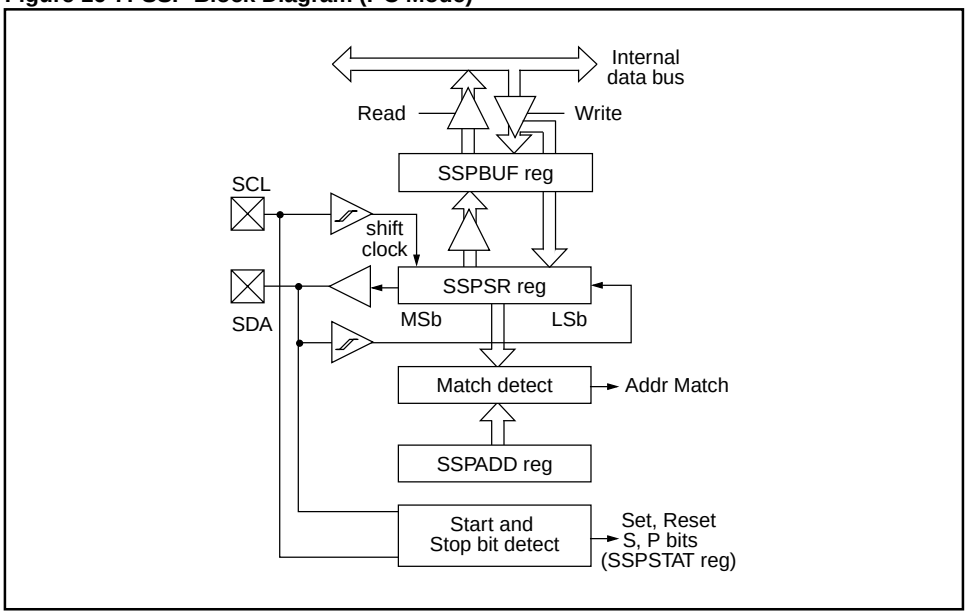
16.4 SSP I²C Operation

The SSP module in I²C mode fully implements all slave functions, except General Call Support, and provides interrupts on start and stop bits in hardware to facilitate software implementations of the master functions. The SSP module implements the standard and fast mode specifications as well as 7-bit and 10-bit addressing. Appendix A gives an overview of the I²C bus specification.

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 user must configure these pins as inputs through the TRIS bits. The SSP module functions are enabled by setting SSP Enable bit, SSPEN (SSPCON<5>).

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 16-7: SSP Block Diagram (I²C Mode)



The SSP module has five registers for I^2C operation. They are:

- SSP Control Register (SSPCON)
- SSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer (SSPBUF)
- SSP Shift Register (SSPSR) - Not directly accessible
- SSP Address Register (SSPADD)

The SSPCON register allows control of the I^2C operation. Four mode selection bits (SSPCON<3:0>) allow one of the following I^2C modes to be selected:

- I^2C Slave mode (7-bit address)
- I^2C Slave mode (10-bit address)
- I^2C Firmware controlled Multi-Master mode, 7-bit address (start and stop bit interrupts enabled)
- I^2C Firmware controlled Multi-Master mode, 10-bit address (start and stop bit interrupts enabled)
- I^2C Firmware controlled Master mode, slave is idle

Before selecting any I^2C mode, the SCL and SDA pins must be programmed to inputs by setting the appropriate TRIS bits. Selecting an I^2C mode, by setting the SSPEN bit, enables the SCL and SDA pins to be used as the clock and data lines in I^2C 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 SSPSTAT register is read only.

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 the SSPIF flag bit is set. If another complete byte is received before the SSPBUF register is read, a receiver overflow has occurred and bit SSPOV (SSPCON<6>) is set.

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).

16.4.1 Slave Mode

In slave mode, the SCL and SDA pins must be configured as inputs (TRIS bits set). The SSP 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 SSP 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 (SSPCON<6>), was set before the transfer was received.

In this case, the SSPSR register value is not loaded into the SSPBUF, but bit SSPIF and SSPOV bits are set. Table 16-2 shows what happens when a data transfer byte is received, given the status of the BF and SSPOV bits. The shaded cells show the condition where user software did not properly clear the overflow condition. The BF flag bit is cleared by reading the SSPBUF register while the SSPOV bit 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 SSP module are given in parameter 100 and parameter 101 of the **"Electrical Specifications"** section.

16.4.1.1 Addressing

Once the SSP 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 eight SCL pulse.

b) The buffer full bit, BF, is set on the falling edge of the eight SCL pulse.

c) An $\overline{ACK}$ pulse is generated.

d) SSP interrupt flag bit, SSPIF, is set (interrupt is generated if enabled) - on the falling edge of the ninth 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. The R/$\overline{W}$ bit (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 (the SSPIF, BF, and UA (SSPSTAT<1>) bits are set).
2. Update the SSPADD register with second (low) byte of Address (clears the UA bit and releases the SCL line).
3. Read the SSPBUF register (clears the BF bit) and clear the SSPIF flag bit.
4. Receive second (low) byte of Address (the SSPIF, BF, and UA bits are set).
5. Update the SSPADD register with the first (high) byte of Address. This will clear the UA bit and release the SCL line.
6. Read the SSPBUF register (clears the BF bit) and clear the SSPIF flag bit.
7. Receive repeated START condition.
8. Receive first (high) byte of Address (the SSPIF and BF bits are set).
9. Read the SSPBUF register (clears the BF bit) and clear the SSPIF flag bit.

Note:	Following the RESTART 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.

Table 16-2: Data Transfer Received Byte Actions

Status bits as data transfer is received				
BF	**SSPOV**	**SSPSR → SSPBUF**	**Generate $\overline{ACK}$ pulse**	**Set bit SSPIF (SSP Interrupt occurs if enabled)**
0	0	Yes	Yes	Yes
1	0	No	No	Yes
1	1	No	No	Yes
0	1	Yes	No	Yes

Note: Shaded cells show the conditions where the user software did not properly clear the overflow condition

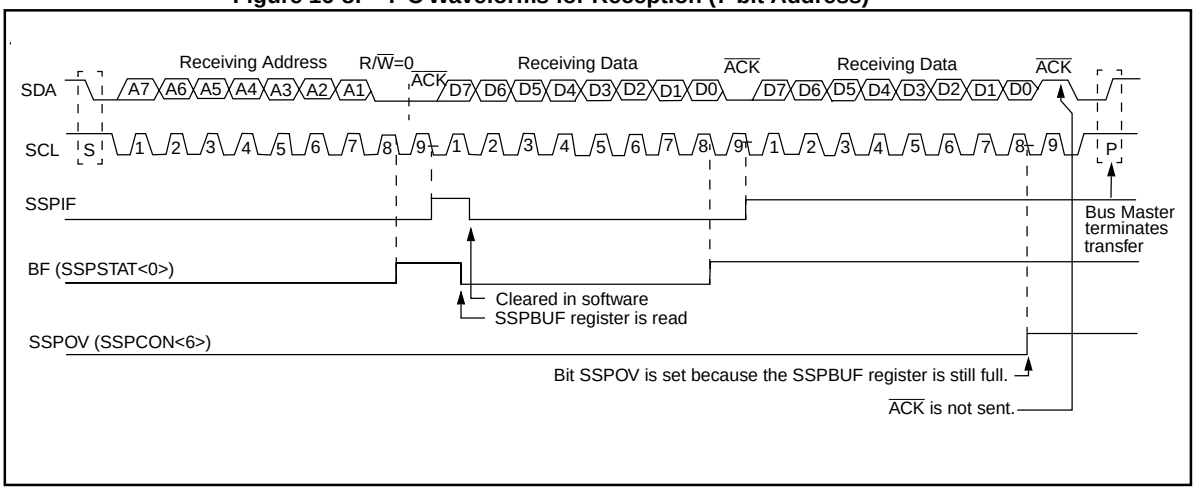
16.4.1.2 Reception

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 the BF bit (SSPSTAT<0>) is set or the SSPOV bit (SSPCON<6>) is set.

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.

Figure 16-8: I^2C Waveforms for Reception (7-bit Address)



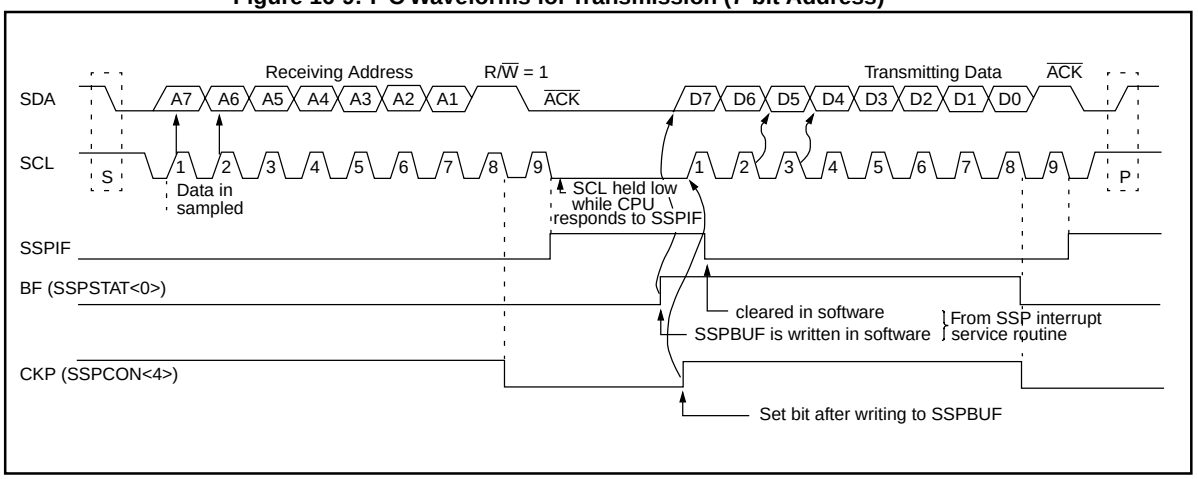
16.4.1.3 Transmission

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 the SCL pin should be enabled by setting the CKP bit (SSPCON<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 16-9).

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 16-9: I^2C Waveforms for Transmission (7-bit Address)



16.4.1.4 Clock Arbitration

Clock arbitration has the SCL pin to inhibit the master device from sending the next clock pulse. The SSP module in I^2C slave mode will hold the SCL pin low when the CPU needs to respond to the SSP interrupt (SSPIF bit is set and the CKP bit is cleared). The data that needs to be transmitted will need to be written to the SSPBUF register, and then the CKP bit will need to be set to allow the master to generate the required clocks.

16.4.2 Master Mode (Firmware)

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 SSP 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 clearing the corresponding TRIS bit(s). The output level is always low, irrespective of the value(s) in PORT. So when transmitting data, a '1' data bit must have the TRIS bit set (input) and a '0' data bit must have the TRIS bit cleared (output). The same scenario is true for the SCL line with the TRIS bit.

The following events will cause the SSPIF Interrupt Flag bit to be set (SSP Interrupt if enabled):

- START condition
- STOP condition
- Data transfer byte transmitted/received

Master mode of operation can be done with either the slave mode idle (SSPM3:SSPM0 = 1011) or with the slave active (SSPM3:SSP0 = 1110 or 1111). When the slave modes are enabled, the software needs to differentiate the source(s) of the interrupt.

16.4.3 Multi-Master Mode (Firmware)

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 SSP module is disabled. Control of the I^2C bus may be taken when the P bit (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 to see if the signal level is the expected output level. This check only needs to be done when a high level is output. If a high level is expected and a low level is present, the device needs to release the SDA and SCL lines (set the TRIS bits). There are two stages where this arbitration can be lost, they are:

- Address Transfer
- Data Transfer

When the slave logic is enabled, the slave continues to receive. If arbitration was lost during the address transfer stage, communication to the device may be in progress. If addressed an $\overline{\text{ACK}}$ pulse will be generated. If arbitration was lost during the data transfer stage, the device will need to re-transfer the data at a later time.

16.4.4 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).

16.4.5 Effect of a Reset

A reset disables the SSP module and terminates the current transfer.

Table 16-3: Registers Associated with I^2C Operation

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
INTCON	GIE	PEIE	T0IE	INTE	RBIE(2)	T0IF	INTF	RBIF(2)	0000 000x	0000 000u
PIR	SSPIF (1)								0	0
PIE	SSPIE (1)								0	0
SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	0000 0000
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
SSPSTAT	—	—	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'.
Shaded cells are not used by SSP in I^2C mode.
Note 1: The position of these bits is device dependent.
2: These bits can also be named GPIE and GPIF.

16.5 Initialization

Example 16-2: SPI Master Mode Initialization

```
    CLRF   STATUS       ; Bank 0
    CLRF   SSPSTAT      ; Clear status bits
    MOVLW  0x31         ; Set up SPI port, Master mode, CLK/16,
    MOVWF  SSPCON       ;   Data xmit on rising edge
                        ;   Data sampled in middle
    BSF    STATUS, RP0  ; Bank 1
    BSF    PIE1, SSPIE   ; Enable SSP interrupt
    BCF    STATUS, RP0  ; Bank 0
    BSF    INTCON, GIE  ; Enable, enabled interrupts
    MOVLW  DataByte     ; Data to be Transmitted
                        ;   Could move data from RAM location
    MOVWF  SSPBUF       ; Start Transmission
```

16.5.1 SSP Module / Basic SSP Module Compatibility

When changing from the SSP Module to the Basic SSP module, the SSPSTAT register contains two additional control bits. These bits are:

- SMP, SPI data input sample phase
- CKE, SPI Clock Edge Select

To be compatible with the SPI of the Basic SSP module, these bits must be appropriately configured. If these bits are not at the states shown in Table 16-4, improper SPI communication should be expected. If the SSP module uses a different configuration then shown in Table 16-4, the Basic SSP module can not be used to implement that mode. That mode may be implemented in software.

Table 16-4: New Bit States for Compatibility

Basic SSP Module	SSP Module		
CKP	CKP	CKE	SMP
1	1	0	0
0	0	0	0

16.6 Design Tips

Question 1: ***Using SPI mode, I do not seem able to talk to an SPI device.***

Answer 1:

Ensure that you are using the correct SPI mode for that device. This SPI supports two of the four SPI modes so ensure that the SPI device that you are trying to interface to is compatible with one of these two modes. Check the clock polarity and the clock phase.

If the device is not compatible, switch to one of the Microchip devices that has the SSP module, and that should solve this.

Question 2: ***Using I^2C mode, I do not seem able to make the master mode work.***

Answer 2:

This SSP module does not have master mode fully automated in hardware, see Application Note AN578 for software which uses the SSP module to implement master mode. If you require a fully automated Hardware implementation of I^2C master mode, please refer to the Microchip Line Card for devices that have the Master SSP module.

Note:	At the time of printing only the High-end family of devices (PIC17CXXX) have devices with the Master SSP module implemented.

Question 3: ***Using I^2C mode, I write data to the SSPBUF register, but the data did not transmit.***

Answer 3:

Ensure that you set the CKP bit to release the I^2C clock.

16.7 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to this section are:

Title	Application Note #
Use of the SSP Module in the I ^{2}C Multi-Master Environment.	AN578
Using Microchip 93 Series Serial EEPROMs with Microcontroller SPI Ports	AN613
Software Implementation of I^{2}C Bus Master	AN554
Use of the SSP module in the Multi-Master Environment	AN578
Interfacing PIC16C64/74 to Microchip SPI Serial EEPROM	AN647
Interfacing a Microchip PIC16C92x to Microchip SPI Serial EEPROM	AN668

16.8 Revision History

Revision A

This is the initial revision of the Basic SSP module description.

Section 17. Master Synchronous Serial Port (MSSP)

HIGHLIGHTS

This section of the manual contains the following major topics:

Note: At present NO Mid-Range MCU devices are available with this module. Devices are planned, but there is no schedule for availability. Please refer to Microchip's Web site or BBS for release of Product Briefs. You will be able to find out the details and the features for new devices.

This module is available on Microchip's High End family (PIC17CXXX). Please refer to Microchip's Web site, BBS, Regional Sales Office, or Factory Representatives.

I^2C is a trademark of Philips Corporation.

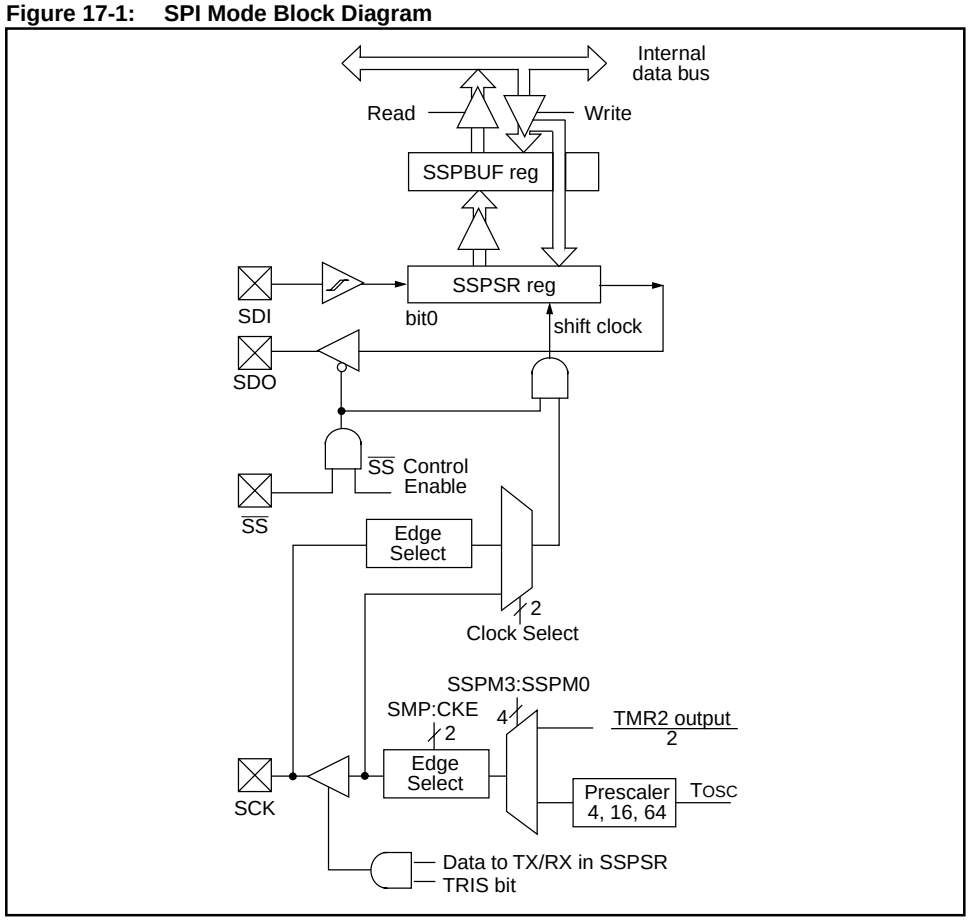
17.1 Introduction

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™)
  - Full Master Mode
  - Slave mode (with general address call)

Figure 17-1 shows a block diagram for the SPI mode, while Figure 17-2, and Figure 17-3 show the block diagrams for the two different I^2C modes of operation.

Figure 17-1: SPI Mode Block Diagram



SPI is a trademark of Motorola Corporation.
I^2C is a trademark of Philips Corporation.

Figure 17-2: I^2C Slave Mode Block Diagram

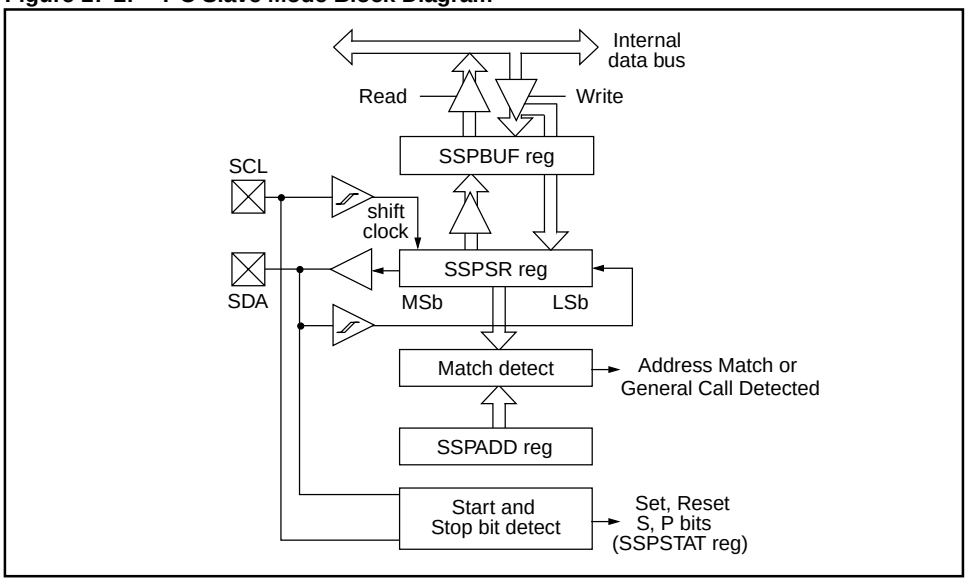
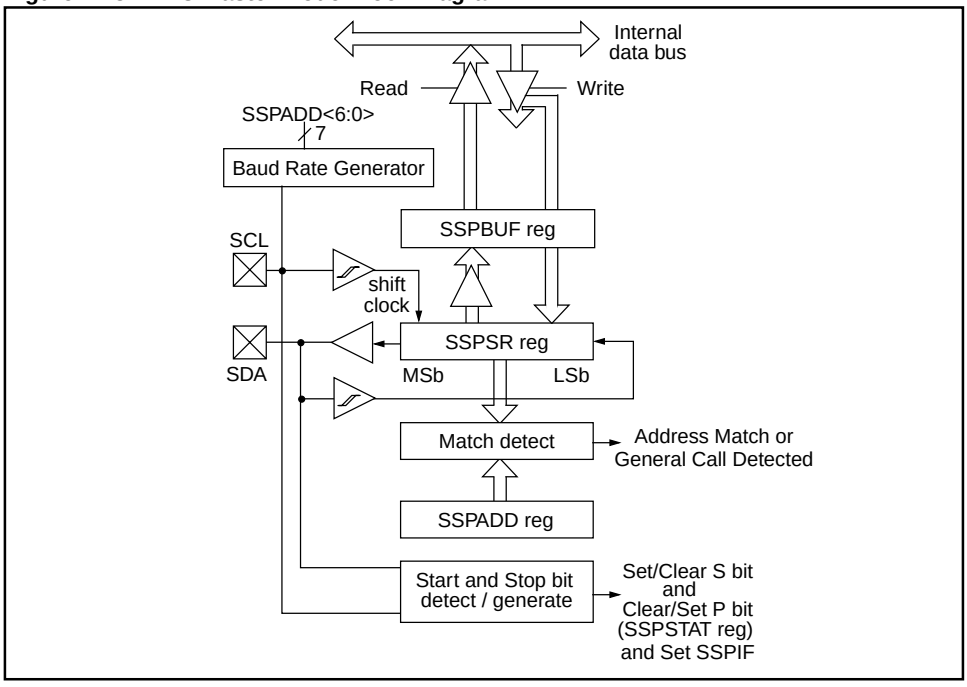


Figure 17-3: I^2C Master Mode Block Diagram



17.2 Control Register

Register 17-1: SSPSTAT: SSP Status Register

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

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 SSP 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 SSP 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 $\overline{ACK}$ 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 SSP 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

Register 17-1: SSPSTAT: SSP Status Register (Cont'd)

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

Register 17-2: SSPCON1: SSP Control Register1

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	SSPM1	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.

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. SSPOV must be cleared in software in either 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 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

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

Register 17-2: SSPCON1: SSP Control Register1 (Cont'd)

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. $\overline{SS}$ pin control enabled.
0101 = SPI slave mode, clock = SCK pin. $\overline{SS}$ pin control disabled. $\overline{SS}$ 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

Register 17-3: SSPCON2: SSP Control Register2

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

17.3 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 ($\overline{SS}$)

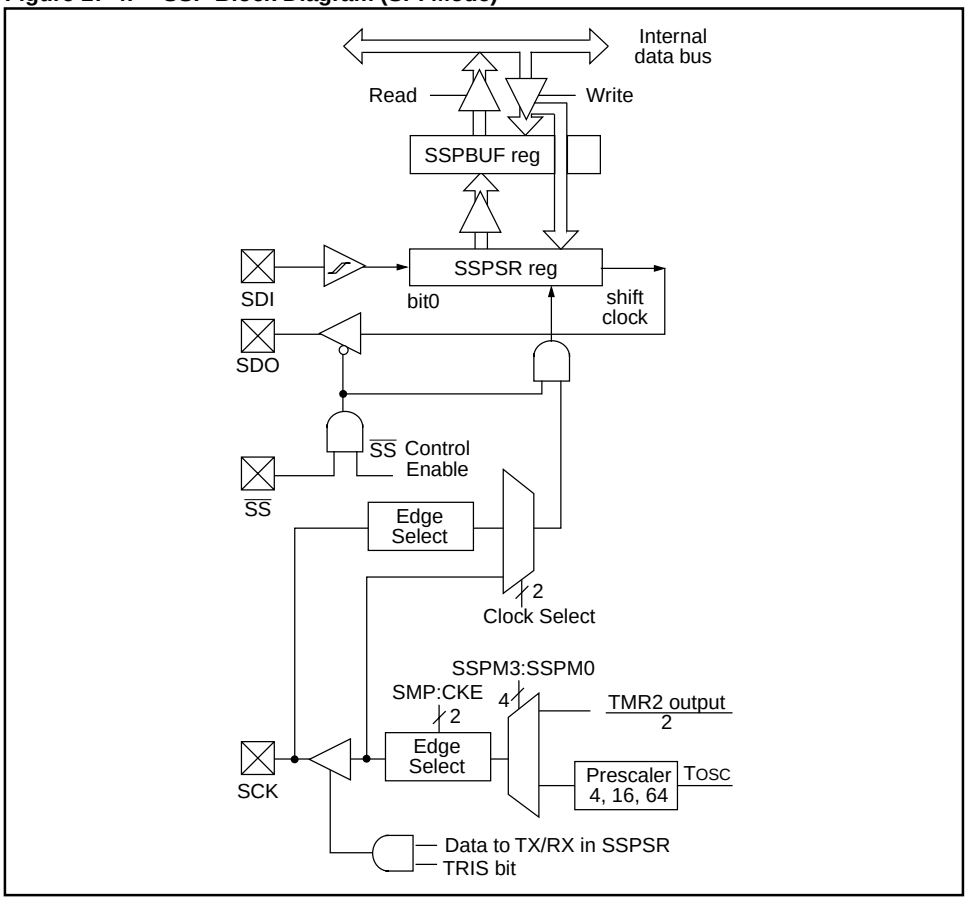
17.3.1 Operation

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 17-4 shows the block diagram of the SSP module, when in SPI mode.

Figure 17-4: SSP Block Diagram (SPI Mode)



The SSP 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, 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, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally the SSP 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 17-1 shows the loading of the SSPBUF (SSPSR) for data transmission.

Example 17-1: Loading the SSPBUF (SSPSR) Register

```
     BCF   STATUS, RP1    ;Specify Bank1
     BSF   STATUS, RP0    ;
LOOP BTFSS SSPSTAT, BF    ;Has data been received (transmit complete)?
     GOTO  LOOP           ;No
     BCF   STATUS, RP0    ;Specify Bank0
     MOVF  SSPBUF, W      ;W reg = contents of SSPBUF
     MOVWF RXDATA         ;Save in user RAM, if data is meaningful
     MOVF  TXDATA, W      ;W reg = contents of TXDATA
     MOVWF 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 SSP status register (SSPSTAT) indicates the various status conditions.

17.3.2 Enabling SPI I/O

To enable the serial port, SSP Enable bit, SSPEN (SSPCON1<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize the SSPCON registers, and then set the SSPEN bit. 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 TRIS register) appropriately programmed. That is:

- SDI is automatically controlled by the SPI module
- SDO must have TRIS bit cleared
- SCK (Master mode) must have TRIS bit cleared
- SCK (Slave mode) must have TRIS bit set
- $\overline{SS}$ must have TRIS bit set

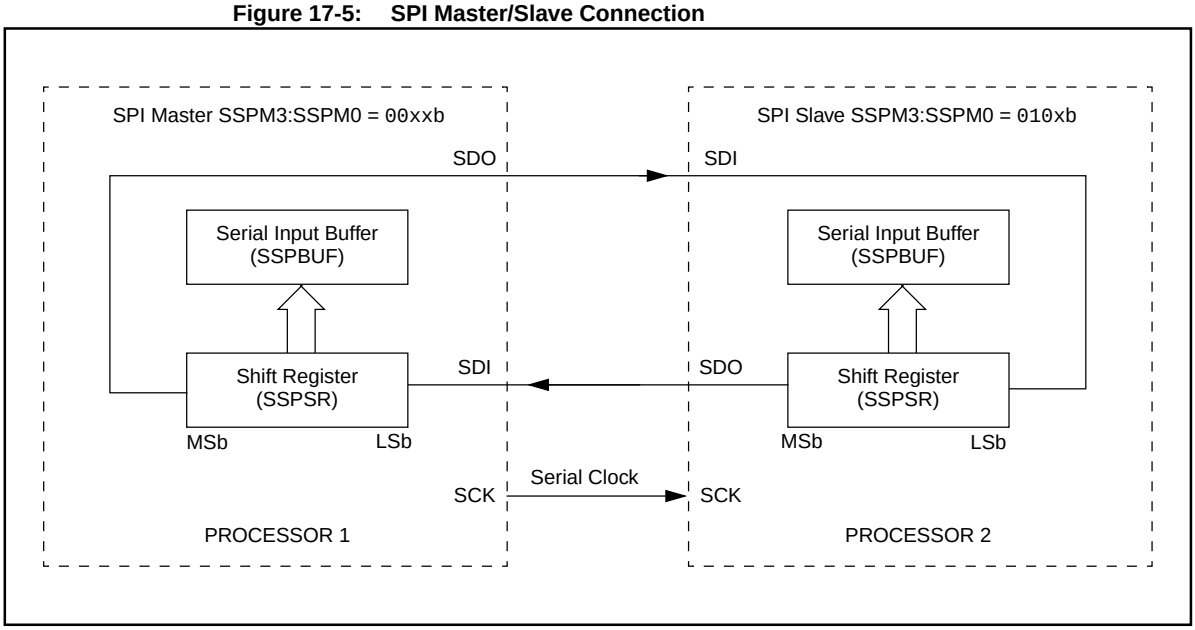
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

17.3.3 Typical Connection

Figure 17-5 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 17-5: SPI Master/Slave Connection



17.3.4 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 17-5) 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.

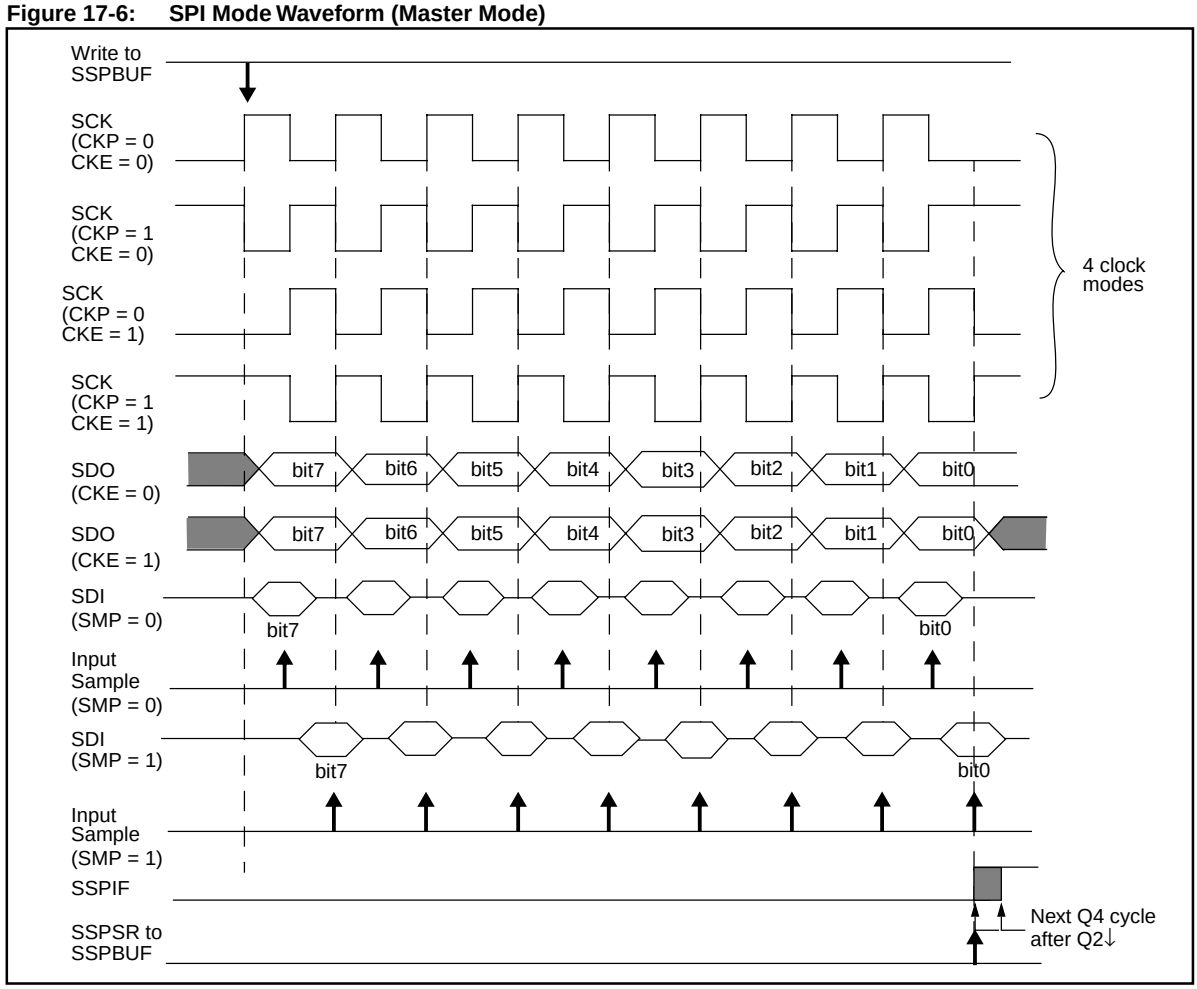
The clock polarity is selected by appropriately programming the CKP bit (SSPCON1<4>). This then would give waveforms for SPI communication as shown in Figure 17-6, Figure 17-8, and Figure 17-9 where the MSb is transmitted first. In master mode, the SPI clock rate (bit rate) is user programmable to be 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 data rate (at 20 MHz) of 8.25 Mbps.

Figure 17-6 Shows the waveforms for master mode. When the CKE bit is set, 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 17-6: SPI Mode Waveform (Master Mode)



17.3.5 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 SSPIF interrupt flag bit 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 receive the device will wake-up from sleep.

17.3.6 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 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 VDD.

Note 2: If the SPI is used in Slave Mode with CKE is set, 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 by 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 17-7: Slave Synchronization Waveform

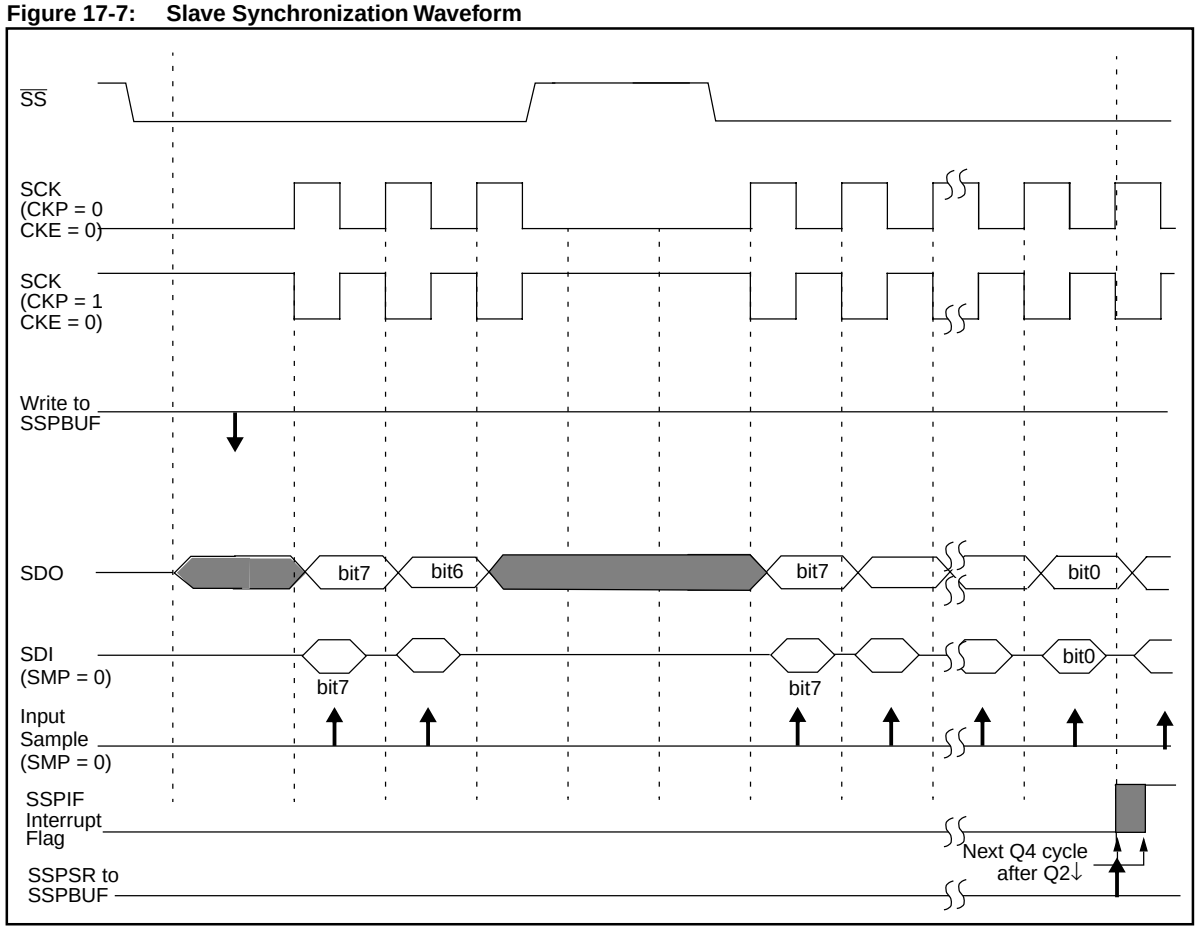


Figure 17-8: SPI Mode Waveform (Slave Mode with CKE = 0)

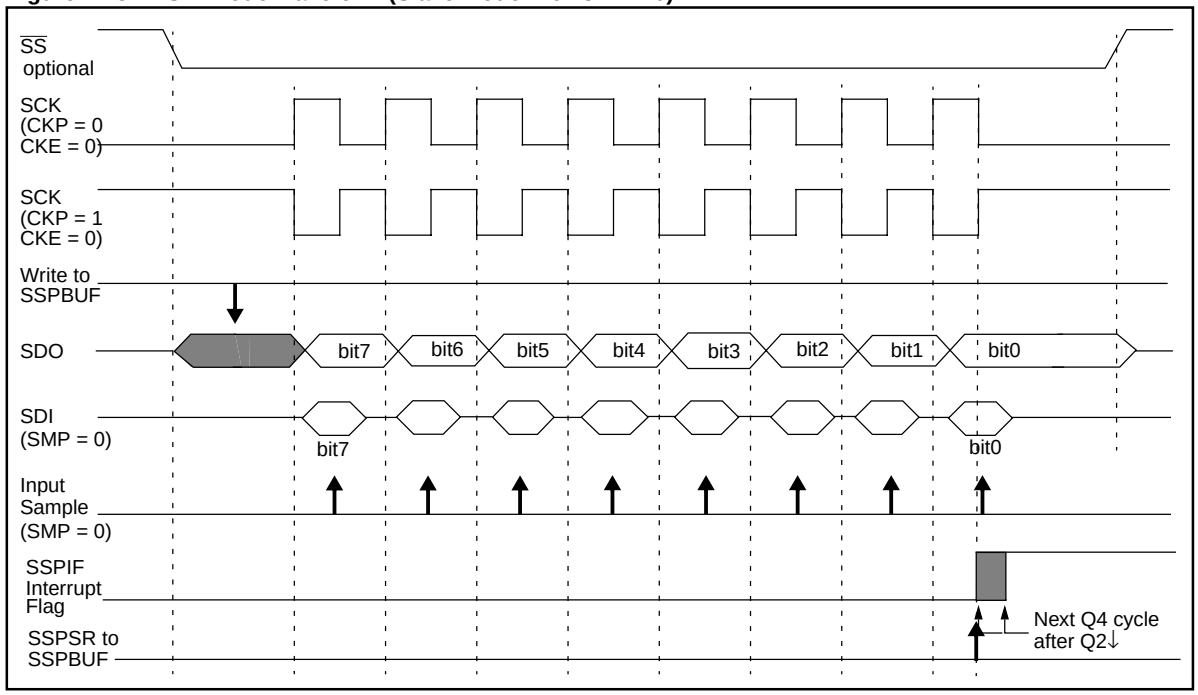
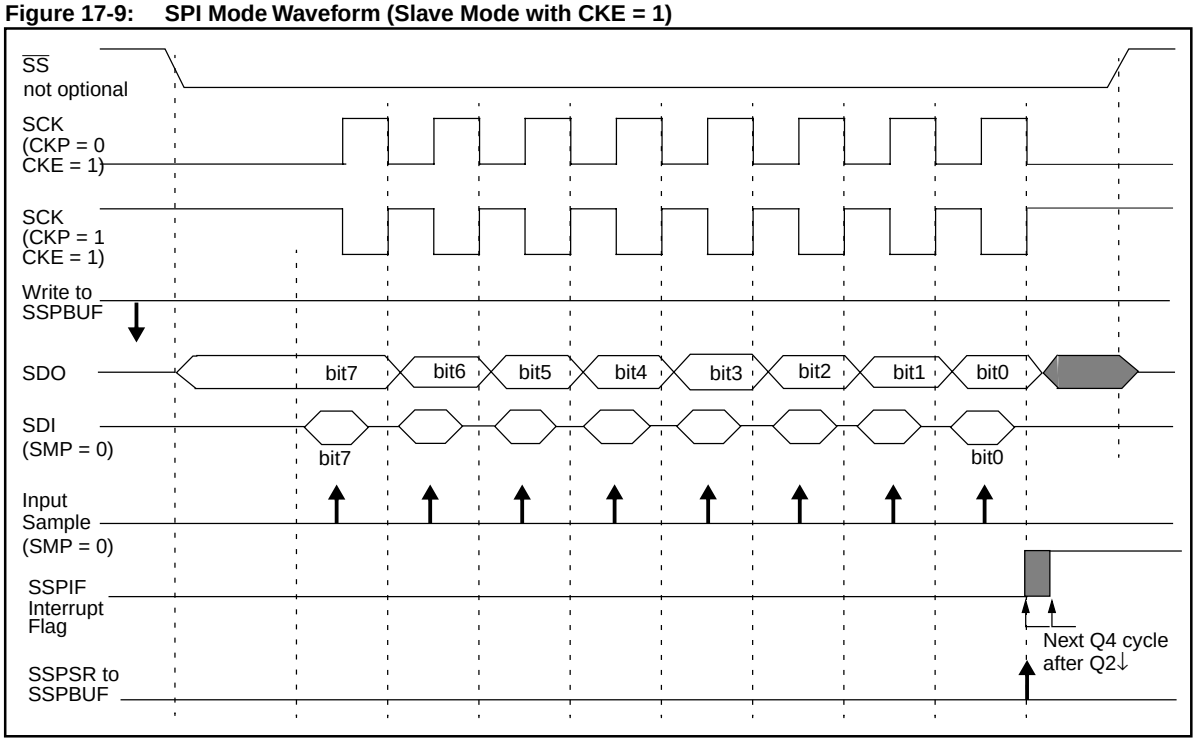


Figure 17-9: SPI Mode Waveform (Slave Mode with CKE = 1)



17.3.7 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.

17.3.8 Effects of a Reset

A reset disables the MSSP module and terminates the current transfer.

Table 17-1: Registers Associated with SPI Operation

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other resets
INTCON	GIE	PEIE	T0IE	INTE	RBIE(2)	T0IF	INTF	RBIF(2)	0000 0000	0000 0000
PIR	SSPIF (1)								0	0
PIE	SSPIE (1)								0	0
SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
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 SPI mode.
Note 1: The position of this bit is device dependent.
2: These bits may also be named GPIE and GPIF.

17.4 SSP 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 SSP module implements the standard mode specifications as well as 7-bit and 10-bit addressing. Appendix A gives an overview of the I²C bus specification.

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 17-10: I²C Slave Mode Block Diagram

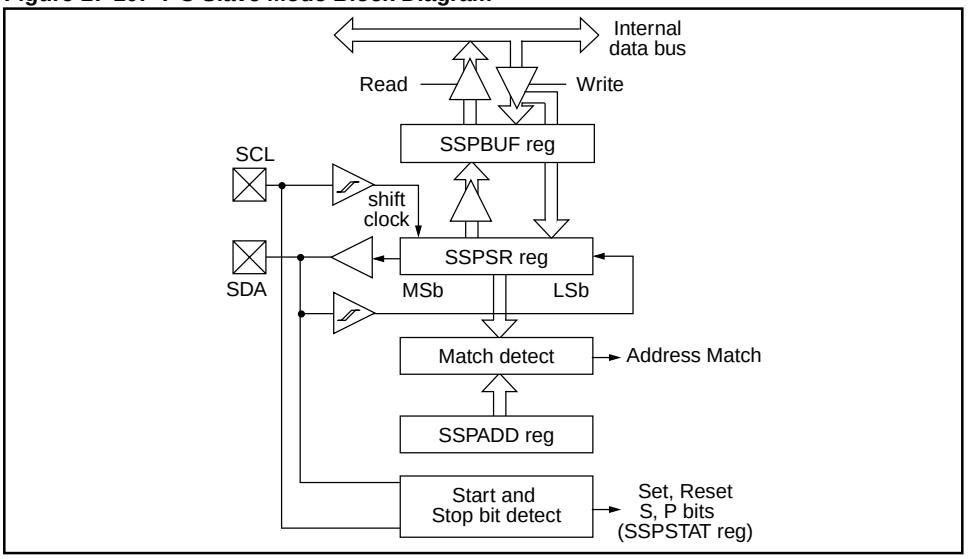
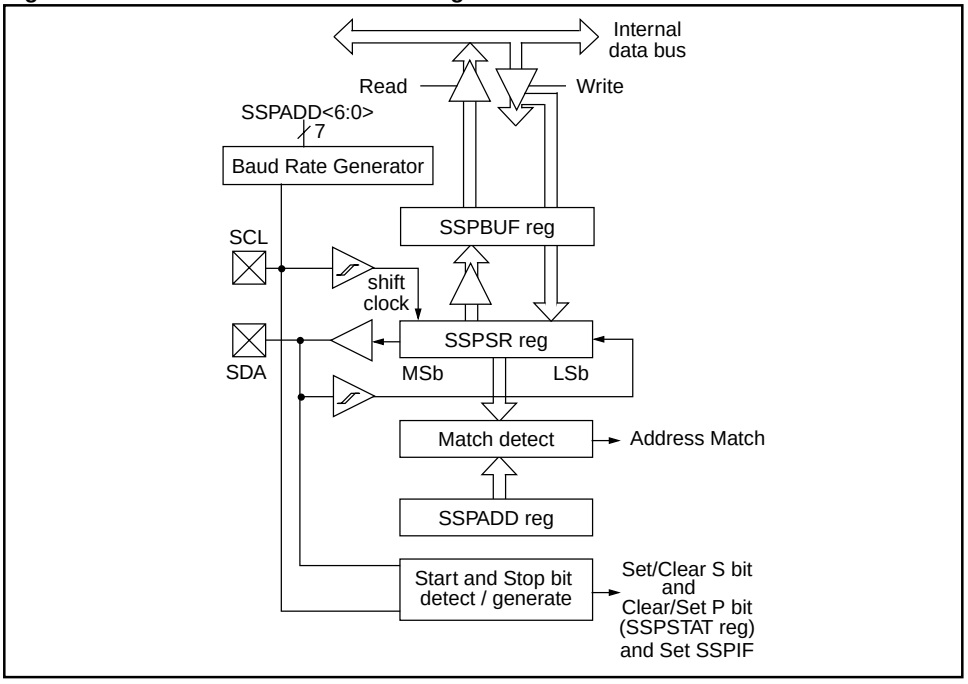


Figure 17-11: I²C Master Mode Block Diagram



Two pins are used for data transfer. These are the SCL pin, which is the clock, and the SDA pin, which is the data. Pins that are on the port are automatically configured when the I^2C mode is enabled. The SSP module functions are enabled by setting SSP Enable bit, SSPEN (SSPCON1<5>).

The SSP module has six registers for I^2C operation. They 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 I^2C operation. Four mode selection bits (SSPCON1<3:0>) allow one of the following I^2C modes to be selected:

- I^2C Slave mode (7-bit address)
- I^2C Slave mode (10-bit address)
- I^2C Master mode, clock = OSC/4 (SSPADD +1)

Before selecting any I^2C mode, the SCL and SDA pins must be programmed to inputs by setting the appropriate TRIS bits. Selecting an I^2C mode, by setting the SSPEN bit, enables the SCL and SDA pins to be used as the clock and data lines in I^2C 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 double 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 the SSPOV bit (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).

17.4.1 Slave Mode

In slave mode, the SCL and SDA pins must be configured as inputs. The SSP 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 SSP 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 the SSPIF and SSPOV bits are set. Table 17-2 shows what happens when a data transfer byte is received, given the status of the BF and SSPOV bits. The shaded cells show the condition where user software did not properly clear the overflow condition. The BF flag bit 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 SSP module is shown in timing parameters 100 and 101 of the **"Electrical Specifications"** section.

17.4.1.1 Addressing

Once the SSP 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 eighth SCL pulse.

b) The buffer full bit, BF, is set on the falling edge of the eighth SCL pulse.

c) An $\overline{ACK}$ pulse is generated.

d) SSP interrupt flag bit, SSPIF, is set (interrupt is generated if enabled) - on the falling edge of the ninth 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. The R/$\overline{W}$ bit (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 (the SSPIF, BF, and UA (SSPSTAT<1>) bits are set).
2. Update the SSPADD register with second (low) byte of Address (clears the UA bit and releases the SCL line).
3. Read the SSPBUF register (clears the BF bit) and clear flag bit SSPIF.
4. Receive second (low) byte of Address (the SSPIF, BF, and UA bits are set).
5. Update the SSPADD register with the first (high) byte of Address. This will clear the UA bit and release the SCL line.
6. Read the SSPBUF register (clears the BF bit) and clear the SSPIF flag bit.
7. Receive repeated START condition.
8. Receive first (high) byte of Address (the SSPIF and BF bits are set).
9. Read the SSPBUF register (clears the BF bit) and clear the SSPIF flag bit.

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.

Table 17-2: Data Transfer Received Byte Actions

Status Bits as Data Transfer is Received				
BF	**SSPOV**	**SSPSR → SSPBUF**	**Generate $\overline{ACK}$ Pulse**	**Set bit SSPIF (SSP Interrupt occurs if enabled)**
0	0	Yes	Yes	Yes
1	0	No	No	Yes
1	1	No	No	Yes
0	1	Yes	No	Yes

Note: Shaded cells show the conditions where the user software did not properly clear the overflow condition

17.4.1.2 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 the BF bit (SSPSTAT<0>) is set or the SSPOV bit (SSPCON1<6>) is set.

An SSP interrupt is generated for each data transfer byte. The SSPIF flag bit 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 bit 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.

17.4.1.3 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 the SCL pin should be enabled by setting the CKP bit (SSPCON1<4>). The master should 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 17-13).

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 17-12: I²C Slave Mode Waveforms for Reception (7-bit Address)

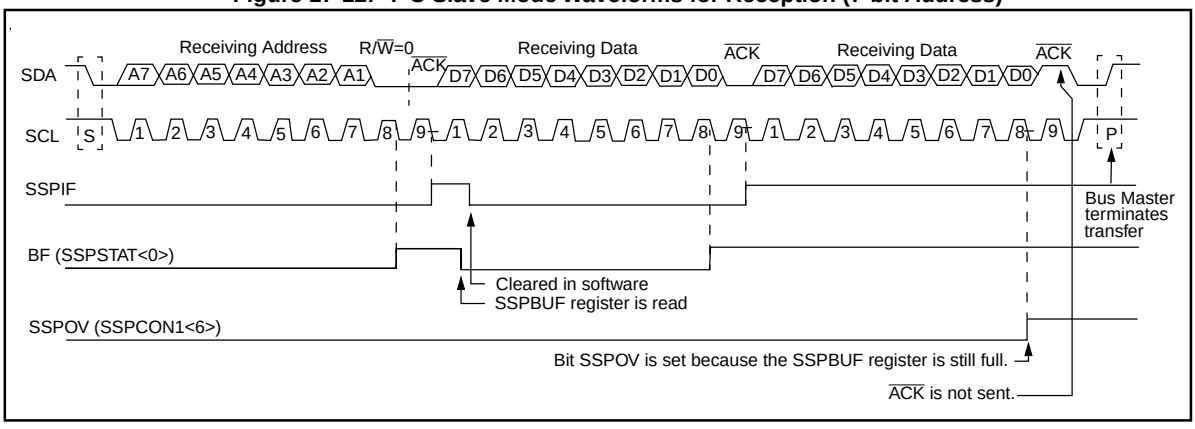


Figure 17-13: I²C Slave Mode Waveforms for Transmission (7-bit Address)

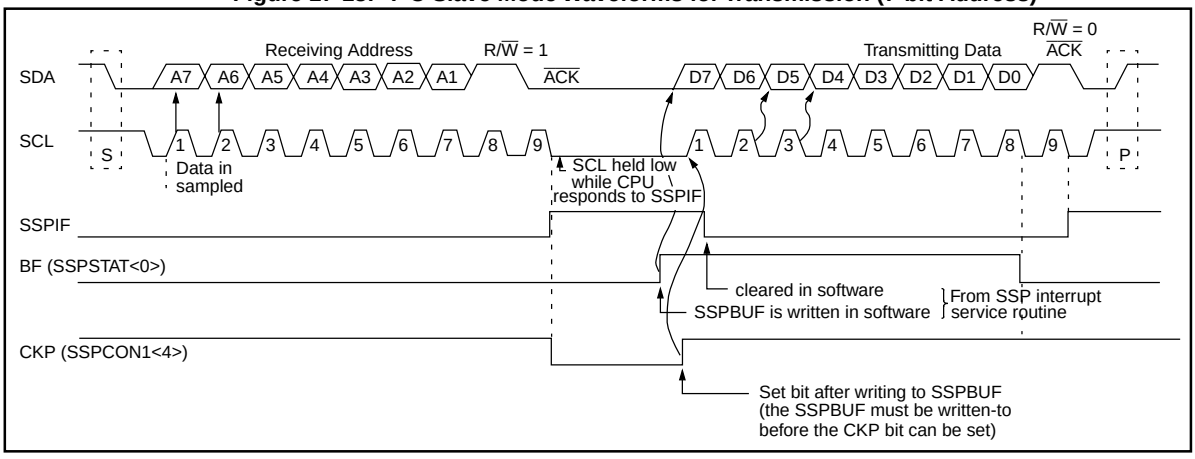


Figure 17-14: I^2C Slave Mode Waveform (Transmission 10-bit Address)

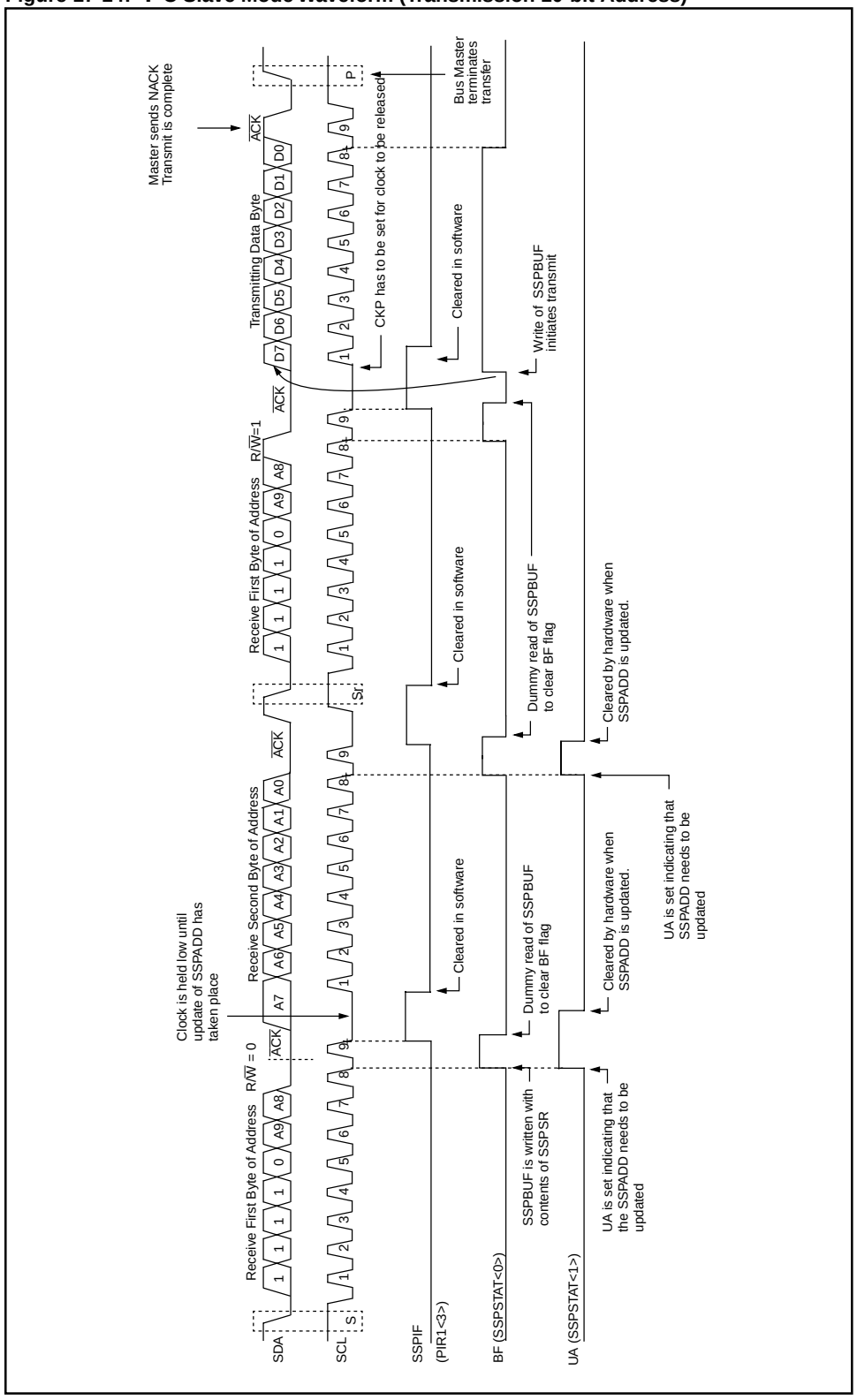
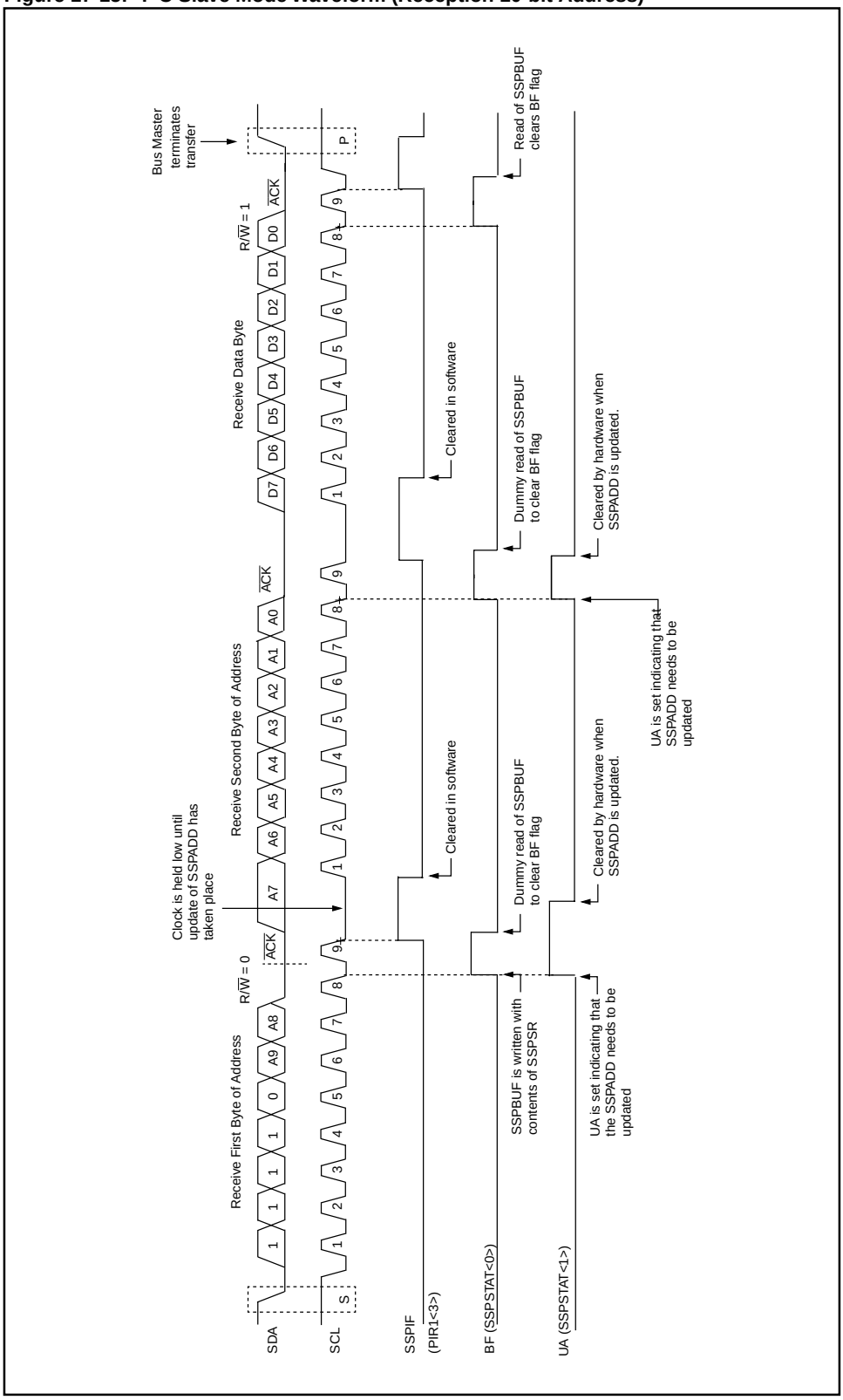


Figure 17-15: I^2C Slave Mode Waveform (Reception 10-bit Address)



17.4.2 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.

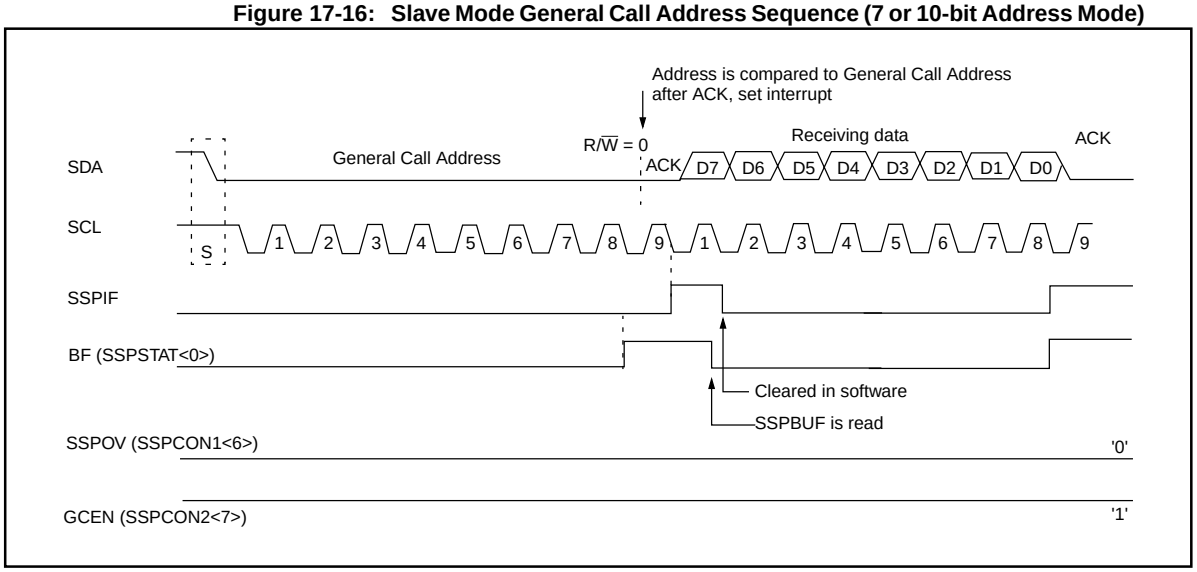
The general call address is recognized when the General Call Enable bit (GCEN) is enabled (SSPCON2<7> set). Following a start-bit detect, 8-bits are shifted into SSPSR and the address is compared against SSPADD, and is also compared to the general call address, fixed in hardware.

If the general call address matches, the SSPSR is transferred to the SSPBUF, the BF flag bit is set (eight bit), and on the falling edge of the ninth bit ($\overline{ACK}$ bit) the SSPIF interrupt flag bit 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 the GCEN bit 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 17-16).

Figure 17-16: Slave Mode General Call Address Sequence (7 or 10-bit Address Mode)



17.4.3 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 MSSP interrupt is enabled).

17.4.4 Effect of a Reset

A reset disables the MSSP module and terminates the current transfer.

Table 17-3: Registers Associated with I^2C Operation

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other resets
INTCON	GIE	PEIE	T0IE	INTE	RBIE(2)	T0IF	INTF	RBIF(2)	0000 0000	0000 0000
PIR	SSPIF, BCLIF (1)								0, 0	0, 0
PIE	SSPIE, BCLIF (1)								0, 0	0, 0
SSPADD	Synchronous Serial Port (I^2C mode) Address Register (slave mode)/Baud Rate Generator (master mode)								0000 0000	0000 0000
SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	0000 0000
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.
Note 1: The position of these bits is device dependent.
2: These bits may also be named GPIE and GPIF.

17.4.5 Master Mode

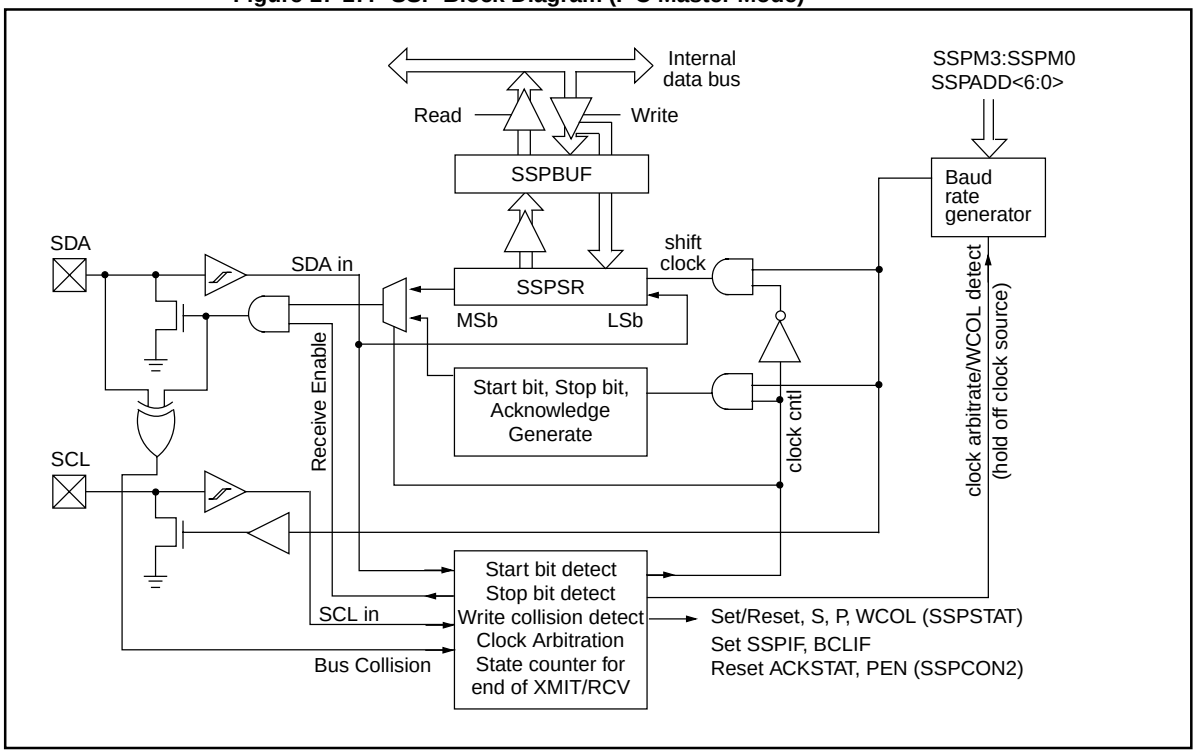
Master mode of operation is supported by interrupt generation on the detection of the START and STOP conditions. The STOP (P) and START (S) bits are cleared from a reset or when the SSP module is disabled. Control of the I^2C bus may be taken when the P bit is set, or the bus is idle with both the S and P bits clear.

In master mode the SCL and SDA lines are manipulated by the SSP 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 17-17: SSP Block Diagram (I^2C Master Mode)



17.4.6 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 SSP module is disabled. Control of the I^2C bus may be taken when the P bit (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

17.4.7 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.

1. Assert a start condition on SDA and SCL.
2. Assert a Repeated Start condition on SDA and SCL.
3. Write to the SSPBUF register initiating transmission of data/address.
4. Generate a stop Condition on SDA and SCL.
5. Configure the I^2C port to receive data.
6. Generate an acknowledge condition at the end of a received byte of data.

Note:	The SSP 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 imitate 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.

17.4.7.1 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 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.

A typical transmit sequence would go as follows:

a) The user generates a Start Condition by setting the START enable bit, SEN (SSPCON2<0>).
b) SSPIF is set. The SSP module will wait the required start time before any other operation takes place.
c) The user loads the SSPBUF with the address to transmit.
d) Address is shifted out the SDA pin until all 8 bits are transmitted.
e) The SSP Module shifts in the ACK bit from the slave device, and writes its value into the SSPCON2 register (SSPCON2<6>).
f) The SSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
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 SSP Module shifts in the ACK bit from the slave device, and writes its value into the SSPCON2 register (SSPCON2<6>).
j) The SSP 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 (SSPCON2<2>).
l) Interrupt is generated once the stop condition is complete.

17.4.8 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 17-18). When the BRG is loaded with this value, the BRG counts down to 0 and stops until another reload has taken place. 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 17-19).

Figure 17-18: Baud Rate Generator Block Diagram

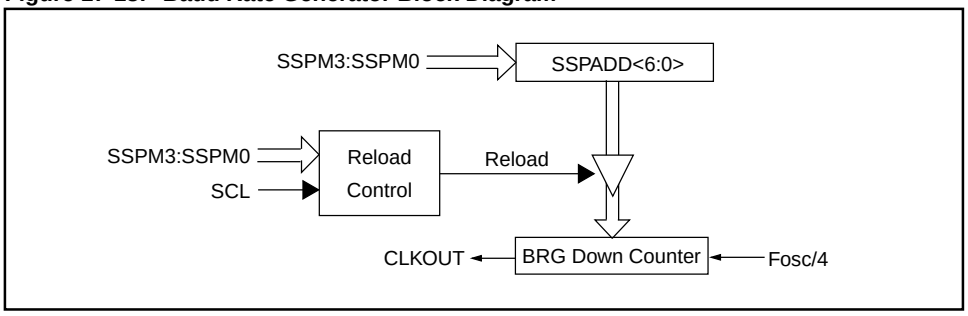
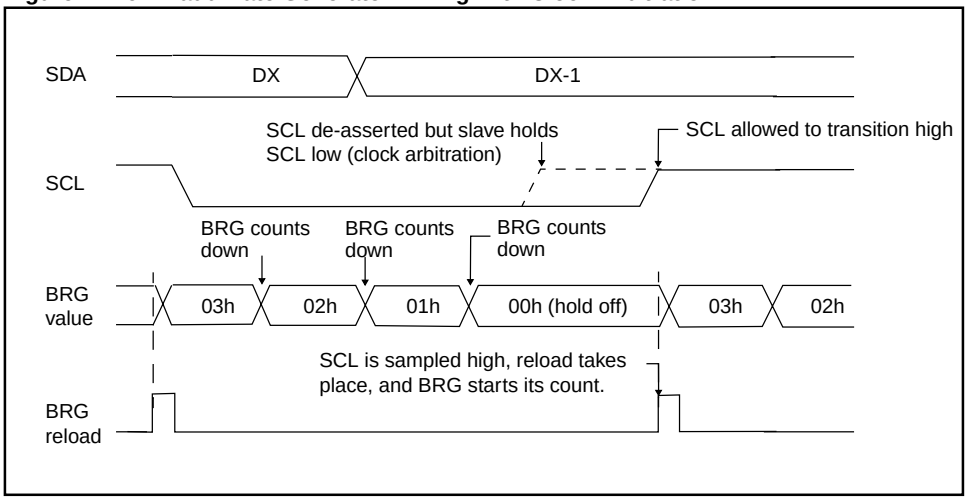


Figure 17-19: Baud Rate Generator Timing With Clock Arbitration



17.4.9 I^2C 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 re-loaded 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 (T_{BRG}), 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 (T_{BRG}) 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 I^2C module is reset into its IDLE state.

17.4.9.1 WCOL Status Flag

If the user writes the SSPBUF when an 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 17-20: First Start Bit Timing

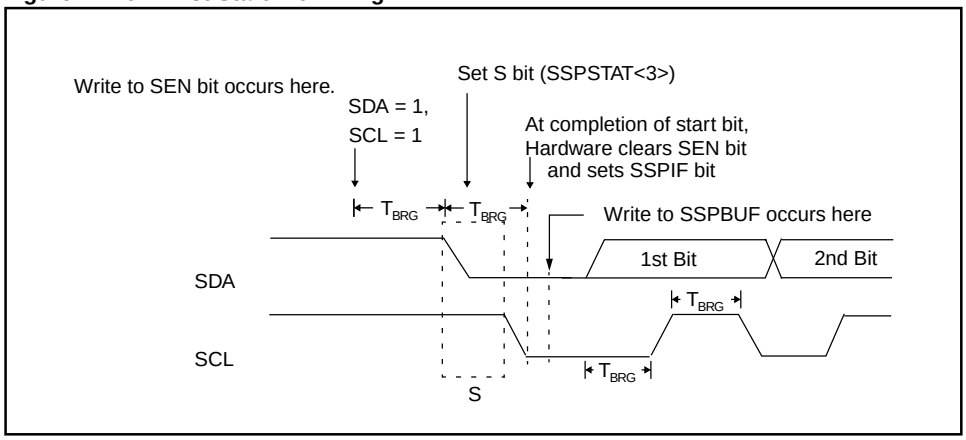
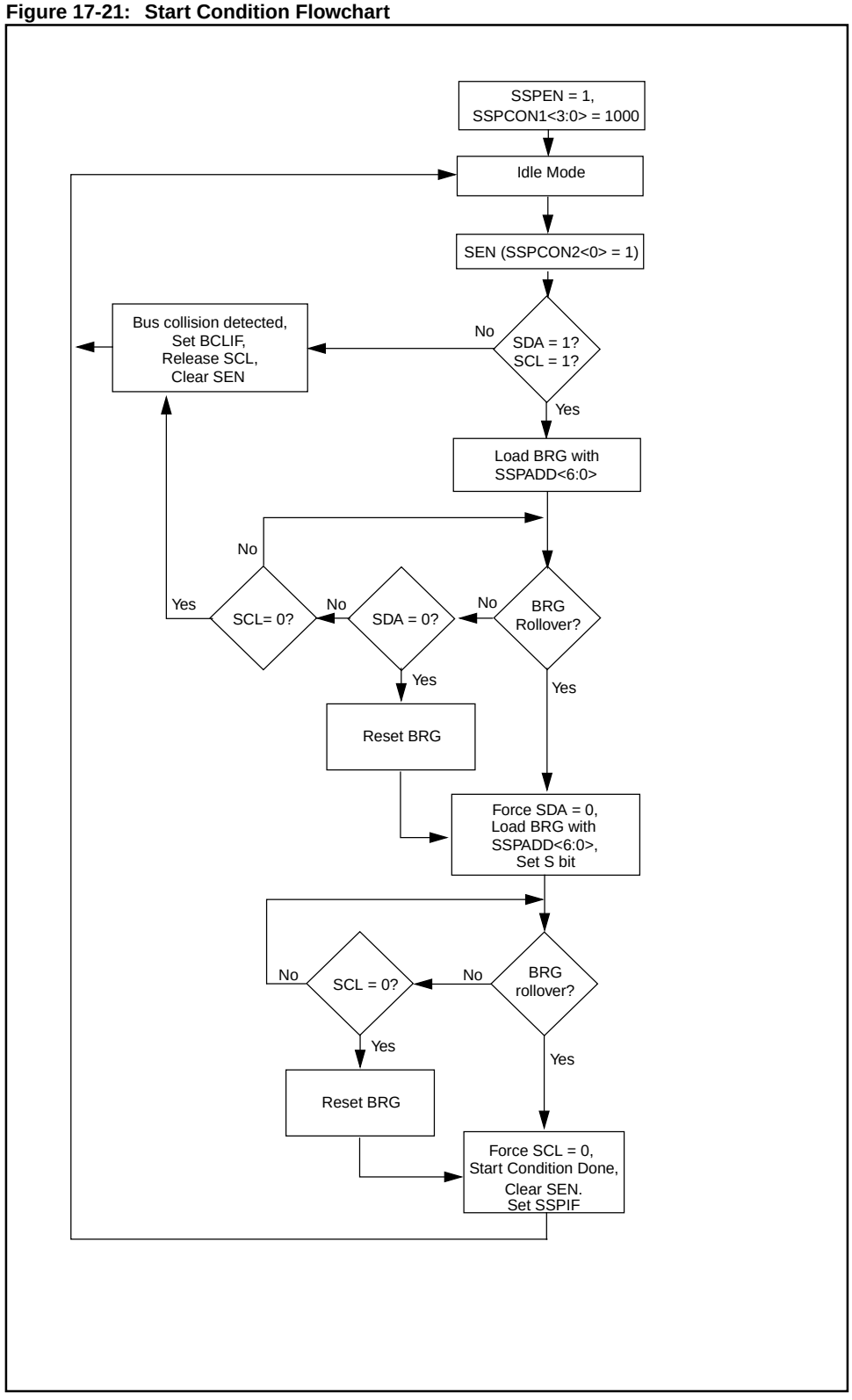


Figure 17-21: Start Condition Flowchart



17.4.10 I^2C Master Mode Repeated Start Condition Timing

A Repeated Start condition occurs when the RSEN bit (SSPCON2<1>) is programmed high and the I^2C logic 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<5:0>, and begins counting. The SDA pin is released (brought high) for one baud rate generator count (T_{BRG}). 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 re-loaded with the contents of SSPADD<6:0> and begins counting. SDA and SCL must be sampled high for one T_{BRG}. This action is then followed by assertion of the SDA pin (SDA = 0) for one T_{BRG} while SCL is high. Following this, the RSEN bit (SSPCON2<1>) 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 RSEN is programmed while any other event is in progress, it will not take effect.

Note 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 indicates 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).

17.4.10.1 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 17-22: Repeat Start Condition Waveform

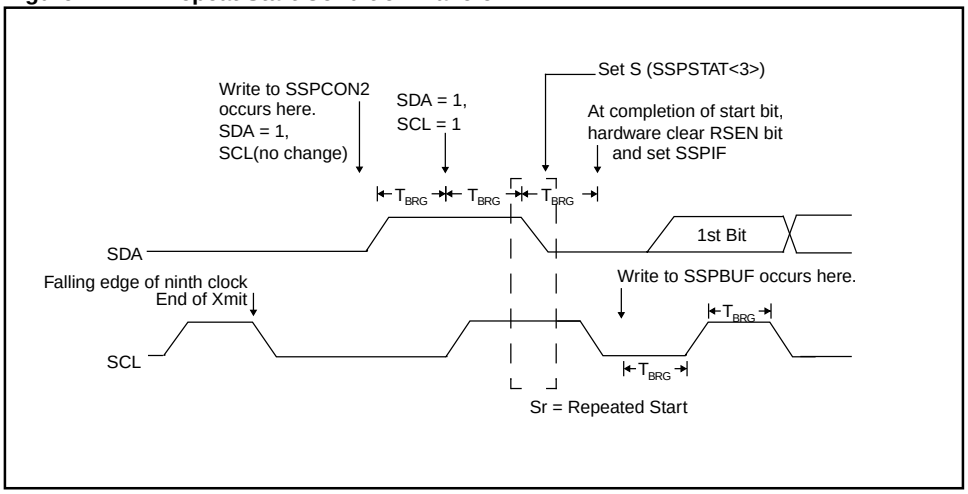


Figure 17-23: Repeated Start Condition Flowchart (part 1 of 2)

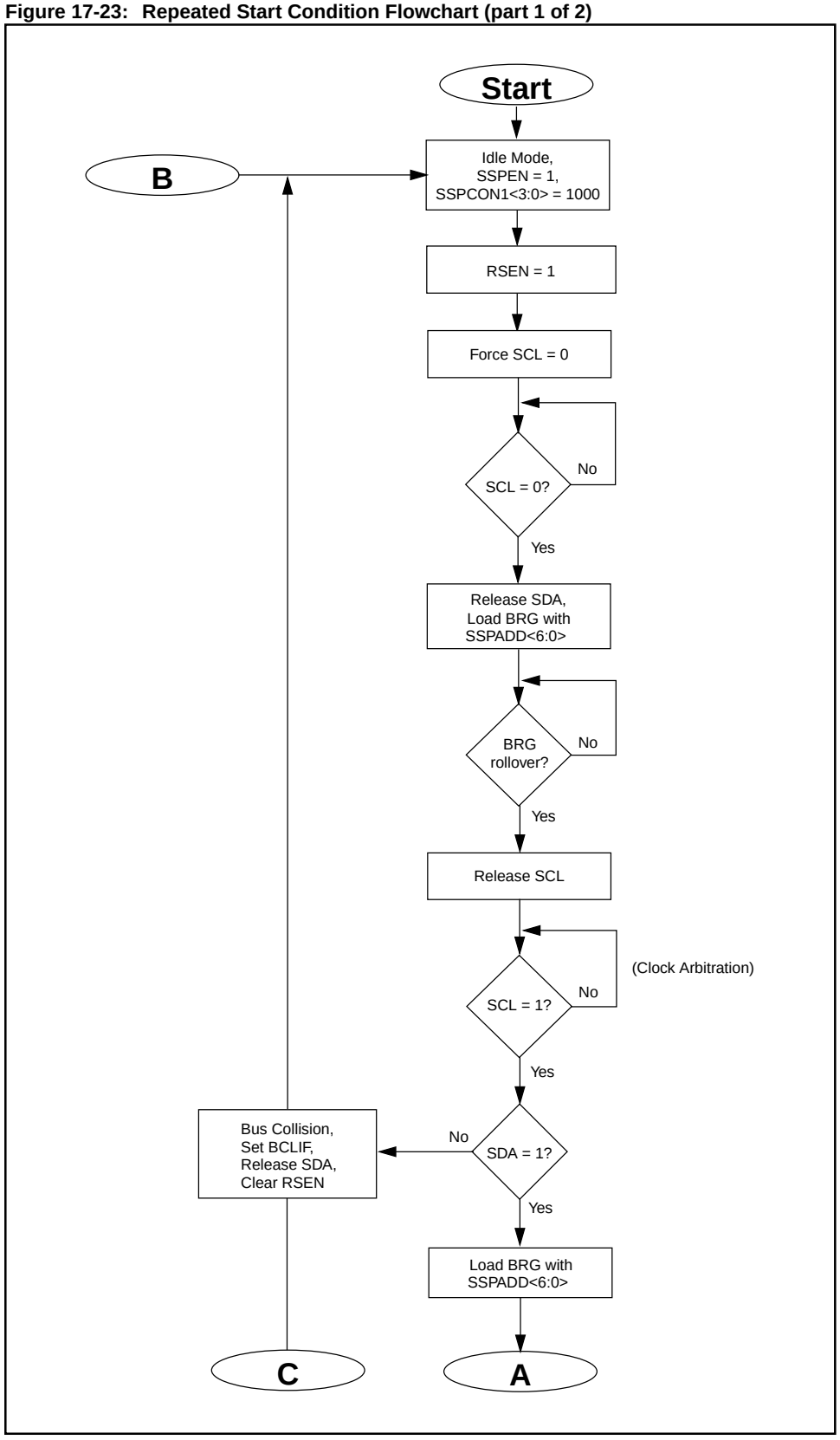
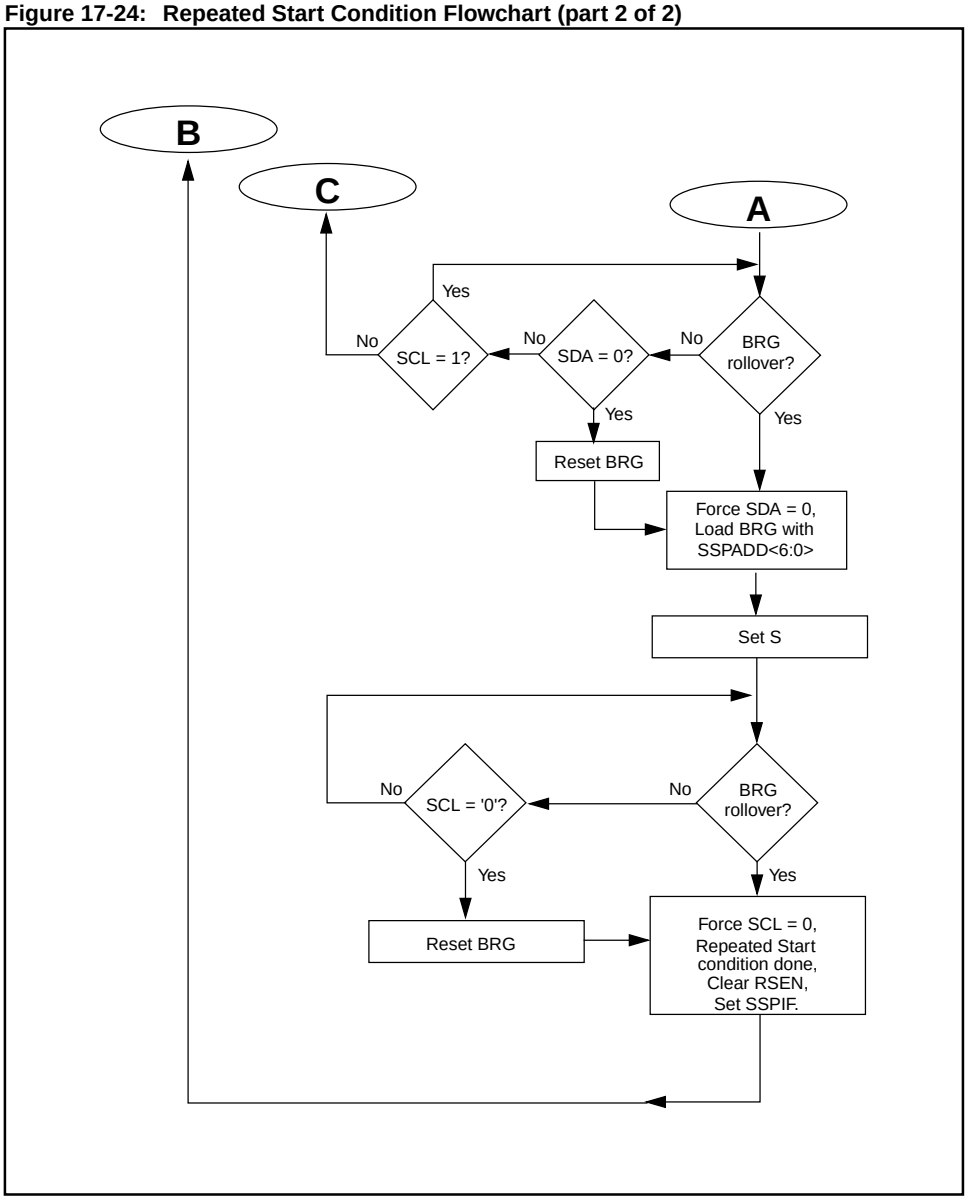


Figure 17-24: Repeated Start Condition Flowchart (part 2 of 2)



17.4.11 I^2C Master Mode Transmission

Transmission of a data byte, a 7-bit address, or the 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 bit, 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 specification parameters 106). SCL is held low for one baud rate generator roll over count (T_{BRG}). Data should be valid before SCL is released high (see Data setup time specification parameters 107). When the SCL pin is released high, it is held that way for T_{BRG}, 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 written into the ACKDT bit on the falling edge of the ninth clock. If the master receives an acknowledge, the acknowledge status bit, ACKSTAT, is cleared. If not, the bit is set. After the ninth clock the SSPIF bit 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 17-26).

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.

17.4.11.1 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.

17.4.11.2 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.

17.4.11.3 ACKSTAT Status Flag

In transmit mode, the ACKSTAT 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 17-25: Master Transmit Flowchart

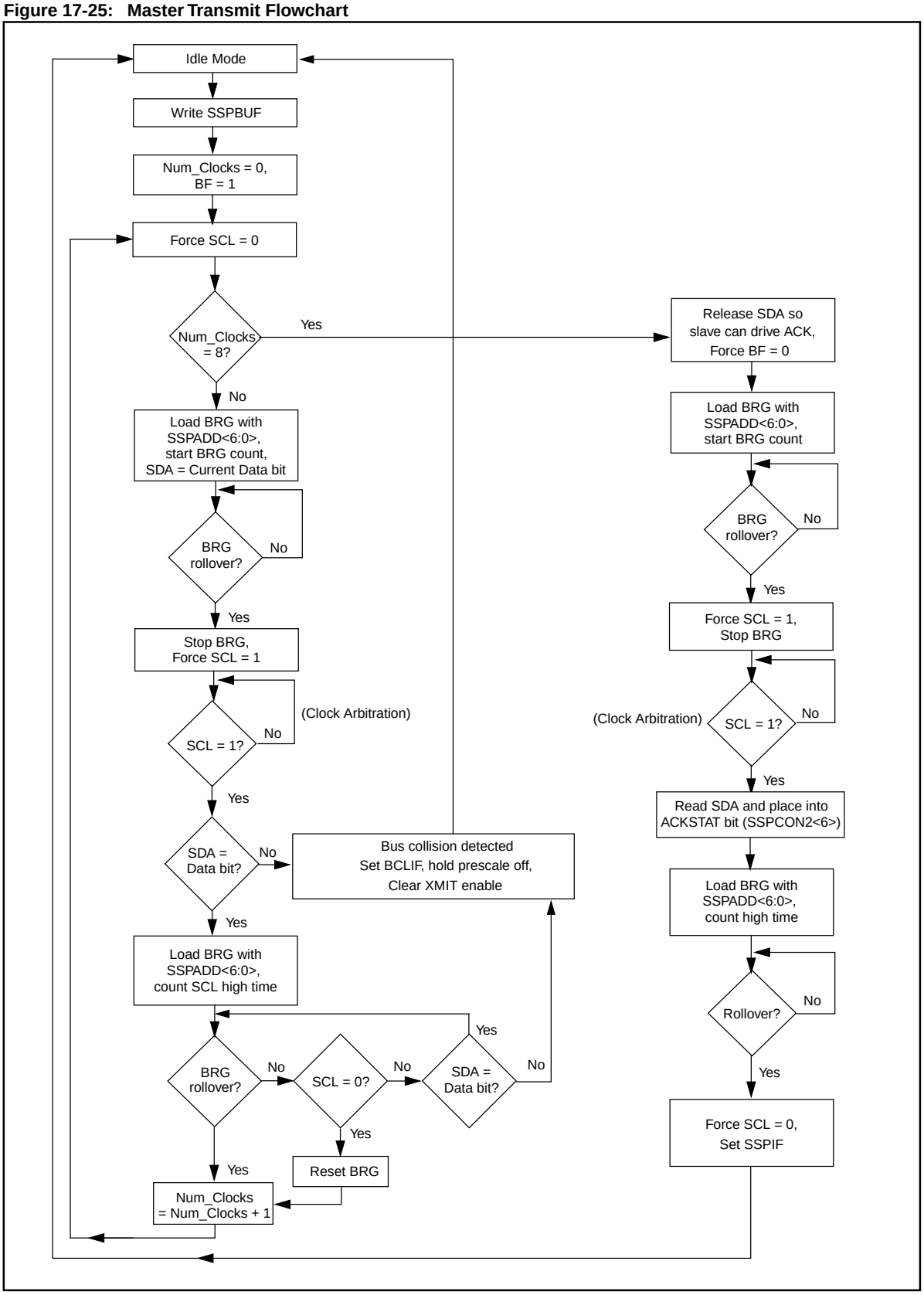
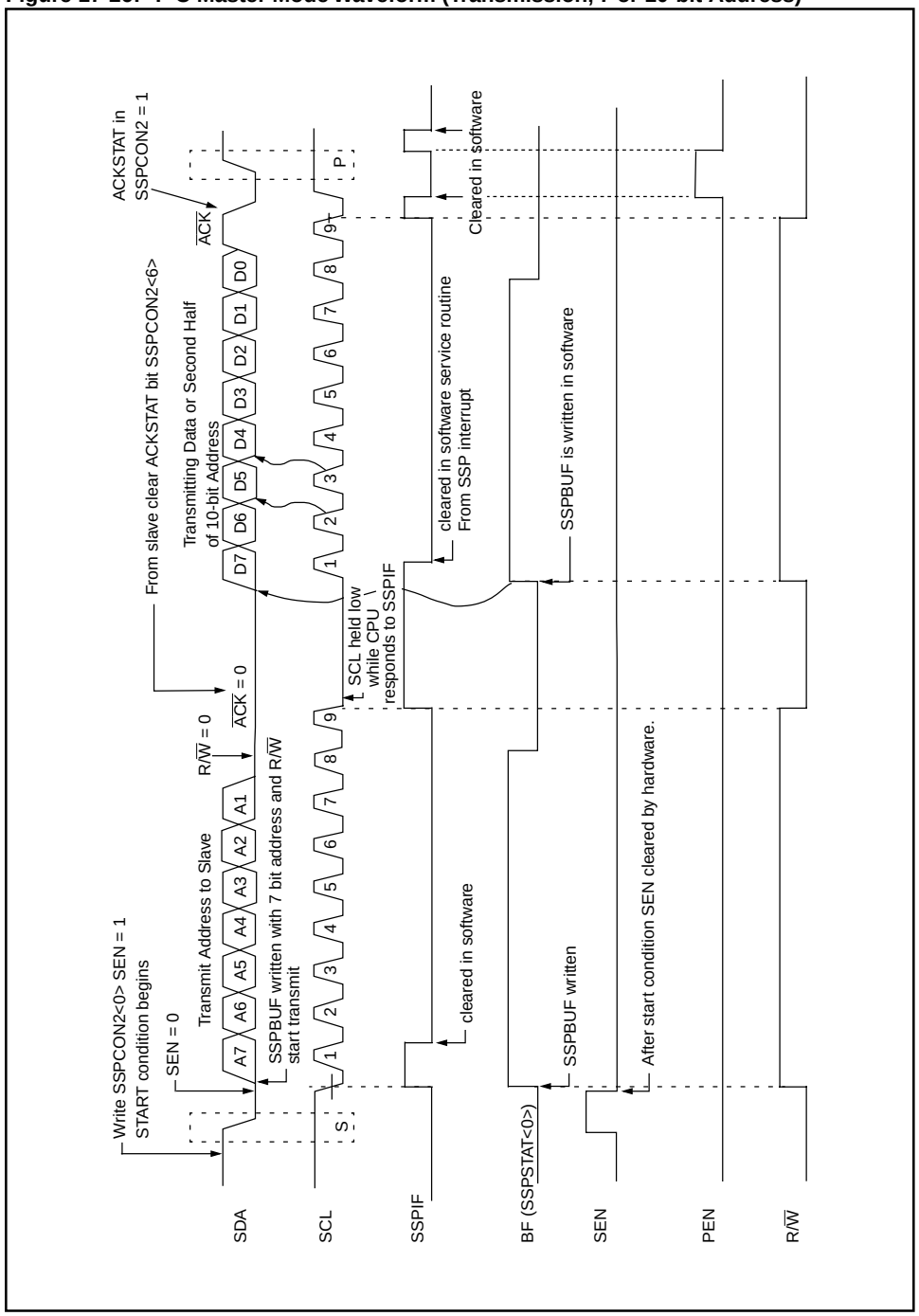


Figure 17-26: I^2C Master Mode Waveform (Transmission, 7 or 10-bit Address)



17.4.12 I²C 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 bit is set, the SSPIF flag bit 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 bit 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>).

17.4.12.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPBUF from SSPSR. It is cleared when the SSPBUF register is read.

17.4.12.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when 8 bits are received into the SSPSR, and the BF flag bit is already set from a previous reception.

17.4.12.3 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 the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

Figure 17-27: Master Receiver Flowchart

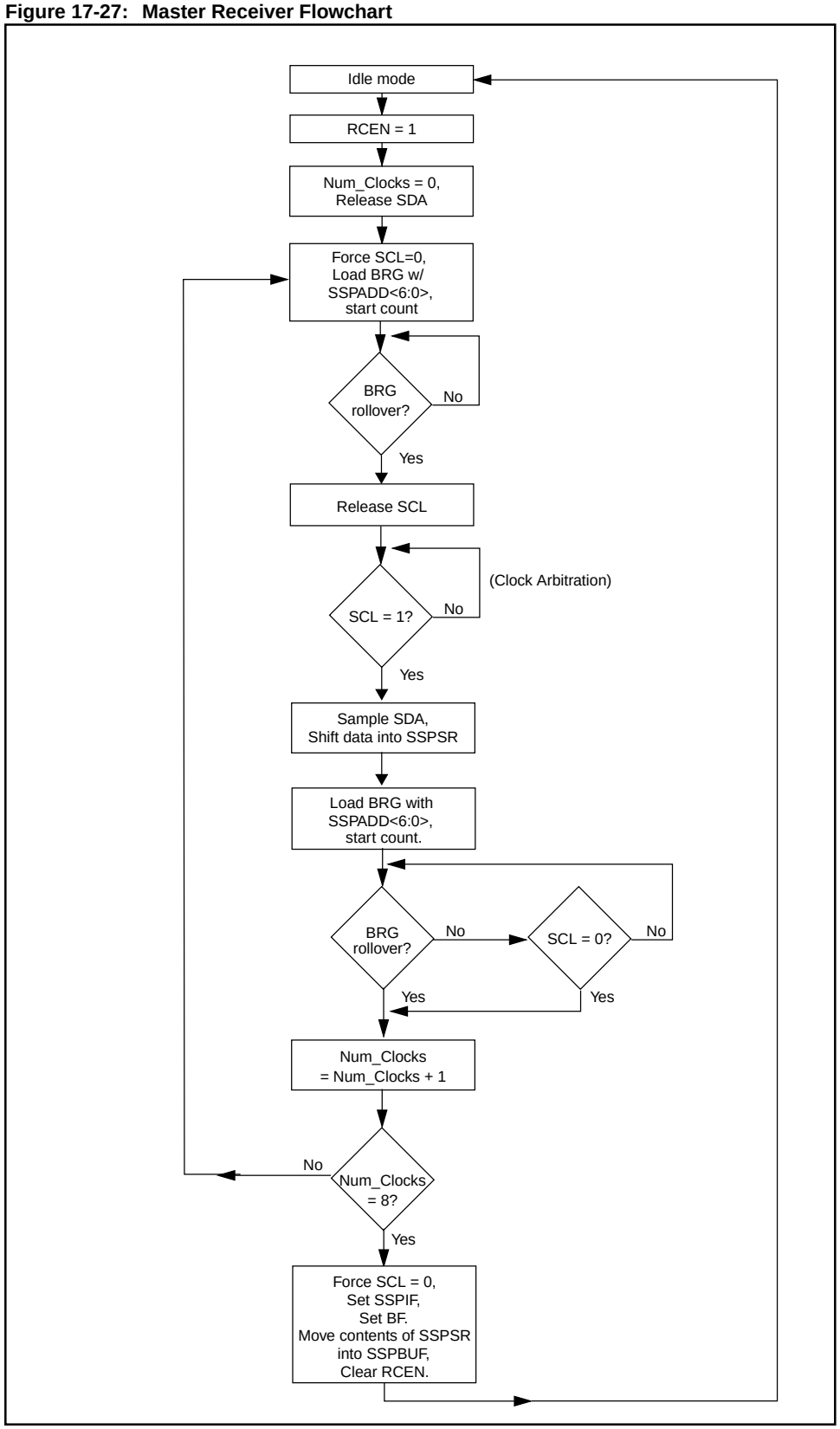
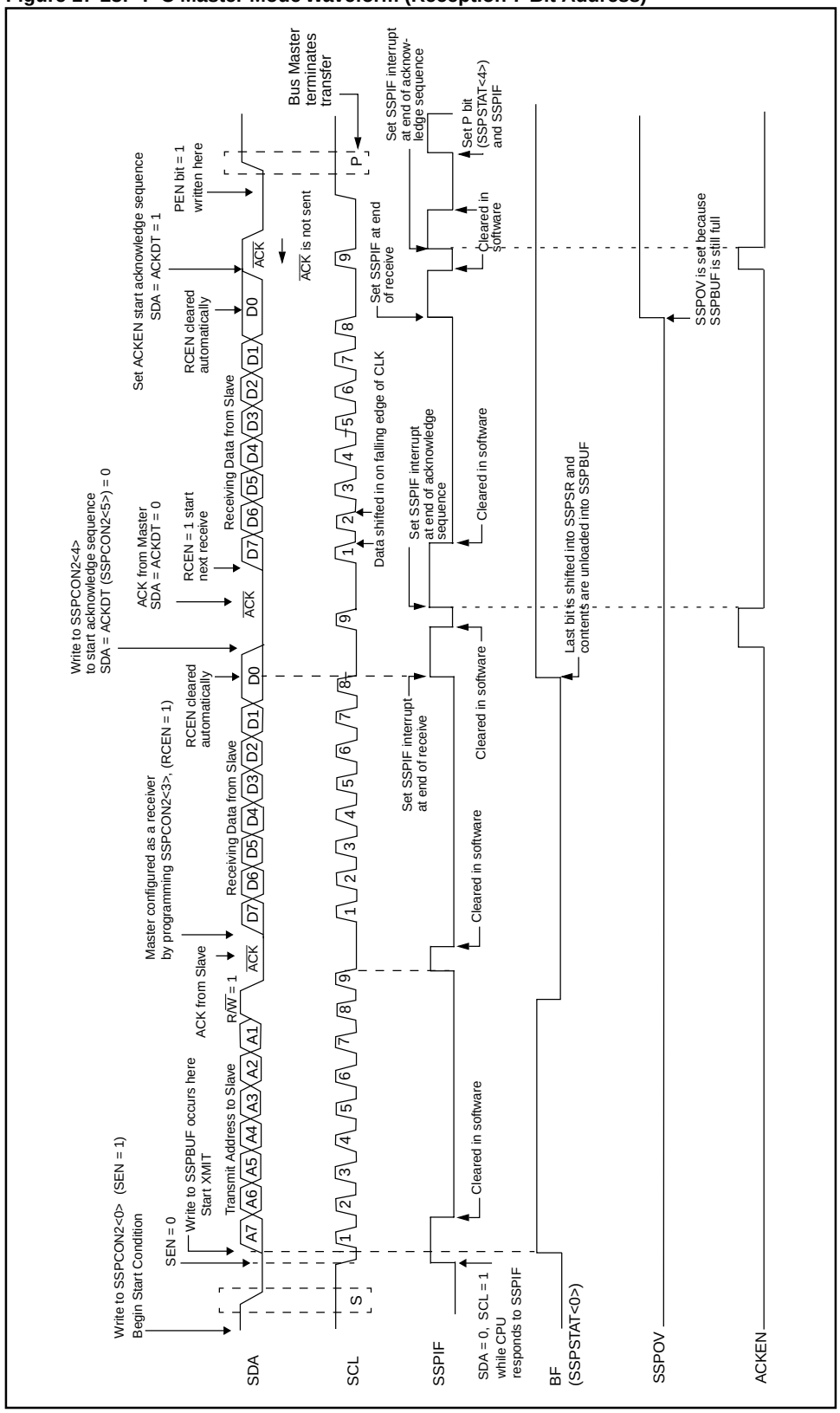


Figure 17-28: I²C Master Mode Waveform (Reception 7-Bit Address)



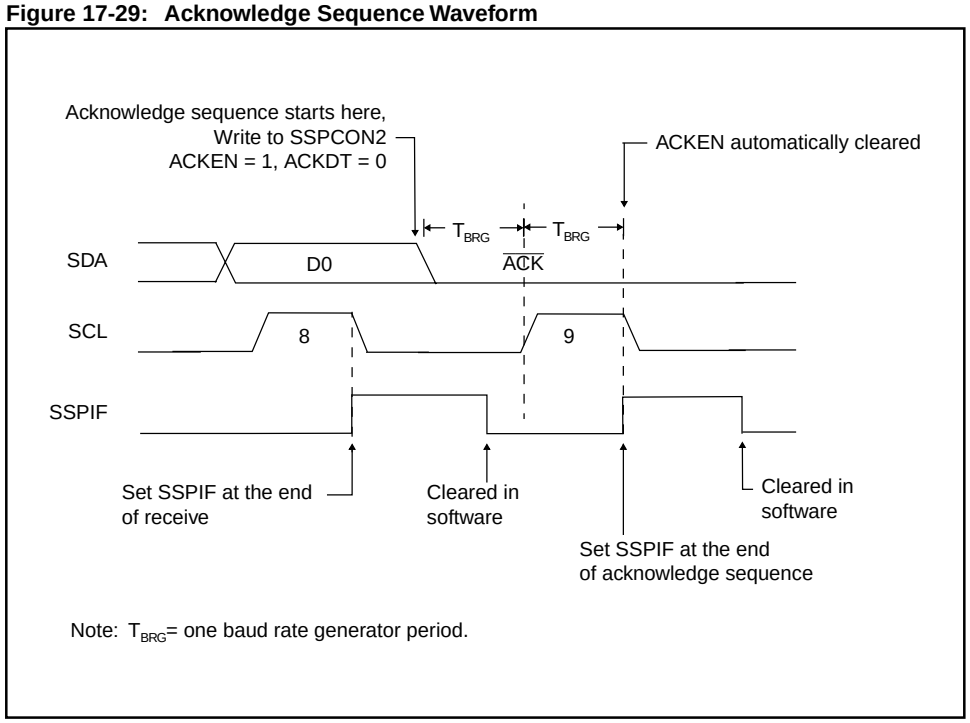
17.4.13 Acknowledge Sequence Timing

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 (T_{BRG}), and the SCL pin is de-asserted (pulled high). When the SCL pin is sampled high (clock arbitration), the baud rate generator counts for T_{BRG}. 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 17-29).

17.4.13.1 WCOL Status Flag

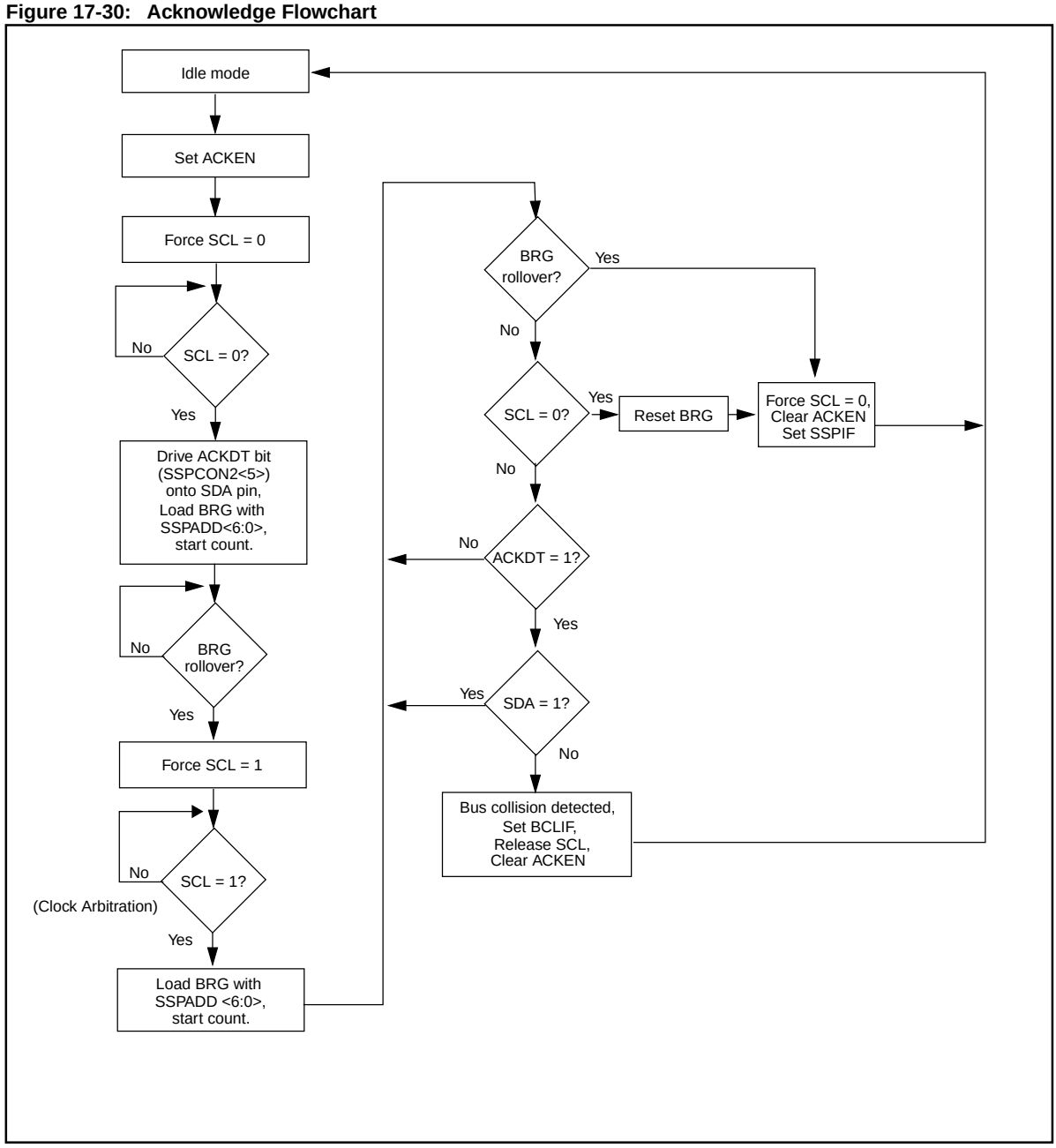
If the user writes the SSPBUF when an acknowledge sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

Figure 17-29: Acknowledge Sequence Waveform



Note: T_{BRG}= one baud rate generator period.

Figure 17-30: Acknowledge Flowchart



17.4.14 Stop Condition Timing

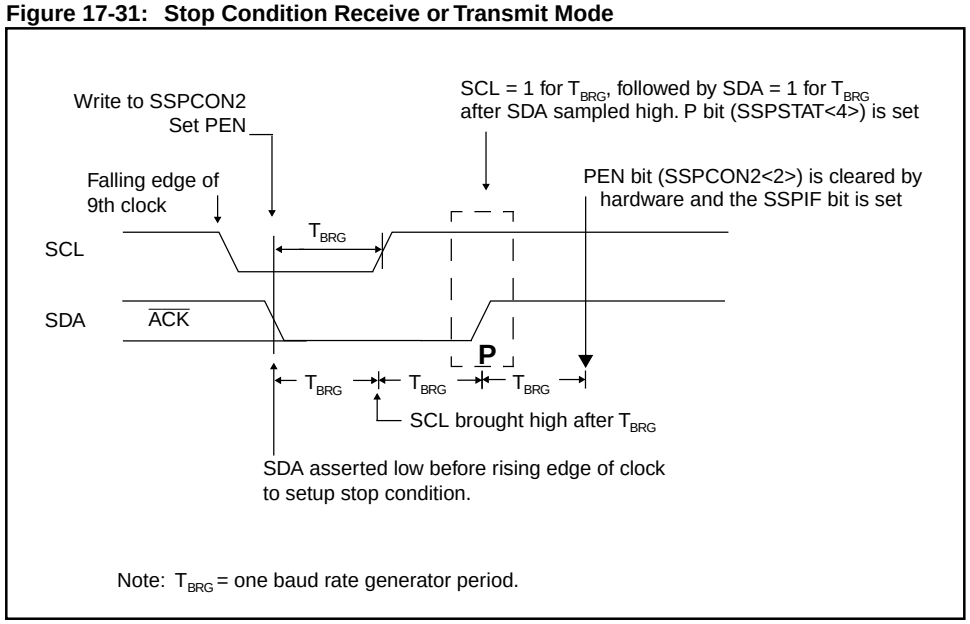
A stop bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop sequence enable bit, PEN (SSPCON2<2>). At the end of a receive/transmit the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When 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 T_{BRG} (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 17-31).

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).

17.4.14.1 WCOL Status Flag

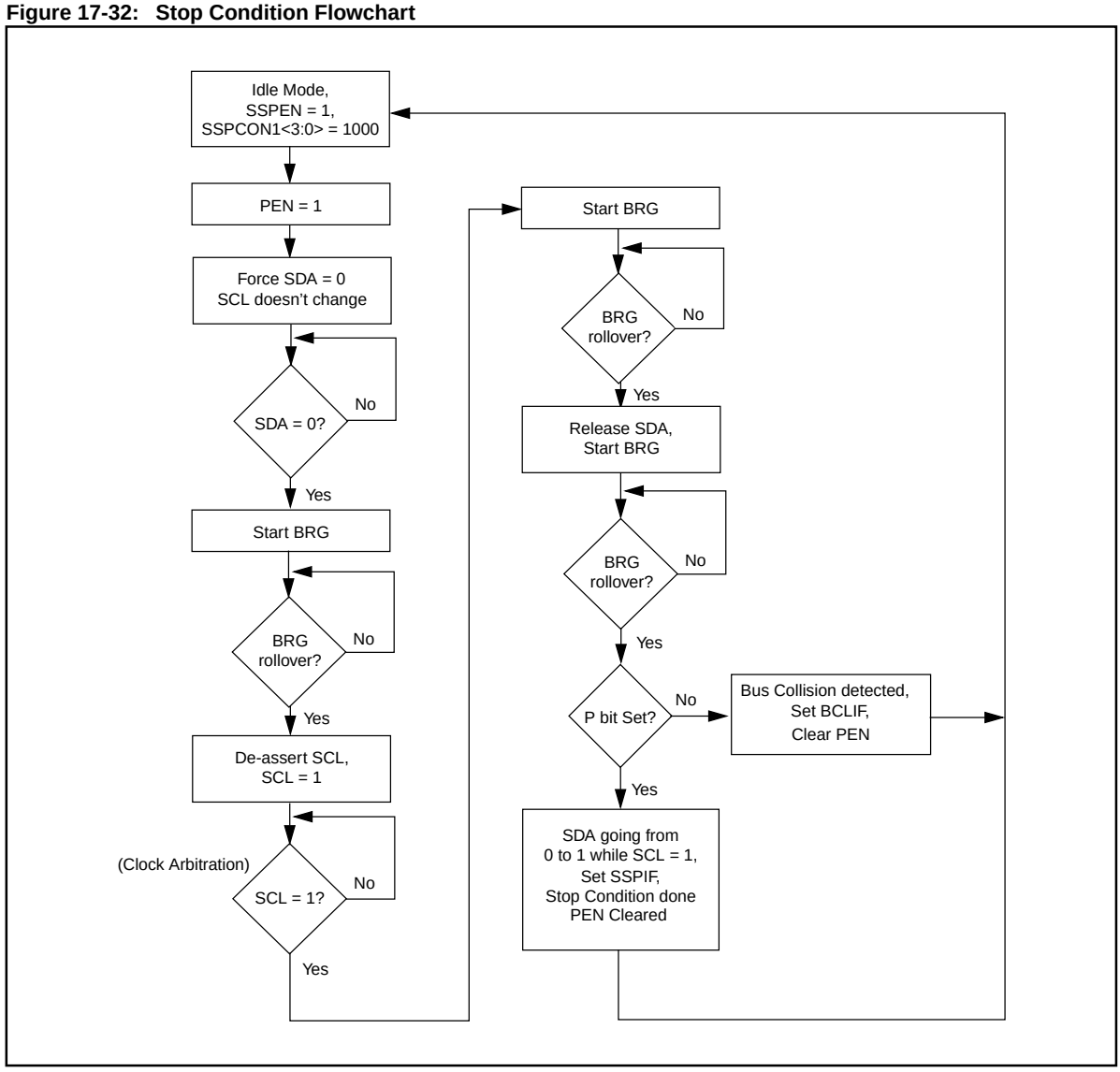
If the user writes the SSPBUF when a STOP sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

Figure 17-31: Stop Condition Receive or Transmit Mode



Note: T_{BRG} = one baud rate generator period.

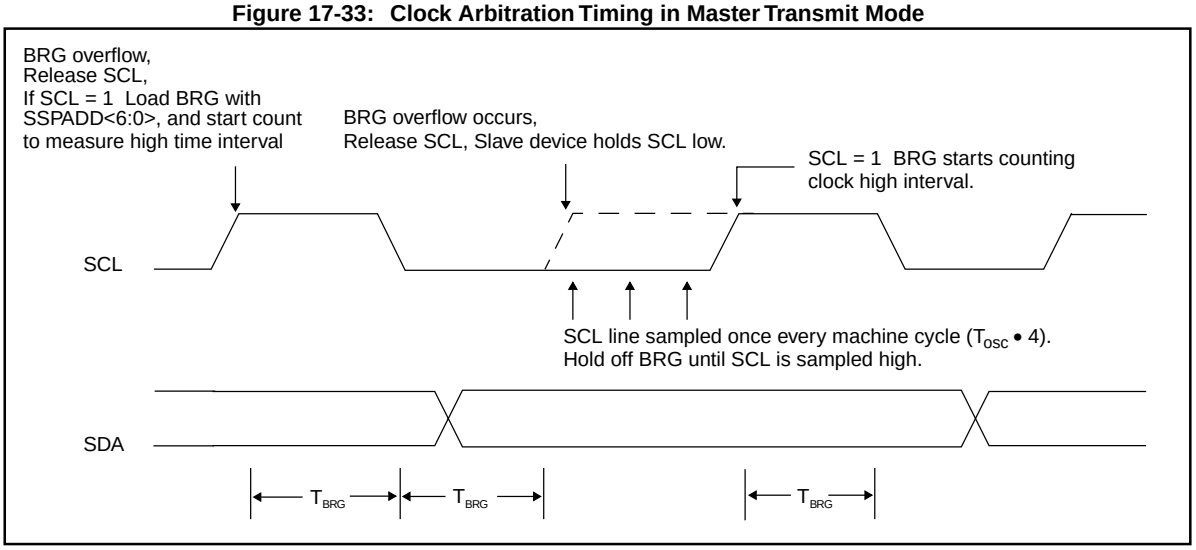
Figure 17-32: Stop Condition Flowchart



17.4.15 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit, or Repeated Start/stop condition de-asserts 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 17-33).

Figure 17-33: Clock Arbitration Timing in Master Transmit Mode



17.4.16 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 MSSP interrupt is enabled).

17.4.17 Effect of a Reset

A reset disables the MSSP module and terminates the current transfer.

17.4.18 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 17-34).

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.

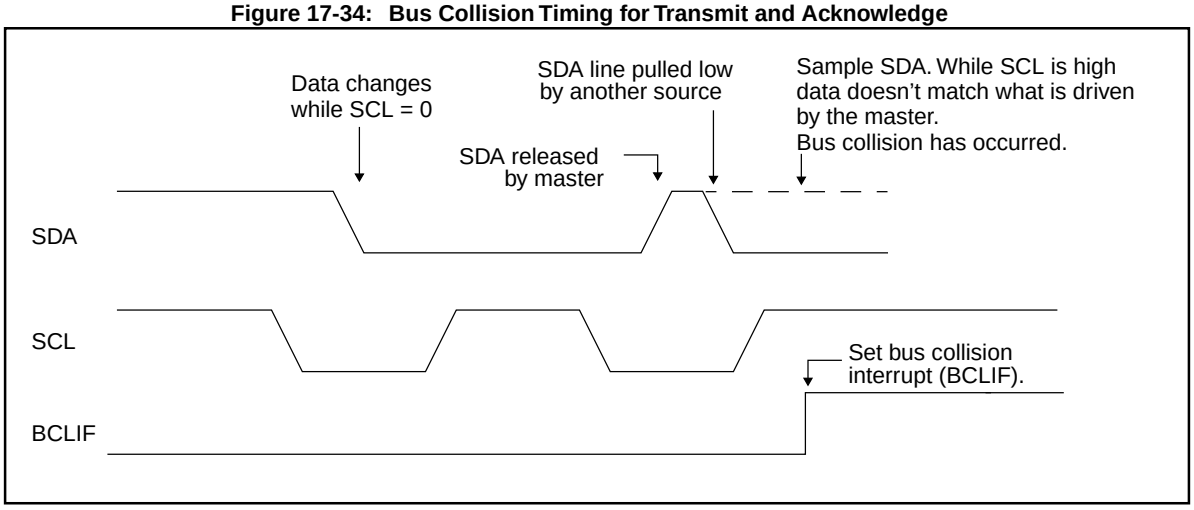
If a START, Repeated Start, STOP, or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are de-asserted, and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision interrupt service 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 17-34: Bus Collision Timing for Transmit and Acknowledge



17.4.18.1 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 17-35).

b) SCL is sampled low before SDA is asserted low (Figure 17-36).

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 17-35).

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 17-37). 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 pins 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 17-35: Bus Collision During Start Condition (SDA only)

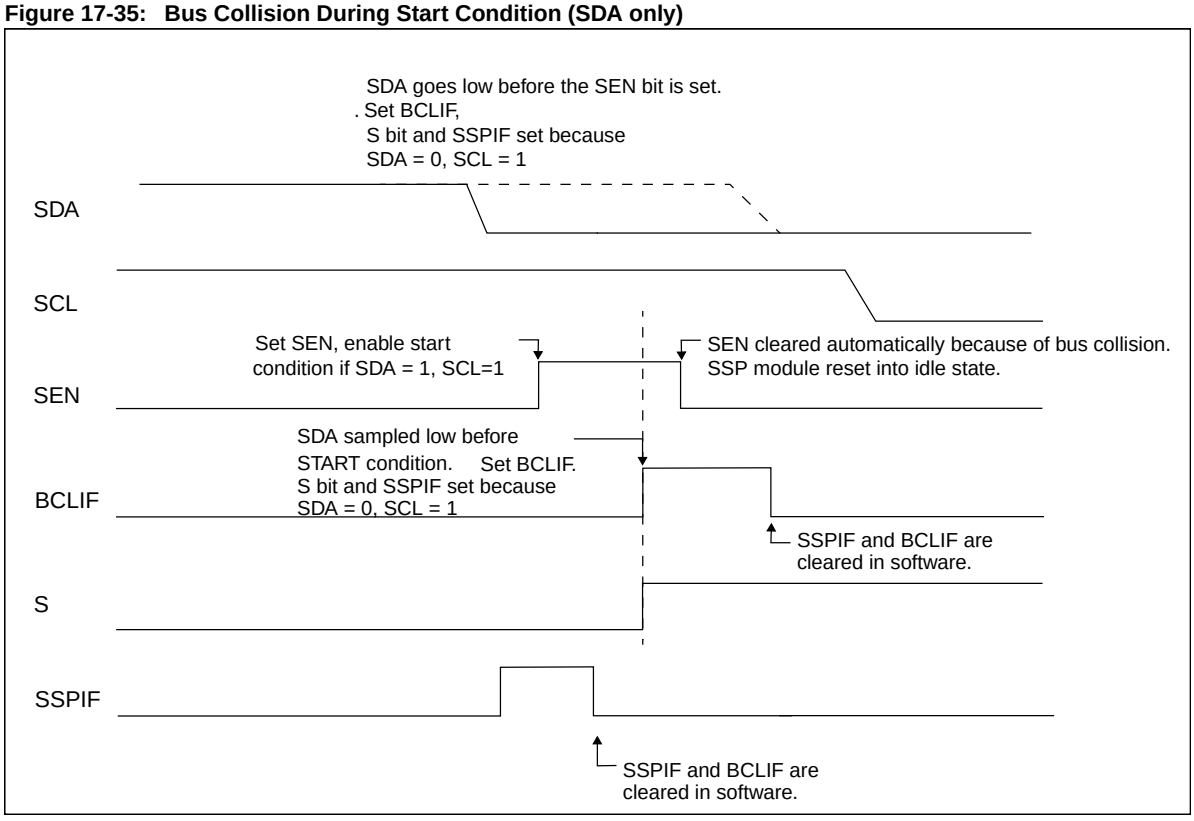


Figure 17-36: Bus Collision During Start Condition (SCL = 0)

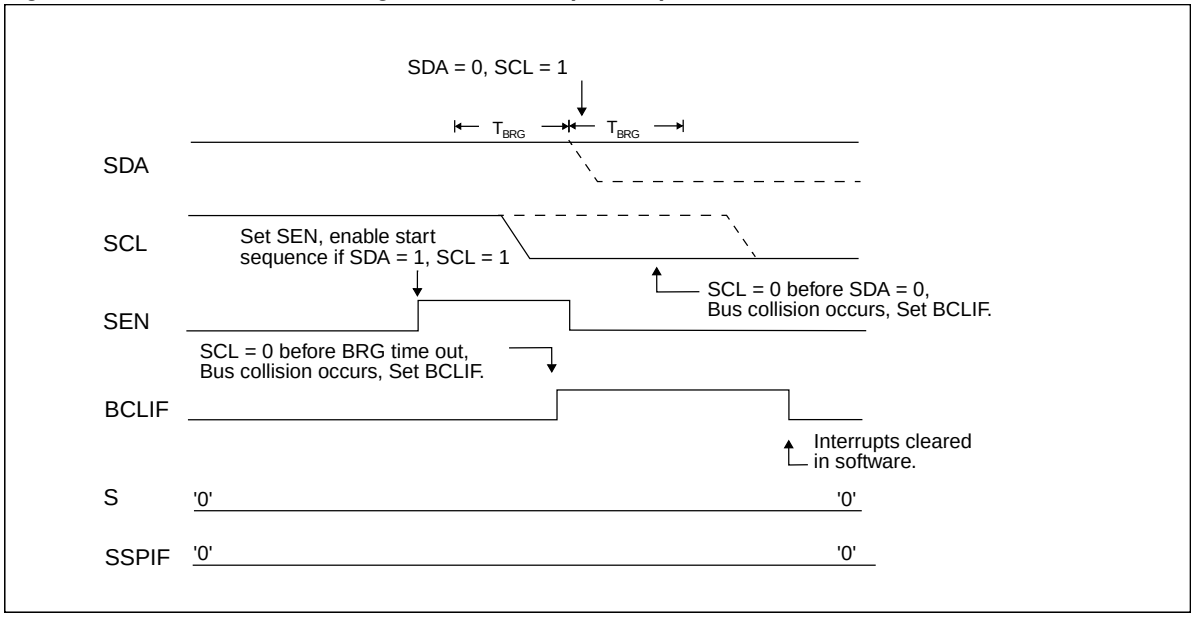
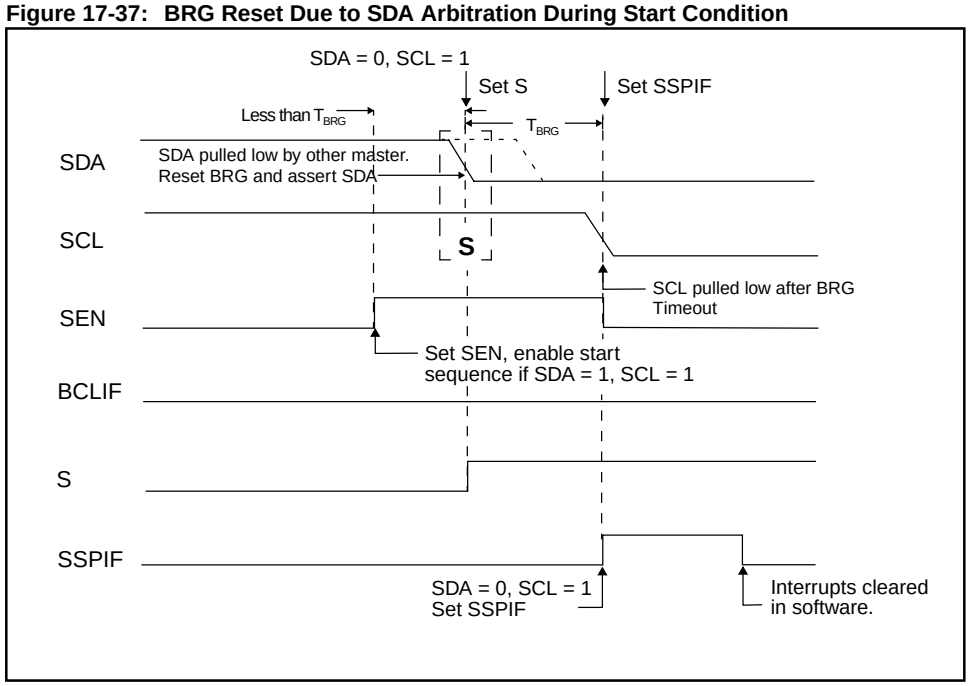


Figure 17-37: BRG Reset Due to SDA Arbitration During Start Condition



17.4.18.2 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 zero. 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, Figure 17-38, 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, Figure 17-39.

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 17-38: Bus Collision During a Repeated Start Condition (Case 1)

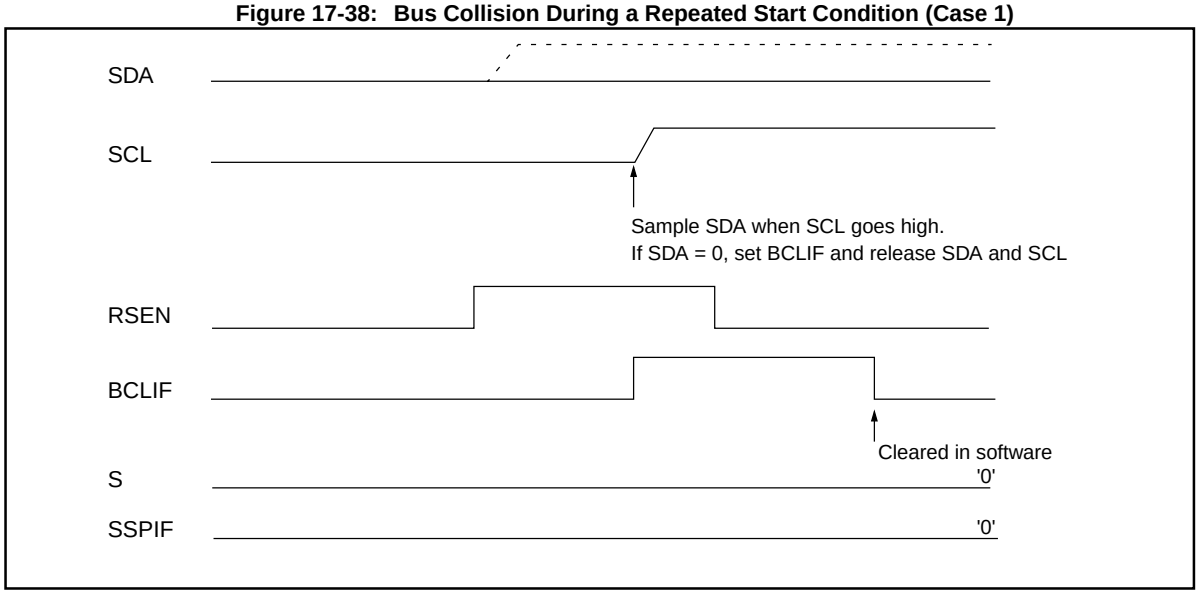
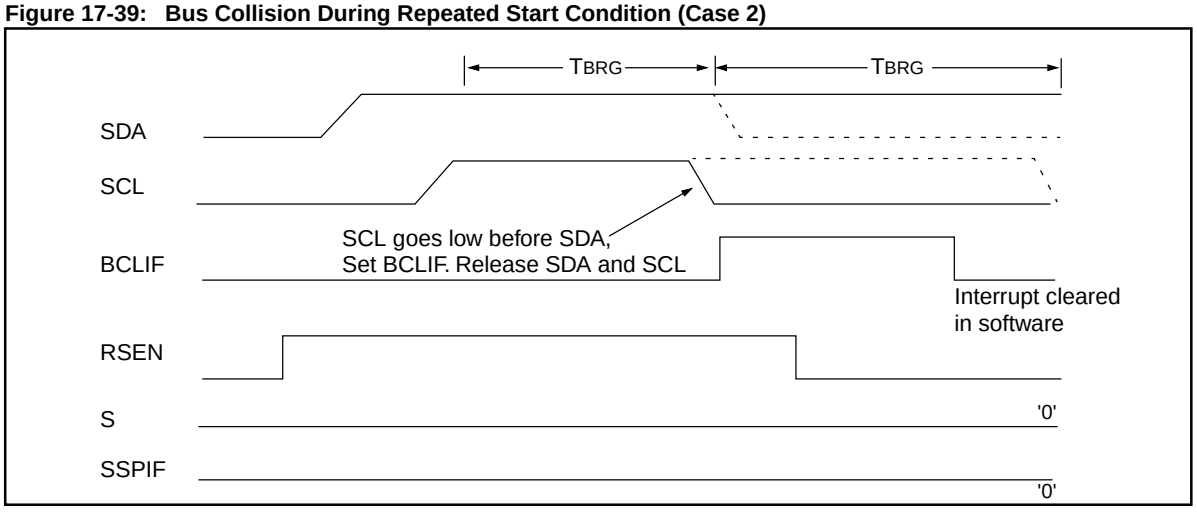


Figure 17-39: Bus Collision During Repeated Start Condition (Case 2)



17.4.18.3 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 allow 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' (Figure 17-40). 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 17-41).

Figure 17-40: Bus Collision During a STOP Condition (Case 1)

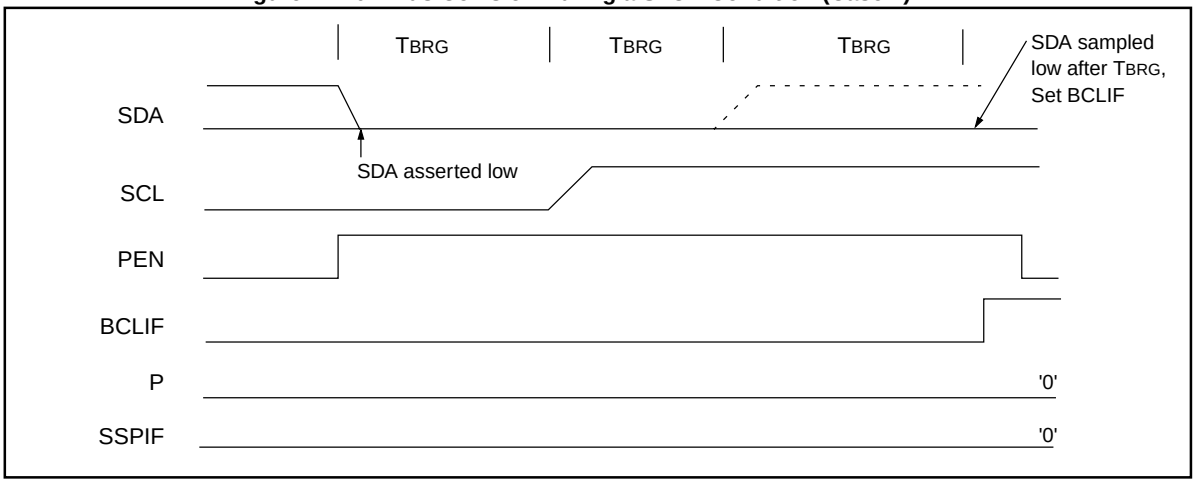
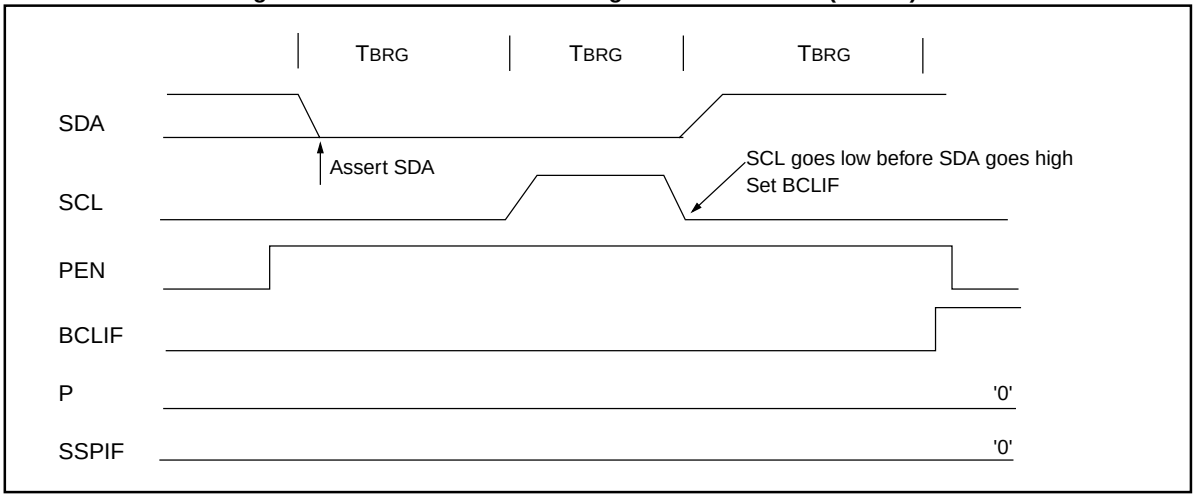


Figure 17-41: Bus Collision During a STOP Condition (Case 2)



17.5 Connection Considerations for I²C Bus

For standard-mode I²C bus devices, the values of resistors ***RP*** and ***RS*** in Figure 17-42 depends on the following parameters:

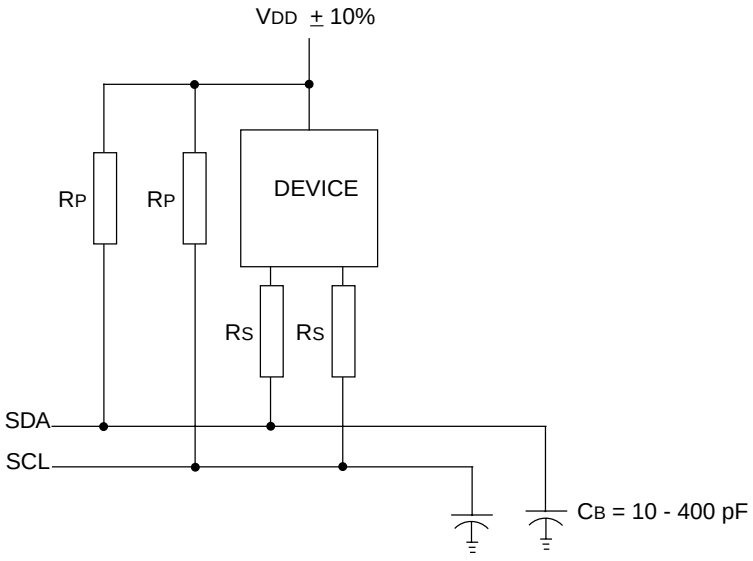
- Supply voltage
- Bus capacitance
- Number of connected devices (input current + leakage current)

The supply voltage limits the minimum value of resistor ***RP*** due to the specified minimum sink current of 3 mA at VOLMAX = 0.4V for the specified output stages. For example, with a supply voltage of VDD = 5V±10% and VOLMAX = 0.4V at 3 mA, RPMIN = (5.5-0.4)/0.003 = 1.7 kΩ. VDD as a function of ***RP*** is shown in Figure 17-42. The desired noise margin of 0.1VDD for the low level, limits the maximum value of ***RS***. 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 ***RP*** due to the specified rise time (Figure 17-42).

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 17-42: 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.

17.6 Initialization

Example 17-2: SPI Master Mode Initialization

```
    CLRF   STATUS       ; Bank 0
    CLRF   SSPSTAT      ; SMP = 0, CKE = 0, and clear status bits
    BSF    SSPSTAT, CKE ; CKE = 1
    MOVLW  0x31         ; Set up SPI port, Master mode, CLK/16,
    MOVWF  SSPCON       ;   Data xmit on falling edge (CKE=1 & CKP=1)
                        ;   Data sampled in middle (SMP=0 & Master mode)
    BSF    STATUS, RP0  ; Bank 1
    BSF    PIE, SSPIE   ; Enable SSP interrupt
    BCF    STATUS, RP0  ; Bank 0
    BSF    INTCON, GIE  ; Enable, enabled interrupts
    MOVLW  DataByte     ; Data to be Transmitted
                        ;   Could move data from RAM location
    MOVWF  SSPBUF       ; Start Transmission
```

17.6.1 Master SSP Module / Basic SSP Module Compatibility

When changing from the SPI in the Basic SSP module, the SSPSTAT register contains two additional control bits. These bits are:

- SMP, SPI data input sample phase
- CKE, SPI Clock Edge Select

To be compatible with the SPI of the Master SSP module, these bits must be appropriately configured. If these bits are not at the states shown in Table 17-4, improper SPI communication may occur.

Table 17-4: New bit States for Compatibility

Basic SSP Module	Master SSP Module		
CKP	CKP	CKE	SMP
1	1	0	0
0	0	0	0

17.7 Design Tips

Question 1: ***Using SPI mode, I do not seem able to talk to an SPI device.***

Answer 1:

Ensure that you are using the correct SPI mode for that device. This SPI supports all 4 SPI modes so you should be able to get it to function. Check the clock polarity and the clock phase.

Question 2: ***Using I^2C mode, I write data to the SSPBUF register, but the data did not transmit.***

Answer 2:

Ensure that you set the CKP bit to release the I^2C clock.

17.8 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the Master SSP module are:

Title	Application Note #
Use of the SSP Module in the I ^{2}C Multi-Master Environment.	AN578
Using Microchip 93 Series Serial EEPROMs with Microcontroller SPI Ports	AN613
Interfacing PIC16C64/74 to Microchip SPI Serial EEPROM	AN647
Interfacing a Microchip PIC16C92x to Microchip SPI Serial EEPROM	AN668

17.9 Revision History

Revision A

This is the initial released revision of the Master SSP module description.

Section 18. USART

HIGHLIGHTS

This section of the manual contains the following major topics:

18.1 Introduction

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules (other is the SSP module). The USART is also known as a Serial Communications Interface or SCI. 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 bit (RCSTA<7>), and the TRIS bits, have to be set in order to configure the TX/CK and RX/DT pins for the USART.

18.2 Control Registers

Register 18-1: TXSTA: Transmit Status and Control Register

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode
Don't care

Synchronous mode
1 = Master mode (Clock generated internally from BRG)
0 = Slave mode (Clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled

Note: SREN/CREN overrides TXEN in SYNC mode.

bit 4 **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

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

bit 2 **BRGH**: High Baud Rate Select bit
Asynchronous mode
1 = High speed
0 = Low speed

Synchronous mode
Unused in this mode

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of transmit data. Can be parity bit.

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

Register 18-2: RCSTA: Receive Status and Control Register

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R-0	R-0	R-0
SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D
bit 7							bit 0

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

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

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode
Don't care

Synchronous mode - master
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.

Synchronous mode - slave
Unused in this mode

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode
1 = Enables continuous receive
0 = Disables continuous receive

Synchronous mode
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

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

bit 2 **FERR**: Framing Error bit
1 = Framing error (Can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

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

bit 0 **RX9D:** 9th bit of received data, can be parity bit.

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

18.3 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. In asynchronous mode bit BRGH (TXSTA<2>) also controls the baud rate. In synchronous mode bit BRGH is ignored. Table 18-1 shows the formula for computation of the baud rate for different USART modes which only apply in master mode (internal clock).

Given the desired baud rate and Fosc, the nearest integer value for the SPBRG register can be calculated using the formula in Table 18-1, where X equals the value in the SPBRG register (0 to 255). From this, the error in baud rate can be determined.

Table 18-1: Baud Rate Formula

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = FOSC/(64(X+1))	Baud Rate= FOSC/(16(X+1))
1	(Synchronous) Baud Rate = FOSC/(4(X+1))	NA

X = value in SPBRG (0 to 255)

Example 18-1 shows the calculation of the baud rate error for the following conditions:

FOSC = 16 MHz
Desired Baud Rate = 9600
BRGH = 0
SYNC = 0

Example 18-1: 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%

It may be advantageous to use the high baud rate (BRGH = 1) even for slower baud clocks. This is because the FOSC / (16(X + 1)) equation can reduce the baud rate error in some cases.

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

Table 18-2: Registers Associated with Baud Rate Generator

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other resets
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented read as '0'. Shaded cells are not used by the BRG.

Table 18-3: Baud Rates for Synchronous Mode

BAUD RATE (Kbps)	Fosc = 20 MHz			16 MHz			10 MHz			7.15909 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	-	-	9.766	+1.73	255	9.622	+0.23	185
19.2	19.53	+1.73	255	19.23	+0.16	207	19.23	+0.16	129	19.24	+0.23	92
76.8	76.92	+0.16	64	76.92	+0.16	51	75.76	-1.36	32	77.82	+1.32	22
96	96.15	+0.16	51	95.24	-0.79	41	96.15	+0.16	25	94.20	-1.88	18
300	294.1	-1.96	16	307.69	+2.56	12	312.5	+4.17	7	298.3	-0.57	5
500	500	0	9	500	0	7	500	0	4	NA	-	-
HIGH	5000	-	0	4000	-	0	2500	-	0	1789.8	-	0
LOW	19.53	-	255	15.625	-	255	9.766	-	255	6.991	-	255

BAUD RATE (Kbps)	Fosc = 5.0688 MHz			4 MHz			3.579545 MHz			1 MHz			32.768 kHz		
	KBAUD	% ERROR	SPBRG value (decimal)	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	-	-	0.303	+1.14	26
1.2	NA	-	-	NA	-	-	NA	-	-	1.202	+0.16	207	1.170	-2.48	6
2.4	NA	-	-	NA	-	-	NA	-	-	2.404	+0.16	103	NA	-	-
9.6	9.6	0	131	9.615	+0.16	103	9.622	+0.23	92	9.615	+0.16	25	NA	-	-
19.2	19.2	0	65	19.231	+0.16	51	19.04	-0.83	46	19.24	+0.16	12	NA	-	-
76.8	79.2	+3.13	15	76.923	+0.16	12	74.57	-2.90	11	83.34	+8.51	2	NA	-	-
96	97.48	+1.54	12	1000	+4.17	9	99.43	+3.57	8	NA	-	-	NA	-	-
300	316.8	+5.60	3	NA	-	-	298.3	-0.57	2	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	1267	-	0	100	-	0	894.9	-	0	250	-	0	8.192	-	0
LOW	4.950	-	255	3.906	-	255	3.496	-	255	0.9766	-	255	0.032	-	255

Table 18-4: Baud Rates for Asynchronous Mode (BRGH = 0)

BAUD RATE (Kbps)	FOSC = 20 MHz			16 MHz			10 MHz			7.15909 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	1.221	+1.73	255	1.202	+0.16	207	1.202	+0.16	129	1.203	+0.23	92
2.4	2.404	+0.16	129	2.404	+0.16	103	2.404	+0.16	64	2.380	-0.83	46
9.6	9.469	-1.36	32	9.615	+0.16	25	9.766	+1.73	15	9.322	-2.90	11
19.2	19.53	+1.73	15	19.23	+0.16	12	19.53	+1.73	7	18.64	-2.90	5
76.8	78.13	+1.73	3	83.33	+8.51	2	78.13	+1.73	1	NA	-	-
96	104.2	+8.51	2	NA	-	-	NA	-	-	NA	-	-
300	312.5	+4.17	0	NA	-	-	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	312.5	-	0	250	-	0	156.3	-	0	111.9	-	0
LOW	1.221	-	255	0.977	-	255	0.6104	-	255	0.437	-	255

BAUD RATE (Kbps)	FOSC = 5.0688 MHz			4 MHz			3.579545 MHz			1 MHz			32.768 kHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	0.31	+3.13	255	0.3005	-0.17	207	0.301	+0.23	185	0.300	+0.16	51	0.256	-14.67	1
1.2	1.2	0	65	1.202	+1.67	51	1.190	-0.83	46	1.202	+0.16	12	NA	-	-
2.4	2.4	0	32	2.404	+1.67	25	2.432	+1.32	22	2.232	-6.99	6	NA	-	-
9.6	9.9	+3.13	7	NA	-	-	9.322	-2.90	5	NA	-	-	NA	-	-
19.2	19.8	+3.13	3	NA	-	-	18.64	-2.90	2	NA	-	-	NA	-	-
76.8	79.2	+3.13	0	NA	-	-	NA	-	-	NA	-	-	NA	-	-
96	NA	-	-	NA	-	-	NA	-	-	NA	-	-	NA	-	-
300	NA	-	-	NA	-	-	NA	-	-	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-	NA	-	-	NA	-	-
HIGH	79.2	-	0	62.500	-	0	55.93	-	0	15.63	-	0	0.512	-	0
LOW	0.3094	-	255	3.906	-	255	0.2185	-	255	0.0610	-	255	0.0020	-	255

Table 18-5: Baud Rates for Asynchronous Mode (BRGH = 1)

BAUD RATE (Kbps)	FOSC = 20 MHz			16 MHz			10 MHz			7.15909 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
9.6	9.615	+0.16	129	9.615	+0.16	103	9.615	+0.16	64	9.520	-0.83	46
19.2	19.230	+0.16	64	19.230	+0.16	51	18.939	-1.36	32	19.454	+1.32	22
38.4	37.878	-1.36	32	38.461	+0.16	25	39.062	+1.7	15	37.286	-2.90	11
57.6	56.818	-1.36	21	58.823	+2.12	16	56.818	-1.36	10	55.930	-2.90	7
115.2	113.636	-1.36	10	111.111	-3.55	8	125	+8.51	4	111.860	-2.90	3
250	250	0	4	250	0	3	NA	-	-	NA	-	-
625	625	0	1	NA	-	-	625	0	0	NA	-	-
1250	1250	0	0	NA	-	-	NA	-	-	NA	-	-

BAUD RATE (Kbps)	FOSC = 5.0688 MHz			4 MHz			3.579545 MHz			1 MHz			32.768 kHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
9.6	9.6	0	32	NA	-	-	9.727	+1.32	22	8.928	-6.99	6	NA	-	-
19.2	18.645	-2.94	16	1.202	+0.17	207	18.643	-2.90	11	20.833	+8.51	2	NA	-	-
38.4	39.6	+3.12	7	2.403	+0.13	103	37.286	-2.90	5	31.25	-18.61	1	NA	-	-
57.6	52.8	-8.33	5	9.615	+0.16	25	55.930	-2.90	3	62.5	+8.51	0	NA	-	-
115.2	105.6	-8.33	2	19.231	+0.16	12	111.860	-2.90	1	NA	-	-	NA	-	-
250	NA	-	-	NA	-	-	223.721	-10.51	0	NA	-	-	NA	-	-
625	NA	-	-	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1250	NA	-	-	NA	-	-	NA	-	-	NA	-	-	NA	-	-

18.4 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 transmits and receives the LSb first. 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 either x16 or x64 of the bit shift rate, depending on the BRGH bit (TXSTA<2>). Parity is not supported by the hardware, but can be implemented in software (stored as the ninth data bit). Asynchronous mode is stopped during SLEEP.

Asynchronous mode is selected by clearing the SYNC bit (TXSTA<4>).

The USART Asynchronous module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver

18.4.1 USART Asynchronous Transmitter

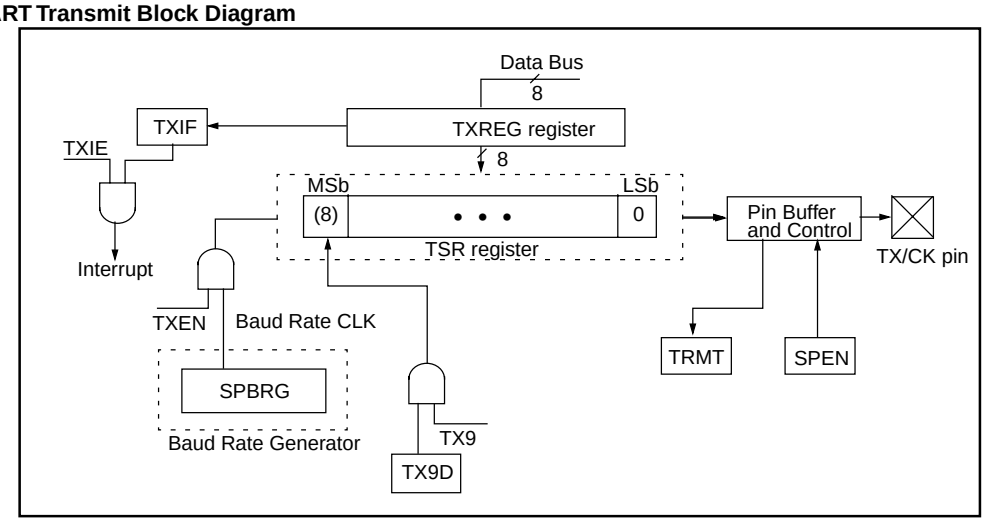
The USART transmitter block diagram is shown in Figure 18-1. The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until 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 register (if available). Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and the TXIF flag bit is set. This interrupt can be enabled/disabled by setting/clearing the TXIE enable bit. The TXIF flag bit will be set regardless of the state of the TXIE enable bit and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While the TXIF flag bit indicated the status of the TXREG register, the TRMT bit (TXSTA<1>) shows the status of the TSR register. The TRMT status bit is a read only bit which is set when the TSR register 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 register is empty.

Note 1:	The TSR register is not mapped in data memory so it is not available to the user.
Note 2:	When the TXEN bit is set, the TXIF flag bit will also be set since the transmit buffer is not yet full (still can move transmit data to the TXREG register).

Transmission is enabled by setting the TXEN enable bit (TXSTA<5>). The actual transmission will not occur until the TXREG register has been loaded with data and the baud rate generator (BRG) has produced a shift clock (Figure 18-1). The transmission can also be started by first loading the TXREG register and then setting the TXEN enable bit. Normally when transmission is first started, the TSR register is empty, so a transfer to the TXREG register will result in an immediate transfer to TSR resulting in an empty TXREG. A back-to-back transfer is thus possible (Figure 18-3). Clearing the TXEN enable bit during a transmission will cause the transmission to be aborted and will reset the transmitter. As a result the TX/CK pin will revert to hi-impedance.

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

Figure 18-1: USART Transmit Block Diagram



Steps to follow when setting up a Asynchronous Transmission:

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set the BRGH bit. (Subsection **18.3 "USART Baud Rate Generator (BRG)"**)
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, GIE and PEIE bits.
4. If 9-bit transmission is desired, then set the TX9 bit.
5. Enable the transmission by setting the TXEN bit, which will also set the TXIF bit.
6. If 9-bit transmission is selected, the ninth bit should be loaded in the TX9D bit.
7. Load data to the TXREG register (starts transmission).

Figure 18-2: Asynchronous Master Transmission

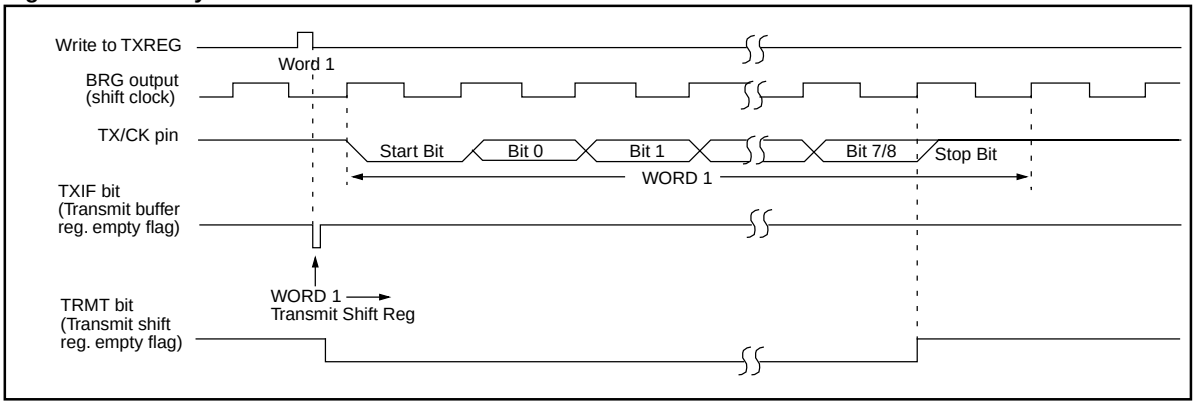


Figure 18-3: Asynchronous Master Transmission (Back to Back)

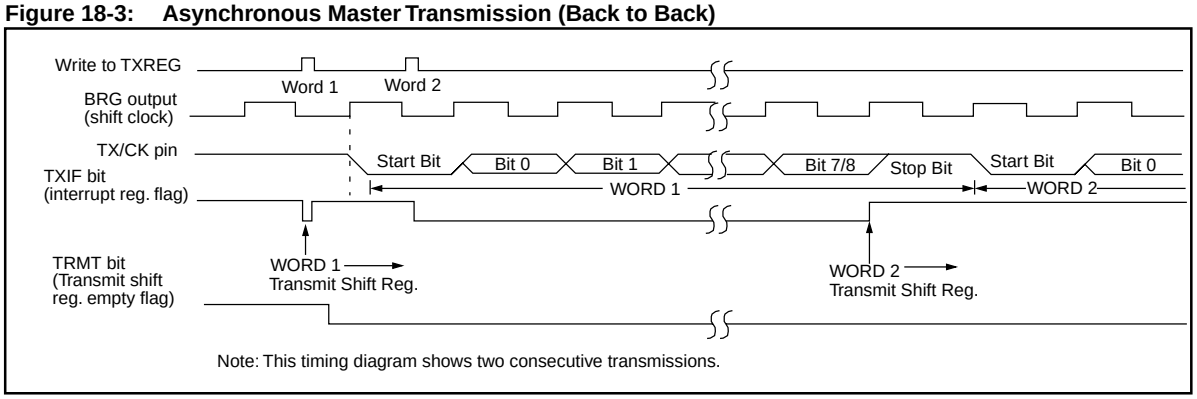


Table 18-6: Registers Associated with Asynchronous Transmission

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
PIR	TXIF (1)								0	0
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	0000 0000	0000 0000
PIE	TXIE (1)								0	0
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'.
Shaded cells are not used for Asynchronous Transmission.
Note 1: The position of this bit is device dependent.

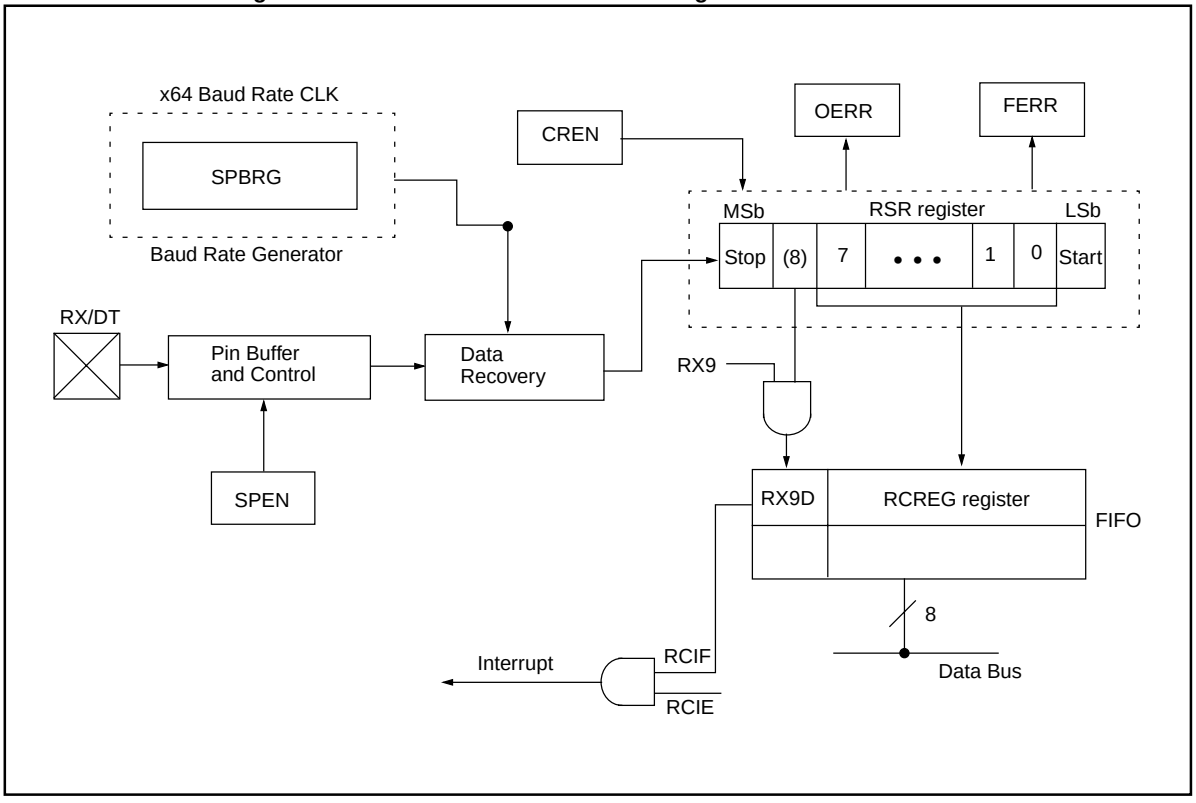
18.4.2 USART Asynchronous Receiver

The receiver block diagram is shown in Figure 18-4. The data is received on the RX/DT pin and drives the data recovery block. The data recovery block is actually a high speed shifter operating at x16 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 the CREN bit (RCSTA<4>).

The heart of the receiver is the receive (serial) shift register (RSR). After sampling the RX/TX pin for the STOP bit, the received data in the RSR is transferred to the RCREG register (if it is empty). If the transfer is complete, the RCIF flag bit is set. The actual interrupt can be enabled/disabled by setting/clearing the RCIE enable bit. The RCIF flag bit is a read only bit which is cleared by the hardware. It is cleared when the RCREG register has been read and is empty. The 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 register. On the detection of the STOP bit of the third byte, if the RCREG register is still full then overrun error bit, OERR (RCSTA<1>), will be set. The word in the RSR will be lost. The RCREG register 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 resetting the receive logic (the CREN bit is cleared and then set). If the OERR bit is set, transfers from the RSR register to the RCREG register are inhibited, so it is essential to clear the OERR bit if it is set. Framing error bit, FERR (RCSTA<2>), is set if a stop bit is detected as a low level. The FERR bit and the 9th receive bit are buffered the same way as the receive data. Reading the RCREG will load the RX9D and FERR bits with new values, therefore it is essential for the user to read the RCSTA register before reading the next RCREG register in order not to lose the old (previous) information in the FERR and RX9D bits.

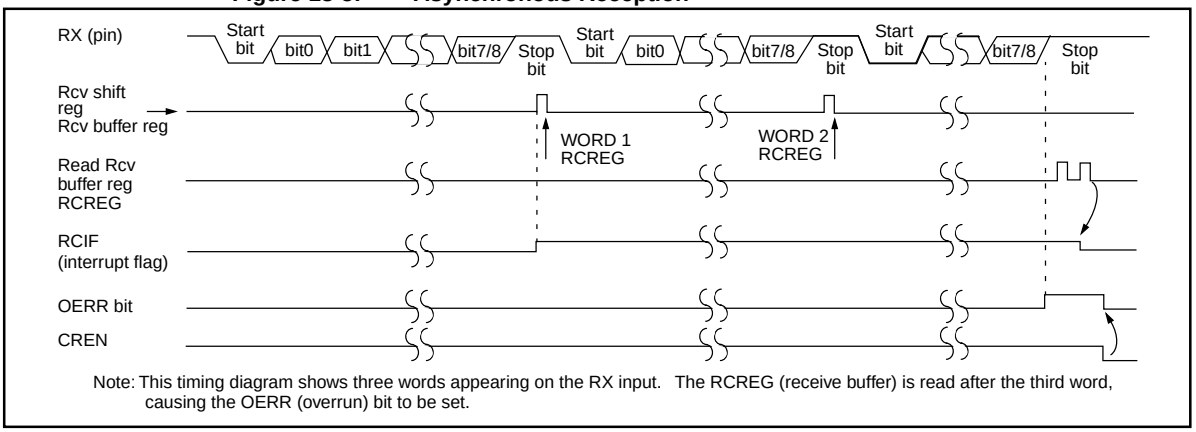
Figure 18-4: USART Receive Block Diagram



Steps to follow when setting up an Asynchronous Reception:

1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH. (Subsection **18.3 "USART Baud Rate Generator (BRG)"**).
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, GIE and PEIE bits.
4. If 9-bit reception is desired, then set the RX9 bit.
5. Enable the reception by setting the CREN bit.
6. The RCIF flag bit will be set when reception is complete and an interrupt will be generated if the RCIE bit was set.
7. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG register.
9. If any error occurred, clear the error by clearing the CREN bit.

Figure 18-5: Asynchronous Reception

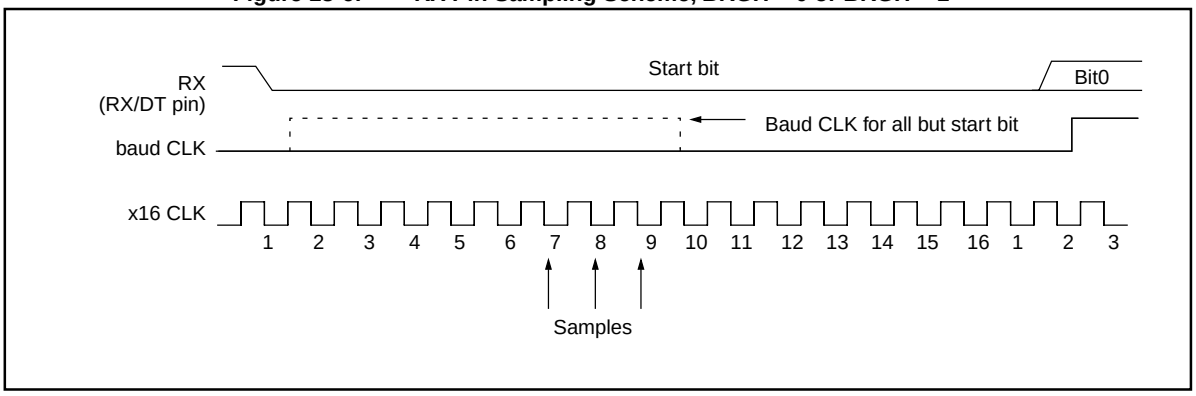


Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

18.4.3 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 pin. Figure 18-6 shows the waveform for the sampling circuit. The sampling operates the same regardless of the state of the BRGH bit, only the source of the x16 clock is different.

Figure 18-6: RX Pin Sampling Scheme, BRGH = 0 or BRGH = 1



18.4.3.1 Device Exceptions

All new devices will use the sampling scheme shown in Figure 18-6. Devices that have an exception to the above sampling scheme are:

- PIC16C63
- PIC16C65
- PIC16C65A
- PIC16C73
- PIC16C73A
- PIC16C74
- PIC16C74A

These devices have a sampling circuitry that works as follows. If the BRGH bit (TXSTA<2>) is clear (i.e., at the low baud rates), the sampling is done on the seventh, eighth and ninth falling edges of a x16 clock (Figure 18-7). If bit BRGH is set (i.e., at the high baud rates), the sampling is done on the 3 clock edges preceding the second rising edge after the first falling edge of a x4 clock (Figure 18-8 and Figure 18-9).

Figure 18-7: RX Pin Sampling Scheme (BRGH = 0)

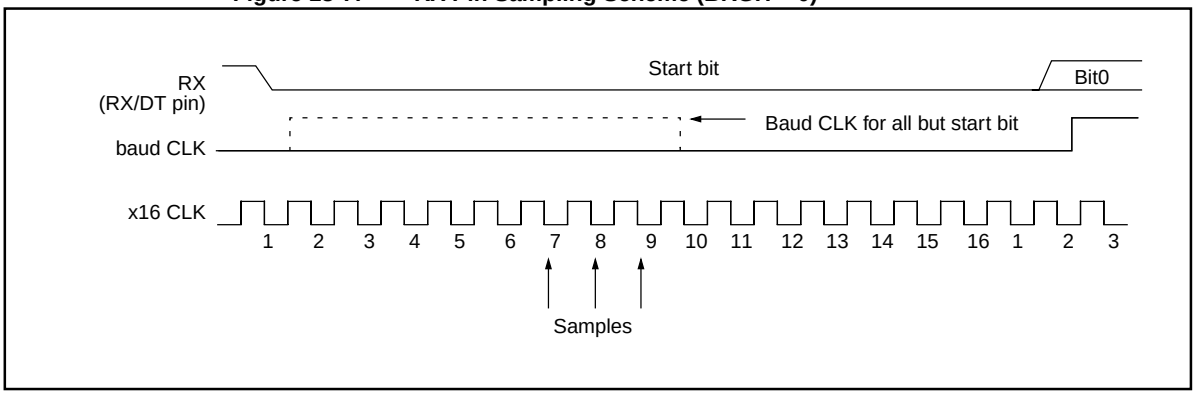


Figure 18-8: RX Pin Sampling Scheme (BRGH = 1)

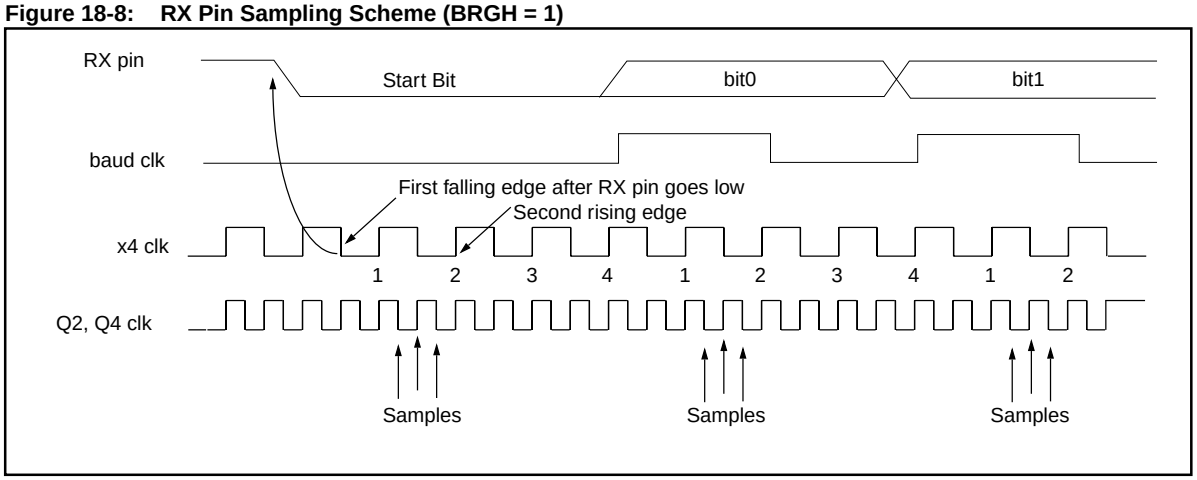


Figure 18-9: RX Pin Sampling Scheme (BRGH = 1)

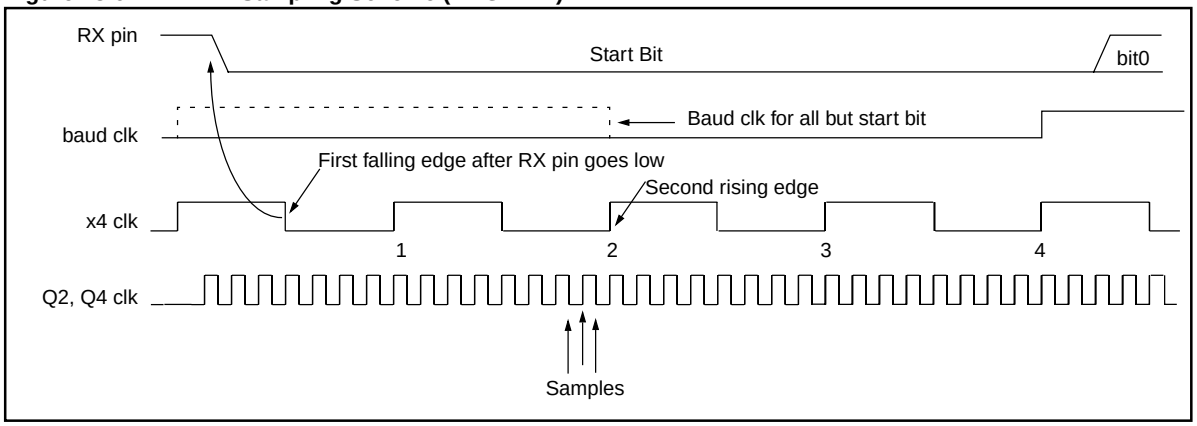


Table 18-7: Registers Associated with Asynchronous Reception

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
PIR	RCIF (1)								0	0
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	0000 0000	0000 0000
PIE	RCIE (1)								0	0
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'.
Shaded cells are not used for Asynchronous Reception.
Note 1: The position of this bit is device dependent.

18.5 USART Synchronous Master Mode

In Synchronous Master 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. Synchronous mode is entered by setting the SYNC bit (TXSTA<4>). In addition, the SPEN enable bit (RCSTA<7>) is set in order to configure the TX/CK and RX/DT 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 bit (TXSTA<7>).

18.5.1 USART Synchronous Master Transmission

The USART transmitter block diagram is shown in Figure 18-1. The heart of the transmitter is the transmit (serial) shift register (TSR). The shift register obtains its data from the read/write transmit buffer register TXREG. The TXREG register is loaded with data in software. The TSR register 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 the TXREG (if available). Once the TXREG register transfers the data to the TSR register (occurs in one Tcycle), the TXREG is empty and the TXIF interrupt flag bit is set. The interrupt can be enabled/disabled by setting/clearing enable the TXIE bit. The TXIF flag bit will be set regardless of the state of the TXIE enable bit and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While the TXIF flag bit indicates the status of the TXREG register, the TRMT bit (TXSTA<1>) shows the status of the TSR register. The TRMT bit 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 register 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 bit (TXSTA<5>). The actual transmission will not occur until the TXREG register has been loaded with data. The first data bit will be shifted out on the next available rising edge of the clock on the CK line. Data out is stable at the falling edge of the synchronous clock (Figure 18-10). The transmission can also be started by first loading the TXREG register and then setting the TXEN bit. This is advantageous when slow baud rates are selected, since the 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 register is empty, so a transfer to the TXREG register will result in an immediate transfer to TSR resulting in an empty TXREG. Back-to-back transfers are possible.

Clearing the TXEN bit during a transmission will cause the transmission to be aborted and will reset the transmitter. The DT and CK pins will revert to hi-impedance. If either of the CREN or SREN bits are set during a transmission, the transmission is aborted and the DT pin reverts to a hi-impedance state (for a reception). The 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 on-going transmission and receive a single word), then after the single word is received, the SREN bit 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 the TXEN bit should be cleared.

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

Steps to follow when setting up a Synchronous Master Transmission:

1. Initialize the SPBRG register for the appropriate baud rate (Subsection **18.3 "USART Baud Rate Generator (BRG)"**).
2. Enable the synchronous master serial port by setting the SYNC, SPEN, and CSRC bits.
3. If interrupts are desired, then set the TXIE bit.
4. If 9-bit transmission is desired, then set the TX9 bit.
5. Enable the transmission by setting the TXEN bit.
6. If 9-bit transmission is selected, the ninth bit should be loaded in the TX9D bit.
7. Start transmission by loading data to the TXREG register.

Table 18-8: Registers Associated with Synchronous Master Transmission

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
PIR	TXIF (1)								0	0
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	0000 0000	0000 0000
PIE	TXIE (1)								0	0
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'.
Shaded cells are not used for Synchronous Master Transmission.
Note 1: The position of this bit is device dependent.

Figure 18-10: Synchronous Transmission

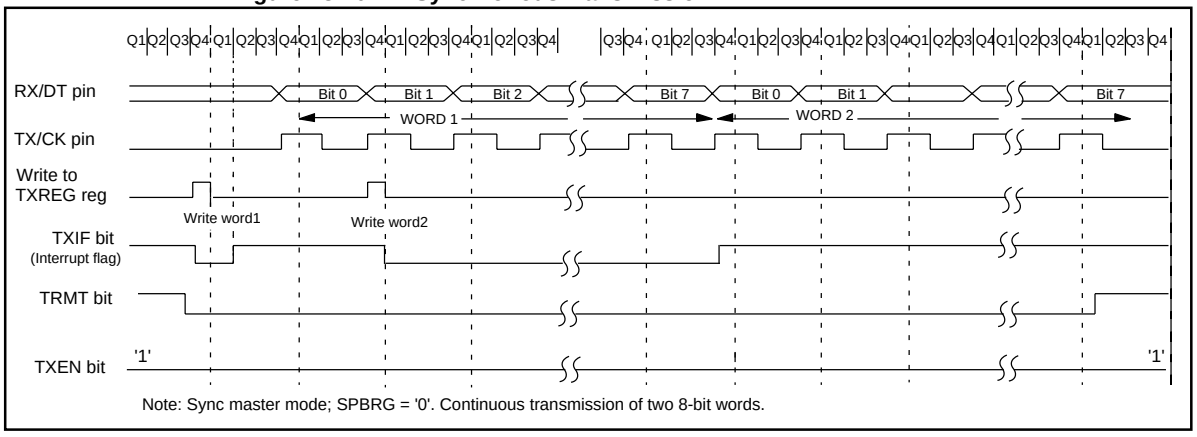
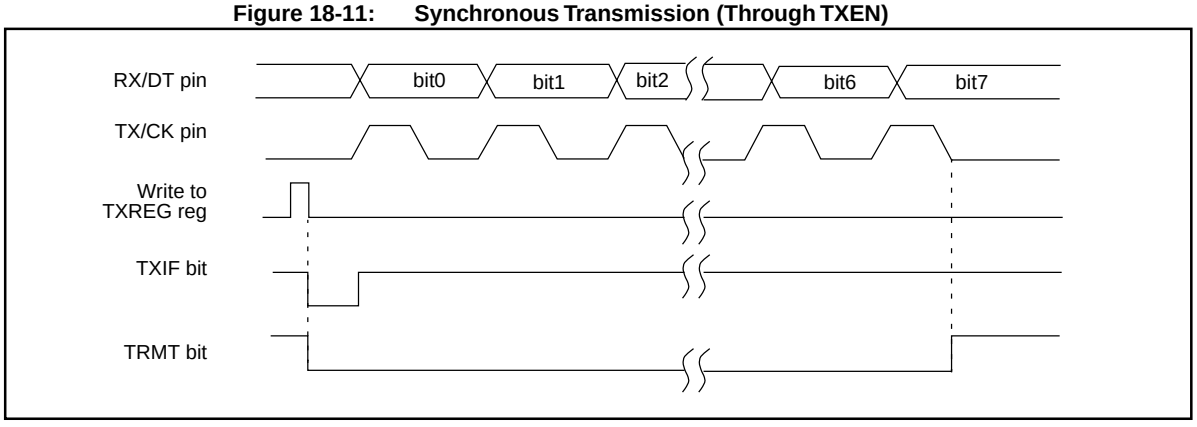


Figure 18-11: Synchronous Transmission (Through TXEN)



18.5.2 USART Synchronous Master Reception

Once Synchronous mode is selected, reception is enabled by setting either of the SREN (RCSTA<5>) or CREN (RCSTA<4>) bits. Data is sampled on the RX/DT pin on the falling edge of the clock. If the SREN bit is set, then only a single word is received. If the CREN bit is set, the reception is continuous until the CREN bit is cleared. If both bits are set then the CREN bit takes precedence. After clocking the last serial data bit, the received data in the Receive Shift Register (RSR) is transferred to the RCREG register (if it is empty). When the transfer is complete, the RCIF interrupt flag bit is set. The actual interrupt can be enabled/disabled by setting/clearing the RCIE enable bit. The RCIF flag bit is a read only bit which is cleared by the hardware. In this case it is cleared when the RCREG register has been read and is empty. The 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 register. On the clocking of the last bit of the third byte, if the RCREG register is still full then overrun error bit, OERR (RCSTA<1>), is set and the word in the RSR is lost. The RCREG register can be read twice to retrieve the two bytes in the FIFO. The OERR bit has to be cleared in software (by clearing the CREN bit). If the OERR bit is set, transfers from the RSR to the 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 load the RX9D bit with a new value, therefore it is essential for the user to read the RCSTA register before reading RCREG in order not to lose the old (previous) information in the RX9D bit.

Steps to follow when setting up a Synchronous Master Reception:

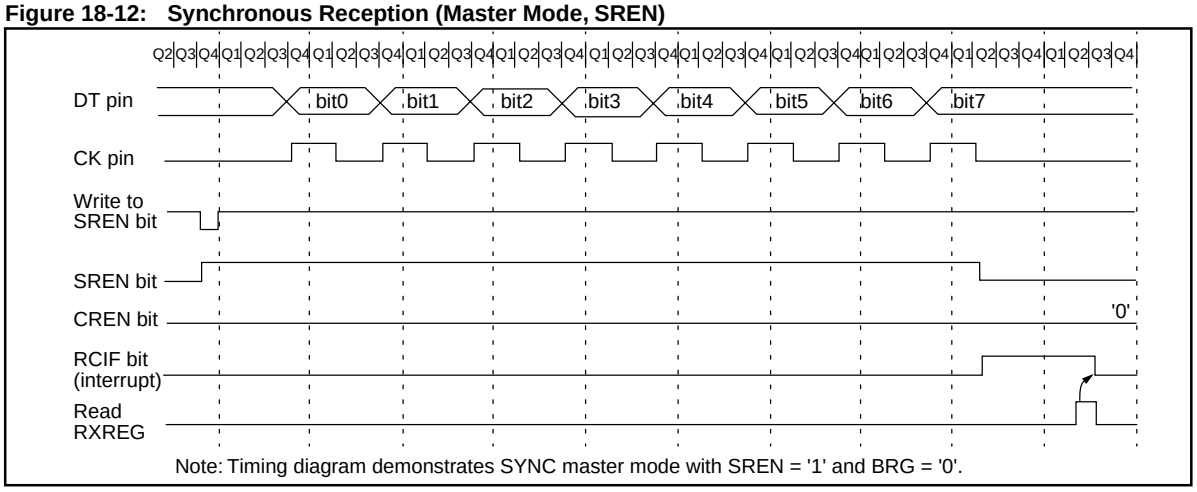
1. Initialize the SPBRG register for the appropriate baud rate. (Subsection **18.3 "USART Baud Rate Generator (BRG)"**)
2. Enable the synchronous master serial port by setting the SYNC, SPEN, and CSRC bits.
3. Ensure that the CREN and SREN bits are clear.
4. If interrupts are desired, then set the RCIE bit.
5. If 9-bit reception is desired, then set the RX9 bit.
6. If a single reception is required, set the SREN bit. For continuous reception set the CREN bit.
7. The RCIF bit will be set when reception is complete and an interrupt will be generated if the RCIE bit was set.
8. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
9. Read the 8-bit received data by reading the RCREG register.
10. If any error occurred, clear the error by clearing the CREN bit.

Table 18-9: Registers Associated with Synchronous Master Reception

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
PIR	RCIF (1)								0	0
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	0000 0000	0000 0000
PIE	RCIE (1)								0	0
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented read as '0'.
Shaded cells are not used for Synchronous Master Reception.
Note 1: The position of this bit is device dependent.

Figure 18-12: Synchronous Reception (Master Mode, SREN)



Note: Timing diagram demonstrates SYNC master mode with SREN = '1' and BRG = '0'.

18.6 USART Synchronous Slave Mode

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 master mode). This allows the device to transfer or receive data while in SLEEP mode. Slave mode is entered by clearing the CSRC bit (TXSTA<7>).

18.6.1 USART Synchronous Slave Transmit

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

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

a) The first word will immediately transfer to the TSR register and transmit.
b) The second word will remain in TXREG register.
c) The TXIF flag bit will not be set.
d) When the first word has been shifted out of TSR, the TXREG register will transfer the second word to the TSR and the TXIF flag bit will now be set.
e) If the TXIE enable bit is set, the interrupt will wake the chip from SLEEP and if the global interrupt is enabled, the program will branch to the interrupt vector (0004h).

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 and SREN bits.
3. If interrupts are desired, then set the TXIE enable bit.
4. If 9-bit transmission is desired, then set the TX9 bit.
5. Enable the transmission by setting the TXEN enable bit.
6. If 9-bit transmission is selected, the ninth bit should be loaded into the TX9D bit.
7. Start transmission by loading data to the TXREG register.

Table 18-10: Registers Associated with Synchronous Slave Transmission

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
PIR	TXIF (1)								0	0
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	0000 0000	0000 0000
PIE	TXIE (1)								0	0
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented read as '0'.
Shaded cells are not used for Synchronous Slave Transmission.
Note 1: The position of this bit is device dependent.

18.6.2 USART Synchronous Slave Reception

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

If receive is enabled, by setting the CREN bit, prior to the SLEEP instruction, then a word may be received during SLEEP. On completely receiving the word, the RSR register will transfer the data to the RCREG register and if the RCIE enable bit 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 (0004h).

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 enable bit.
3. If 9-bit reception is desired, then set the RX9 bit.
4. To enable reception, set the CREN enable 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 the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREG register.
8. If any error occurred, clear the error by clearing the CREN bit.

Table 18-11: Registers Associated with Synchronous Slave Reception

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
PIR	RCIF (1)								0	0
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	0000 0000	0000 0000
PIE	RCIE (1)								0	0
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented read as '0'.
Shaded cells are not used for Synchronous Slave Reception.
Note 1: The position of this bit is device dependent.

18.7 Initialization

Example 18-2 is an initialization routine for asynchronous Transmitter/Receiver mode. Example 18-3 is for the synchronous mode. In both examples the data is 8-bits, and the value to load into the SPBRG register is dependent on the desired baud rate and the device frequency.

Example 18-2: Asynchronous Transmitter/Receiver

```
    BSF    STATUS,RP0     ; Go to Bank1
    MOVLW  <baudrate>     ; Set Baud rate
    MOVWF  SPBRG
    MOVLW  0x40           ; 8-bit transmit, transmitter enabled,
    MOVWF  TXSTA          ; asynchronous mode, low speed mode
    BSF    PIE1,TXIE      ; Enable transmit interrupts
    BSF    PIE1,RCIE      ; Enable receive interrupts
    BCF    STATUS,RP0     ; Go to Bank 0
    MOVLW  0x90           ; 8-bit receive, receiver enabled,
    MOVWF  RCSTA          ; serial port enabled
```

Example 18-3: Synchronous Transmitter/Receiver

```
    BSF    STATUS,RP0      ; Go to Bank 1
    MOVLW  <baudrate>      ; Set Baud Rate
    MOVWF  SPBRG
    MOVLW  0xB0            ; Synchronous Master,8-bit transmit,
    MOVWF  TXSTA           ; transmitter enabled, low speed mode
    BSF    PIE1,TXIE       ; Enable transmit interrupts
    BSF    PIE1,RCIE       ; Enable receive interrupts
    BCF    STATUS,RP0      ; Go to Bank 0
    MOVLW  0x90            ; 8-bit receive, receiver enabled,
    MOVWF  RCSTA           ; continuous receive, serial port enabled
```

18.8 Design Tips

Question 1: ***Using the Asynchronous mode I am getting a lot of transmission errors.***

Answer 1:

The most common reasons are

1. You are using the high speed mode (BRGH is set) on one of the devices which has an errata for this mode (PIC16C65/65A/73/73A/74/74A).
2. You have incorrectly calculated the value to load in to the SPBRG register
3. The sum of the baud errors for the transmitter and receiver is too high.

18.9 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to this section are:

Title	Application Note #
Serial Port Utilities	AN547
Servo Control of a DC Brushless Motor	AN543

18.10 Revision History

Revision A

This is the initial released revision of the USART module description.

Section 19. Voltage Reference

HIGHLIGHTS

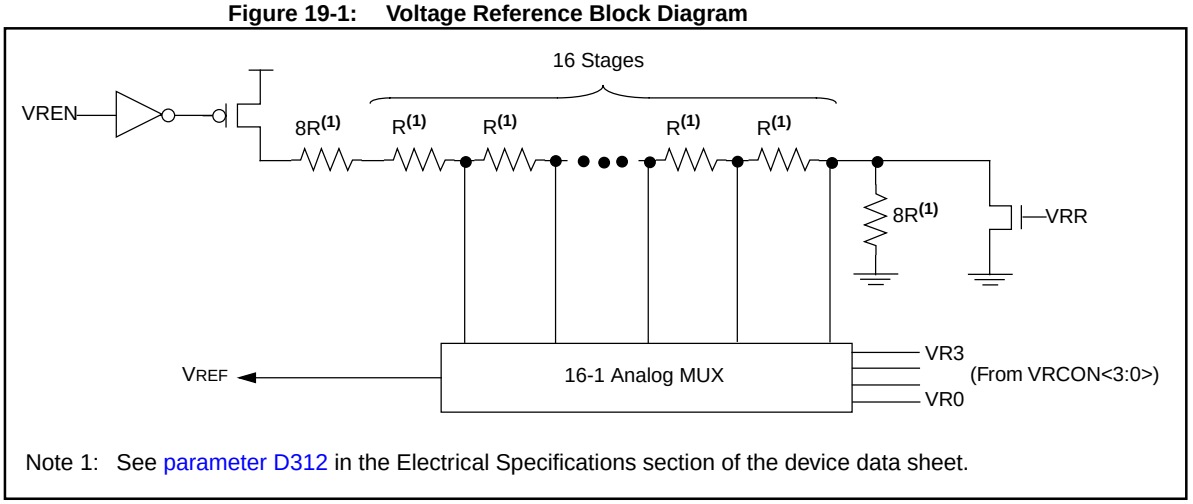
This section of the manual contains the following major topics:

19.1 Introduction

The Voltage Reference module is typically used in conjunction with the Comparator module. The comparator module's inputs do not require very large drive, and therefore the drive capability of the Voltage Reference is limited.

The Voltage Reference is a 16-tap resistor ladder network that provides a selectable voltage reference. The resistor ladder is segmented to provide two ranges of VREF values and has a power-down function to conserve power when the reference is not being used. The VRCON register controls the operation of the reference as shown in Figure 19-1. The block diagram is given in Figure 19-1. Within each range, the 16 steps are monotonic (i.e. each increasing code will result in an increasing output).

Figure 19-1: Voltage Reference Block Diagram



Note 1: See parameter D312 in the Electrical Specifications section of the device data sheet.

Table 19-1: Typical Voltage Reference with VDD = 5.0V

VR3:VR0	VREF	
	VRR = 1	VRR = 0
0000	0.00 V	1.25 V
0001	0.21 V	1.41 V
0010	0.42 V	1.56 V
0011	0.63 V	1.72 V
0100	0.83 V	1.88 V
0101	1.04 V	2.03 V
0110	1.25 V	2.19 V
0111	1.46 V	2.34 V
1000	1.67 V	2.50 V
1001	1.88 V	2.66 V
1010	2.08 V	2.81 V
1011	2.29 V	2.97 V
1100	2.50 V	3.13 V
1101	2.71 V	3.28 V
1110	2.92 V	3.44 V
1111	3.13 V	3.59 V

19.2 Control Register

Register 19-1: VRCON Register

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
VREN	VROE	VRR	—	VR3	VR2	VR1	VR0
bit 7							bit 0

bit 7 **VREN:** VREF Enable
1 = VREF circuit powered on
0 = VREF circuit powered down

bit 6 **VROE:** VREF Output Enable
1 = VREF is internally connected to Comparator module's VREF. This voltage level is also output on the VREF pin
0 = VREF is not connected to the comparator module. This voltage is disconnected from the VREF pin

bit 5 **VRR:** VREF Range selection
1 = 0V to 0.75 VDD, with VDD/24 step size
0 = 0.25 VDD to 0.75 VDD, with VDD/32 step size

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

bit 3:0 **VR3:VR0:** VREF value selection $0 \le VR3:VR0 \le 15$

When VRR = 1:

$V_{REF} = (VR<3:0>/ 24) \bullet V_{DD}$

When VRR = 0:

$V_{REF} = 1/4 * V_{DD} + (VR3:VR0/ 32) \bullet V_{DD}$

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

19.3 Configuring the Voltage Reference

The Voltage Reference can output 16 distinct voltage levels for each range.

The equations used to calculate the output of the Voltage Reference are as follows:

if VRR = 1: $V_{REF} = (VR3:VR0/24) \times V_{DD}$

if VRR = 0: $V_{REF} = (V_{DD} \times 1/4) + (VR3:VR0/32) \times V_{DD}$

The settling time of the Voltage Reference must be considered when changing the V_{REF} output. Example 19-1 shows an example of how to configure the Voltage Reference for an output voltage of 1.25V with V_{DD} = 5.0V.

Generally the V_{REF} and V_{DD} of the system will be known and you need to determine the value to load into VR3:VR0. Equation 19-1 shows how to calculate the VR3:VR0 value. There will be some error since VR3:VR0 can only be an integer, and the V_{REF} and V_{DD} levels must be chosen so that the result is not greater then 15.

Equation 19-1: Calculating VR3:VR0

When VRR = 1

$$VR3:VR0 = \frac{V_{REF}}{V_{DD}} \quad X\ 24$$

When VRR = 0

$$VR3:VR0 = \frac{V_{REF} - V_{DD}/4}{V_{DD}} \quad X\ 32$$

19.4 Voltage Reference Accuracy/Error

The full range of VSS to VDD cannot be realized due to the construction of the module. The transistors on the top and bottom of the resistor ladder network (Figure 19-1) keep VREF from approaching VSS or VDD. The Voltage Reference is VDD derived and therefore, the VREF output changes with fluctuations in VDD. The absolute accuracy of the Voltage Reference can be found in the Device Data Sheets electrical specification parameter D311.

19.5 Operation During Sleep

When the device wakes up from sleep through an interrupt or a Watchdog Timer time-out, the contents of the VRCON register are not affected. To minimize current consumption in SLEEP mode, the Voltage Reference should be disabled.

19.6 Effects of a Reset

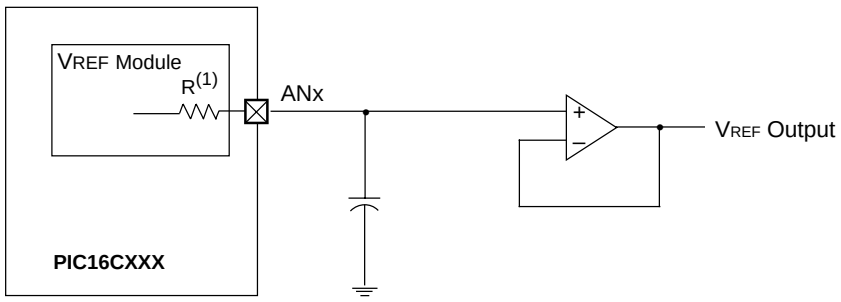
A device reset disables the Voltage Reference by clearing the VREN bit (VRCON<7>). This reset also disconnects the reference from the VREF pin by clearing the VROE bit (VRCON<6>) and selects the high voltage range by clearing the VRR bit (VRCON<5>). The VREF value select bits, VRCON<3:0>, are also cleared.

19.7 Connection Considerations

The Voltage Reference Module operates independently of the comparator module. The output of the reference generator may be connected to the VREF pin if the corresponding TRIS bit is set and the VROE bit (VRCON<6>) is set. Enabling the Voltage Reference output onto the VREF pin with an input signal present will increase current consumption. Configuring the VREF as a digital output with VREF enabled will also increase current consumption.

The VREF pin can be used as a simple D/A output with limited drive capability. Due to the limited drive capability, a buffer must be used in conjunction with the Voltage Reference output for external connections to VREF. Figure 19-2 shows an example buffering technique.

Figure 19-2: Voltage Reference Output Buffer Example



Note 1: R is the Voltage Reference Output Impedance and is dependent upon the Voltage Reference Configuration VRCON<3:0> and VRCON<5>.

19.8 Initialization

Example 19-1 shows the steps to configure the voltage reference module.

Example 19-1: Voltage Reference Configuration

```
    MOVLW   0x02        ; 4 Inputs Muxed to 2 comparators
    MOVWF   CMCON       ;
    BSF     STATUS,RP0  ; go to Bank1
    MOVLW   0x07        ; RA3:RA0 are outputs
    MOVWF   TRISA       ; outputs
    MOVLW   0xA6        ; enable VREF
    MOVWF   VRCON       ; low range set VR3:VR0 = 6
    BCF     STATUS,RP0  ; go to Bank0
    CALL    DELAY10     ; 10 µs delay
```

19.9 Design Tips

Question 1: ***My VREF is not what I expect.***

Answer 1:

Any variation of the device VDD will translate directly onto the VREF pin. Also ensure that you have correctly calculated (specified) the VDD divider which generates the VREF.

Question 2: ***I am connecting VREF into a low impedance circuit, and the VREF is not at the expected level.***

Answer 2:

The Voltage Reference module is not intended to drive large loads. A buffer must be used between the PICmicro's VREF pin and the load.

19.10 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to Voltage Reference are:

Title	Application Note #
Resistance and Capacitance Meter using a PIC16C622	AN611

19.11 Revision History

Revision A

This is the initial released revision of the Voltage Reference description.

Section 20. Comparator

HIGHLIGHTS

This section of the manual contains the following major topics:

20.1 Introduction

The comparator module contains two analog comparators. The inputs to the comparators are multiplexed with the I/O pins. The on-chip Voltage Reference (see the **"Voltage Reference"** section) can also be an input to the comparators.

The CMCON register, shown in Figure 20-1, controls the comparator input and output multiplexers. A block diagram of the comparator is shown in Figure 20-1.

20.2 Control Register

Register 20-1: CMCON Register

R-0	R-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
C2OUT	C1OUT	—	—	CIS	CM2	CM1	CM0
bit 7							bit 0

bit 7 **C2OUT**: Comparator2 Output Indicator bit
1 = C2 VIN+ > C2 VIN–
0 = C2 VIN+ < C2 VIN–

bit 6 **C1OUT**: Comparator1 Output Indicator bit
1 = C1 VIN+ > C1 VIN–
0 = C1 VIN+ < C1 VIN–

bit 5:4 **Unimplemented:** Read as '0'

bit 3 **CIS**: Comparator Input Switch bit

When CM2:CM0: = 001:

1 = C1 VIN– connects to AN3
0 = C1 VIN– connects to AN0

When CM2:CM0 = 010:

1 = C1 VIN– connects to AN3
C2 VIN– connects to AN2
0 = C1 VIN– connects to AN0
C2 VIN– connects to AN1

bit 2:0 **CM2:CM0**: Comparator Mode Select bits
See Figure 20-1.

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

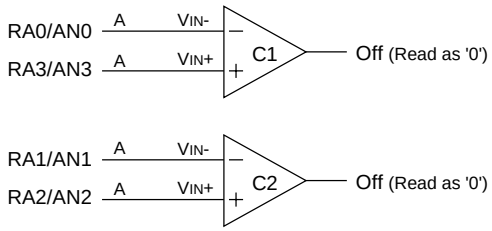
20.3 Comparator Configuration

There are eight modes of operation for the comparators. The CMCON register is used to select the mode. Figure 20-1 shows the eight possible modes. The TRIS register controls the data direction of the comparator I/O pins for each mode. If the comparator mode is changed, the comparator output level may not be valid for the new mode for the delay specified in the electrical specifications of the device.

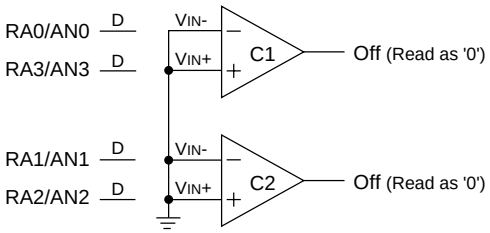
Note:	Comparator interrupts should be disabled during a comparator mode change, otherwise a false interrupt may occur.

Figure 20-1: Comparator I/O Operating Modes

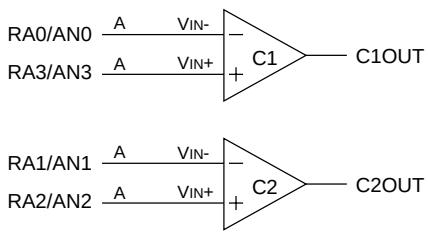
CM2:CM0 = 000
Comparators Reset (POR Default Value)



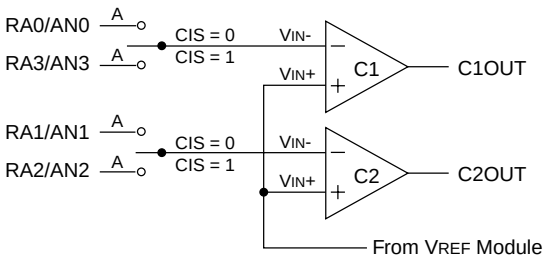
CM2:CM0 = 111
Comparators Off



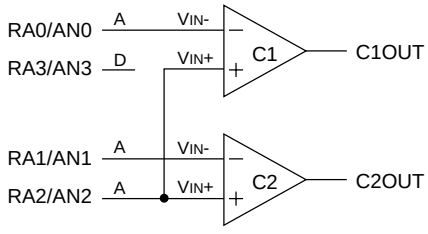
CM2:CM0 = 100
Two Independent Comparators



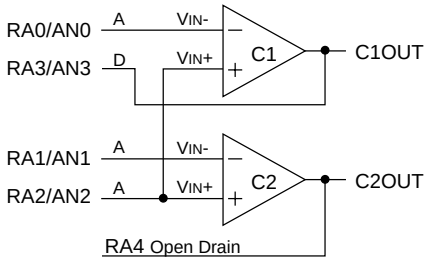
CM2:CM0 = 010
Four Inputs Multiplexed to Two Comparators



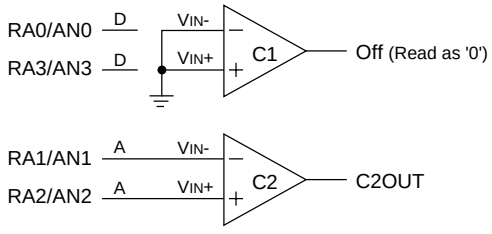
CM2:CM0 = 011
Two Common Reference Comparators



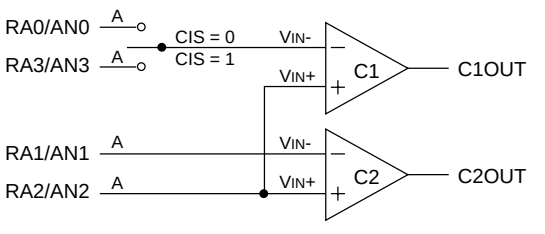
CM2:CM0 = 110
Two Common Reference Comparators with Outputs



CM2:CM0 = 101
One Independent Comparator



CM2:CM0 = 001
Three Inputs Multiplexed to Two Comparators



A = Analog Input, port reads as zeros always.
D = Digital Input.
CIS (CMCON<3>) is the Comparator Input Switch.

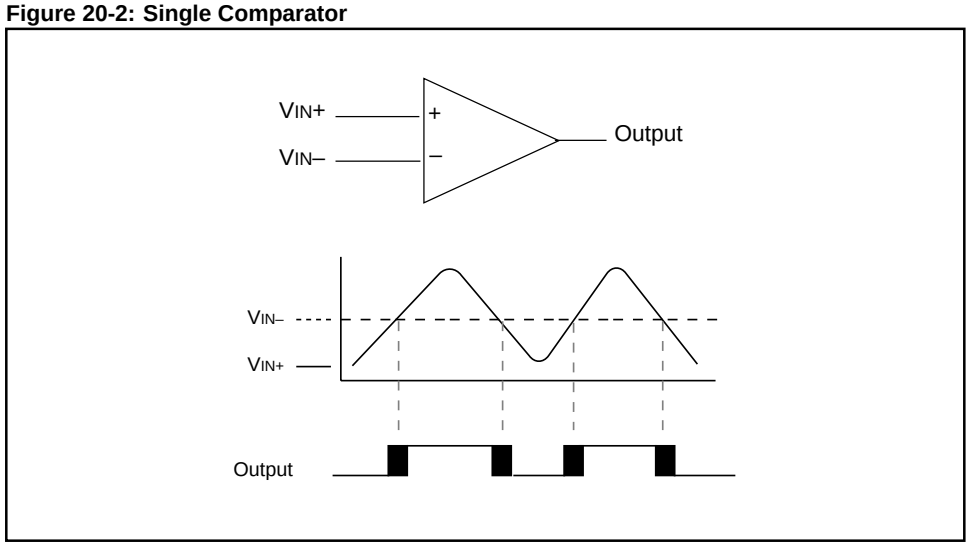
20.4 Comparator Operation

A single comparator is shown in Figure 20-2 along with the relationship between the analog input levels and the digital output. When the analog input at VIN+ is less than the analog input VIN–, the output of the comparator is a digital low level. When the analog input at VIN+ is greater than the analog input VIN–, the output of the comparator is a digital high level. The shaded areas of the output of the comparator in Figure 20-2 represent the uncertainty due to input offsets and response time.

20.5 Comparator Reference

An external or internal reference signal may be used depending on the comparator operating mode. The analog signal that is present at VIN– is compared to the signal at VIN+, and the digital output of the comparator is adjusted accordingly (Figure 20-2).

Figure 20-2: Single Comparator



20.5.1 External Reference Signal

When external voltage references are used, the comparator module can be configured to have the comparators operate from the same or different reference sources. The reference signal must be between VSS and VDD, and can be applied to either pin of the comparator(s).

20.5.2 Internal Reference Signal

The comparator module also allows the selection of an internally generated voltage reference for the comparators. The **"Voltage Reference"** section contains a detailed description of the Voltage Reference Module that provides this signal. The internal reference signal is used when the comparators are in mode CM2:CM0 = 010 (Figure 20-1). In this mode, the internal voltage reference is applied to the VIN+ input of both comparators.

The internal voltage reference may be used in any comparator mode. When used in this fashion the I/O/VREF pin may be used for I/O. The voltage reference is connected to the VREF pin.

20.6 Comparator Response Time

Response time is the minimum time, after selecting a new reference voltage or input source, before the comparator output is guaranteed to have a valid level. If the internal reference is changed, the maximum settling time of the internal voltage reference must be considered when using the comparator outputs. Otherwise the maximum response time of the comparators should be used.

20.7 Comparator Outputs

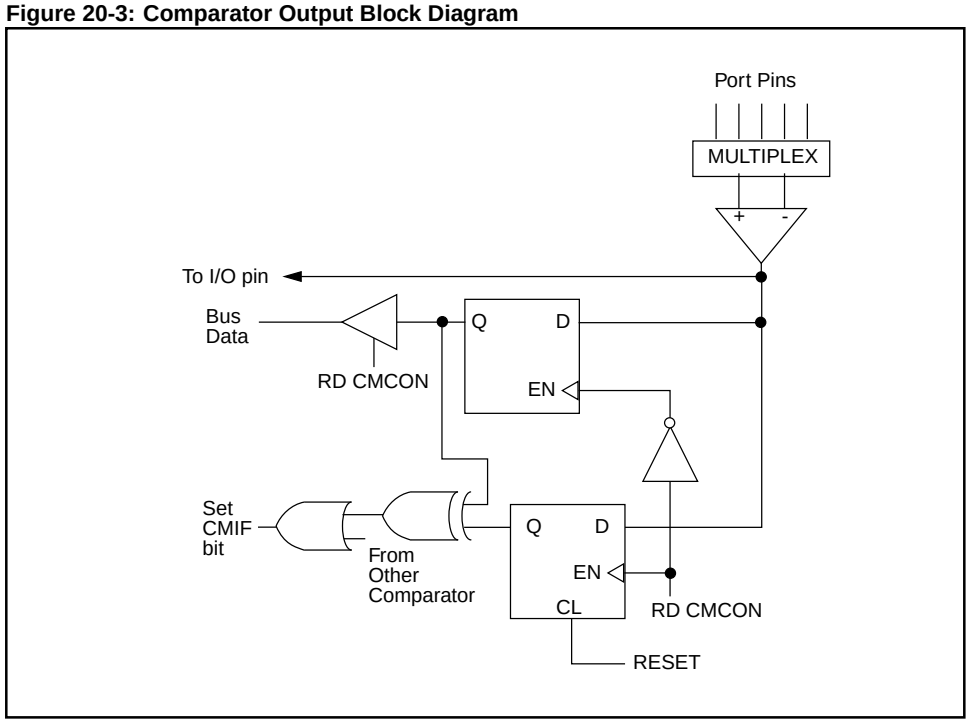
The comparator outputs are read through the CMCON register. These bits are read only. The comparator outputs may also be directly output to the I/O pins. When CM2:CM0 = 110, multiplexors in the output path of the I/O pins will switch and the output of each pin will be the unsynchronized output of the comparator. The uncertainty of each of the comparators is related to the input offset voltage and the response time given in the specifications. Figure 20-3 shows the comparator output block diagram.

The TRIS bits will still function as the output enable/disable for the I/O pins while in this mode.

Note 1: When reading the Port register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert an analog input according to the Schmitt Trigger input specification.

Note 2: Analog levels on any pin that is defined as a digital input may cause the input buffer to consume more current than is specified.

Figure 20-3: Comparator Output Block Diagram



20.8 Comparator Interrupts

The comparator interrupt flag is set whenever the comparators value changes relative to the last value loaded into CMxOUT bits. Software will need to maintain information about the status of the output bits, as read from CMCON<7:6>, to determine the actual change that has occurred. The CMIF bit, is the comparator interrupt flag. The CMIF bit must be cleared. Since it is also possible to set this bit, a simulated interrupt may be initiated.

The CMIE bit and the PEIE bit (INTCON<6>) must be set to enable the interrupt. In addition, the GIE bit must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMIF bit will still be set if an interrupt condition occurs.

The user, in the interrupt service routine, can clear the interrupt in the following manner:

a) Any read or write of the CMCON register. This will load the CMCON register with the new value with the CMxOUT bits.

b) Clear the CMIF flag bit.

An interrupt condition will continue to set the CMIF flag bit. Reading CMCON will end the interrupt condition, and allow the CMIF flag bit to be cleared.

20.9 Comparator Operation During SLEEP

When a comparator is active and the device is placed in SLEEP mode, the comparator remains active and the interrupt is functional if enabled. This interrupt will wake up the device from SLEEP mode when enabled. While the comparator is powered-up, each comparator that is operational will consume additional current as shown in the comparator specifications. To minimize power consumption while in SLEEP mode, turn off the comparators, CM2:CM0 = 111, before entering sleep. If the device wakes-up from sleep, the contents of the CMCON register are not affected.

20.10 Effects of a RESET

A device reset forces the CMCON register to its reset state. This forces the comparator module to be in the comparator reset mode, CM2:CM0 = 000. This ensures that all potential inputs are analog inputs. Device current is minimized when analog inputs are present at reset time. The comparators will be powered-down during the reset interval.

20.11 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 20-4. Since the analog pins are connected to a digital output, they have reverse biased diodes to VDD and VSS. The analog input therefore, must be between VSS and VDD. If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up may occur. A maximum source impedance of 10 kΩ is recommended for the analog sources.

Figure 20-4: Analog Input Model

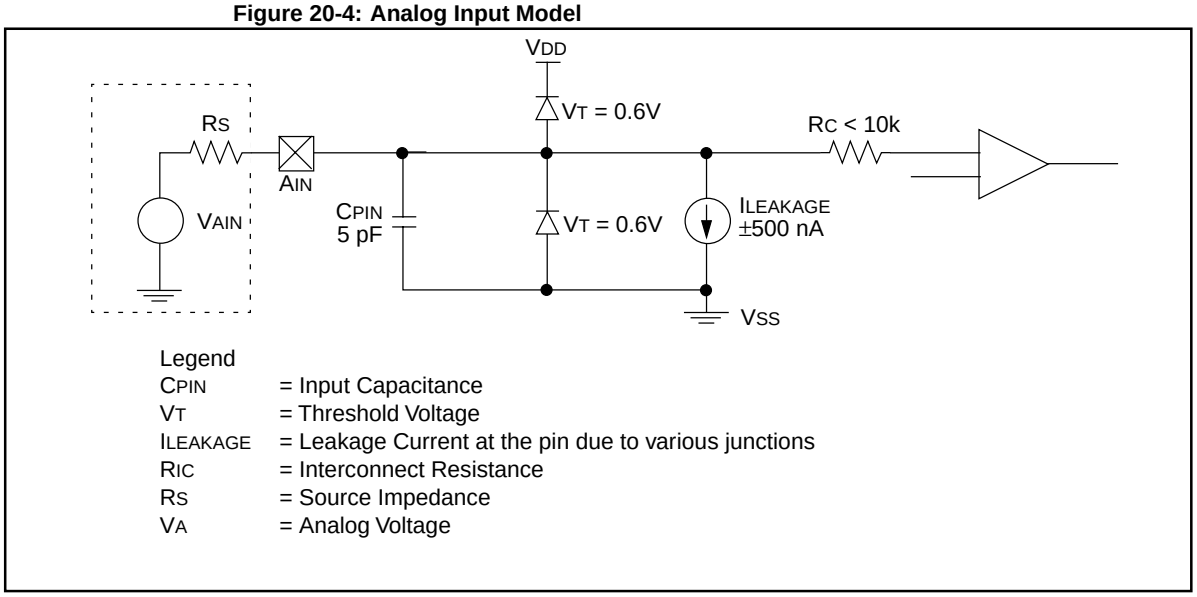


Table 20-1: Registers Associated with Comparator Module

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other Resets
CMCON	C2OUT	C1OUT	—	—	CIS	CM2	CM1	CM0	00-- 0000	00-- 0000
VRCON	VREN	VROE	VRR	—	VR3	VR2	VR1	VR0	000- 0000	000- 0000
INTCON	GIE	PEIE	T0IE	INTE	RBIE(2)	T0IF	INTF	RBIF(2)	0000 000x	0000 000x
PIR	CMIF (1)								0	0
PIE	CMIE (1)								0	0

Legend: x = unknown, - = unimplemented locations read as '0'.
Shaded cells are not used for Comparator Module.
Note 1: The position of this bit is device dependent.
2: These bits can also be named GPIE and GPIF.

20.12 Initialization

The code in Example 20-1 depicts example steps required to configure the comparator module of the PIC16C62X devices. RA3 and RA4 are configured as digital output. RA0 and RA1 are configured as the V- inputs and RA2 as the V+ input to both comparators.

Example 20-1: Initializing Comparator Module (PIC16C62X)

```
FLAG_REG   EQU    0X20
;
   CLRF    FLAG_REG      ; Init flag register
   CLRF    PORTA         ; Init PORTA
   ANDLW   0xC0          ; Mask comparator bits
   IORWF   FLAG_REG,F    ; Store bits in flag register
   MOVLW   0x03          ; Init comparator mode
   MOVWF   CMCON         ; CM<2:0> = 011
   BSF     STATUS,RP0    ; Select Bank1
   MOVLW   0x07          ; Initialize data direction
   MOVWF   TRISA         ; Set RA<2:0> as inputs, RA<4:3> as outputs,
                         ;    TRISA<7:5> always read '0'
   BCF     STATUS,RP0    ; Select Bank0
   CALL    DELAY 10      ; 10µs delay
   MOVF    CMCON,F       ; Read    CMCON to end change condition
   BCF     PIR1,CMIF     ; Clear pending interrupts
   BSF     STATUS,RP0    ; Select Bank1
   BSF     PIE1,CMIE     ; Enable comparator interrupts
   BCF     STATUS,RP0    ; Select Bank0
   BSF     INTCON,PEIE   ; Enable peripheral interrupts
   BSF     INTCON,GIE    ; Global interrupt enable
```

20.13 Design Tips

Question 1: ***My program appears to lock up.***

Answer 1:

You may be getting stuck in an infinite loop with the comparator interrupt service routine if you did not follow the proper sequence to clear the CMIF flag bit. First you must read the CMCON register, and then you can clear the CMIF flag bit.

20.14 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the comparator module are:

Title	Application Note #
Resistance and Capacitance Meter using a PIC16C622	AN611

20.15 Revision History

Revision A

This is the initial released revision of the Comparator module description.

Section 21. 8-bit A/D Converter

HIGHLIGHTS

This section of the manual contains the following major topics:

Note:	Please refer to Appendix C.3 or device Data Sheet to determine which devices use this module.

21.1 Introduction

The analog-to-digital (A/D) converter module has up to eight analog inputs.

The A/D allows conversion of an analog input signal to a corresponding 8-bit digital number. The output of the sample and hold is the input into the converter, which generates the result via successive approximation. The analog reference voltage is software selectable to either the device's positive supply voltage (V_{DD}) or the voltage level on the V_{REF} pin. The A/D converter has a unique feature of being able to operate while the device is in SLEEP mode.

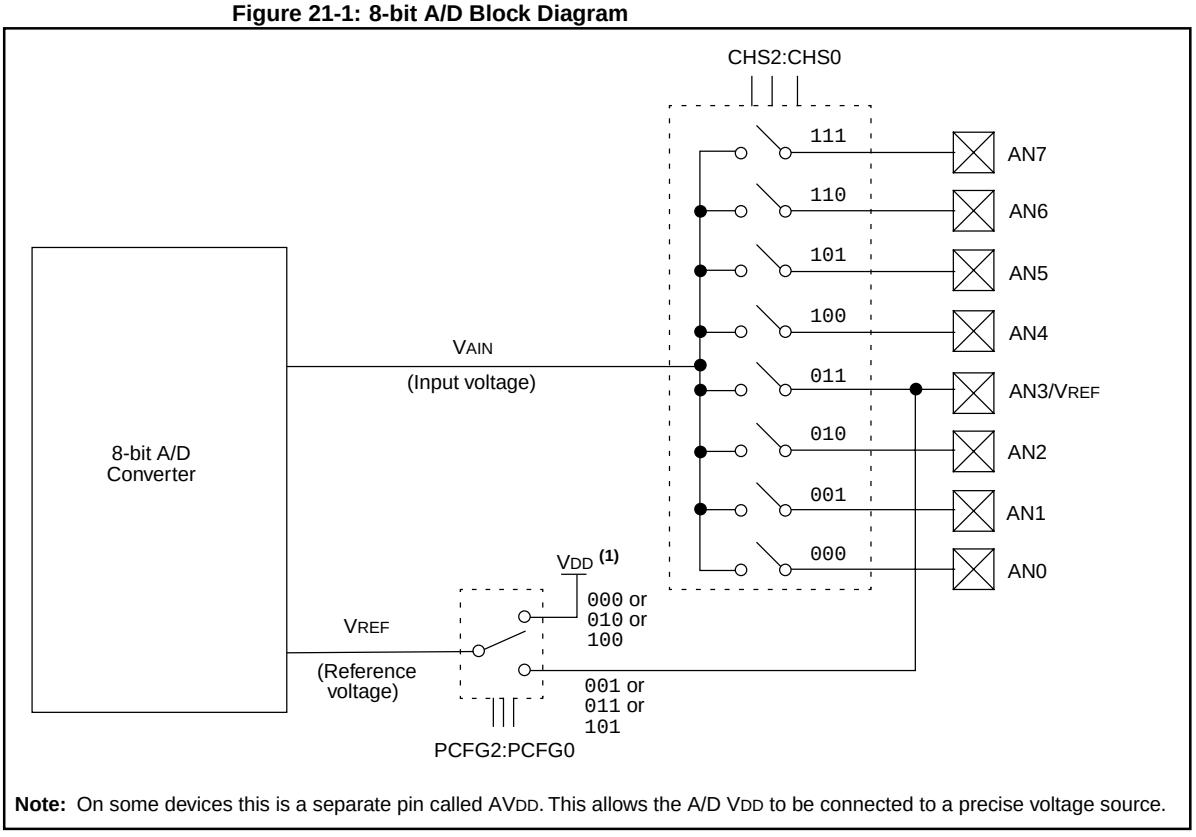
The A/D module has three registers. These registers are:

- A/D Result Register (ADRES)
- A/D Control Register0 (ADCON0)
- A/D Control Register1 (ADCON1)

The ADCON0 register, shown in Figure 21-1, controls the operation of the A/D module. The ADCON1 register, shown in Figure 21-2, configures the functions of the port pins. The I/O pins can be configured as analog inputs (one I/O can also be a voltage reference) or as digital I/O.

The block diagram of the A/D module is shown in Figure 21-1.

Figure 21-1: 8-bit A/D Block Diagram



Note: On some devices this is a separate pin called AV_{DD}. This allows the A/D V_{DD} to be connected to a precise voltage source.

21.2 Control Registers

Register 21-1: ADCON0 Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	Resv	ADON
bit 7							bit 0

bit 7:6 **ADCS1:ADCS0**: A/D Conversion Clock Select bits
00 = FOSC/2
01 = FOSC/8
10 = FOSC/32
11 = FRC (clock derived from the internal A/D RC oscillator)

bit 5:3 **CHS2:CHS0**: Analog Channel Select bits

000 = channel 0, (AN0)
001 = channel 1, (AN1)
010 = channel 2, (AN2)
011 = channel 3, (AN3)
100 = channel 4, (AN4)
101 = channel 5, (AN5)
110 = channel 6, (AN6)
111 = channel 7, (AN7)

Note: For devices that do not implement the full 8 A/D channels, the unimplemented selections are reserved. Do not select any unimplemented channels.

bit 2 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

When ADON = 1

1 = A/D conversion in progress
(Setting this bit starts the A/D conversion. This bit is automatically cleared by hardware when the A/D conversion is complete)

0 = A/D conversion not in progress

bit 1 **Reserved:** Always maintain this bit cleared.

bit 0 **ADON**: A/D On bit

1 = A/D converter module is operating
0 = A/D converter module is shutoff and consumes no operating current

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

Register 21-2: ADCON1 Register

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7:3 **Unimplemented**: Read as '0'

bit 2:0 **PCFG2:PCFG0**: A/D Port Configuration Control bits

PCFG2:PCFG0	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
000	A	A	A	A	A	A	A	A
001	A	A	A	A	VREF	A	A	A
010	D	D	D	A	A	A	A	A
011	D	D	A	A	VREF	A	A	A
100	D	D	D	D	A	D	A	A
101	D	D	D	D	VREF	D	A	A
11x	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O

Note: When AN3 is selected as VREF, the A/D reference is the voltage on the AN3 pin. When AN3 is selected as an analog input (A), then the voltage reference for the A/D is the device VDD.

Legend

R = Readable bit W = Writable bit

U = Unimplemented bit, read as '0' - n = Value at POR reset

Note 1: On any device reset, the Port pins multiplexed with analog functions (ANx) are forced to be an analog input.

21.3 Operation

When the A/D conversion is complete, the result is loaded into the ADRES register, the GO/$\overline{\text{DONE}}$ bit (ADCON0<2>) is cleared, and A/D interrupt flag bit, ADIF, is set.

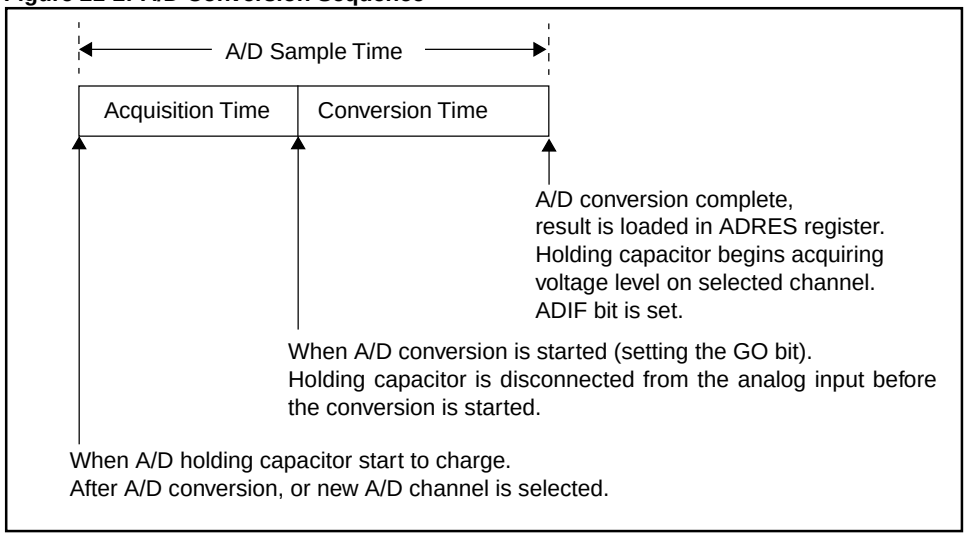
After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. To determine acquisition time, see Subsection **21.4 "A/D Acquisition Requirements."** After 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:
   - Configure analog pins / voltage reference / and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D conversion clock (ADCON0)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear the ADIF bit
   - Set the ADIE bit
   - Set the GIE bit
3. Wait the required acquisition time.
4. Start conversion:
   - Set the GO/$\overline{\text{DONE}}$ bit (ADCON0)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared

   OR
   - Waiting for the A/D interrupt
6. Read A/D Result register (ADRES), clear the ADIF bit, 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 21-2 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 10 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 21-2: A/D Conversion Sequence



21.4 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 21-3. 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 21-3). **The maximum recommended impedance for analog sources is 10 kΩ.** After the analog input channel is selected (changed) the acquisition must be done before the conversion can be started.

To calculate the minimum acquisition time, Equation 21-1 may be used. This equation assumes that 1/2 LSb error is used (512 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

Equation 21-1: Acquisition Time

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \\ &\quad \text{Holding Capacitor Charging Time} + \\ &\quad \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \end{aligned}$$

Equation 21-2: A/D Minimum Charging Time

$$V_{HOLD} = (V_{REF} - (V_{REF}/512)) \cdot (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$

or

$$T_C = -(51.2\ pF)(1\ k\Omega + R_{SS} + R_S) \ln(1/511)$$

Example 21-1 shows the calculation of the minimum required acquisition time TACQ. This calculation is based on the following system assumptions.

Rs	=	10 kΩ	
Conversion Error	≤	1/2 LSb	
VDD	=	5V → Rss = 7 kΩ	(see graph in Figure 21-3)
Temperature	=	50°C (system max.)	
VHOLD	=	0V @ time = 0	

Example 21-1: Calculating the Minimum Required Acquisition Time

$$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$$

$$T_{ACQ} = 5\ \mu s + T_C + [(Temp - 25°C)(0.05\ \mu s/°C)]$$

$$\begin{aligned} T_C &= -C_{HOLD}(R_{IC} + R_{SS} + R_S) \ln(1/512) \\ &= -51.2\ pF\ (1\ k\Omega + 7\ k\Omega + 10\ k\Omega) \ln(0.0020) \\ &= -51.2\ pF\ (18\ k\Omega) \ln(0.0020) \\ &= -0.921\ \mu s\ (-6.2146) \\ &= 5.724\ \mu s \end{aligned}$$

$$\begin{aligned} T_{ACQ} &= 5\ \mu s + 5.724\ \mu s + [(50°C - 25°C)(0.05\ \mu s/°C)] \\ &= 10.724\ \mu s + 1.25\ \mu s \\ &= 11.974\ \mu s \end{aligned}$$

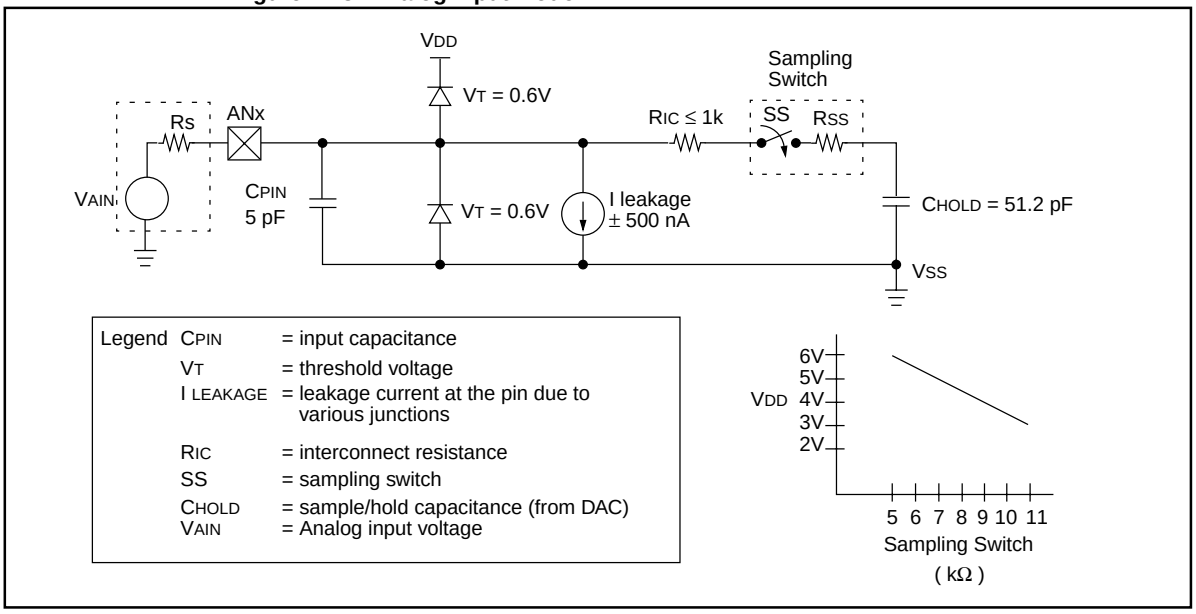
Note 1: The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

Note 2: The charge holding capacitor (CHOLD) is not discharged after each conversion.

Note 3: The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

Note 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.

Figure 21-3: Analog Input Model



21.5 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 9.5 TAD per 8-bit conversion. The source of the A/D conversion clock is software selected. The four possible options for TAD are:

- 2TOSC
- 8TOSC
- 32TOSC
- 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 for all devices, as shown in parameter 130 of the devices electrical specifications.

Table 21-1 and Table 21-2 show the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

Table 21-1: TAD vs. Device Operating Frequencies (for Standard, C, Devices)

AD Clock Source (TAD)		Device Frequency			
Operation	**ADCS1:ADCS0**	**20 MHz**	**5 MHz**	**1.25 MHz**	**333.33 kHz**
2TOSC	00	100 ns(2)	400 ns(2)	1.6 µs	6 µs
8TOSC	01	400 ns(2)	1.6 µs	6.4 µs	24 µs(3)
32TOSC	10	1.6 µs	6.4 µs	25.6 µs(3)	96 µs(3)
RC	11	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1)

Legend: Shaded cells are outside of recommended range.
Note 1: The RC source has a typical TAD time of 4 µs.
2: These values violate the minimum required TAD time.
3: For faster conversion times, the selection of another clock source is recommended.
4: For device frequencies above 1 MHz, the device must be in SLEEP for the entire conversion, or the A/D accuracy may be out of specification.

Table 21-2: TAD vs. Device Operating Frequencies (for Extended, LC, Devices)

AD Clock Source (TAD)		Device Frequency			
Operation	**ADCS1:ADCS0**	**4 MHz**	**2 MHz**	**1.25 MHz**	**333.33 kHz**
2TOSC	00	500 ns(2)	1.0 µs(2)	1.6 µs(2)	6 µs
8TOSC	01	2.0 µs(2)	4.0 µs	6.4 µs	24 µs(3)
32TOSC	10	8.0 µs	16.0 µs	25.6 µs(3)	96 µs(3)
RC	11	3 - 9 µs(1,4)	3 - 9 µs(1,4)	3 - 9 µs(1,4)	3 - 9 µs(1)

Legend: Shaded cells are outside of recommended range.
Note 1: The RC source has a typical TAD time of 6 µs.
2: These values violate the minimum required TAD time.
3: For faster conversion times, the selection of another clock source is recommended.
4: For device frequencies above 1 MHz, the device must be in SLEEP for the entire conversion, or the A/D accuracy may be out of specification.

21.6 Configuring Analog Port Pins

ADCON1 and the corresponding TRIS registers control the operation of the A/D port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bits set (input). If the TRIS 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 TRIS bits.

Note 1: When reading the port register, all pins configured as analog input channels 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.

Note 2: Analog levels on any pin that is defined as a digital input (including the AN7:AN0 pins), may cause the input buffer to consume current that is out of the devices specification.

21.7 A/D Conversions

Example 21-2 show how to perform an A/D conversion. The I/O pins are configured as analog inputs. The analog reference (VREF) is the device VDD. The A/D interrupt is enabled, and the A/D conversion clock is FRC. The conversion is performed on the AN0 channel.

Note:	The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D, due to the required acquition time requirement.

Clearing the GO/$\overline{\text{DONE}}$ bit during a conversion will abort the current conversion. The ADRES register will NOT be updated with the partially completed A/D conversion sample. That is, the ADRES register will continue to contain the value of the last completed conversion (or the last value written to the ADRES register). After the A/D conversion is aborted, a 2TAD wait is required before the next acquisition is started. After this 2TAD wait, an acquisition is automatically started on the selected channel.

Example 21-2: Doing an A/D Conversion

```
    BSF     STATUS, RP0      ; Select Bank1
    CLRF    ADCON1           ; Configure A/D inputs
    BSF     PIE1,   ADIE     ; Enable A/D interrupts
    BCF     STATUS, RP0      ; Select Bank0
    MOVLW   0xC1             ; RC Clock, A/D is on, Channel 0 is selected
    MOVWF   ADCON0           ;
    BCF     PIR1,   ADIF     ; Clear A/D interrupt flag bit
    BSF     INTCON, PEIE     ; Enable peripheral interrupts
    BSF     INTCON, GIE      ; Enable all interrupts
;
;  Ensure that the required sampling time for the selected input
;  channel has elapsed. Then the conversion may be started.
;
    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 21-4: A/D Conversion TAD Cycles

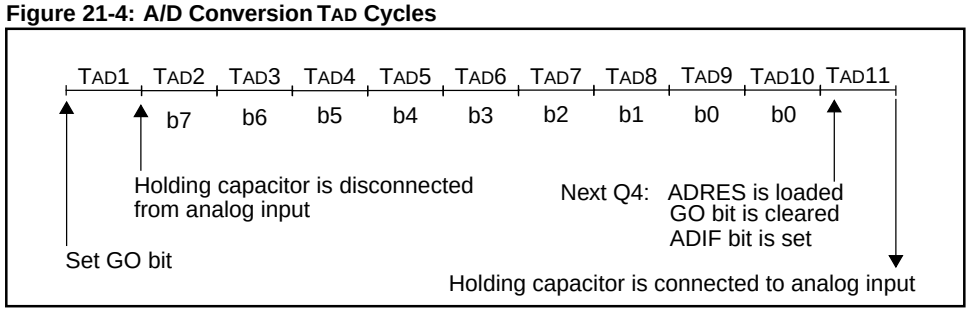
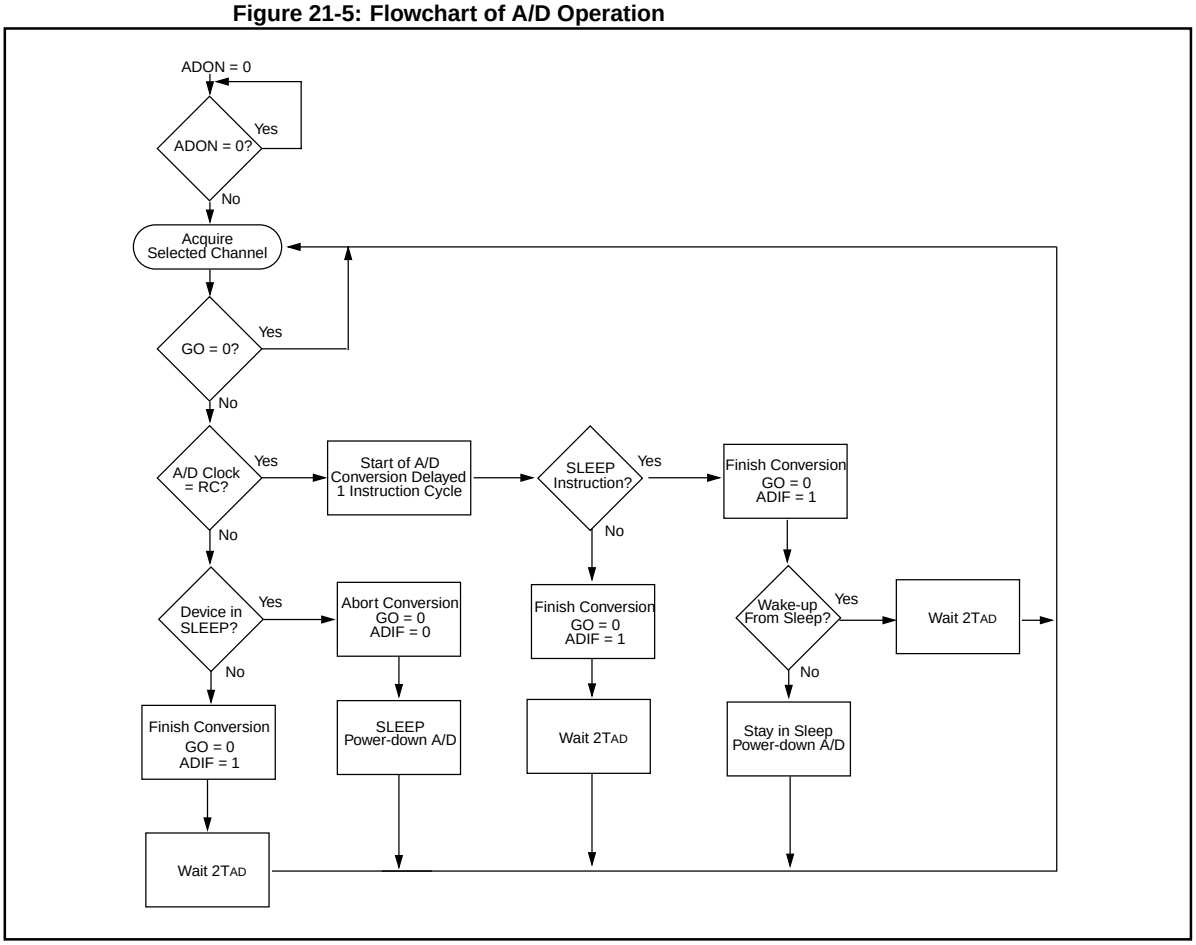


Figure 21-5: Flowchart of A/D Operation



21.7.1 Faster Conversion - Lower Resolution Trade-off

Not all applications require a result with 8-bits of resolution, but may instead require a faster conversion time. The A/D module allows users to make the trade-off of conversion speed to resolution. Regardless of the resolution required, the acquisition time is the same. To speed up the conversion, the clock source of the A/D module may be switched so that the TAD time violates the minimum specified time (see the applicable electrical specification). Once the TAD time violates the minimum specified time, all the following A/D result bits are not valid (see A/D Conversion Timing in the Electrical Specifications section). The clock sources may only be switched between the three oscillator versions (cannot be switched from/to RC). The equation to determine the time before the oscillator can be switched is as follows:

Conversion time = TAD + N • TAD + (10 - N)(2TOSC)
Where: N = number of bits of resolution required.

Since the TAD is based from the device oscillator, the user must use some method (a timer, software loop, etc.) to determine when the A/D oscillator may be changed. Example 21-3 shows a comparison of time required for a conversion with 4-bits of resolution, versus the 8-bit resolution conversion. The example is for devices operating at 20 MHz (The A/D clock is programmed for 32TOSC), and assumes that immediately after 5TAD, the A/D clock is programmed for 2TOSC.

The 2TOSC violates the minimum TAD time since the last 4-bits will not be converted to correct values.

Example 21-3: 4-bit vs. 8-bit Conversion Times

	Freq. (MHz)(1)	Resolution	
		4-bit	8-bit
TAD	20	1.6 µs	1.6 µs
TOSC	20	50 ns	50 ns
TAD + N • TAD + (10 - N)(2TOSC)	20	8.6 µs	17.6 µs

Note 1: A minimum TAD time of 1.6 µs is required.
2: If the full 8-bit conversion is required, the A/D clock source should not be changed.

21.8 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 internal 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 (to conserve power), 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 perform an A/D conversion in SLEEP, the GO/$\overline{\text{DONE}}$ bit must be set, followed by the SLEEP instruction.

21.9 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 VDD diverges from VREF.

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.

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 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.

21.10 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 ADRES register is not modified for a Power-on Reset. The ADRES register will contain unknown data after a Power-on Reset.

21.11 Use of the CCP Trigger

An A/D conversion may be started by the “special event trigger” of a CCP module. This requires that the CCPxM3:CCPxM0 bits (CCPxCON<3:0>) be programmed as 1011 and that the A/D module is enabled (ADON bit is set). When the trigger occurs, the GO/$\overline{\text{DONE}}$ bit will be set, starting the A/D conversion, and the Timer1 counter will be reset to zero. Timer1 is reset to automatically repeat the A/D acquisition period with minimal software overhead (moving the ADRES to the desired location). The appropriate analog input channel must be selected and the minimum acquisition done before the “special event trigger” sets the GO/$\overline{\text{DONE}}$ bit (starts a conversion).

If the A/D module is not enabled (ADON is cleared), then the “special event trigger” will be ignored by the A/D module, but will still reset the Timer1 counter.

21.12 Connection Considerations

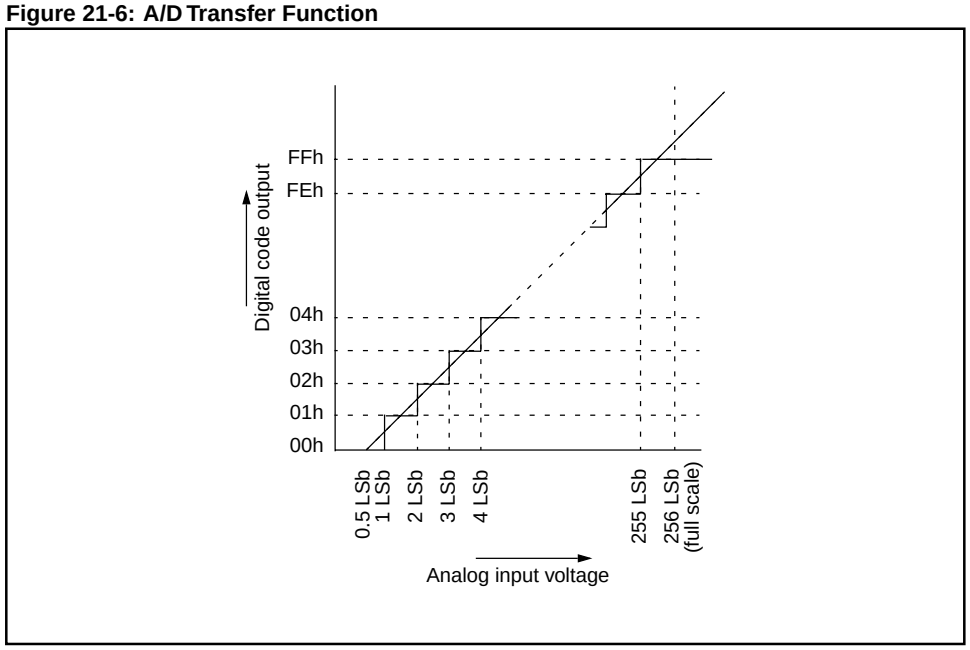
If the input voltage exceeds the rail values (V_{SS} or V_{DD}) by greater than 0.3V, then the accuracy of the conversion is out of specification.

An external RC filter can sometimes be 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.

21.13 Transfer Function

The ideal transfer function of the A/D converter is as follows: the first transition occurs when the analog input voltage (V_{AIN}) is 1 LSb (or Analog V_{REF} / 256) (Figure 21-6).

Figure 21-6: A/D Transfer Function



21.14 Initialization

Example 21-4 shows the initialization of the A/D module for the PIC16C74A

Example 21-4: A/D Initialization (for PIC16C74A)

```
    BSF     STATUS, RP0      ; Select Bank1
    CLRF    ADCON1           ; Configure A/D inputs
    BSF     PIE1, ADIE       ; Enable A/D interrupts
    BCF     STATUS, RP0      ; Select Bank0
    MOVLW   0xC1             ; RC Clock, A/D is on, Channel 0 is selected
    MOVWF   ADCON0           ;
    BCF     PIR1, ADIF       ; Clear A/D interrupt flag bit
    BSF     INTCON, PEIE     ; Enable peripheral interrupts
    BSF     INTCON, GIE      ; Enable all interrupts
;
;  Ensure that the required sampling time for the selected input
;  channel has elapsed. Then the conversion may be started.
;
    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.
```

21.15 Design Tips

Question 1: ***I am using one of your PIC16C7X devices, and I find that the Analog to Digital Converter result is not always accurate. What can I do to improve accuracy?***

Answer 1:

1. Make sure you are meeting all of the timing specifications. If you are turning the A/D module off and on, there is a minimum delay you must wait before taking a sample, if you are changing input channels, there is a minimum delay you must wait for this as well, and finally there is Tad, which is the time selected for each bit conversion. This is selected in ADCON0 and should be between 2 and 6μs. If TAD is too short, the result may not be fully converted before the conversion is terminated, and if TAD is made too long the voltage on the sampling capacitor can droop before the conversion is complete. These timing specifications are provided in the data book in a table or by way of a formula, and should be looked up for your specific part and circumstances.
2. Often the source impedance of the analog signal is high (greater than 1k ohms) so the current drawn from the source to charge the sample capacitor can affect accuracy. If the input signal does not change too quickly, try putting a 0.1 μF capacitor on the analog input. This capacitor will charge to the analog voltage being sampled, and supply the instantaneous current needed to charge the 51.2 pf internal holding capacitor.
3. Finally, straight from the data book: “In systems where the device frequency is low, use of the A/D clock derived from the device oscillator is preferred...this reduces, to a large extent, the effects of digital switching noise.” and “In systems where the device will enter SLEEP mode after start of A/D conversion, the RC clock source selection is required. This method gives the highest accuracy.”

Question 2: ***After starting an A/D conversion may I change the input channel (for my next conversion)?***

Answer 2:

After the holding capacitor is disconnected from the input channel, one TAD after the GO bit is set, the input channel may be changed.

Question 3: ***Do you know of a good reference on A/D’s?***

Answer 3:

A very good reference for understanding A/D conversions is the “Analog-Digital Conversion Handbook” third edition, published by Prentice Hall (ISBN 0-13-03-2848-0).

21.16 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the 8-bit A/D are:

Title	Application Note #
Using the Analog to Digital Converter	AN546
Four Channel Digital Voltmeter with Display and Keyboard	AN557

21.17 Revision History

Revision A

This is the initial released revision of the 8-bit A/D module description.

Section 22. Basic 8-bit A/D Converter

HIGHLIGHTS

This section of the manual contains the following major topics:

Note: Please refer to Appendix C.2 or the device Data Sheet to determine which devices use this module.

22.1 Introduction

This Analog-to-Digital (A/D) converter module has four analog inputs.

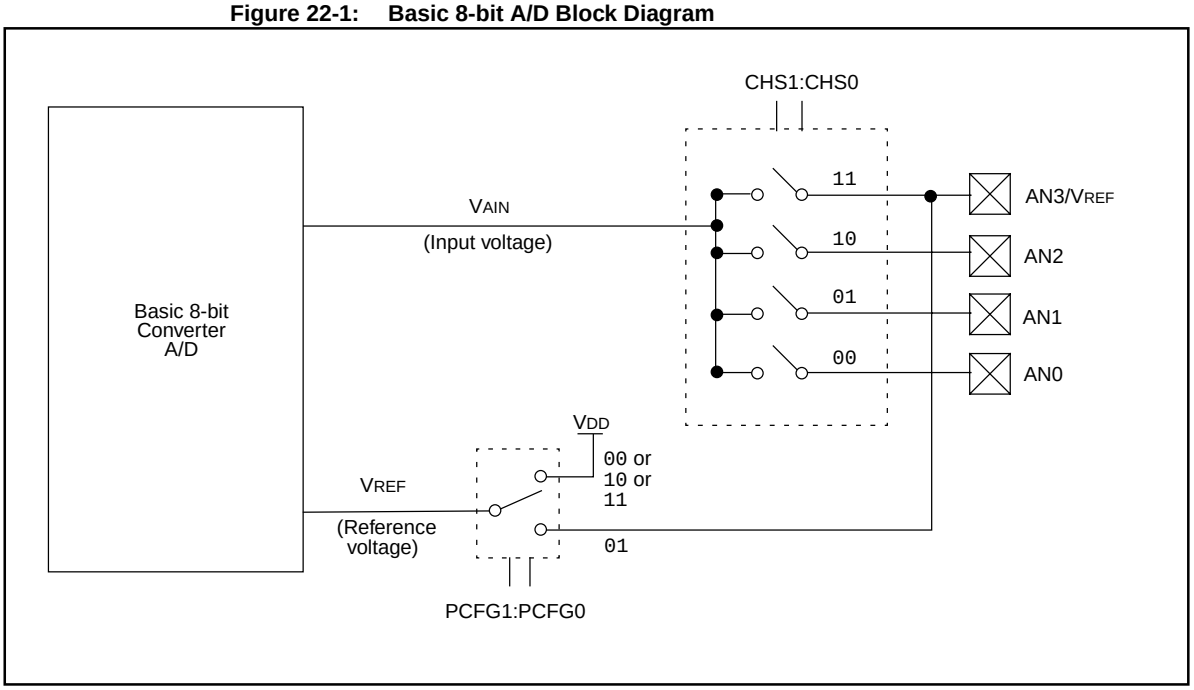
The A/D allows conversion of an analog input signal to a corresponding 8-bit digital number. The output of the sample and hold is the input into the converter, which generates the result via successive approximation. The analog reference voltage is software selectable to either the device's positive supply voltage (V_{DD}) or the voltage level on the AN3/V_{REF} pin. The A/D converter has a unique feature of being able to operate while the device is in SLEEP mode.

The A/D module has three registers. These registers are:

- A/D Result Register (ADRES)
- A/D Control Register0 (ADCON0)
- A/D Control Register1 (ADCON1)

The ADCON0 register, shown in Figure 22-1 controls the operation of the A/D module. The ADCON1 register, shown in Figure 22-2, configures the functions of the port pins. The port pins can be configured as analog inputs (or a voltage reference) or as digital I/O.

Figure 22-1: Basic 8-bit A/D Block Diagram



22.2 Control Registers

Register 22-1: ADCON0 Register

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADCS1	ADCS0	— (1)	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADIF / — (2)	ADON
bit 7							bit 0

bit 7:6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits
00 = FOSC/2
01 = FOSC/8
10 = FOSC/32
11 = FRC (clock derived from the internal A/D RC oscillator)

bit 5 **Unimplemented**: Read as '0'.

bit 4:3 **CHS1:CHS0**: Analog Channel Select bits
00 = channel 0, (AN0)
01 = channel 1, (AN1)
10 = channel 2, (AN2)
11 = channel 3, (AN3)

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)
0 = A/D conversion not in progress (This bit is automatically cleared by hardware when the A/D conversion is complete)

bit 1 **ADIF (2):** A/D Conversion Complete Interrupt Flag bit
1 = conversion is complete (must be cleared in software)
0 = conversion is not complete

bit 0 **ADON:** A/D On bit
1 = A/D converter module is operating
0 = A/D converter module is shutoff and consumes no operating current

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

Note 1: For the PIC16C71, Bit5 of ADCON0 is a General Purpose R/W bit. For the PIC16C710/711/715, this bit is unimplemented, read as '0'.

Note 2: For the PIC12CXXX devices, this bit is reserved. The ADIF bit is implemented in the PIR register. Use of this bit a a general purpose R/W bit is not recommended. Always maintain this bit cleared.

Register 22-2: ADCON1 Register

U-0	U-0	U-0	U-0	U-0	U-0 / R/W-0	R/W-0	R/W-0
—	—	—	—	—	— / PCFG2 (1)	PCFG1	PCFG0
bit 7							bit 0

bit 7:2 **Unimplemented:** Read as '0'

Note: Some devices implement bit2 as the PCFG2 bit.

bit 1:0 **PCFG1:PCFG0**: A/D Port Configuration Control bits

PCFG1:PCFG0	AN3	AN2	AN1	AN0
00	A	A	A	A
01	VREF+	A	A	A
10	D	D	A	A
11	D	D	D	D

A = Analog input

D = Digital I/O

Note: When AN3 is selected as VREF+, the A/D reference is the voltage on the AN3 pin. When AN3 is selected as an analog input (A), then the voltage reference for the A/D is the device VDD.

bit 2:0 **PCFG2:PCFG0**: A/D Port Configuration Control bits (1)

PCFG2:PCFG0	AN3	AN2	AN1	AN0
000	A	A	A	A
001	A	A	VREF+	A
010	D	A	A	A
011	D	A	VREF+	A
100	D	D	A	A
101	D	D	VREF+	A
110	D	D	D	A
111	D	D	D	D

A = Analog input

D = Digital I/O

Note: When AN1 is selected as VREF+, the A/D reference is the voltage on the AN1 pin. When AN1 is selected as an analog input (A), then the voltage reference for the A/D is the device VDD.

Legend

R = Readable bit W = Writable bit

U = Unimplemented bit, read as '0' - n = Value at POR reset

Note 1: Some devices add an additional Port configuration bit (PCFG2). This allows the minimum number of analog channels to be one. This is of most benefit to the 8-pin devices with the A/D converter, since in an 8-pin device I/O is a premium resource. In the other devices this bit is unimplemented, and read as '0'.

Note 2: On any device reset, the Port pins multiplexed with analog functions (ANx) are forced to be an analog input.

The ADRES register contains the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRES register, the GO/$\overline{\text{DONE}}$ bit (ADCON0<2>) is cleared, and A/D interrupt flag bit ADIF is set. The block diagram of the A/D module is shown in Figure 22-1.

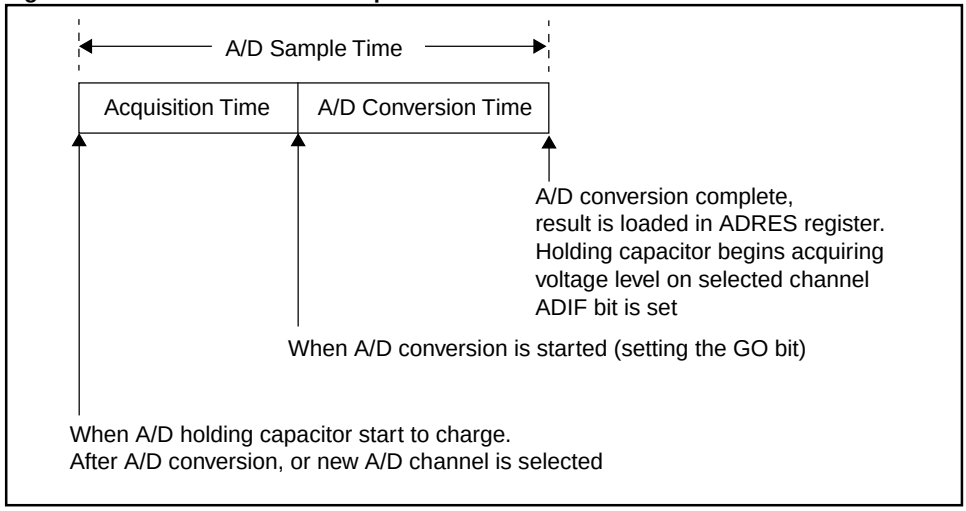
After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. To determine sample time, see Subsection **22.3 "A/D Acquisition Requirements"** After this 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:
   - Configure analog pins / voltage reference / and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D conversion clock (ADCON0)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear the ADIF bit
   - Set the ADIE bit
   - Set the GIE bit
3. Wait the required acquisition time.
4. Start conversion:
   - Set the GO/$\overline{\text{DONE}}$ bit (ADCON0)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared

   OR
   - Waiting for the A/D interrupt
6. Read A/D Result register (ADRES), clear the ADIF bit, 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 22-2 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 10 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 22-2: A/D Conversion Sequence



22.3 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 22-3. 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), see Figure 22-3. **The maximum recommended impedance for analog sources is 10 kΩ.** 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 22-1 may be used. This equation assumes that 1/2 LSb error is used (512 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

Equation 22-1:Acquisition Time

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \\ &\quad \text{Holding Capacitor Charging Time} + \\ &\quad \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \end{aligned}$$

Equation 22-2:A/D Minimum Charging Time

$$\begin{aligned} V_{HOLD} &= (V_{REF} - (V_{REF}/512)) \bullet (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))}) \\ \text{or} & \\ T_C &= -(51.2\ pF)(1\ k\Omega + R_{SS} + R_S) \ln(1/511) \end{aligned}$$

Example 22-1 shows the calculation of the minimum required acquisition time TACQ. This calculation is based on the following system assumptions.

Rs	=	10 kΩ	
Conversion Error	≤	1/2 LSb	
VDD	=	5V → Rss = 7 kΩ	(see graph in Figure 22-3)
Temperature	=	50°C (system max.)	
VHOLD	=	0V @ time = 0	

Example 22-1: Calculating the Minimum Required Acquisition Time

$$\begin{aligned} T_{ACQ} &= T_{AMP} + T_C + T_{COFF} \\ T_{ACQ} &= 5\ \mu s + T_C + [(Temp - 25°C)(0.05\ \mu s/°C)] \\ T_C &= -C_{HOLD}(R_{IC} + R_{SS} + R_S) \ln(1/512) \\ &= -51.2\ pF\ (1\ k\Omega + 7\ k\Omega + 10\ k\Omega) \ln(0.0020) \\ &= -51.2\ pF\ (18\ k\Omega) \ln(0.0020) \\ &= -0.921\ \mu s\ (-6.2146) \\ &= 5.724\ \mu s \\ T_{ACQ} &= 5\ \mu s + 5.724\ \mu s + [(50°C - 25°C)(0.05\ \mu s/°C)] \\ &= 10.724\ \mu s + 1.25\ \mu s \\ &= 11.974\ \mu s \end{aligned}$$

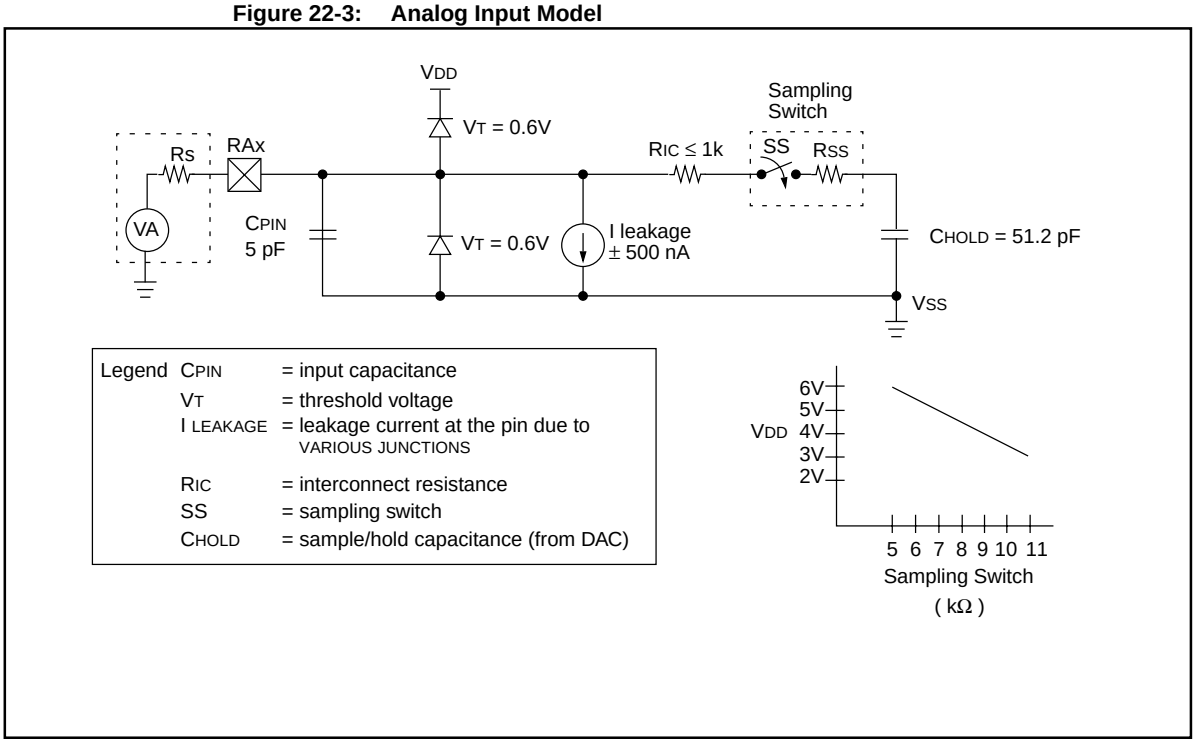
Note 1: The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

Note 2: The charge holding capacitor (CHOLD) is not discharged after each conversion.

Note 3: The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

Note 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.

Figure 22-3: Analog Input Model



22.4 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 9.5 TAD per 8-bit conversion. The source of the A/D conversion clock is software selected. The four possible options for TAD are:

- 2TOSC
- 8TOSC
- 32TOSC
- Internal RC oscillator

For correct A/D conversions, the A/D conversion clock (TAD) must be selected to ensure a minimum TAD time of:

2.0 µs for the PIC16C71, as shown in parameter 130 of devices electrical specifications.

1.6 µs for all other devices, as shown in parameter 130 of devices electrical specifications.

Table 22-1 through Table 22-4 show the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

Table 22-1: TAD vs. Device Operating Frequencies, All Devices (except PIC16C71) (C Devices)

AD Clock Source (TAD)		Device Frequency			
Operation	**ADCS1:ADCS0**	**20 MHz**	**5 MHz**	**1.25 MHz**	**333.33 kHz**
2TOSC	00	100 ns(2)	400 ns(2)	1.6 µs	6 µs
8TOSC	01	400 ns(2)	1.6 µs	6.4 µs	24 µs(3)
32TOSC	10	1.6 µs	6.4 µs	25.6 µs(3)	96 µs(3)
RC(5)	11	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1)

Note 1: The RC source has a typical TAD time of 4 µs.
2: These values violate the minimum required TAD time.
3: For faster conversion times, the selection of another clock source is recommended.
4: For device frequencies above 1 MHz, the device must be in SLEEP for the entire conversion, or the A/D accuracy may be out of specification.

Table 22-2: TAD vs. Device Operating Frequencies, All Devices (except PIC16LC71) (LC Devices)

AD Clock Source (TAD)		Device Frequency			
Operation	**ADCS1:ADCS0**	**4 MHz**	**2 MHz**	**1.25 MHz**	**333.33 kHz**
2TOSC	00	500 ns(2)	1.0 µs(2)	1.6 µs(2)	6 µs
8TOSC	01	2.0 µs(2)	4.0 µs	6.4 µs	24 µs(3)
32TOSC	10	8.0 µs	16.0 µs	25.6 µs(3)	96 µs(3)
RC(5)	11	3 - 9 µs(1,4)	3 - 9 µs(1,4)	3 - 9 µs(1,4)	3 - 9 µs(1)

Note 1: The RC source has a typical TAD time of 6 µs.
2: These values violate the minimum required TAD time.
3: For faster conversion times, the selection of another clock source is recommended.
4: For device frequencies above 1 MHz, the device must be in SLEEP for the entire conversion, or the A/D accuracy may be out of specification.

Table 22-3: TAD vs. Device Operating Frequencies, PIC16C71 (C Devices)

AD Clock Source (TAD)		Device Frequency				
Operation	**ADCS1:ADCS0**	**20 MHz**	**16 MHz**	**4 MHz**	**1 MHz**	**333.33 kHz**
2TOSC	00	100 ns(2)	125 ns(2)	500 ns(2)	2.0 µs	6 µs
8TOSC	01	400 ns(2)	500 ns(2)	2.0 µs	8.0 µs	24 µs(3)
32TOSC	10	1.6 µs(2)	2.0 µs	8.0 µs	32.0 µs(3)	96 µs(3)
RC	11	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1)	2 - 6 µs(1)

Legend: Shaded cells are outside of recommended range.
Note 1: The RC source has a typical TAD time of 4 µs.
2: These values violate the minimum required TAD time.
3: For faster conversion times, the selection of another clock source is recommended.
4: For device frequencies above 1 MHz, the device must be in SLEEP for the entire conversion, or the A/D accuracy may be out of specification.

Table 22-4: TAD vs. Device Operating Frequencies, PIC16LC71 (LC Devices)

AD Clock Source (TAD)		Device Frequency			
Operation	**ADCS1:ADCS0**	**4 MHz**	**2 MHz**	**1.25 MHz**	**333.33 kHz**
2TOSC	00	500 ns(2)	1.0 µs(2)	1.6 µs(2)	6 µs
8TOSC	01	2.0 µs(2)	4.0 µs	6.4 µs	24 µs(3)
32TOSC	10	8.0 µs	16.0 µs	25.6 µs(3)	96 µs(3)
RC	11	3 - 9 µs(1,4)	3 - 9 µs(1,4)	3 - 9 µs(1,4)	3 - 9 µs(1)

Legend: Shaded cells are outside of recommended range.
Note 1: The RC source has a typical TAD time of 6 µs.
2: These values violate the minimum required TAD time.
3: For faster conversion times, the selection of another clock source is recommended.
4: For device frequencies above 1 MHz, the device must be in SLEEP for the entire conversion, or the A/D accuracy may be out of specification.

22.5 Configuring Analog Port Pins

The ADCON1 and TRISA registers control the operation of the A/D port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bits set (input). If the TRIS 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 CHS1:CHS0 bits and the TRIS bits.

Note 1: When reading the port register, all pins configured as 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.

Note 2: Analog levels on any pin that is defined as a digital input (including the AN3:AN0 pins), may cause the input buffer to consume current that is out of the devices specification.

22.6 A/D Conversions

Example 22-2 show how to perform an A/D conversion. The RA pins are configured as analog inputs. The analog reference (VREF) is the device VDD. The A/D interrupt is enabled, and the A/D conversion clock is FRC. The conversion is performed on the RA0 channel.

Note:	The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D, due to the required acquisition time.

Clearing the GO/$\overline{\text{DONE}}$ bit during a conversion will abort the current conversion. The ADRES register will NOT be updated with the partially completed A/D conversion sample. That is, the ADRES register will continue to contain the value of the last completed conversion (or the last value written to the ADRES register). After the A/D conversion is aborted, a 2TAD wait is required before the next acquisition is started. After this 2TAD wait, an acquisition is automatically started on the selected channel.

Example 22-2: Doing an A/D Conversion

```
     BSF     STATUS, RP0    ; Select Bank1
     CLRF    ADCON1         ; Configure A/D inputs
     BCF     STATUS, RP0    ; Select Bank0
     MOVLW   0xC1           ; RC Clock, A/D is on, Channel 0 selected
     MOVWF   ADCON0         ;
     BSF     INTCON, ADIE   ; Enable A/D Interrupt
     BSF     INTCON, GIE    ; Enable all interrupts
;
;  Ensure that the required sampling time for the selected input
;  channel has elapsed. Then the conversion may be started.
;
     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 22-4: A/D Conversion TAD Cycles

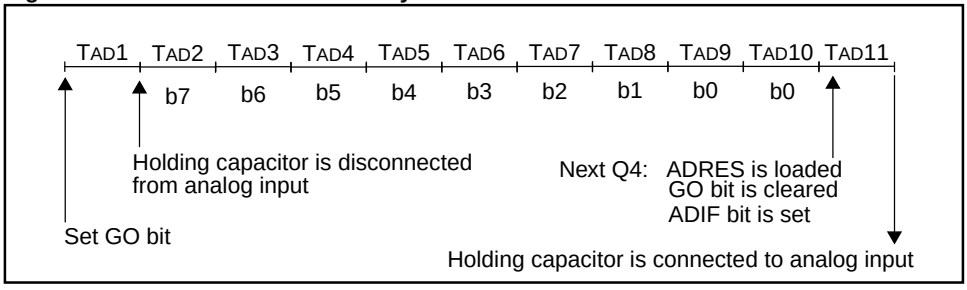
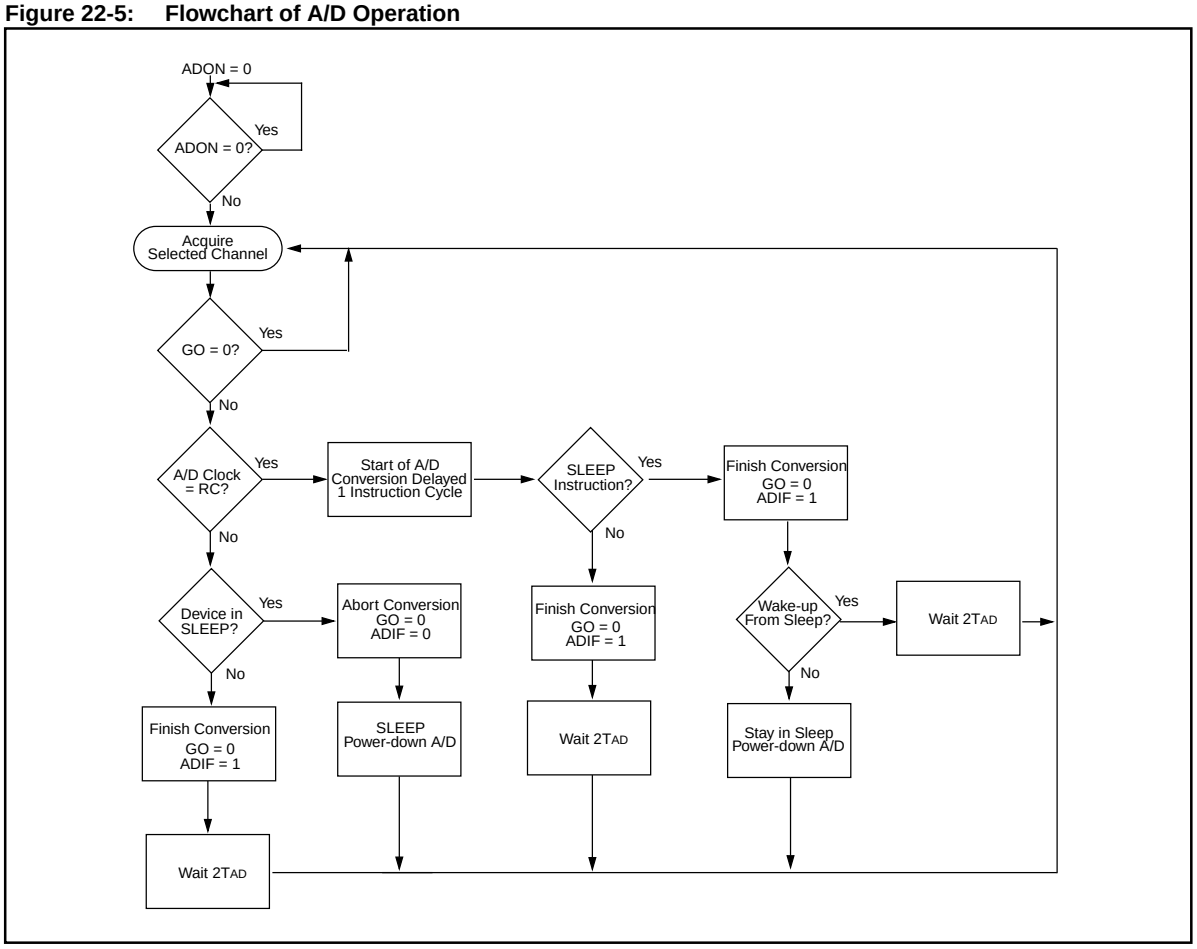


Figure 22-5: Flowchart of A/D Operation



22.6.1 Faster Conversion - Lower Resolution Trade-off

Not all applications require a result with 8-bits of resolution, but may instead require a faster conversion time. The A/D module allows users to make the trade-off of conversion speed to resolution. Regardless of the resolution required, the acquisition time is the same. To speed up the conversion, the clock source of the A/D module may be switched so that the TAD time violates the minimum specified time (see the applicable electrical specification). Once the TAD time violates the minimum specified time, all the following A/D result bits are not valid (see A/D Conversion Timing in the Electrical Specifications section.) The clock sources may only be switched between the three oscillator versions (cannot be switched from/to RC). The equation to determine the time before the oscillator can be switched is as follows:

Conversion time = TAD + N • TAD + (10 - N)(2TOSC)
Where: N = number of bits of resolution required.

Since the TAD is based from the device oscillator, the user must use some method (a timer, software loop, etc.) to determine when the A/D oscillator may be changed. Example 22-3 shows a comparison of time required for a conversion with 4-bits of resolution, versus the 8-bit resolution conversion. The example is for devices operating at 20 MHz and 16 MHz (The A/D clock is programmed for 32TOSC), and assumes that immediately after 5TAD, the A/D clock is programmed for 2TOSC.

The 2TOSC violates the minimum TAD time since the last 4-bits will not be converted to correct values.

Example 22-3: 4-bit vs. 8-bit Conversion Times

	Freq. (MHz)(1)	Resolution	
		4-bit	8-bit
TAD	20	1.6 µs	1.6 µs
	16	2.0 µs	2.0 µs
TOSC	20	50 ns	50 ns
	16	62.5 ns	62.5 ns
TAD + N • TAD + (10 - N)(2TOSC)	20	8.6 µs	17.6 µs
	16	10.75 µs	22 µs

Note 1: The PIC16C71 has a minimum TAD time of 2.0 µs.
All other devices have a minimum TAD time of 1.6 µs.
2: If the full 8-bit conversion is required, the A/D clock source should not be changed.

22.7 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 internal 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 perform an A/D conversion in SLEEP, the GO/$\overline{\text{DONE}}$ bit must be set, followed by the SLEEP instruction.

22.8 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 VDD diverges from VREF.

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.

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 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.

22.9 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 ADRES register is not modified for a Power-on Reset. The ADRES register will contain unknown data after a Power-on Reset.

22.10 Connection Considerations

If the input voltage exceeds the rail values (V_{SS} or V_{DD}) by greater than 0.2V, then the accuracy of the conversion is out of specification.

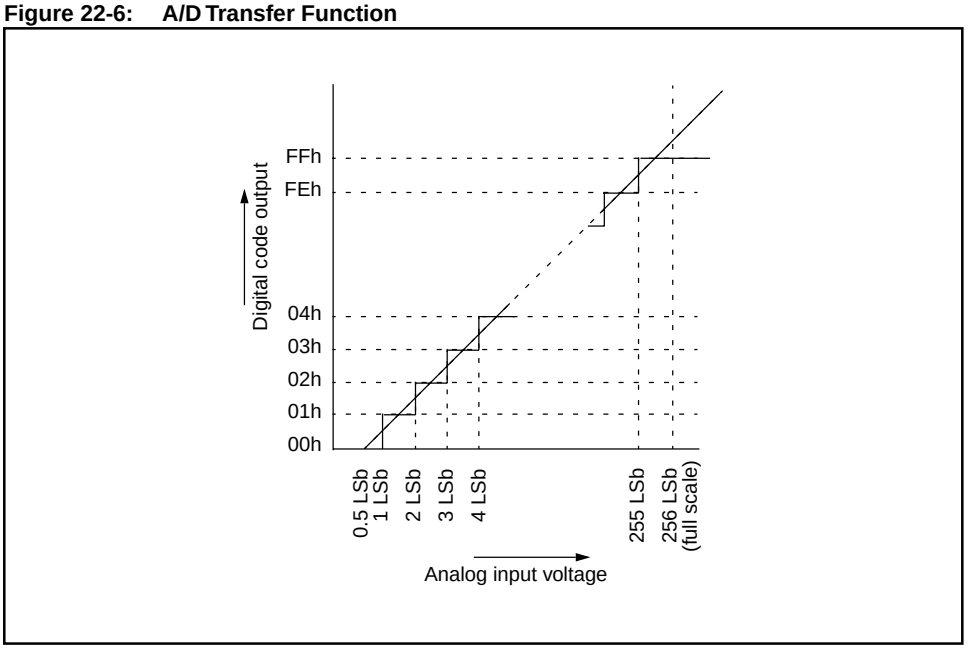
Note:	Care must be taken when using the RA0 pin in A/D conversions due to its proximity to the OSC1 pin.

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.

22.11 Transfer Function

The ideal transfer function of the A/D converter is as follows: the first transition occurs when the analog input voltage (V_{AIN}) is 1 LSb (or Analog V_{REF} / 256) (Figure 22-6).

Figure 22-6: A/D Transfer Function



22.12 Initialization

Example 22-4 shows the initialization of the A/D module in the PIC16C711.

Example 22-4: A/D Initialization (for PIC16C711)

```
     BSF     STATUS, RP0    ; Select Bank1
     CLRF    ADCON1         ; Configure A/D inputs
     BCF     STATUS, RP0    ; Select Bank0
     MOVLW   0xC1           ; RC Clock, A/D is on, Channel 0 selected
     MOVWF   ADCON0         ;
     BSF     INTCON, ADIE   ; Enable A/D Interrupt
     BSF     INTCON, GIE    ; Enable all interrupts
;
;  Ensure that the required sampling time for the selected input
;  channel has elapsed. Then the conversion may be started.
;
     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.
```

22.13 Design Tips

Question 1: ***I am using one of your PIC16C7X devices, and I find that the Analog to Digital Converter result is not always accurate. What can I do to improve accuracy?***

Answer 1:

1. Make sure you are meeting all of the timing specifications. If you are turning the ADC off and on, there is a minimum delay you must wait before taking a sample, if you are changing input channels, there is a minimum delay you must wait for this as well, and finally there is TAD, which is the time selected for each bit conversion. This is selected in ADCON0 and should be between 2 and 6 μs. If TAD is too short, the result may not be fully converted before the conversion is terminated, and if Tad is made too long the voltage on the sampling capacitor can droop before the conversion is complete. These timing specifications are provided in the data book in a table or by way of a formula, and should be looked up for your specific part and circumstances.
2. Often the source impedance of the analog signal is high (greater than 1k ohms) so the current drawn from the source to charge the sample capacitor can affect accuracy. If the input signal does not change too quickly, try putting a 0.1 μF capacitor on the analog input. This capacitor will charge to the analog voltage being sampled, and supply the instantaneous current needed to charge the 51.2 pf internal holding capacitor.
3. On the PIC16C71, one of the analog input pins is next to an oscillator pin. Naturally if these traces are next to each other some noise can couple from the oscillator to the analog circuit. This is especially true when the clock source is an external canned oscillator, since its output is a square wave with a high frequency component to its sharp edge, as opposed to a crystal circuit which provides a slower rise sine wave. Again, decoupling the analog pin can help, or if you can spare it, turn the pin into an output and drive it low. This will really help eliminate cross coupling into the analog circuit.
4. Finally, straight from the data book: “In systems where the device frequency is low, use of the A/D clock derived from the device oscillator is preferred...this reduces, to a large extent, the effects of digital switching noise.” and “In systems where the device will enter SLEEP mode after start of A/D conversion, the RC clock source selection is required. This method gives the highest accuracy.”

Question 2: ***After starting an A/D conversion may I change the input channel (for my next conversion)?***

Answer 2:

After the holding capacitor is disconnected from the input channel, one TAD after the GO bit is set, the input channel may be changed.

Question 3: ***Do you know of a good reference on A/D’s?***

Answer 3:

A very good reference for understanding A/D conversions is the “Analog-Digital Conversion Handbook” third edition, published by Prentice Hall (ISBN 0-13-03-2848-0).

22.14 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the Basic 8-bit A/D module are:

Title	Application Note #
Using the Analog to Digital Converter	AN546
Four Channel Digital Voltmeter with Display and Keyboard	AN557

22.15 Revision History

Revision A

This is the initial released revision of the Basic 8-bit A/D Converter module description.

Section 23. 10-bit A/D Converter

HIGHLIGHTS

This section of the manual contains the following major topics:

Note 1: At present NO released mid-range MCU devices are available with this module. Devices are planned, but there is no schedule for availability. Please refer to Microchip's Web site or BBS for release of Product Briefs which detail the features of devices.

If your current design requires a 10-bit A/D, please look at the PIC17C756 which has a 12-channel 10-bit A/D. This A/D has characteristics which are identical to this module's description.

23.1 Introduction

The analog-to-digital (A/D) converter module can have up to eight analog inputs for a device.

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 (AVDD, AVss) or the voltage level on the AN3/VREF+ and AN2/VREF- pins.

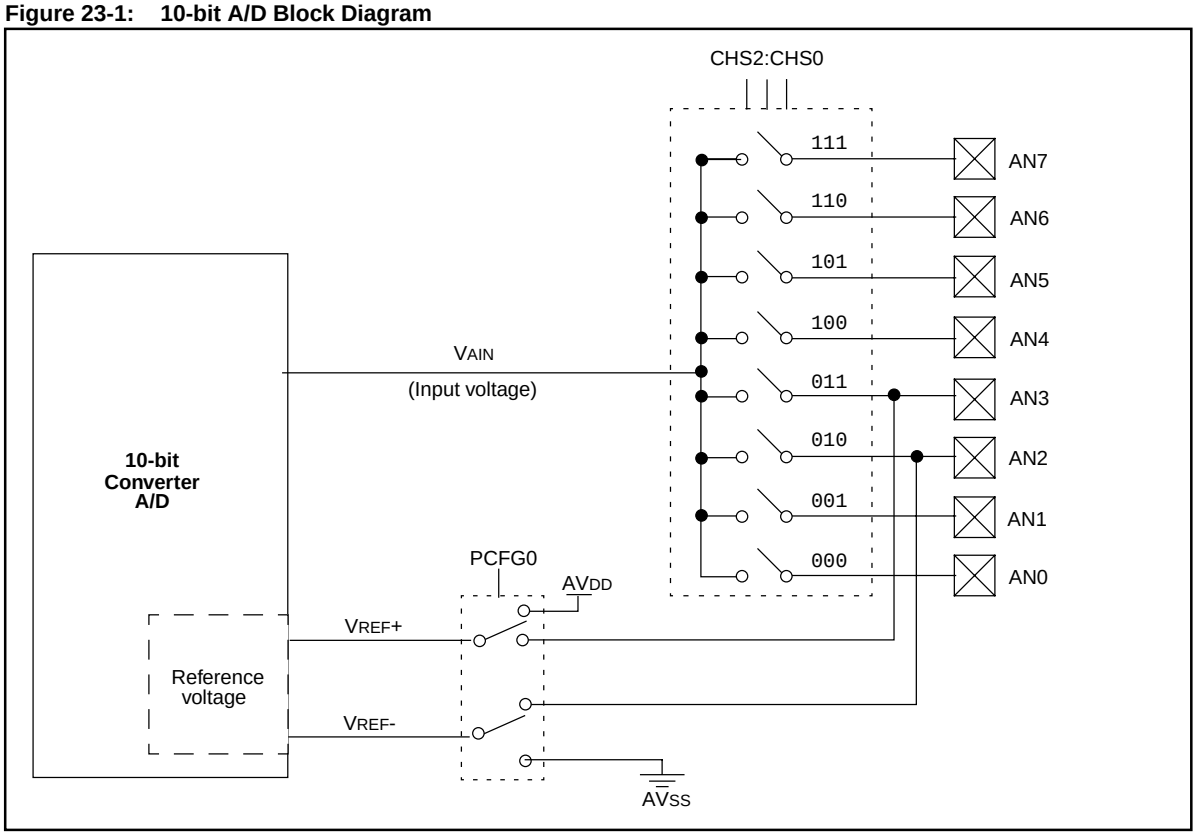
The A/D converter has a unique feature of being able to operate while the device is in SLEEP mode.

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 Figure 23-1, controls the operation of the A/D module. The ADCON1 register, shown in Figure 23-2, configures the functions of the port pins. The port pins can be configured as analog inputs (AN3 and AN2 can also be the voltage references) or as digital I/O.

Figure 23-1: 10-bit A/D Block Diagram



23.2 Control Register

Register 23-1: ADCON0 Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7:6 **ADCS1:ADCS0**: A/D Conversion Clock Select bits

00 = FOSC/2
01 = FOSC/8
10 = FOSC/32
11 = FRC (clock derived from the internal A/D RC oscillator)

bit 5:3 **CHS2:CHS0**: Analog Channel Select bits

000 = channel 0, (AN0)
001 = channel 1, (AN1)
010 = channel 2, (AN2)
011 = channel 3, (AN3)
100 = channel 4, (AN4)
101 = channel 5, (AN5)
110 = channel 6, (AN6)
111 = channel 7, (AN7)

Note: For devices that do not implement the full 8 A/D channels, the unimplemented selections are reserved. Do not select any unimplemented channel.

bit 2 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

When 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 powered up
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

Register 23-2: ADCON1 Register

U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	ADFM	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

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

bit 5 **ADFM**: A/D Result format select (also see Figure 23-6).

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 3:0 **PCFG3:PCFG0**: A/D Port Configuration Control bits

PCFG	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C / R
0000	A	A	A	A	A	A	A	A	AVDD	AVSS	8 / 0
0001	A	A	A	A	VREF+	A	A	A	AN3	AVSS	7 / 1
0010	D	D	D	A	A	A	A	A	AVDD	AVSS	5 / 0
0011	D	D	D	A	VREF+	A	A	A	AN3	AVSS	4 / 1
0100	D	D	D	D	A	D	A	A	AVDD	AVSS	3 / 0
0101	D	D	D	D	VREF+	D	A	A	AN3	AVSS	2 / 1
011x	D	D	D	D	D	D	D	D	—	—	0 / 0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6 / 2
1001	D	D	A	A	A	A	A	A	AVDD	AVSS	6 / 0
1010	D	D	A	A	VREF+	A	A	A	AN3	AVSS	5 / 1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4 / 2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3 / 2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2 / 2
1110	D	D	D	D	D	D	D	A	AVDD	AVSS	1 / 0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1 / 2

A = Analog input D = Digital I/O
C/R = # of analog input channels / # of A/D voltage references

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

Note 1: On any device reset, the port pins that are multiplexed with analog functions (ANx) are forced to be an analog input.

23.3 Operation

The ADRESH:ADRESL registers contains 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 23-1.

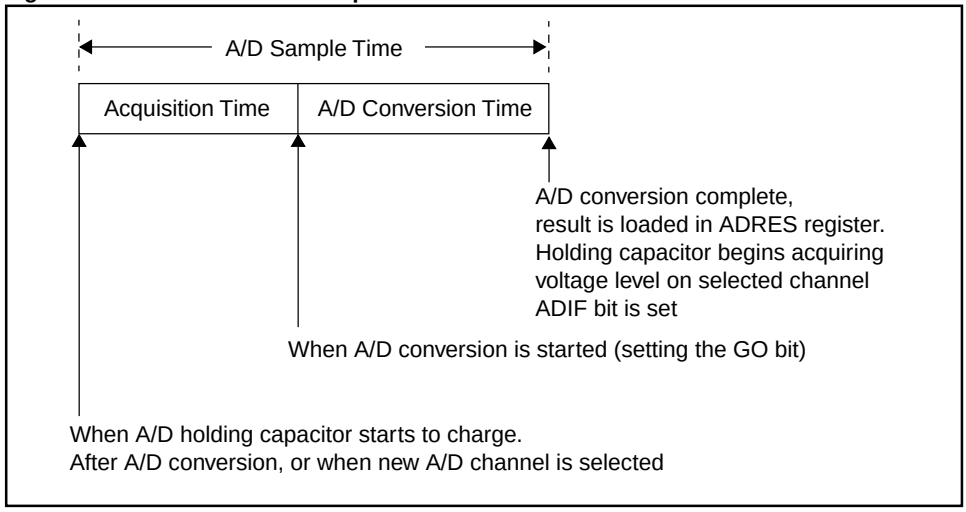
After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as inputs. To determine sample time, see Subsection **23.4 "A/D Acquisition Requirements."** After this 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:
   - Configure analog pins / voltage reference/ and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D conversion clock (ADCON0)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear the ADIF bit
   - Set the ADIE bit
   - Set the GIE bit
3. Wait the required acquisition time.
4. Start conversion:
   - Set the GO/$\overline{\text{DONE}}$ bit (ADCON0)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared or ADIF bit to be set

   OR
   - Waiting for the A/D interrupt
6. Read A/D Result register pair (ADRESH:ADRESL), clear the ADIF bit, if required.
7. For next conversion, go to step 1 or step 2 as required.

Figure 23-2 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 23-2: A/D Conversion Sequence



23.4 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 23-3. 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 23-3. **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 23-1 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.

Equation 23-1: 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 23-2: A/D Minimum Charging Time

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \bullet (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$

or

$$T_C = -(120\ pF)(1\ k\Omega + R_{SS} + R_S) \ln(1/2047)$$

Example 23-1 shows the calculation of the minimum required acquisition time TACQ. This calculation 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 23-3)
Temperature	=	50°C (system max.)	
VHOLD	=	0V @ time = 0	

Example 23-1: Calculating the Minimum Required Acquisition Time (Case 1)

$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$

Temperature coefficient is only required for temperatures > 25°C.

$T_{ACQ} = 2\ \mu s + T_C + [(Temp - 25°C)(0.05\ \mu s/°C)]$

$$\begin{aligned} T_C &= -C_{HOLD}(R_{IC} + R_{SS} + R_S) \ln(1/2047) \\ &= -120\ pF\ (1\ k\Omega + 7\ k\Omega + 10\ k\Omega) \ln(0.0004885) \\ &= -120\ pF\ (18\ k\Omega) \ln(0.0004885) \\ &= -2.16\ \mu s\ (-7.6241) \\ &= 16.47\ \mu s \end{aligned}$$

$$\begin{aligned} T_{ACQ} &= 2\ \mu s + 16.47\ \mu s + [(50°C - 25°C)(0.05\ \mu s/°C)] \\ &= 18.47\ \mu s + 1.25\ \mu s \\ &= 19.72\ \mu s \end{aligned}$$

Now to get an idea what happens to the acquisition time when the source impedance is a minimal value (*Rs* = 50 Ω). Example 23-2 shows the same conditions as in Example 23-1 with only the source impedance made a minimal value (*Rs* = 50 Ω).

Example 23-2: Calculating the Minimum Required Acquisition Time (Case 2)

TACQ =	TAMP + TC + TCOFF
Temperature coefficient is only required for temperatures > 25°C.	
TACQ =	2 µs + Tc + [(Temp - 25°C)(0.05 µs/°C)]
TC =	-CHOLD (RIC + RSS + RS) ln(1/2047) -120 pF (1 kΩ + 7 kΩ + 50 Ω) ln(0.0004885) -120 pF (8050 Ω) ln(0.0004885) -0.966 µs (-7.6241) 7.36 µs
TACQ =	2 µs + 16.47 µs + [(50°C - 25°C)(0.05 µs/°C)] 9.36 µs + 1.25 µs 10.61 µs

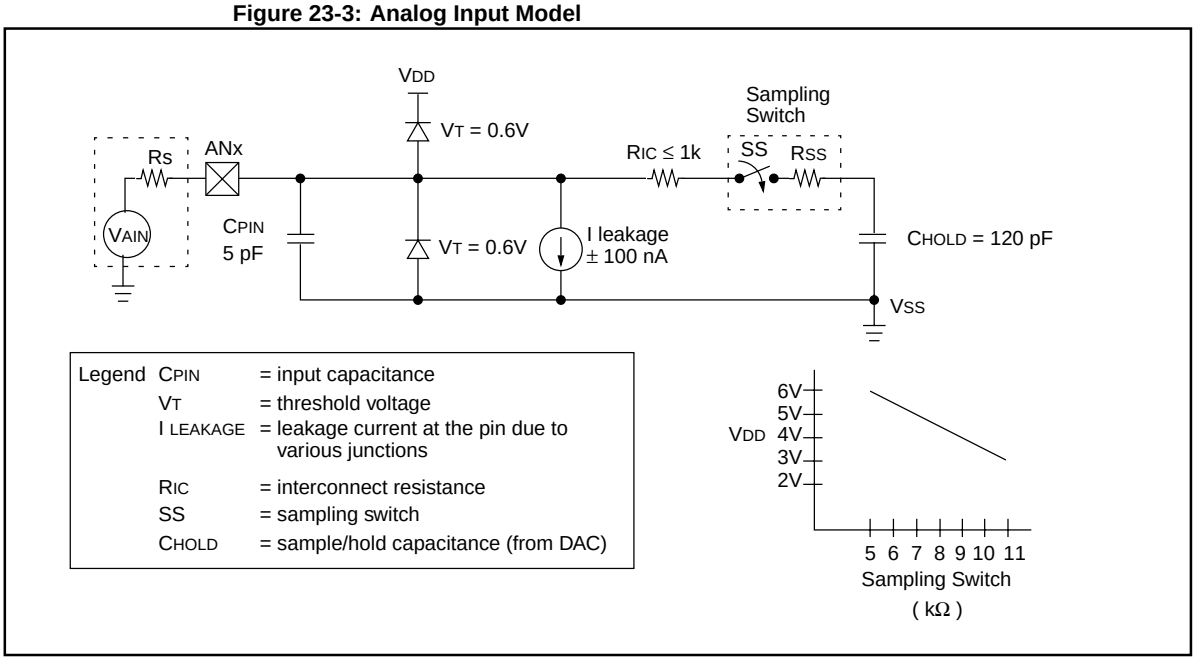
Note 1: The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

Note 2: The charge holding capacitor (CHOLD) is not discharged after each conversion.

Note 3: The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

Note 4: After a conversion has completed, a 2.0TAD delay must complete before acquisition can begin again. During this time the holding capacitor is not connected to the selected A/D input channel.

Figure 23-3: Analog Input Model



23.5 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 11.5TAD per 10-bit conversion. The source of the A/D conversion clock is software selected. The four possible options for TAD are:

- 2TOSC
- 8TOSC
- 32TOSC
- 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 as shown in parameter 130 of the **"Electrical Specifications"** section.

Table 23-1 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 23-1: TAD vs. Device Operating Frequencies (for Standard, C, Devices)

AD Clock Source (TAD)		Device Frequency			
Operation	**ADCS1:ADCS0**	**20 MHz**	**5 MHz**	**1.25 MHz**	**333.33 kHz**
2TOSC	00	100 ns(2)	400 ns(2)	1.6 µs	6 µs
8TOSC	01	400 ns(2)	1.6 µs	6.4 µs	24 µs(3)
32TOSC	10	1.6 µs	6.4 µs	25.6 µs(3)	96 µs(3)
RC	11	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1)

Legend: Shaded cells are outside of recommended range.
Note 1: The RC source has a typical TAD time of 4 µs.
2: These values violate the minimum required TAD time.
3: For faster conversion times, the selection of another clock source is recommended.
4: For device frequencies above 1 MHz, the device must be in SLEEP for the entire conversion, or the A/D accuracy may be out of specification.

Table 23-2: TAD vs. Device Operating Frequencies (for Extended, LC, Devices)

AD Clock Source (TAD)		Device Frequency			
Operation	**ADCS1:ADCS0**	**4 MHz**	**2 MHz**	**1.25 MHz**	**333.33 kHz**
2TOSC	00	500 ns(2)	1.0 µs(2)	1.6 µs(2)	6 µs
8TOSC	01	2.0 µs(2)	4.0 µs	6.4 µs	24 µs(3)
32TOSC	10	8.0 µs	16.0 µs	25.6 µs(3)	96 µs(3)
RC	11	3 - 9 µs(1,4)	3 - 9 µs(1,4)	3 - 9 µs(1,4)	3 - 9 µs(1,4)

Legend: Shaded cells are outside of recommended range.
Note 1: The RC source has a typical TAD time of 6 µs.
2: These values violate the minimum required TAD time.
3: For faster conversion times, the selection of another clock source is recommended.
4: For device frequencies above 1 MHz, the device must be in SLEEP for the entire conversion, or the A/D accuracy may be out of specification.

23.6 Configuring Analog Port Pins

The ADCON1 and TRIS registers control the operation of the A/D port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bits set (input). If the TRIS 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 TRIS 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.

Note 2: Analog levels on any pin that is defined as a digital input (including the AN7:AN0 pins), may cause the input buffer to consume current that is out of the devices specification.

23.7 A/D Conversions

Example 23-3 shows how to perform an A/D conversion for the PIC17C756. 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 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, due to the required acquisition time requirement.

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.

Example 23-3: A/D Conversion

```
    BSF     STATUS, RP0       ; Select Bank1
    CLRF    ADCON1            ; Configure A/D inputs,
                              ;    result is left justified
    BSF     PIE1, ADIE        ; Enable A/D interrupts
    BCF     STATUS, RP0       ; Select Bank0
    MOVLW   0xC1              ; RC Clock, A/D is on, Channel 0 is selected
    MOVWF   ADCON0            ;
    BCF     PIR1, ADIF        ; Clear A/D interrupt flag bit
    BSF     INTCON, PEIE      ; Enable peripheral interrupts
    BSF     INTCON, GIE       ; Enable all interrupts
;
;  Ensure that the required sampling time for the selected input
;  channel has elapsed. Then the conversion may be started.
;
    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 23-4: A/D Conversion T_{AD} Cycles

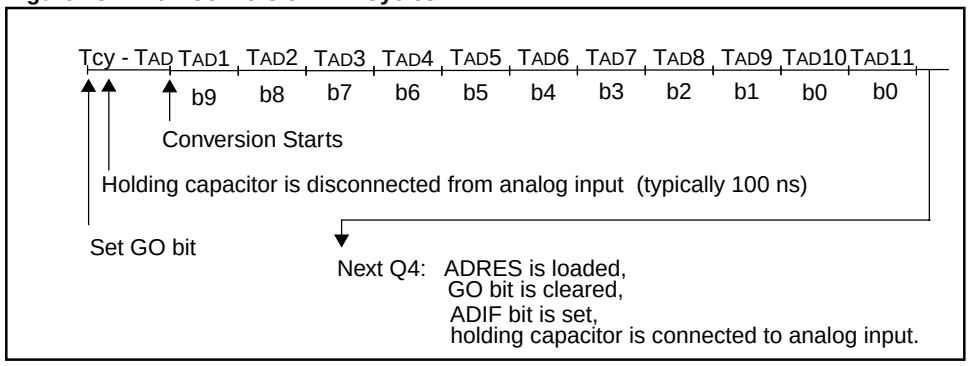
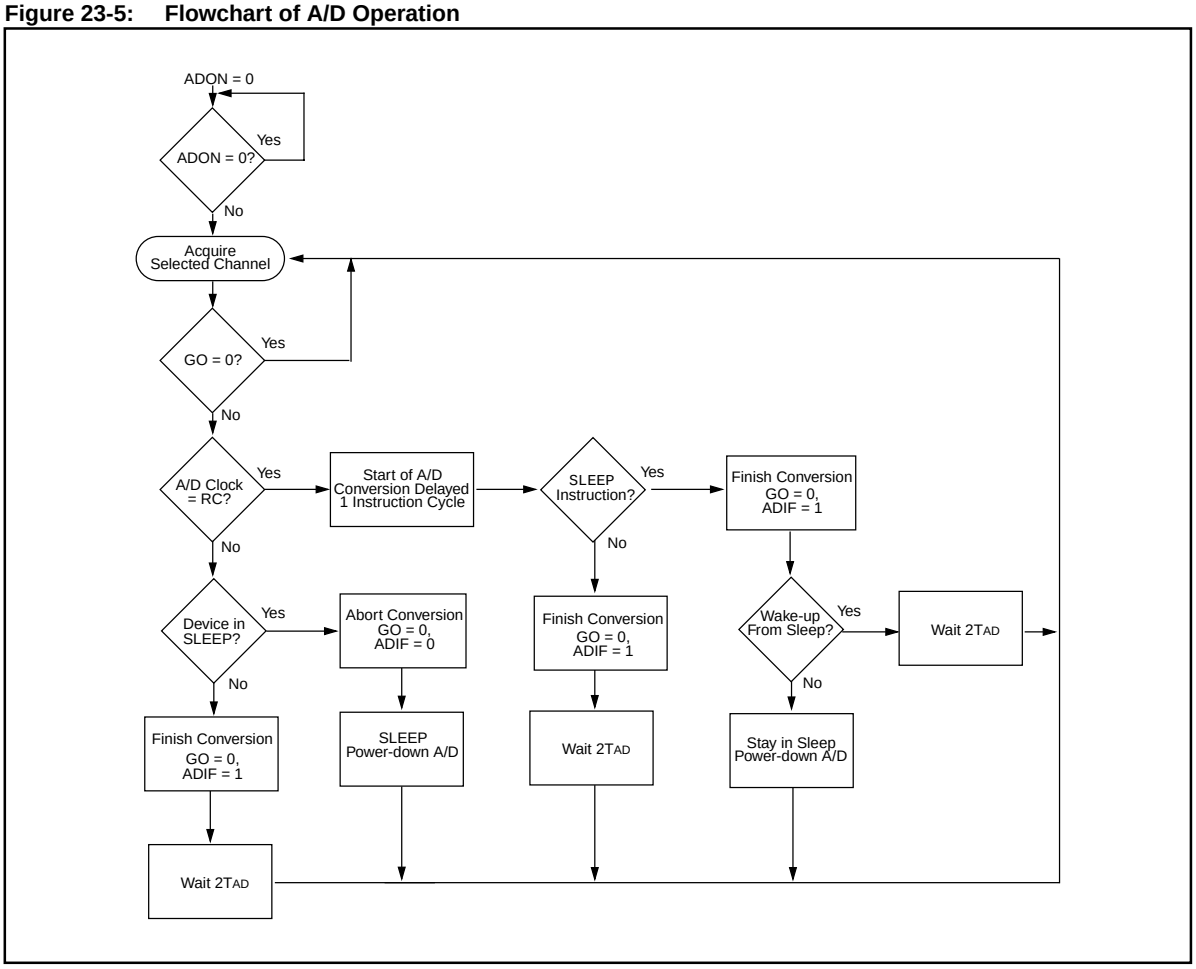


Figure 23-5: Flowchart of A/D Operation



23.7.1 Faster Conversion - Lower Resolution Trade-off

Not all applications require a result with 10-bits of resolution, but may instead require a faster conversion time. The A/D module allows users to make the trade-off of conversion speed to resolution. Regardless of the resolution required, the acquisition time is the same. To speed up the conversion, the clock source of the A/D module may be switched so that the TAD time violates the minimum specified time (see the applicable electrical specification). Once the TAD time violates the minimum specified time, all the following A/D result bits are not valid (see A/D Conversion Timing in the Electrical Specifications section). The clock sources may only be switched between the three oscillator versions (cannot be switched from/to RC). The equation to determine the time before the oscillator can be switched is as follows:

Conversion time = TAD + N • TAD + (11 - N)(2TOSC)
Where: N = number of bits of resolution required.

Since the TAD is based from the device oscillator, the user must use some method (a timer, software loop, etc.) to determine when the A/D oscillator may be changed. Example 23-4 shows a comparison of time required for a conversion with 4-bits of resolution, versus the 10-bit resolution conversion. The example is for devices operating at 20 MHz (The A/D clock is programmed for 32TOSC), and assumes that immediately after 6TAD, the A/D clock is programmed for 2TOSC.

The 2TOSC violates the minimum TAD time since the last 4 bits will not be converted to correct values.

Example 23-4: 4-bit vs. 8-bit Conversion Times

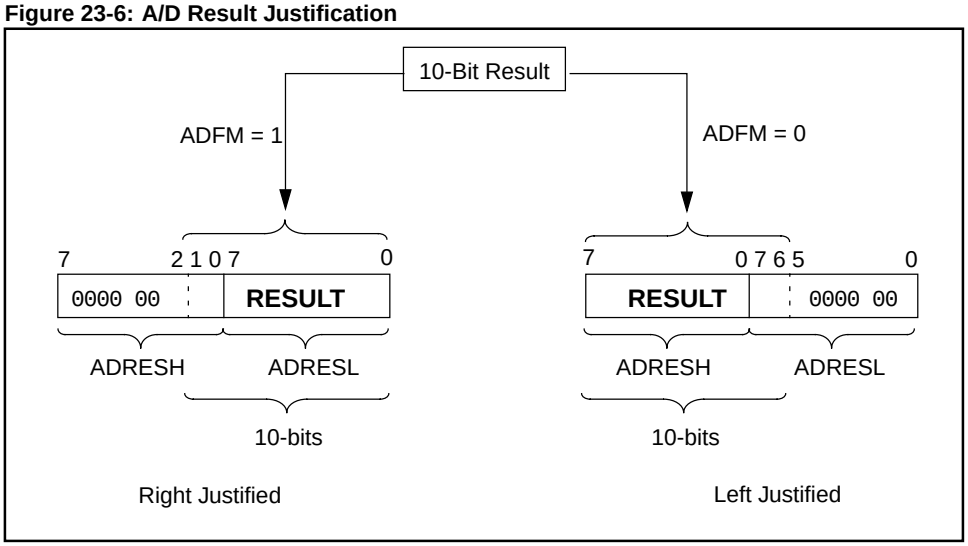
	Freq. (MHz)(1)	Resolution	
		4-bit	10-bit
TAD	20	1.6 µs	1.6 µs
TOSC	20	50 ns	50 ns
2TAD + N • TAD + (11 - N)(2TOSC)	20	8.7 µs	17.6 µs

Note 1: A minimum TAD time of 1.6 µs is required.
2: If the full 8-bit conversion is required, the A/D clock source should not be changed.

23.7.2 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 23-6 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.

Figure 23-6: A/D Result Justification



23.8 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 internal digital switching noise from the conversion. When the conversion is completed the GO/$\overline{\text{DONE}}$ bit will be cleared, and the result is 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 (to conserve power), 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.

23.9 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.

23.10 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 VDD diverges from VREF.

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.

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 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.

23.11 Connection Considerations

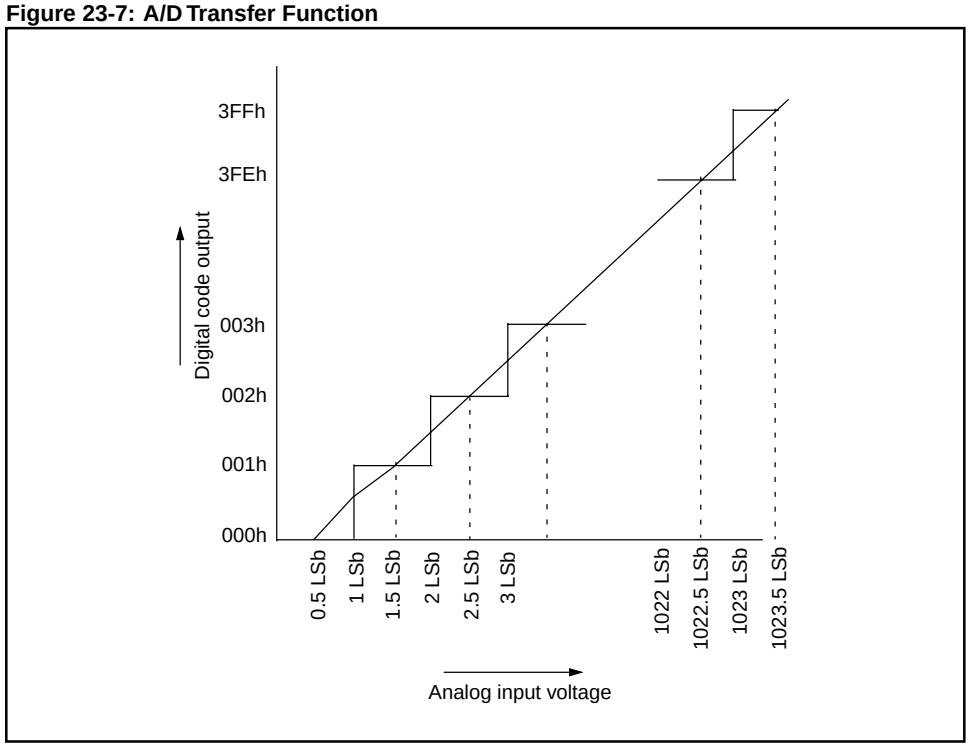
If the input voltage exceeds the rail values (V_{SS} or V_{DD}) by greater than 0.3V, then the accuracy of the conversion is out of specification.

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.

23.12 Transfer Function

The ideal transfer function of the A/D converter is as follows: the first transition occurs when the analog input voltage (V_{AIN}) is 1 LSb (or Analog V_{REF} / 1024) (Figure 23-7).

Figure 23-7: A/D Transfer Function



23.13 Initialization

Example 23-5 shows an initialization of the A/D module.

Example 23-5: A/D Initialization

```
    BSF     STATUS, RP0      ; Select Bank1
    CLRF    ADCON1           ; Configure A/D inputs
    BSF     PIE1, ADIE       ; Enable A/D interrupts
    BCF     STATUS, RP0      ; Select Bank0
    MOVLW   0xC1             ; RC Clock, A/D is on, Channel 0 is selected
    MOVWF   ADCON0           ;
    BCF     PIR1, ADIF       ; Clear A/D interrupt flag bit
    BSF     INTCON, PEIE     ; Enable peripheral interrupts
    BSF     INTCON, GIE      ; Enable all interrupts
;
;  Ensure that the required sampling time for the selected input
;  channel has elapsed. Then the conversion may be started.
;
    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.
```

23.14 Design Tips

Question 1: ***I find that the Analog to Digital Converter result is not always accurate. What can I do to improve accuracy?***

Answer 1:

1. Make sure you are meeting all of the timing specifications. If you are turning the module off and on, there is a minimum delay you must wait before taking a sample. If you are changing input channels, there is a minimum delay you must wait for this as well, and finally there is TAD, which is the time selected for each bit conversion. This is selected in ADCON0 and should be between 1.6 and 6 μs. If TAD is too short, the result may not be fully converted before the conversion is terminated, and if TAD is made too long the voltage on the sampling capacitor can droop before the conversion is complete. These timing specifications are provided in the **"Electrical Specifications"** section. See the device data sheet for device specific information.
2. Often the source impedance of the analog signal is high (greater than 1k ohms) so the current drawn from the source to charge the sample capacitor can affect accuracy. If the input signal does not change too quickly, try putting a 0.1 μF capacitor on the analog input. This capacitor will charge to the analog voltage being sampled and supply the instantaneous current needed to charge the 120 pF internal holding capacitor.
3. Finally, straight from the data book: "In systems where the device frequency is low, use of the A/D clock derived from the device oscillator is preferred...this reduces, to a large extent, the effects of digital switching noise." and "In systems where the device will enter SLEEP mode after start of A/D conversion, the RC clock source selection is required.This method gives the highest accuracy."

Question 2: ***After starting an A/D conversion may I change the input channel (for my next conversion)?***

Answer 2:

After the holding capacitor is disconnected from the input channel, typically 100 ns after the GO bit is set, the input channel may be changed.

Question 3: ***Do you know of a good reference on A/D's?***

Answer 3:

A very good reference for understanding A/D conversions is the "Analog-Digital Conversion Handbook" third edition, published by Prentice Hall (ISBN 0-13-03-2848-0).

23.15 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the 10-bit A/D module are:

Title	Application Note #
Using the Analog to Digital Converter	AN546
Four Channel Digital Voltmeter with Display and Keyboard	AN557

23.16 Revision History

Revision A

This is the initial released revision of the 10-bit A/D module description.

Section 24. Slope A/D

HIGHLIGHTS

This section of the manual contains the following major topics:

24.1 Introduction

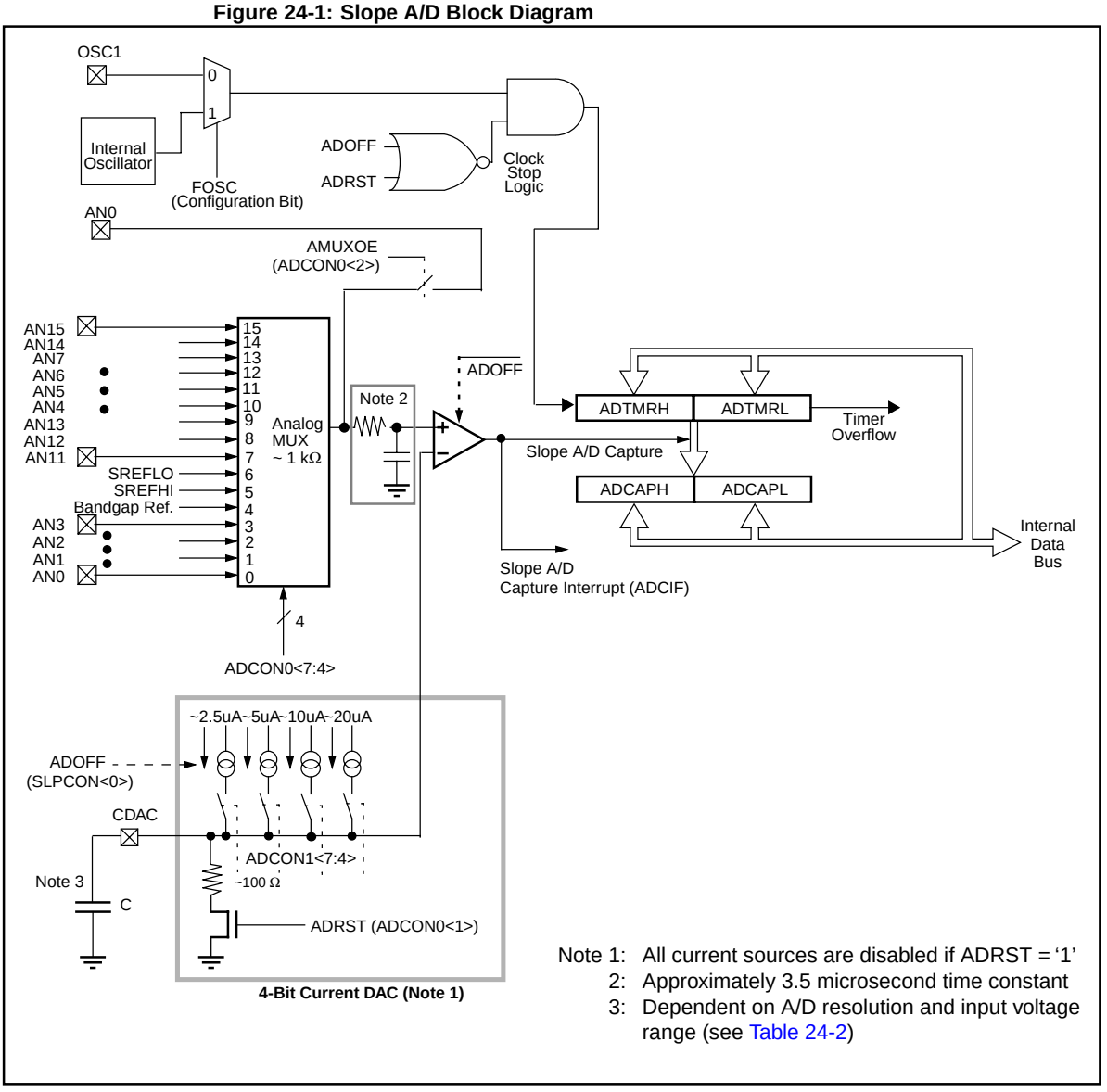
The components required to create a Slope A/D converter include:

- Precision comparator
- 4-bit programmable current source
- 16-channel analog MUX
- 16-bit timer with capture register

This section will discuss using these components for a Slope A/D.

Each analog input channel is multiplexed to a single analog input source to be converted by means of a slope conversion method (using a single precision comparator). The programmable current source feeds an external capacitor to generate the ramp voltage used in the conversion.

Figure 24-1: Slope A/D Block Diagram



Note 1: All current sources are disabled if ADRST = '1'
2: Approximately 3.5 microsecond time constant
3: Dependent on A/D resolution and input voltage range (see Table 24-2)

24.2 Control Registers

Two A/D control registers are provided to control the conversion process. They are ADCON0 and ADCON1. Both registers are readable and writable.

Register 24-1: ADCON0 Register

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-1	R/W-0
ADCS3	ADCS2	ADCS1	ADCS0	—	AMUXOE	ADRST	ADZERO
bit 7							bit 0

bit 7-4: **ADCS3:ADCS0**: Analog Channel Select bits

0000 = AN0 input
0001 = AN1 input
0010 = AN2 input
0011 = AN3 input
0100 = Bandgap reference voltage input
0101 = Slope reference SREFHI input
0110 = Slope reference SREFLO input
0111 = AN11 input
1000 = AN12 input
1001 = AN13 input
1010 = AN4 input
1011 = AN5 input
1100 = AN6 input
1101 = AN7 input
1110 = AN14 input
1111 = AN15 input

Note: For devices that do not use the full 16 A/D input channels, the unimplemented selections are reserved. Do not select any unimplemented channels.

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

bit 2: **AMUXOE:** Analog MUX Output Enable

1 = Connect AMUX Output to AN0 pin (overrides TRIS setting)
0 = AN0 pin normal

bit 1: **ADRST:** A/D Reset Control Bit

1 = Stop the A/D Timer, discharge CDAC capacitor
0 = Normal operation (A/D running)

bit 0: **ADZERO**: A/D Zero Select Control

1 = Enable zeroing operation on AN1 and AN5
0 = Normal operation, sample AN1 and AN5 pins

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

Register 24-2: ADCON1 Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADDAC3	ADDAC2	ADDAC1	ADDAC0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7-4: **ADDAC3:ADDAC0**: Programmable Current Source Select bits

0000 = OFF - all current sources disabled
0001 = 2.25 µA
0010 = 4.5 µA
0011 = 6.75 µA
0100 = 9 µA
0101 = 11.25 µA
0110 = 13.5 µA
0111 = 15.75 µA
1000 = 18 µA
1001 = 20.25 µA
1010 = 22.5 µA
1011 = 24.75 µA
1100 = 27 µA
1101 = 29.25 µA
1110 = 31.5 µA
1111 = 33.75 µA

bit 3-0: **PCFG3:PCFG0**: Port Configuration Selects

PCFG3:PCFG2	AN4	AN5	AN6	AN7
00	A	A	A	A
01	A	A	A	D
10	A	A	D	D
11	D	D	D	D

PCFG1:PCFG0	AN0	AN1	AN2	AN3
00	A	A	A	A
01	A	A	A	D
10	A	A	D	D
11	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

Note: On any device reset, all port pins multiplexed with analog functions (ANx pins), are forced to be an analog input.

Register 24-3: SLPCON Register

R/W-0	U-0	R/W-1	R/W-1	U-1	R/W-1	R/W-1	R/W-1
Resv	—	REFOFF	Resv	OSCOFF	Resv	Resv	ADOFF
bit 7							bit 0

bit 7: **Reserved:** Always maintain this bit cleared.

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

bit 5: **REFOFF:** Slope A/D Voltage Reference Power Control bit

1 = Voltage references are disabled (not consuming current)
0 = Voltage references are powered (consuming current)

bit 4: **Reserved:** Always maintain this bit cleared.

bit 3: **OSCOFF:** Slope A/D Oscillator Sleep Control bit

1 = Slope A/D Oscillator is disabled during SLEEP mode (not consuming current)
0 = Slope A/D Oscillator is enabled during SLEEP mode (consuming current)

bit 2: **Reserved:** Always maintain this bit cleared.

bit 1: **Reserved:** Always maintain this bit cleared.

bit 0: **ADOFF:** Slope A/D Power Control bit

1 = Slope A/D is disabled (not consuming current)
0 = Slope A/D is powered (consuming current)

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

24.3 Conversion Process

There are two methods for performing a conversion. To determine the end of conversion, the first method uses the ADTMR overflow interrupt (OVFIF bit). The second method uses the A/D Capture Interrupt (ADCIF bit). At the end of conversion both bits are used to determine if an overrange condition has occurred.

Method 1 uses a fixed conversion time, this means that the capacitor voltage always ramps to the full scale voltage. Immediately after the overflow of the ADTMR, we recommend that the ADRST bit is set to discharge the external capacitor. This will ensure that the residual voltage on the external capacitor, due to dielectric absorption, is independent of input voltage or previous conversions.

Method 2 uses a variable conversion time, which results in faster conversions for lower input voltages.

Steps for Method 1 (“fixed conversion time”):

1. Initialize the Slope A/D module:
   a) Clear the REFOFF bit (SLPCON<5>)
   b) Clear the ADOFF bit (SLPCON<0>)
   c) Initialize ADCON1<7:4> to select the programmable current source.
2. Set the ADRST bit (ADCON0<1>), until the ramp capacitor reaches ground. This is capacitor dependent. A minimum of 200 µs is recommended.
3. Select Input Channel
4. Clear the OVFIF and ADCIF bits.
5. Initialize Slope A/D Timer (ADTMR). ADTMR value depends on bits of resolution required (see Table 24-1).
6. To start a conversion, clear the ADRST bit, this allows the ramp capacitor to begin charging and the ADTMR to increment.
7. Conversion is complete when the Slope A/D timer (ADTMR) overflows from FFFFh to 0000h. This causes the OVFIF bit to be set.
8. Check if the ADCIF bit is set. If this bit is set, the value in the capture register ADCAP is valid. This method depends on minimum latency to verify the capture interrupt flag bit after the ADTMR overflows. If the ADCIF bit is cleared, then the input voltage was out of the A/D input range.
9. Set the ADRST bit (ADCON0<1>) to stop ADTMR and discharge external capacitor
10. Do Conversion Calculations
11. Goto Step 2

Steps for Method 2 (“variable conversion time”):

1. Initialize the Slope A/D module:
   a)Clear the REFOFF bit (SLPCON<5>).
   b)Clear the ADOFF bit (SLPCON<0>).
   c)Initialize ADCON1<7:4> to select the programmable current source.
2. Set the ADRST bit (ADCON0<1>), until the ramp capacitor reaches ground. This is capacitor dependent. A minimum of 200 µs is recommended.
3. Select Input Channel.
4. Clear the OVFIF and ADCIF bits.
5. Initialize Slope A/D Timer (ADTMR). ADTMR value depends on bits of resolution required (see Table 24-1).
6. To start a conversion, clear the ADRST bit, this allows the ramp capacitor to begin charging and the ADTMR to increment.
7. Conversion is complete when the ramp voltage exceeds the analog input so the comparator output changes from high to low. This causes the ADCIF bit to be set.
8. Check if the ADTMR did not increment more counts than the maximum resolution allowed. If there were more counts, then the input voltage was out of the A/D input range.
9. Set the ADRST bit (ADCON0<1>) to stop ADTMR and discharge external capacitor
10. Do Conversion Calculations.
11. Go to Step 2.

Note:	The Slope A/D timer continues to run following a capture event.

The maximum Slope A/D timer count is 65,536. It can be clocked by the on-chip or external oscillator. At a 4 MHz oscillation frequency, the maximum conversion time is 16.38 ms for a full count. A typical conversion should complete before full-count is reached. The timer overflow flag is set once the timer rolls over (FFFFh to 0000h), and an interrupt occurs, if enabled.

End-user calibration is simplified or eliminated by making use of the on-chip EPROM. Internal component values are measured at factory final test and stored in the memory for use by the application firmware.

Periodic conversion cycles should be performed on the bandgap and slope references (described in Subsection **24.4 “Other Analog Modules”**) to compensate for Slope A/D component drift. Measurements for the reference voltage counts are equated to the voltage value stored into EPROM during calibration. Since all measurements are relative to the reference, offset voltages inherent in the comparator are minimized. The Slope A/D clock source does not require a precise frequency, only a stable frequency.

See AN624, “PIC14000 Slope A/D Theory and Implementation” for further details of Slope A/D operation.

24.3.1 Slope A/D Timer (ADTMR)

The Slope A/D timer (ADTMR) is comprised of a 16-bit timer (ADTMRH:ADTMRL), which is incremented every oscillator cycle. The ADTMR registers are cleared by a power-on reset; otherwise the software must initialize it after each conversion. A separate 16-bit capture register (ADCAPH:ADCAPL) is used to capture the ADTMR count if a Slope A/D capture event occurs (see below). Both the Slope A/D timer and capture registers are readable and writable. The 16-bit timer is a read/write register and is cleared on any device reset.

Note 1:	Reading or writing the ADTMR register during an Slope A/D conversion cycle can produce unpredictable results and is not recommended.
Note 2:	The correct sequence for writing the ADTMR register is HI byte followed by LO byte. Reversing this order will prevent the Slope A/D timer from running.

During a conversion one or both of the following events will occur:

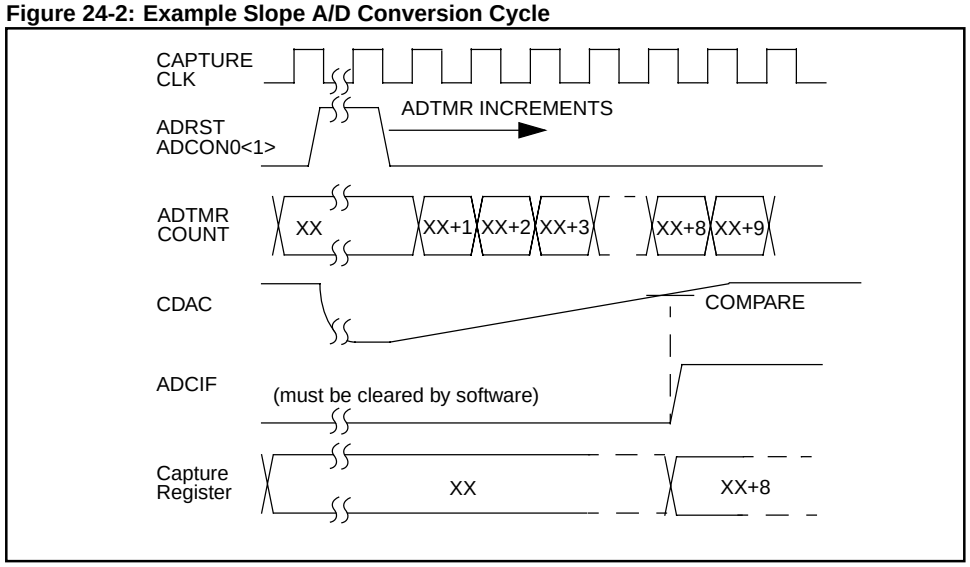
- capture event
- timer overflow

In a capture event, the comparator trips when the slope voltage on the CDAC output exceeds the input voltage from the selected Slope A/D channel, causing the comparator output to transition from high to low. This causes a transfer of the current timer count to the capture register and sets the ADCIF flag bit.

An interrupt will be generated if the ADCIE bit is set (interrupt enabled). In addition, GIE and PIE bits must be set. Software is responsible for clearing the ADCIF flag bit prior to the next conversion cycle. This interrupt can only occur once per conversion cycle.

In a timer overflow condition, the timer rolls over from FFFFh to 0000h, and a capture overflow flag (OVFIF) is asserted. The timer continues to increment following a timer overflow. An interrupt can be generated if bit OVFIE is set (interrupt enabled). In addition, the GIE and PIE bits must be set. Software is responsible for clearing the OVFIF flag bit prior to the next conversion cycle.

Figure 24-2: Example Slope A/D Conversion Cycle



24.3.2 Sleep Operation

The Slope A/D may operate when the device is in Sleep mode. For the Slope A/D to do a conversion during Sleep mode, the Slope A/D module must have a device clock. For a clock to be present the OSCOFF bit must be cleared before going to SLEEP. Also the REFOFF and ADOFF bits must be cleared to ensure that the results reflect the voltage on the input channel. By doing an A/D conversion during Sleep mode, the result has improved accuracy due to a reduction of system noise.

When the device clock is disabled, the Slope A/D Timer (ADTMRH:ADTMRL) stops incrementing. Even if the Slope A/D module is not disabled, the slope A/D cannot wake-up the device. This is because the ADCIF bit cannot be set, which is one of the control bits used to wake the device from SLEEP mode. When the device awakes, if the comparator value has tripped, the capture and interrupt will occur. The value in the ADCAP registers is meaningless.

For maximum power savings, all analog components of the Slope A/D module should be disabled (no conversion in progress).

24.3.3 Effects of a Reset

After any device reset, the Slope A/D module is disabled (lowest current state) and the device I/O are configured as analog channels.

24.3.4 Slope A/D Comparator

This module includes a high gain comparator for Slope A/D conversions. The non-inverting input terminal of the Slope A/D comparator is connected to the output of an analog MUX through an RC low-pass filter. The inverting input terminal is connected to the external ramp capacitor.

The output of the comparator is used to cause the capture event to occur. This causes the value in the ADTMR registers to be loaded into the ADCAP registers. This output will also cause the ADCIF bit to be set.

24.3.5 Analog MUX

A total of 16 channels are internally multiplexed to the single Slope A/D comparator positive input. Four configuration bits (ADCON0<7:4>) select the channel to be converted.

24.3.6 Programmable Current Source

Four configuration bits (ADCON1<7:4>) are used to control a programmable current source for generating the ramp voltage to the Slope A/D comparator. This allows compensation for full-scale input voltage, clock frequency and the external capacitor tolerance variations.

Setting the ADRST bit disconnects the current source from the CDAC pin. Current flow begins when the ADRST bit is cleared.

The programmable current source output is tied to the CDAC pin. This current source is used to charge an external capacitor, which generates the ramp voltage for the Slope A/D comparator (Figure 24-1).

24.3.7 Slope A/D Resolution, Speed, Voltage Range, and Capacitor Selection

The Slope A/D module allows many trade-offs. For a conversion the user needs to make the following Trade-offs:

- The Resolution of the Result
- The Speed of the Conversion
- The Analog Input Voltage Range
- The External Capacitor

The resolution is the number of bits that is used by the ADTMR to represent the measured input voltage. This resolution affects the time that the conversion can be completed in. Table 24-1 shows the trade-off between the resolution of the conversion and the maximum conversion time.

The conversion time for the Slope A/D converter can be calculated using the equation:

Conversion Time = (1/Fosc) x 2^N

Where Fosc is the oscillator frequency and N is the number of bits of resolution desired.

Therefore at 4 MHz, the conversion time for 16 bits is 16.384 msec. Conversely, it is 256 µsec for 10-bits.

Table 24-1: ADTMR Initialization Values and Conversion Times

Resolution Bits	Value Loaded into ADTMR	Maximum Conversion Time		
		Cycles	20 MHz	4 MHz
16	0000h	65536 TOSC	3.28 ms	16.38 ms
15	8000h	32768 TOSC	1.64 ms	8.2 ms
14	C000h	16384 TOSC	820 µs	4.1 ms
13	E000h	8192 TOSC	410 µs	2.05 ms
12	F000h	4096 TOSC	204.8 µs	1.03 ms
11	F800h	2048 TOSC	102.4 µs	500 µs
10	FC00h	1024 TOSC	51.2 µs	250 µs

.

The selection of the external capacitor is determined by the desired characteristics of the application. These include

- Input Voltage Range (widest range of all input channels)
- Conversion Time
- Programmable Current Source Output Values

The selection of these values should be done to minimize the time between a comparator trip (ADCIF bit is set) to the ADTMR overflow (OVFIF is set). This ensures that the entire range of the ADTMR is used for the A/D conversion process.

The equation for selecting the ramp capacitor value is:

Capacitor = (conversion time in seconds) X
(current source output in amps) / (full scale in volts)

Table 24-2 provides example capacitor values for the desired Slope A/D resolution, conversion time, and full scale voltage measurement.

This capacitor on the CDAC pin should have a low voltage-coefficient as found in teflon, polypropylene, or polystyrene capacitors, for optimum results. This external capacitor must be discharged at the beginning of each conversion cycle by setting the ADRST bit (ADCON0<1>). The time for the ADRST bit to be set depends on the characteristics of the external capacitor for a complete discharge.

Table 24-2: External Capacitor Selection (@ 4 MHz)

Slope A/D Resolution (Bits)	Conversion Time (ms)	Full Scale (Volts)	Slope A/D Current Source Select		Calculated CDAC Capacitor	CDAC Capacitor Nearest Standard Value
			ADDAC3:ADDC0	Typical Output (μamps)		
16	16.384	3.5	1100	27	0.126 μF	0.12 μF
		2.0	1010	22.5	0.184 μF	0.18 μF
		1.5	1011	24.75	0.270 μF	0.27 μF
14	4.096	3.5	1101	29.25	34 nF	33 nF
		2.0	1011	24.75	50.7 nF	47 nF
		1.5	1100	27	73.7 nF	68 nF
12	1.024	3.5	1101	29.25	8.56 nF	8.2 nF
		2.0	1001	20.25	10.4 nF	10 nF
		1.5	1010	22.5	15.4 nF	15 nF
10	0.256	3.5	1011	24.75	1.81 nF	1.8 nF
		2.0	1010	22.5	2.88 nF	2.7 nF
		1.5	1011	24.75	4.22 nF	3.9 nF

.

24.4 Other Analog Modules

Additional analog modules for mixed signal applications are required. These include:

- bandgap voltage reference
- slope reference voltage divider

24.4.1 Bandgap Voltage Reference

The bandgap reference circuit is used to generate a 1.2V nominal stable voltage reference for the Slope A/D, the low-voltage detector, and the slope reference divider. The bandgap reference voltage is available on the analog MUX. To enable the bandgap reference REFOFF (SLPCON<5>) must be cleared. The bandgap reference must be enabled for any slope A/D conversion.

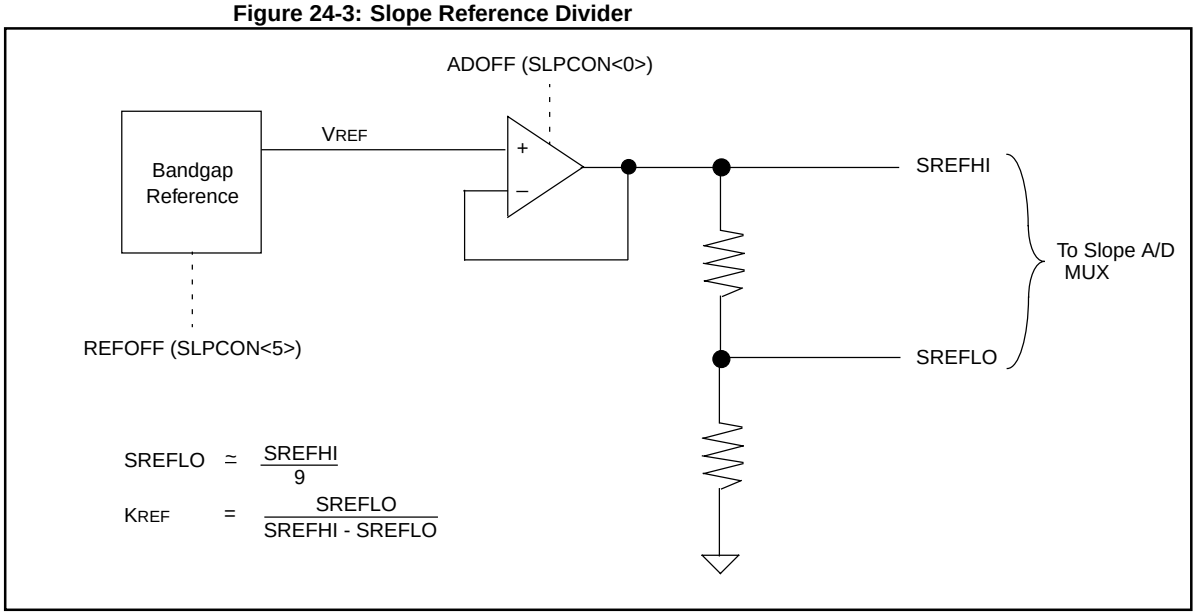
The bandgap reference calibration factor is stored in the calibration space EPROM.

24.4.2 Slope Reference Voltage Divider

The slope reference voltage divider circuit, consisting of a buffer amplifier and resistor divider, is connected to the internal bandgap reference producing two other voltage references called SREFHI and SREFLO (see Figure 24-3). SREFHI is nominally the same as the bandgap voltage, 1.2V, and SREFLO is nominally 0.13V. These reference voltages are available on two of the analog multiplexer channels. The Slope A/D module and firmware can measure the SREFHI and SREFLO voltages, and in conjunction with the KREF and KBG calibration data correct for the ADC's offset and slope errors.

See AN624 for further details.

Figure 24-3: Slope Reference Divider



24.5 Calibration Parameters

The Slope A/D module has several analog components. Like all CMOS circuitry the parametric values vary with process, temperature, voltage, and time. Devices have been designed to minimize the effect of these variations. In addition, each device, with the slope A/D module, is calibrated at factory test by measuring several key parameters and storing these values into EPROM at specified locations. The customer's application program may access this data and use it to mathematically compensate for device variations.

Collectively, these data values are referred to as calibration constants. The calibration constants are listed in Table 24-3. The 32-bit floating point representation has an exponent byte, and three bytes of mantissa. For information on floating point algorithms, refer to AN575.

Table 24-3: Calibration Constants

Parameter	Symbol	Number of Bytes	Representation of Value
A/D Slope reference ratio	KREF	4	32-bit Floating Point
Bandgap reference voltage	KBG	4	32-bit Floating Point

For additional information on using the calibration parameters see Application Note 624.

24.5.1 Using Calibration Data

The calibration constants should be used by the application firmware to obtain the best accuracy. KREF and KBG are used in A/D conversions.

24.5.2 Parameter Variation

Table 24-4 lists the "Maximum Parameter Variation" attainable when the calibration data is not used as well as the "Expected Parameter Variation with Calibration."

If the accuracies without calibration are adequate for the task at hand, no further calibrations of the module are necessary. If greater accuracy is needed, the calibration constants must be used.

Table 24-4: Parameter Variation

Symbol	Parameter	Maximum Variation Without Calibration	Achievable Variation with Calibration
KREF	A/D slope reference ratio	+/- 2.2%	+/- 0.13%
KBG	Bandgap reference voltage	+/- 4.2%	+/- 0.058%

24.5.3 Device Programming

24.5.3.1 Non-Windowed Parts

Non-windowed parts are programmed just like any PIC16CXXX processor. The calibration area is write-protected during factory calibration.

24.5.3.2 Windowed Parts

Caution:

Windowed parts must not be write-protected. If the parts are erased by ultraviolet light, the calibration parameters are lost and cannot be reprogrammed once the part has been write-protected.

Calibration data must be read out and saved before erasing a windowed part. There is no way to recreate these values, so if they are lost the part can no longer be calibrated.

24.6 Design Tips

Question 1: ***What are some recommended Capacitor types?***

Answer 1:

Polypropylene film capacitor is a good trade-off between cost, availability and performance

Question 2: ***Can you suggest some sources for Capacitors***

Answer 2:

A source is:

Southern Electronics Company
Telephone: (203) 876-7488

Question 3: ***I used the recommended capacitor and Programmable Current Source from Table 24-2, and my A/D input range does not match.***

Answer 3:

That table is meant to be a good starting point, but does not include variation that is the result of the device not operating at exactly 4 MHz; tolerance of the external capacitor and variations of the Programmable Current Source, due to process and application temperature.

A conversion on the Bandgap Reference can be used to judge how to adjust the Programmable Current Source Output to ensure proper A/D full scale conversions. Example code (routine ad_optimize, in P14_RV10.ASM) for this adjustment is available with the PICDEM-14A Demo Board, and may be also available on the Microchip web site.

Question 4: ***I am using the PIC14C000 which also has the on-chip Temperature sensor. The sensor results seem to be a little high.***

Answer 4:

This may be caused by self heating of the DIE. Self heating of the DIE may be caused by a few things, including:

- I/O sinking and/or sourcing significant amount of current
- Power dissipation of the device running
  (remember the PIC14C000 can operate in sleep mode)
- Package type due to junction to ambient temperature coefficient of package

For best results the power dissipation should be kept low. Calibration is performed with the device in a low power state.

Question 5: ***My A/D conversion results seem affected by the operation of high current components on my board. What can I do to minimize this?***

Answer 5:

The high current components on your board may cause the ground potential difference across the ground trace or ground plane. To minimize this effect, you should employ two system grounds on the application board. The first ground, analog ground, used for the reference analog signals (Slope A/D external capacitor ground, Resistor Divider ground, and etc.). No high current nor any digital power returns should go through this analog ground system.

The second ground, digital ground, is used for all other digital logic in the system. The application's digital logic will inject noise onto this ground. Proper grounding techniques should be used to minimize this noise.

These two grounds are connected at the PICmicro's ground pin. Ideally the two grounds are implemented using separate ground planes. In most cases, this can still be implemented on a two layer board. One layer is used for both ground systems, where the two planes are separated by a gap. The second layer is used as the trace layer.

24.7 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the Slope A/D are:

Title	Application Note #
PIC14C000 Calibration Parameters	AN621
PIC14C000 A/D Theory and Implementation	AN624
Lead Acid Battery Charger Implementation using the PIC14C000	AN626

24.8 Revision History

Revision A

This is the initial released revision of the Slope A/D module description.

Section 25. LCD

HIGHLIGHTS

This section of the manual contains the following major topics:

25.1 Introduction

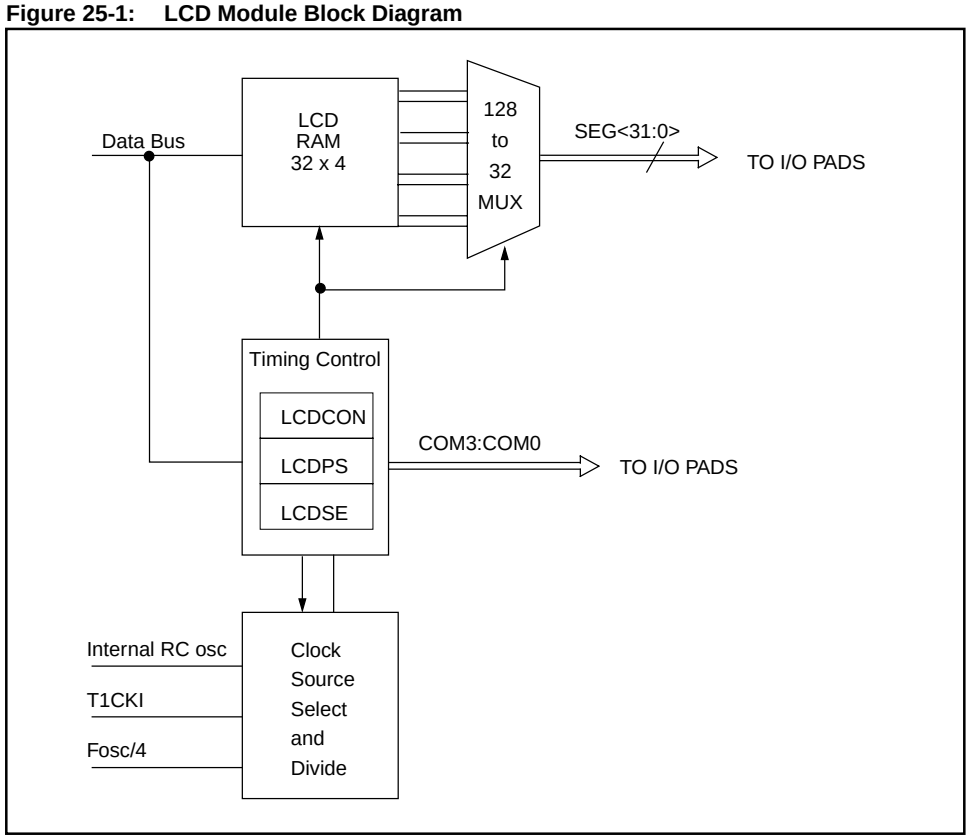
The LCD module generates the timing control to drive a static or multiplexed LCD panel, with support for up to 32 segments multiplexed with up to four commons. It also provides control of the LCD pixel data.

The interface to the module consists of three control registers (LCDCON, LCDSE, and LCDPS) used to define the timing requirements of the LCD panel and up to 16 LCD data registers (LCD00-LCD15) that represent the array of the pixel data. In normal operation, the control registers are configured to match the LCD panel being used. Primarily, the initialization information consists of selecting the number of commons and segments required by the LCD panel, and then specifying the LCD Frame clock rate to be used by the panel.

Once the module is initialized for the LCD panel, the individual bits of the LCD data registers are cleared/set to represent a turned-on pixel respectively.

Once the module is configured, the LCDEN bit (LCDCON<7>) is used to enable or disable the LCD module. The LCD panel can also operate during sleep by clearing the SLPEN bit (LCDCON<6>).

Figure 25-1: LCD Module Block Diagram



25.2 Control Register

Register 25-1: LCDCON Register

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
LCDEN	SLPEN	—	VGEN	CS1	CS0	LMUX1	LMUX0
bit 7							bit 0

bit 7 **LCDEN**: Module Drive Enable bit
1 = LCD drive enabled
0 = LCD drive disabled

bit 6 **SLPEN**: LCD Display Sleep Enable bit
1 = LCD module will stop operating during SLEEP
0 = LCD module will continue to display during SLEEP

bit 5 **Unimplemented**: Read as '0'

bit 4 **VGEN**: Voltage Generator Enable bit
1 = Internal LCD Voltage Generator Enabled, (powered-up)
0 = Internal LCD Voltage Generator powered-down, voltage is expected to be provided externally

bit 3:2 **CS1:CS0**: Clock Source Select bits
00 = Fosc/256
01 = T1CKI (Timer1)
1x = Internal RC oscillator

bit 1:0 **LMUX1:LMUX0**: Common Selection bits
Specifies the number of commons and the bias method

LMUX1:LMUX0	MULTIPLEX		BIAS	Max # of Segments
00	Static	(COM0)	Static	32
01	1/2	(COM0, 1)	1/3	31
10	1/3	(COM0, 1, 2)	1/3	30
11	1/4	(COM0, 1, 2, 3)	1/3	29

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

Register 25-2: LCDPS Register

U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	—	LP3	LP2	LP1	LP0
bit 7							bit 0

bit 7:4 **Unimplemented,** read as '0'

bit 3:0 **LP3:LP0**: Frame Clock Prescale Selection bits

LMUX1:LMUX0	Multiplex	Frame Frequency =
00	Static	Clock source / (128 * (LP3:LP0 + 1))
01	1/2	Clock source / (128 * (LP3:LP0 + 1))
10	1/3	Clock source / (96 * (LP3:LP0 + 1))
11	1/4	Clock source / (128 * (LP3:LP0 + 1))

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

Register 25-3: Generic LCDD (Pixel Data) Register Layout

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SEGs COMc	SEGs COMc	SEGs COMc	SEGs COMc	SEGs COMc	SEGs COMc	SEGs COMc	SEGs COMc
bit 7							bit 0

bit 7:0 **SEGsCOMc**: Pixel Data bit for segment s and common c

1 = Pixel on (dark)
0 = Pixel off (clear)

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

Register 25-4: LCDSE Register

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
SE29	SE27	SE20	SE16	SE12	SE9	SE5	SE0
bit 7							bit 0

bit 7 **SE29**: Pin Function Select bits for COM1/SEG31 - COM3/SEG29
1 = pins have LCD segment driver function
0 = pins have digital Input function

Note: The LMUX1:LMUX0 setting takes precedence over the SE29 bit, causing pins to become common drivers.

bit 6 **SE27**: Pin Function Select for SEG28 and SEG27
1 = pins have LCD segment driver function
0 = pins have digital Input function

bit 5 **SE20**: Pin Function Select bits for SEG26 - SEG20
1 = pins have LCD segment driver function
0 = pins have digital Input function

bit 4 **SE16**: Pin Function Select bits for SEG19 - SEG16
1 = pins have LCD segment driver function
0 = pins have digital Input function

bit 3 **SE12**: Pin Function Select bits for SEG15 - SEG12
1 = pins have LCD segment driver function
0 = pins have digital Input function

bit 2 **SE9**: Pin Function Select bits for SEG11 - SEG09
1 = pins have LCD segment driver function
0 = pins have digital Input function

bit 1 **SE5**: Pin Function Select bits for SEG08 - SEG05
1 = pins have LCD segment driver function
0 = pins have digital Input function

bit 0 **SE0**: Pin Function Select bits for SEG04 - SEG00
1 = pins have LCD segment driver function
0 = pins have digital I/O function

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

Note: On a Power-on Reset, the LCD pins are configured for LCD drive function.

25.3 LCD Timing

The LCD module has 3 possible clock source inputs and supports static, 1/2, 1/3, and 1/4 multiplexing.

25.3.1 Timing Clock Source Selection

The clock sources for the LCD timing generation are:

- Internal RC oscillator — used for device low frequency or sleep operation
- Timer1 oscillator — used for device low frequency or sleep operation
- System clock divided by 256

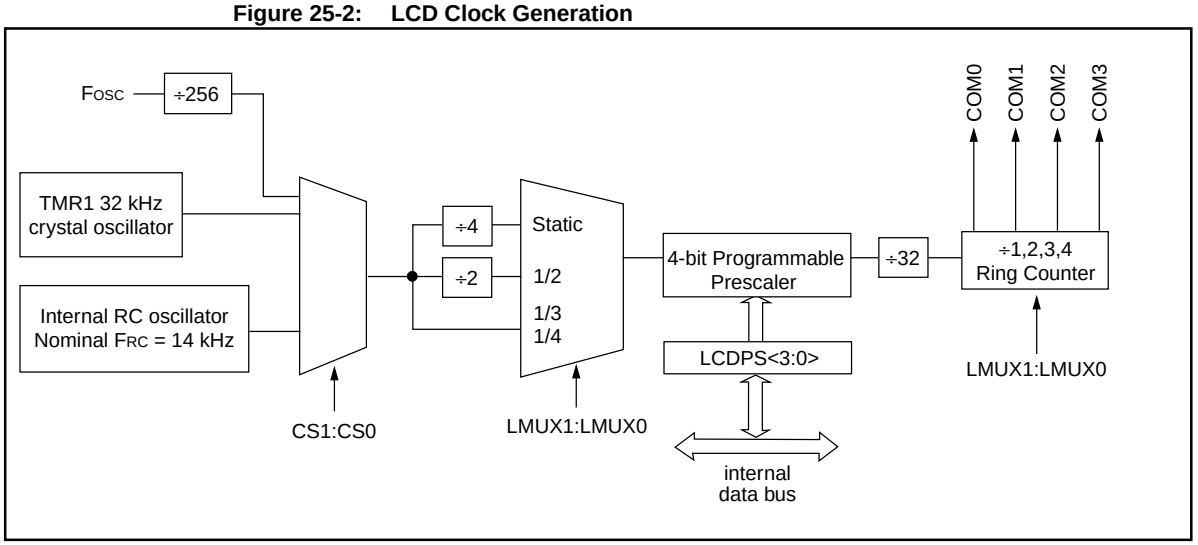
The first timing source is an internal RC oscillator which runs at a nominal frequency of 14 kHz. This oscillator provides a lower speed clock which may be used to continue running the LCD while the processor is in sleep. The RC oscillator will power-down when it is not selected or when the LCD module is disabled.

The second source is the Timer1 external oscillator. This oscillator provides a lower speed clock which may be used to continue running the LCD while the processor is in sleep. It is assumed that the frequency provided on this oscillator will be 32 kHz. To use the Timer1 oscillator as a LCD module clock source, it is only necessary to set the T1OSCEN (T1CON<3>) bit.

The third source is the system clock divided by 256. This divider ratio is chosen to provide about 32 kHz output when the external oscillator is 8 MHz. The divider is not programmable. Instead the LCDPS register is used to set the LCD frame clock rate.

The clock sources are selected with bits CS1:CS0 (LCDCON<3:2>). Refer to Figure 25-1 for details of the register programming.

Figure 25-2: LCD Clock Generation



25.3.2 Multiplex Timing Generation

The timing generation circuitry will generate 1 to 4 common's based on the display mode selected. The mode is specified by bits LMUX1:LMUX0 (LCDCON<1:0>). Table 25-1 shows the formulas for calculating the frame frequency.

Table 25-1: Frame Frequency Formulas

Multiplex	Frame Frequency =
Static	Clock source / (128 * (LP3:LP0 + 1))
1/2	Clock source / (128 * (LP3:LP0 + 1))
1/3	Clock source / (96 * (LP3:LP0 + 1))
1/4	Clock source / (128 * (LP3:LP0 + 1))

Table 25-2: Approximate Frame Frequency in Hz using Timer1 @ 32.768 kHz or Fosc @ 8 MHz

LP3:LP0	Static	1/2	1/3	1/4
2	85	85	114	85
3	64	64	85	64
4	51	51	68	51
5	43	43	57	43
6	37	37	49	37
7	32	32	43	32

Table 25-3: Approximate Frame Frequency in Hz using internal RC osc @ 14 kHz

LP3:LP0	Static	1/2	1/3	1/4
0	109	109	146	109
1	55	55	73	55
2	36	36	49	36
3	27	27	36	27

Figure 25-3: STATIC Waveforms

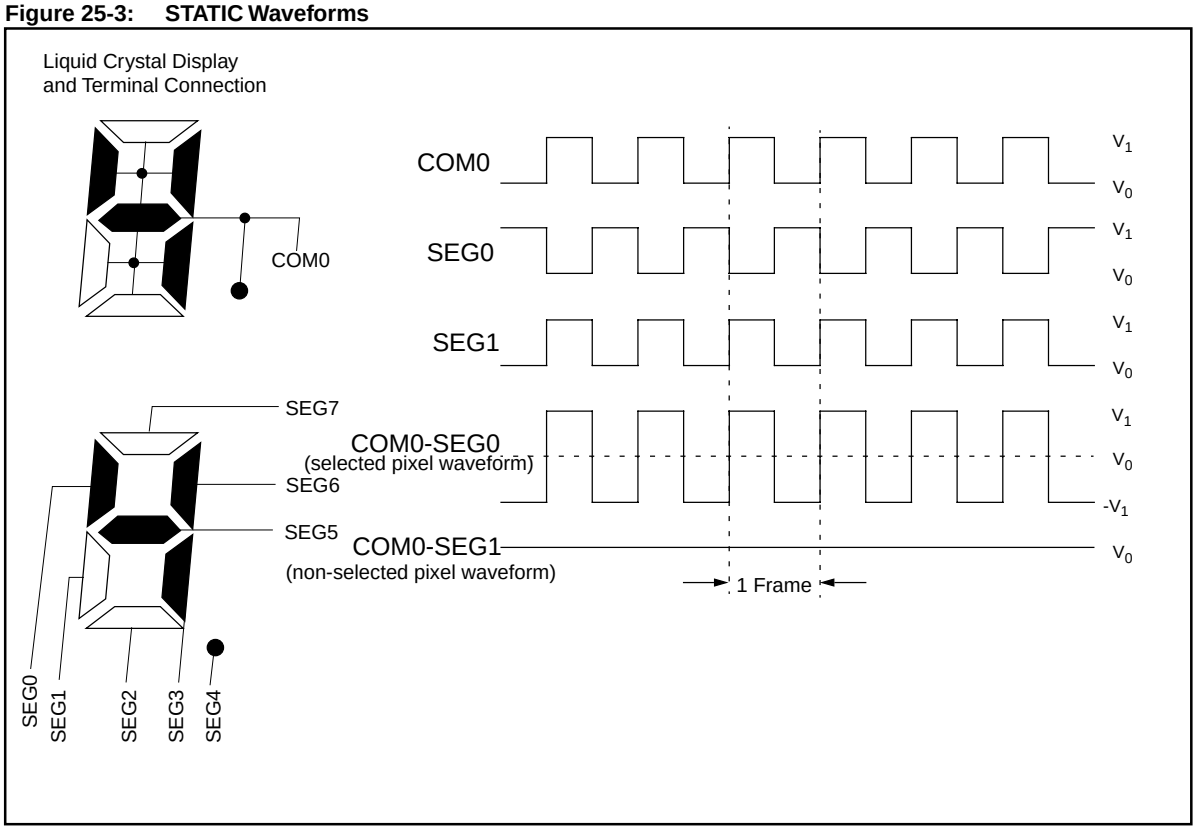


Figure 25-4: 1/2 MUX, 1/3 BIAS Waveform

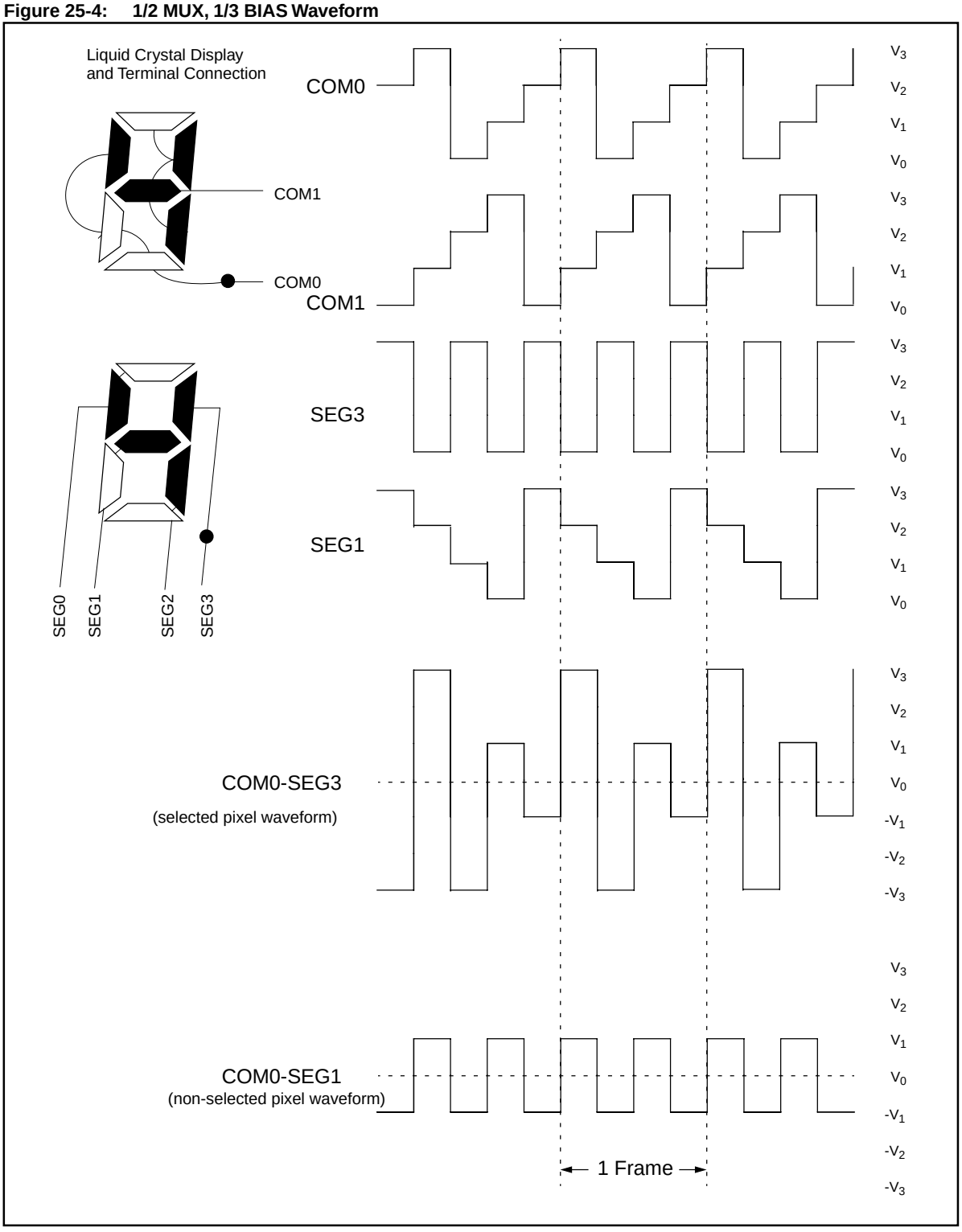


Figure 25-5: 1/3 MUX, 1/3 BIAS Waveform

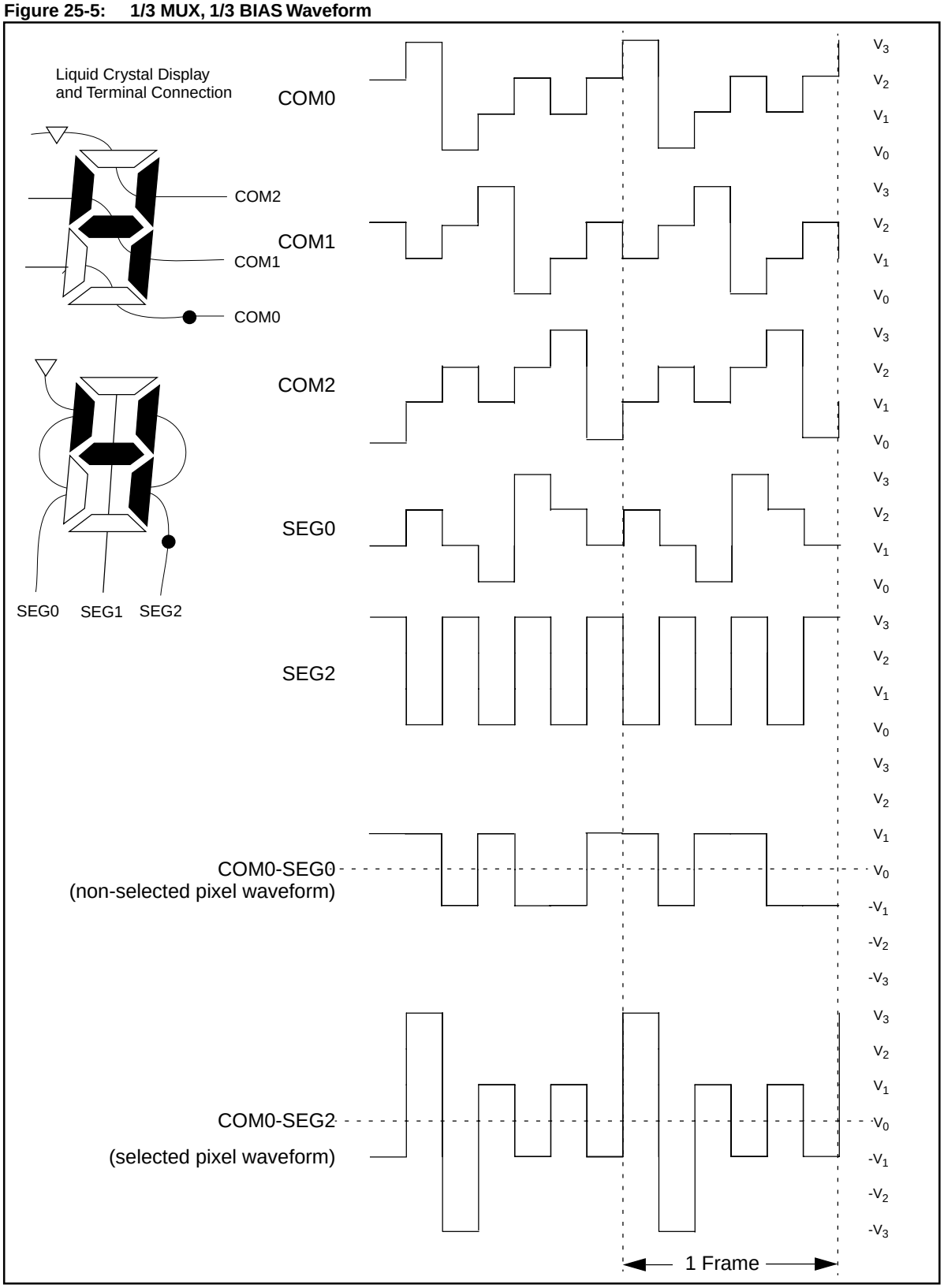
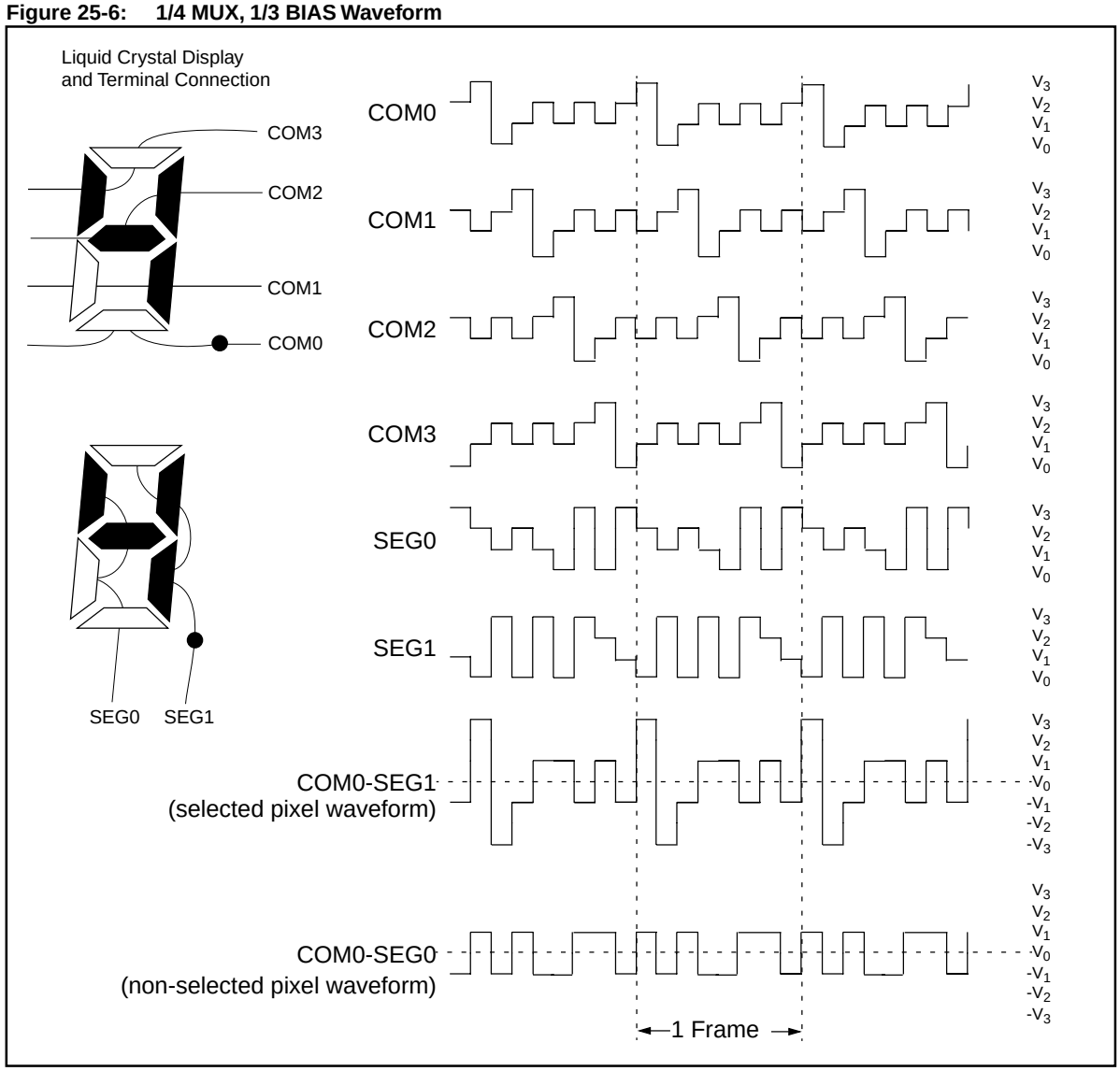


Figure 25-6: 1/4 MUX, 1/3 BIAS Waveform

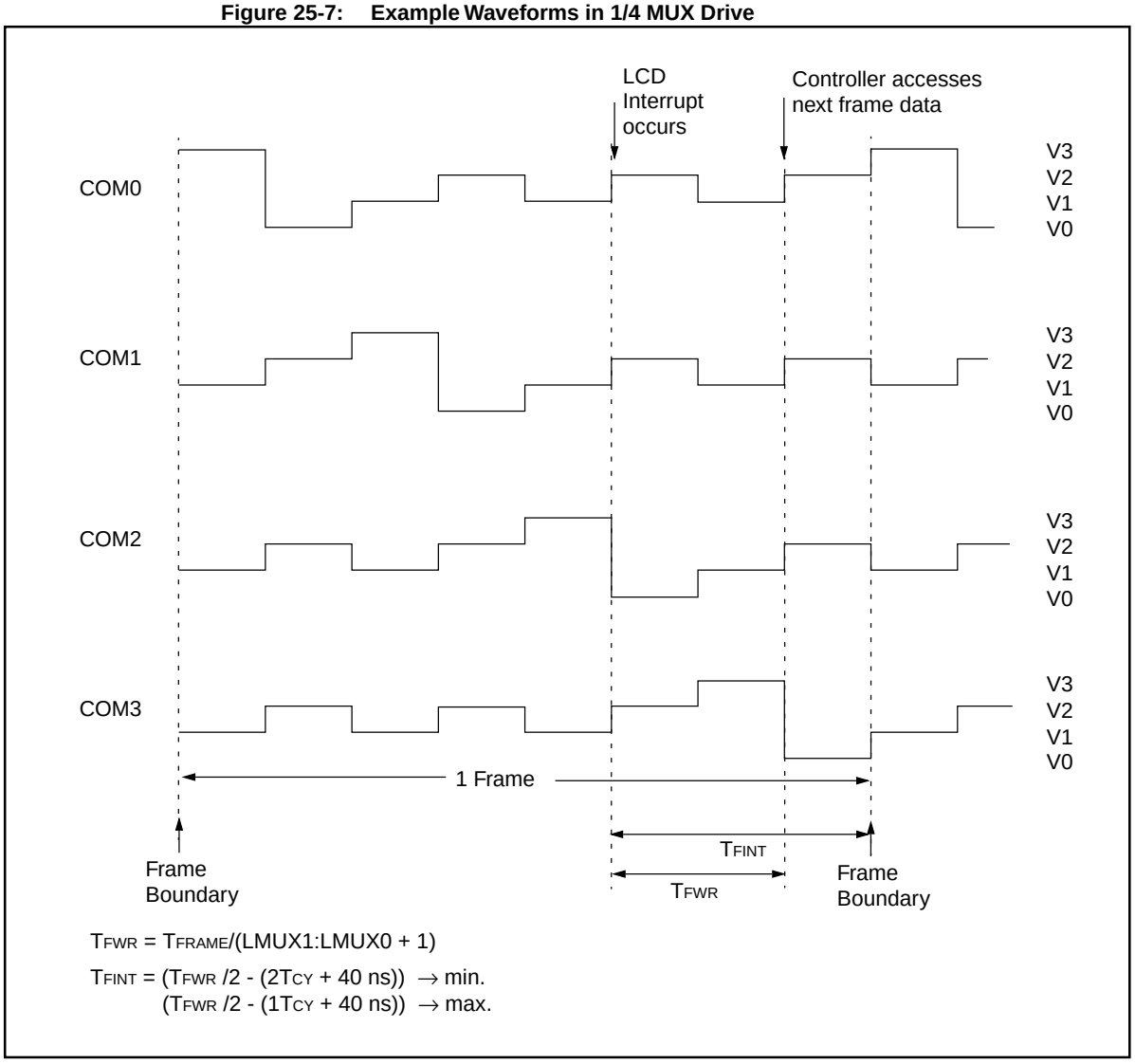


25.4 LCD Interrupts

The LCD timing generation provides an interrupt that defines the LCD frame timing. This interrupt can be used to coordinate the writing of the pixel data with the start of a new frame. Writing pixel data at the frame boundary allows a visually crisp transition of the image. This interrupt can also be used to synchronize external events to the LCD. For example, the interface to an external segment driver, such as a Microchip AY0438, can be synchronized for segment data update to the LCD frame.

A new frame is defined to begin at the leading edge of the COM0 common signal. The interrupt will be set immediately after the LCD controller completes accessing all pixel data required for a frame. This will occur at a certain fixed time before the frame boundary as shown in Figure 25-7. The LCD controller will begin to access data for the next frame within T_{FWR} after the interrupt.

Figure 25-7: Example Waveforms in 1/4 MUX Drive



$T_{FWR} = T_{FRAME}/(LMUX1:LMUX0 + 1)$

$T_{FINT} = (T_{FWR}/2 - (2T_{CY} + 40\ ns)) \rightarrow min.$

$(T_{FWR}/2 - (1T_{CY} + 40\ ns)) \rightarrow max.$

25.5 Pixel Control

25.5.1 LCDD (Pixel Data) Registers

The pixel registers contain bits which define the state of each pixel. Each bit defines one unique pixel.

Table 25-4 shows the correlation of each bit in the LCDD registers to the respective common and segment signals.

Any LCD pixel location not being used for display can be used as general purpose RAM.

Table 25-4: LCDD Registers

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
LCDD00	SEG07 COM0	SEG06 COM0	SEG05 COM0	SEG04 COM0	SEG03 COM0	SEG02 COM0	SEG01 COM0	SEG00 COM0	xxxx xxxx	xxxx xxxx
LCDD01	SEG15 COM0	SEG14 COM0	SEG13 COM0	SEG12 COM0	SEG11 COM0	SEG10 COM0	SEG09 COM0	SEG08 COM0	xxxx xxxx	xxxx xxxx
LCDD02	SEG23 COM0	SEG22 COM0	SEG21 COM0	SEG20 COM0	SEG19 COM0	SEG18 COM0	SEG17 COM0	SEG16 COM0	xxxx xxxx	xxxx xxxx
LCDD03	SEG31 COM0	SEG30 COM0	SEG29 COM0	SEG28 COM0	SEG27 COM0	SEG26 COM0	SEG25 COM0	SEG24 COM0	xxxx xxxx	xxxx xxxx
LCDD04	SEG07 COM1	SEG06 COM1	SEG05 COM1	SEG04 COM1	SEG03 COM1	SEG02 COM1	SEG01 COM1	SEG00 COM1	xxxx xxxx	xxxx xxxx
LCDD05	SEG15 COM1	SEG14 COM1	SEG13 COM1	SEG12 COM1	SEG11 COM1	SEG10 COM1	SEG09 COM1	SEG08 COM1	xxxx xxxx	xxxx xxxx
LCDD06	SEG23 COM1	SEG22 COM1	SEG21 COM1	SEG20 COM1	SEG19 COM1	SEG18 COM1	SEG17 COM1	SEG16 COM1	xxxx xxxx	xxxx xxxx
LCDD07	SEG31 COM**1** (1)	SEG30 COM1	SEG29 COM1	SEG28 COM1	SEG27 COM1	SEG26 COM1	SEG25 COM1	SEG24 COM1	xxxx xxxx	xxxx xxxx
LCDD08	SEG07 COM2	SEG06 COM2	SEG05 COM2	SEG04 COM2	SEG03 COM2	SEG02 COM2	SEG01 COM2	SEG00 COM2	xxxx xxxx	xxxx xxxx
LCDD09	SEG15 COM2	SEG14 COM2	SEG13 COM2	SEG12 COM2	SEG11 COM2	SEG10 COM2	SEG09 COM2	SEG08 COM2	xxxx xxxx	xxxx xxxx
LCDD10	SEG23 COM2	SEG22 COM2	SEG21 COM2	SEG20 COM2	SEG19 COM2	SEG18 COM2	SEG17 COM2	SEG16 COM2	xxxx xxxx	xxxx xxxx
LCDD11	SEG31 COM**2** (1)	SEG30 COM**2** (1)	SEG29 COM2	SEG28 COM2	SEG27 COM2	SEG26 COM2	SEG25 COM2	SEG24 COM2	xxxx xxxx	xxxx xxxx
LCDD12	SEG07 COM3	SEG06 COM3	SEG05 COM3	SEG04 COM3	SEG03 COM3	SEG02 COM3	SEG01 COM3	SEG00 COM3	xxxx xxxx	xxxx xxxx
LCDD13	SEG15 COM3	SEG14 COM3	SEG13 COM3	SEG12 COM3	SEG11 COM3	SEG10 COM3	SEG09 COM3	SEG08 COM3	xxxx xxxx	xxxx xxxx
LCDD14	SEG23 COM3	SEG22 COM3	SEG21 COM3	SEG20 COM3	SEG19 COM3	SEG18 COM3	SEG17 COM3	SEG16 COM3	xxxx xxxx	xxxx xxxx
LCDD15	SEG31 COM**3** (1)	SEG30 COM**3** (1)	SEG29 COM**3** (1)	SEG28 COM30	SEG27 COM3	SEG26 COM30	SEG25 COM3	SEG24 COM3	xxxx xxxx	xxxx xxxx

Note 1: These pixels do not display, but can be used as general purpose RAM.

25.5.2 Segment Enables

The LCDSE register is used to select the pin function for groups of pins. The selection allows each group of pins to operate as either LCD drivers or digital only pins. To configure the pins as a digital port, the corresponding bits in the LCDSE register must be cleared.

If the pin is a digital input the corresponding TRIS bit controls the data direction. Any bit set in the LCDSE register overrides any bit settings in the corresponding TRIS register.

Note 1: On a Power-on Reset, the LCD pins are configured as LCD drivers.

Note 2: The LMUX1:LMUX0 bits take precedence over the LCDSE bit settings for pins RD7, RD6 and RD5.

Example 25-1: Static MUX with 32 Segments

```
    BCF   STATUS,RP0   ; Select Bank2
    BSF   STATUS,RP1   ;
    BCF   LCDCON,LMUX1 ; Select Static MUX
    BCF   LCDCON,LMUX0 ;
    MOVLW 0xFF         ; Make PortD,E,F,G LCD pins
    MOVWF LCDSE        ; configure rest of LCD
```

Example 25-2: 1/3 MUX with 13 Segments

```
    BCF   STATUS,RP0   ; Select Bank2
    BSF   STATUS,RP1   ;
    BSF   LCDCON,LMUX1 ; Select 1/3 MUX
    BCF   LCDCON,LMUX0 ;
    MOVLW 0x87         ; Make PORTD<7:0> & PORTE<6:0> LCD pins
    MOVWF LCDSE        ; configure rest of LCD
```

25.6 Voltage Generation

There are two methods for LCD voltage generation, internal charge pump, or external resistor ladder.

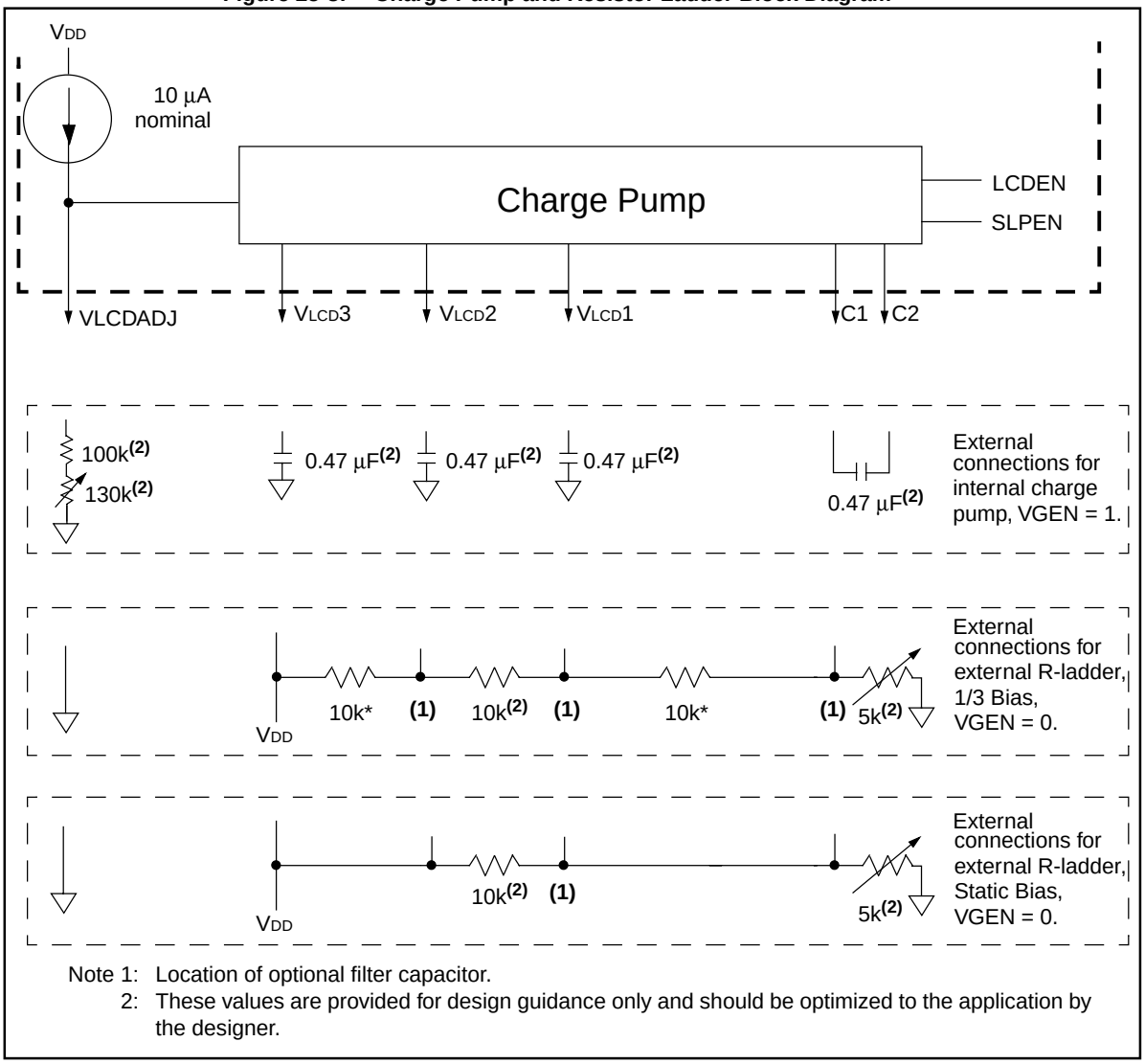
25.6.1 Charge Pump

The LCD charge pump is shown in Figure 25-8. The 1.0V - 2.3V regulator will establish a stable base voltage from the varying battery voltage. This regulator is adjustable through the range by connecting a variable external resistor from VLCDADJ to ground. The potentiometer provides contrast adjustment for the LCD. This base voltage is connected to $V_{LCD}1$ on the charge pump. The charge pump boosts $V_{LCD}1$ into $V_{LCD}2 = 2 * V_{LCD}1$ and $V_{LCD}3 = 3 * V_{LCD}1$. When the charge pump is not operating, $V_{LCD}3$ will be internally tied to V_{DD}. See the Electrical Specifications section for charge pump capacitor and potentiometer values.

25.6.2 External R-Ladder

The LCD module can also use an external resistor ladder (R-Ladder) to generate the LCD voltages. Figure 25-8 shows external connections for static and 1/3 bias. The VGEN (LCDCON<4>) bit must be cleared to use an external R-Ladder.

Figure 25-8: Charge Pump and Resistor Ladder Block Diagram



Note 1: Location of optional filter capacitor.
2: These values are provided for design guidance only and should be optimized to the application by the designer.

25.7 Operation During Sleep

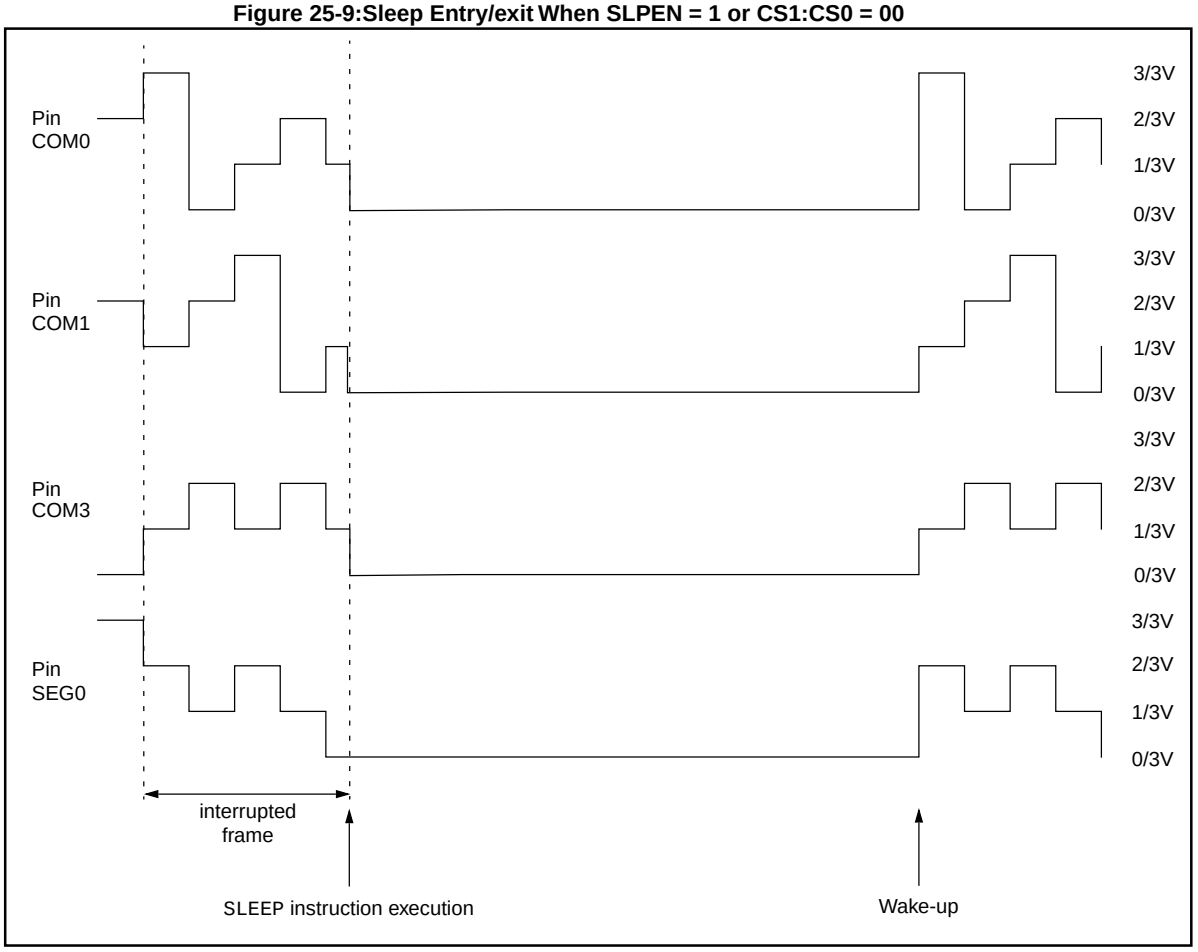
The LCD module can operate during sleep. The selection is controlled by bit SLPEN (LCDCON<6>). Setting the SLPEN bit allows the LCD module to go to sleep. Clearing the SLPEN bit allows the module to continue to operate during sleep.

If a SLEEP instruction is executed and SLPEN = '1', the LCD module will cease all functions and go into a very low current consumption mode. The module will stop operation immediately and drive the minimum LCD voltage on both segment and common lines. Figure 25-9 shows this operation. To ensure that the LCD completes the frame, the SLEEP instruction should be executed immediately after a LCD frame boundary. The LCD interrupt can be used to determine the frame boundary. See **25.4 "LCD Interrupts"** for the formulas to calculate the delay.

If a SLEEP instruction is executed and SLPEN = '0', the module will continue to display the current contents of the LCDD registers. To allow the module to continue operation while in sleep, the clock source must be either the internal RC oscillator or Timer1 external oscillator. While in sleep, the LCD data cannot be changed. The LCD module current consumption will not decrease in this mode, however the overall consumption of the device will be lower due to shutdown of the core and other peripheral functions.

Note:	The internal RC oscillator or external Timer1 oscillator must be used to operate the LCD module during sleep.

Figure 25-9: Sleep Entry/exit When SLPEN = 1 or CS1:CS0 = 00



25.8 Effects of a Reset

The LCD module is disabled, but the LCD pins are configured as LCD drivers. This ensures that the microcontroller does not damage the LCD glass by accidently having a DC voltage across a segment.

25.9 Configuring the LCD Module

The following is the sequence of steps to follow to configure the LCD module.

1. Select the frame clock prescale using the LP3:LP0 bits (LCDPS<3:0>).
2. Configure the appropriate pins to function as segment drivers using the LCDSE register.
3. Configure the LCD module for the following using the LCDCON register.
   - Multiplex mode and Bias, selected by the LMUX1:LMUX0 bits
   - Timing source, selected by the CS1:CS0 bits
   - Voltage generation, enabled by the VGEN bit
   - Sleep mode operation, enabled by the SLPEN bit
4. Write initial values to pixel data registers, LCDD00 through LCDD15.
5. Clear LCD interrupt flag bit, LCDIF, and if desired, enable the interrupt by setting the LCDIE bit.
6. Enable the LCD module, by setting the LCDEN bit (LCDCON<7>).

25.10 Discrimination Ratio

Discrimination ratio is a way to calculate the contrast levels that a panel can achieve. The first example is a static waveform from Figure 25-3. The voltages V_1 and V_0 will be assigned values of 1 and 0. The next step is to construct an equation for one frame to help visualize the DC and RMS voltages present on an individual pixel that is ON and OFF. The rest of the following shows the calculation of the DC, RMS, and Discrimination Ratio.

Example 25-3: Discrimination Ratio Calculation for Static MUX

$$\text{COMx - SEGx [ON]} = 1 - 1, \quad V_{DC} = 0$$

$$\text{COMx - SEGx [OFF]} = 0 + 0, \quad V_{DC} = 0$$

$$V_{RMS}\,[ON] = \Delta V \sqrt{\frac{(1)^2 + (-1)^2}{2}} = 1\Delta V$$

$$V_{RMS}\,[OFF] = \Delta V \sqrt{\frac{(0)^2 + (0)^2}{2}} = 0\Delta V$$

$$D = \frac{V_{RMS}\,[ON]}{V_{RMS}\,[OFF]} = \frac{1\Delta V}{0\Delta V} = \infty$$

See Figure 25-3 for Static waveform.

The next example is for Figure 25-6 which is a 1/4 MUX, 1/3 BIAS waveform. For this example, the values 3, 2, 1 and 0 will be assigned to V_3, V_2, V_1, and V_0 respectively. The frame equation, DC voltage, RMS voltage and discrimination ratio calculations are shown in Example 25-4.

Example 25-4: Discrimination Ratio Calculation 1/4 MUX

COM0 - SEGx [ON] = 3 - 3 + 1 - 1 + 1 - 1 + 1 - 1 $V_{DC} = 0$
COM0 - SEGx [OFF] = 1 - 1 - 1 + 1 - 1 + 1 - 1 + 1 $V_{DC} = 0$

$$V_{RMS}[ON] = \Delta V \sqrt{\frac{(3)^2 + (-3)^2 + (1)^2 + (-1)^2 + (1)^2 + (-1)^2 + (1)^2 + (-1)^2}{8}} = \sqrt{3}\,\Delta V$$

$$V_{RMS}[OFF] = \Delta V \sqrt{\frac{(1)^2 + (-1)^2 + (-1)^2 + (1)^2 + (-1)^2 + (1)^2 + (-1)^2 + (1)^2}{8}} = \Delta V$$

$$D = \frac{V_{RMS}[ON]}{V_{RMS}[OFF]} = \frac{\sqrt{3}\,\Delta V}{1\,\Delta V} = 1.732$$

Note: Refer to Figure 25-6

As shown in these examples, static displays have excellent contrast. The higher the multiplex ratio of the LCD, the lower the discrimination ratio, and therefore, the lower the contrast of the display.

Table 25-5 shows the V_{OFF}, V_{ON} and discrimination ratios of the various combinations of MUX and BIAS.

As the multiplex of the LCD panel increases, the discrimination ratio decreases. The contrast of the panel will also decrease, so to provide better contrast the LCD voltages must be increased to provide greater separation between each level.

Table 25-5: Discrimination Ratio vs. MUX and Bias

	1/3 BIAS		
	V_{OFF}	V_{ON}	D
STATIC	0	1	∞
1/2 MUX	0.333	0.745	2.236
1/3 MUX	0.333	0.638	1.915
1/4 MUX	0.333	0.577	1.732

25.11 LCD Voltage Generation

Among the many ways to generate LCD voltage, two methods stand out above the crowd:

- resistor ladder
- charge pump.

The resistor ladder method, shown in Figure 25-10, is most commonly used for higher VCC voltages. This method uses inexpensive resistors to create the multi-level LCD voltages. Regardless of the number of pixels that are energized the current remains constant. The voltage at point V3 is typically tied to VCC, either internally or externally.

The resistance values are determined by two factors: display quality and power consumption. Display quality is a function of the LCD drive waveforms. Since the LCD panel is a capacitive load, the waveform is distorted due to the charging and discharging currents. This distortion can be reduced by decreasing the value of resistance. However, this change increases the power consumption due to the increased current now flowing through the resistors. As the LCD panel increases in size, the resistance value must be decreased to maintain the image quality of the display.

Sometimes the addition of parallel capacitors to the resistance can reduce the distortion caused by charging/discharging currents. The capacitors act as charge storage to provide current as the display waveform transitions. In general, R is 1 kΩ to 50 kΩ and the potentiometer is 5 kΩ to 200 kΩ.

Figure 25-10: Resistor Ladder

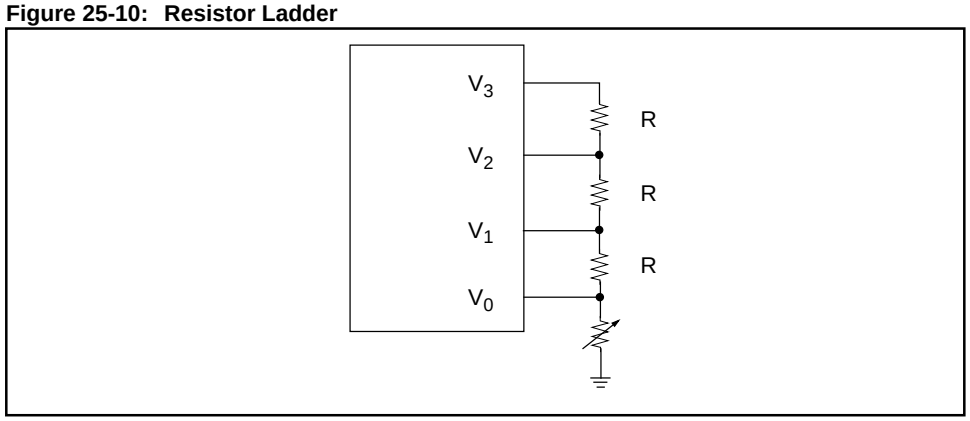
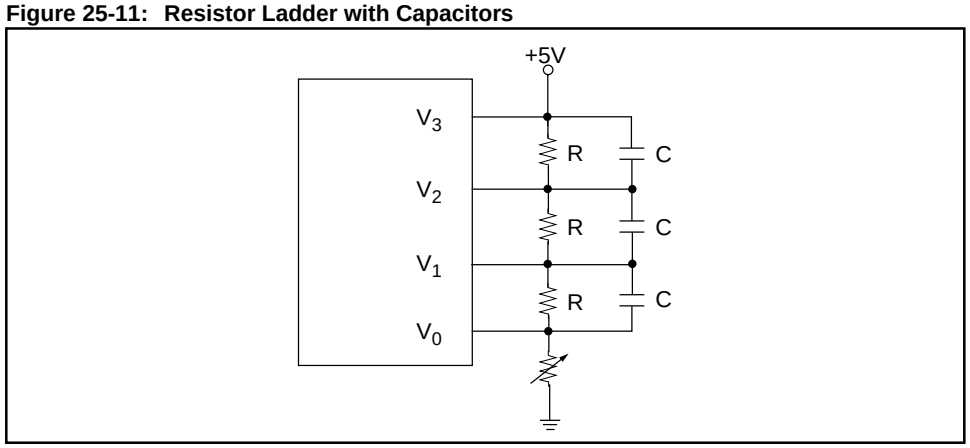
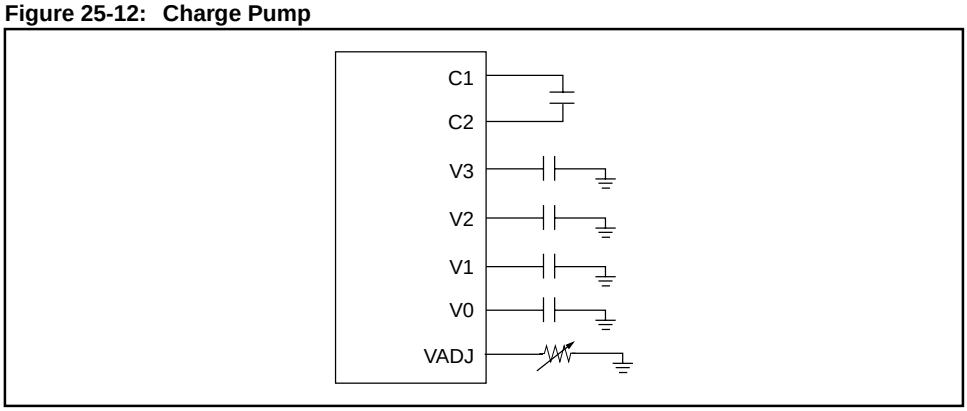


Figure 25-11: Resistor Ladder with Capacitors



A charge pump is ideal for low voltage battery operation because the V_{DD} voltage can be boosted up to drive the LCD panel. The charge pump requires a charging capacitor and filter capacitor for each of the LCD voltages as seen in Figure 25-12. These capacitors are typically low leakage types such as polyester, polypropylene, or polystyrene material. Another feature that makes the charge pump ideal for battery applications is that the current consumption is proportional to the number of pixels that are energized.

Figure 25-12: Charge Pump



25.12 Contrast

Although contrast is heavily dependent on the light source available and the multiplex mode, it also varies with the LCD voltage levels. As previously seen, a potentiometer is used to control the contrast of the LCD panel. The potentiometer sets the separation between each of the LCD voltages. The larger the separation, the better the contrast achievable.

25.13 LCD Glass

The characteristics of the LCD glass vary depending on the materials used. Appendix B gives a list of some LCD manufacturers. Please contact them for the characteristics of your desired glass.

25.14 Initialization

Example 25-5 shows the code for initializing the LCD module with all segments cleared.

Example 25-5: LCD Initialization Code

```
    BCF    PIR1,LCDIF     ; Clear LCD interrupt flag
    BCF    STATUS,RP0     ; Go to Bank2
    BSF    STATUS,RP1
    MOVLW  0x06           ; Set frame freq to ~37Hz
    MOVWF  LCDPS
    MOVLW  0xff           ; Make all pin functions LCD drivers
    MOVWF  LCDSE
    MOVLW  0x17           ; Drive during SLEEP, Charge pump enabled
    MOVWF  LCDCON         ; Timer1 clock source, 1/4 MUX
    CLRF   LCDD00         ; Clear all data registers to turn
    CLRF   LCDD01         ;    all pixels off
    CLRF   LCDD02
    CLRF   LCDD03
    CLRF   LCDD04
    CLRF   LCDD05
    CLRF   LCDD06
    CLRF   LCDD07
    CLRF   LCDD08
    CLRF   LCDD09
    CLRF   LCDD10
    CLRF   LCDD11
    CLRF   LCDD12
    CLRF   LCDD13
    CLRF   LCDD14
    CLRF   LCDD15
    BSF    PIE1,LCDIE      ; Enable LCD interrupts
    BSF    LCDCON,LCDEN    ; Enable LCD Module
    BCF    STATUS,RP1      ; Go to Bank0
```

25.15 Design Tips

Question 1: ***I'm trying to use some of the LCD pins as inputs.***

Answer 1:

Ensure that you have the control bits in the LCDSE properly configured, since these bits override the TRIS bits.

Question 2: ***My LCD panel is flickering.***

Answer 2:

Your frame frequency may be too low. The frame frequency can be changed in the LCDPS register.

Question 3: ***The LCD segments are not very visible.***

Answer 3:

This may be due to misadjusted LCD voltage, some possibilities include:

1. If you are using the R-ladder, try different values of R, vary the R-ladder potentiometer. The VLCDADJ pin should be connected to ground.
2. If you are using the charge pump, adjust the resistance value on the VLCDADJ pin.

25.16 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the LCD drivers are:

Title	Application Note #
Yet Another Clock Using the PIC16C92X	AN649
LCD Fundamentals Using PIC16C92x Microcontrollers	AN658
PICDEM3 Demo Board User's Guide	DS51079

25.17 Revision History

Revision A

This is the initial released revision of the LCD module description.

Section 26. Watchdog Timer and Sleep Mode

HIGHLIGHTS

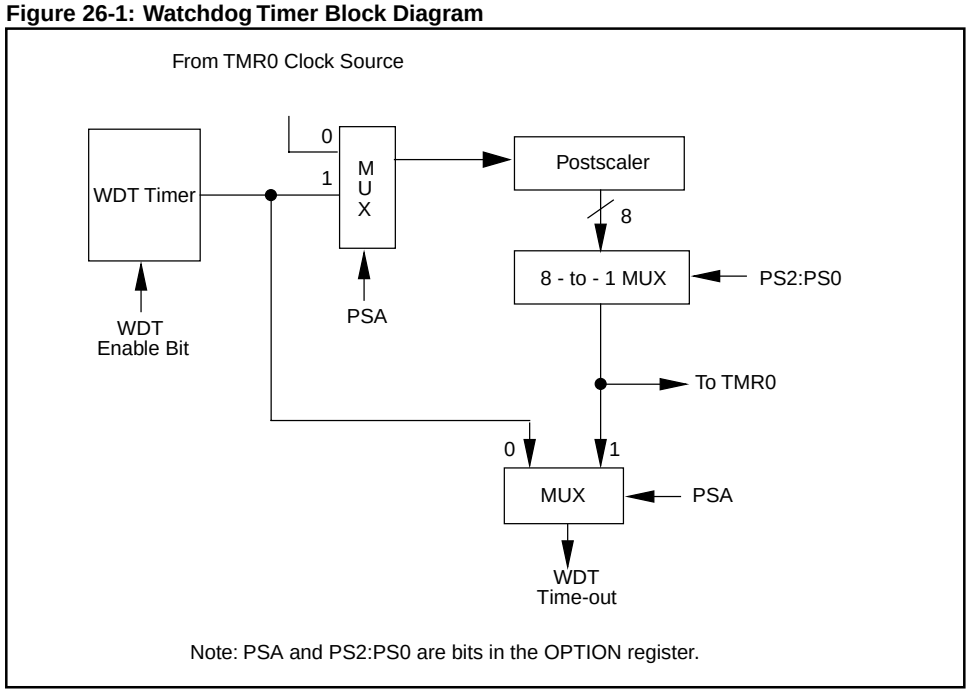
This section of the manual contains the following major topics:

26.1 Introduction

The Watchdog Timer (WDT) is a free running on-chip RC oscillator which does not require any external components. The block diagram is shown in Figure 26-1. This RC oscillator is separate from the device RC oscillator of the OSC1/CLKIN pin. This means that the WDT will run, even if the clock on the OSC1 and OSC2 pins has been stopped, for example, by execution of a SLEEP instruction.

The Watchdog Timer (WDT) is enabled/disabled by a device configuration bit. If the WDT is enabled, software execution may not disable this function.

Figure 26-1: Watchdog Timer Block Diagram



Note: PSA and PS2:PS0 are bits in the OPTION register.

26.2 Control Register

The OPTION_REG register is a readable and writable register which contains various control bits to configure the TMR0 prescaler/WDT postscaler, the External INT Interrupt, TMR0, and the weak pull-ups on PORTB.

Note:	To achieve a 1:1 prescaler assignment for the TMR0 register, assign the prescaler to the Watchdog Timer.

Register 26-1: OPTION_REG Register

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$ (1)	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$ (1):** Weak Pull-up Enable bit
1 = Weak pull-ups are disabled
0 = Weak pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of INT pin
0 = Interrupt on falling edge of INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2:0 **PS2:PS0**: TMR0 Prescaler/WDT Postscaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

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

Note 1:	Some devices call this bit $\overline{\text{GPPU}}$. Devices that have the $\overline{\text{RBPU}}$ bit, have the weak pull-ups on PORTB, while devices that have the $\overline{\text{GPPU}}$ have the weak pull-ups on the GP Port.

26.3 Watchdog Timer (WDT) Operation

During normal operation, a WDT time-out generates a device RESET. If the device is in SLEEP mode, a WDT time-out causes the device to wake-up and continue with normal operation, this is known as a WDT wake-up. The WDT can be permanently disabled by clearing the WDTE configuration bit.

The postscaler assignment is fully under software control, i.e., it can be changed “on the fly” during program execution.

Note:	To avoid an unintended device RESET, the following instruction sequence (shown in Example 26-1) must be executed when changing the prescaler assignment from Timer0 to the postscaler of the WDT. This sequence must be followed even if the WDT is disabled.

In Example 26-1, the first modification of the OPTION_REG does not need to be included if the final desired prescaler is other then 1:1. If the final prescaler value is 1:1, then a temporary prescale value is set (other than 1:1), and the final prescale value is set in the last modification of the OPTION_REG. This sequence must be followed since the value in the TMR0 prescaler is unknown, and is being used as the WDT postscaler. If the OPTION_REG is changed without this code sequence, the time before a WDT reset is unknown.

Example 26-1: Changing Prescaler (Timer0→WDT)

```
    BSF     STATUS, RP0    ; Bank1
    MOVLW   B’xx0x0xxx’    ; Select clock source and postscale value
    MOVWF   OPTION_REG     ;   other than 1:1
    BCF     STATUS, RP0    ; Bank0
    CLRF    TMR0           ; Clear TMR0 & Prescaler
    BSF     STATUS, RP0    ; Bank1
    MOVLW   B’xxxx1xxx’    ; Select WDT, do not change prescale value
    MOVWF   OPTION_REG     ;
    CLRWDT                 ; Clears WDT
    MOVLW   b'xxxx1xxx'    ; Select new prescale value and WDT
    MOVWF   OPTION_REG     ;
    BCF     STATUS, RP0    ; Bank0
```

To change prescaler from the WDT to the Timer0 module use the sequence shown in Example 26-2.

Example 26-2: Changing Prescaler (WDT→Timer0)

```
    CLRWDT                 ; Clear WDT and postscaler
    BSF     STATUS, RP0    ; Bank1
    MOVLW   b'xxxx0xxx'    ; Select TMR0, new prescale
    MOVWF   OPTION_REG     ; value and clock source
    BCF     STATUS, RP0    ; Bank0
```


26.3.1 WDT Period

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

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

The $\overline{TO}$ bit in the STATUS register will be cleared upon a Watchdog Timer time-out (WDT Reset and WDT wake-up).

26.3.2 WDT Programming Considerations

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

Note:	When the postscaler is assigned to the WDT, always execute a CLRWDT instruction before changing the postscale value, otherwise a WDT reset may occur.

Table 26-1: Summary of Watchdog Timer Registers

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Config. bits	MPEEN	BODEN	CP1	CP0	$\overline{PWRTE}$	WDTE	FOSC1	FOSC0
OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

Legend: Shaded cells are not used by the Watchdog Timer.

26.4 SLEEP (Power-Down) Mode

Sleep (Power-down) mode is a mode where the device is placed in it's lowest current consumption state. The device oscillator is turned off, so no system clocks are occurring in the device. Sleep mode is entered by executing a SLEEP instruction.

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

For lowest current consumption in this mode, all I/O pins should be either at V_{DD}, or V_{SS}, with no external circuitry drawing current from the I/O pin and the modules that are specified to have a delta sleep current should be disabled. 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 also be at V_{DD} or V_{SS} for lowest current consumption. The contribution from on-chip pull-ups on PORTB should be considered.

The $\overline{MCLR}$ pin must be at a logic high level (V_{IHMC}).

Some features of the device that consume a delta sleep current are enabled / disabled by device configuration bits. These include the Watchdog Timer (WDT) and Brown-out Reset (BOR) circuitry modules.

26.4.1 Wake-up from SLEEP

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

1. Any device reset.
2. Watchdog Timer Wake-up (if WDT was enabled).
3. Any peripheral module which can set its interrupt flag while in sleep, such as:
   - External INT pin
   - Change on port pin
   - Comparators
   - A/D
   - Timer1
   - LCD
   - SSP
   - Capture

The first event will reset the device upon wake-up. However the latter two events will wake the device and then resume program execution. The $\overline{TO}$ and $\overline{PD}$ bits in the STATUS register can be used to determine the cause of device reset. The $\overline{PD}$ bit, which is set on power-up is cleared when SLEEP is invoked. The $\overline{TO}$ bit is cleared if WDT wake-up occurred.

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 GIE bit. If the GIE bit is clear (disabled), the device continues execution at the instruction after the SLEEP instruction. If the GIE bit is set (enabled), the device executes the instruction after the SLEEP instruction and then branches to the interrupt address (0004h). In cases where the execution of the instruction following SLEEP is not desirable, the user should have an NOP after the SLEEP instruction.

26.4.2 Wake-up Using Interrupts

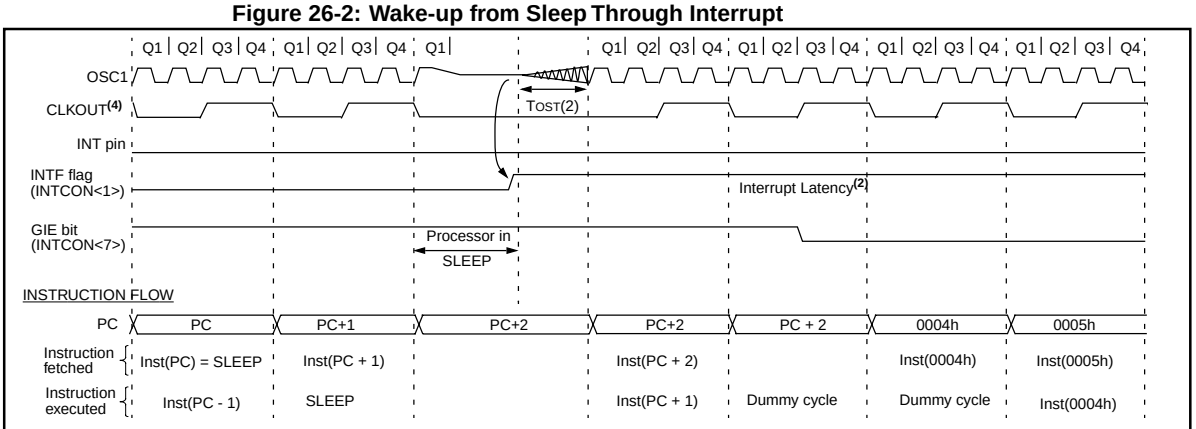
When interrupts are globally disabled (GIE cleared) and any interrupt source has both its interrupt enable bit and interrupt flag set, one of the following events will occur:

- If the interrupt occurs before the execution of a SLEEP instruction, the SLEEP instruction will complete as an NOP. Therefore, the WDT and WDT postscaler will not be cleared, the $\overline{TO}$ bit will not be set and $\overline{PD}$ bit will not be cleared.
- If the interrupt occurs during or after the execution of a SLEEP instruction, the device will immediately wake-up from sleep. The SLEEP instruction will be completely executed before the wake-up. Therefore, the WDT and WDT postscaler will be cleared, the $\overline{TO}$ bit will be set and the $\overline{PD}$ bit will be cleared.

Even if the flag bits were checked before executing a SLEEP instruction, it may be possible for flag bits to become set before the SLEEP instruction completes. To determine whether a SLEEP instruction executed, test the $\overline{PD}$ bit. If the $\overline{PD}$ bit is set, the SLEEP instruction was executed as an NOP.

To ensure that the WDT is clear, a CLRWDT instruction should be executed before a SLEEP instruction.

Figure 26-2: Wake-up from Sleep Through Interrupt



Note 1: XT, HS or LP oscillator mode assumed.
2: TOST = 1024TOSC (drawing not to scale) This delay will not be there for RC osc mode.
3: GIE = '1' assumed. In this case after wake- up, the processor jumps to the interrupt routine. If GIE = '0', execution will continue in-line.
4: CLKOUT is not available in these osc modes, but shown here for timing reference.

26.5 Initialization

No initialization code at this time.

26.6 Design Tips

Question 1: ***My system voltage drops and then returns to the specified device voltage range. The device is not operating correctly and the WDT does not reset and return the device to proper operation.***

Answer 1:

The WDT was not designed to be a recovery from a brown-out condition. It was designed to recover from errant software operation (the device remaining in the specified operating ranges). If your system can be subjected to brown-outs, either the on-chip brown-out circuitry should be enabled or an external brown-out circuit should be implemented.

Question 2: ***Device resets even though I do the CLRWDT instruction in my loop.***

Answer 2:

Make sure that the loop with the CLRWDT instruction meets the minimum specification of the WDT (not the typical).

Question 3: ***Device never gets out of resets.***

Answer 3:

On power-up, you must take into account the Oscillator Start-up time (Tost). Sometimes it helps to put the CLRWDT instruction at the beginning of the loop, since this start-up time may be variable.

26.7 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the WDT and Sleep Mode are:

Title	Application Note #
Power-up Trouble Shooting	AN607

26.8 Revision History

Revision A

This is the initial released revision of the Watchdog Timer and Sleep mode description.

Section 27. Device Configuration Bits

HIGHLIGHTS

This section of the manual contains the following major topics:

27.1 Introduction

The device configuration bits allow each user to customize certain aspects of the device to the needs of the application. When the device powers up, the state of these bits determines the modes that the device uses. Subsection **27.2 "Configuration Word Bits"** discusses the configuration bits, and the modes that they can be configured to. These bits are mapped in program memory location 2007h. This location is not accessible during normal device operation (can be accessed only during programming mode).

The configuration bits can be programmed (read as '0') or left unprogrammed (read as '1') to select various device configurations. The ability to change these settings once they have been programmed depends on the memory technology and the package type.

For Read Only Memory (ROM) devices, these bits are specified at time of ROM code submittal and once the device is masked may not be changed for those devices (would require a new mask code).

For One Time Programmable (OTP) devices, once these bits are programmed ('0'), they may not be changed.

For windowed EPROM devices, once these bits are programmed ('0'), the device must be UV erased to return the configuration word to the erased state. UV erasing the device also erases the program memory.

For Flash devices, these bits may be erased and reprogrammed.

Note:	Microchip does not recommend code protecting windowed devices.

Section 27. Device Configuration Bits

Section **27.2** is forced to the next page for formatting purposes.

27.2 Configuration Word Bits

These configuration bits specify some of the modes of the device, and are programmed by a device programmer, or by using the In-Circuit Serial Programming (ICSP) feature of the midrange devices. The device is not able to read the values of these bits, and there placement is automatically taken care of when you select the device in you device programmer. For additional information, please refer to the Programming Specification of the Device.

Note 1: Always ensure that your device programmer has the same device selected as you are programming.

Note 2: Microchip recommends that the desired configuration bit states be embedded in to the application source code. This is easily done in the MPASM assembler by the use of the CONFIG directive. See Subsection **27.2.1 "MPASM's CONFIG Directive."**

CP1:CP0: Code Protection bits

11 = Code protection off
10 = See device data sheet
01 = See device data sheet
00 = All memory is code protected

Note: Some devices may use more or less bits to determine the code protect. Presently there are also some devices that use only one bit (CP0). For these devices the bit description is:
1 = Code protection off
0 = Code protection on

DP: Data EEPROM Memory Code Protection bit

1 = Code protection off
0 = Data EEPROM Memory is code protected

Note: This bit is used when a device with ROM program memory also has Data EEPROM memory.

BODEN: Brown-out Reset Enable bit

1 = BOR enabled
0 = BOR disabled

Note: Enabling Brown-out Reset automatically enables the Power-up Timer (PWRT) regardless of the value of bit $\overline{\text{PWRTE}}$. Ensure the Power-up Timer is enabled anytime Brown-out Reset is enabled.

$\overline{\text{PWRTE}}$: Power-up Timer Enable bit

1 = PWRT disabled
0 = PWRT enabled

Note 1: Enabling Brown-out Reset automatically enables Power-up Timer (PWRT) regardless of the value of bit $\overline{\text{PWRTE}}$. Ensure the Power-up Timer is enabled anytime Brown-out Reset is enabled.

Note 2: Some original PICmicros have the polarity of this bit reversed.

Note 3:

MCLRE: $\overline{\text{MCLR}}$ Pin Function Select bit

1 = Pin's function is $\overline{\text{MCLR}}$

0 = Pin's function is as a digital I/O. $\overline{\text{MCLR}}$ is internally tied to V_{DD}.

WDTE: Watchdog Timer Enable bit

1 = WDT enabled
0 = WDT disabled

FOSC1:FOSC0: Oscillator Selection bits

11 = RC oscillator
10 = HS oscillator
01 = XT oscillator
00 = LP oscillator

FOSC2:FOSC0: Oscillator Selection bits

111 = EXTRC oscillator, with CLKOUT
110 = EXTRC oscillator
101 = INTRC oscillator, with CLKOUT
100 = INTRC oscillator
011 = Reserved
010 = HS oscillator
001 = XT oscillator
000 = LP oscillator

Unimplemented: Read as '1'

Legend		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value at POR reset		u = Unchanged from programmed state

Note: The bit position of the configuration bits is device dependent. Please refer to the device programming specification for bit placement. The use of a Microchip device programmer does not require you to know the bit locations.

27.2.1 MPASM's CONFIG Directive

Microchip's assembler, MPASM, has a nice feature that allows you to specify, in the source code file, the selected states of the configuration bits for this program. This ensures that when programming a device for an application the required configuration is also programmed. This minimizes the risk of programming the wrong device configuration, and wondering why it no longer works in the application.

Example 27-1 show a template for using the CONFIG directive.

Example 27-1:Using the CONFIG Directive, a Source File Template

```
     LIST   p = p16C77      ; List Directive,
;    Revision History
;
     #INCLUDE    <P16C77.INC>    ; Microchip Device Header File
;
     #INCLUDE    <MY_STD.MAC>    ; File which includes my standard macros
     #INCLUDE    <APP.MAC>       ; File which includes macros specific
                                 ;   to this application
;
; Specify Device Configuration Bits
;
     __CONFIG    _XT_OSC & _PWRTE_ON & _BODEN_OFF & _CP_OFF & _WDT_ON
;
     org   0x00            ; Start of Program Memory
RESET_ADDR   :             ; First instruction to execute after a reset


     end
```

The Symbols that are currently in the Microchip Device Header files that make using the CONFIG directive straight forward are shown in Table 27-1. For the symbols available for your device, please refer to that device's Microchip Include file.

Note:	As long as the correct device is specified (in the LIST and INCLUDE file directives), the correct polarity of all bits is ensured.

Table 27-1: __CONFIG Directive Symbols (From Microchip Header Files)

Feature	SYMBOLS
Oscillators	_RC_OSC
	_EXTRC_OSC
	_EXTRC_OSC_CLKOUT
	_EXTRC_OSC_NOCLKOUT
	_INTRC_OSC
	_INTRC_OSC_CLKOUT
	_INTRC_OSC_NOCLKOUT
	_LP_OSC
	_XT_OSC
	_HS_OSC
Watch Dog Timer	_WDT_ON
	_WDT_OFF
Power-up Timer	_PWRTE_ON
	_PWRTE_OFF
Brown-out Reset	_BODEN_ON
	_BODEN_OFF
Master Clear Enable	_MCLRE_ON
	_MCLRE_OFF
Code Protect	_CP_ALL
	_CP_ON
	_CP_75
	_CP_50
	_CP_OFF
Code Protect Data EEPROM	_DP_ON
	_DP_OFF
Code Protect Calibration Space	_CPC_ON
	_CPC_OFF

Note 1: Not all configuration bit symbols may be available on any one device. Please refer to the MIcrochip include file of that device for available symbols.

27.3 Program Verification/Code Protection

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

Note:	Microchip does not recommend code protecting windowed devices.

27.3.1 ROM Devices

When a ROM device also has Data EEPROM memory, an additional code protect configuration bit may be implemented. The program memory configuration bit is submitted as part of the ROM code submittal. The Data EEPROM memory code protect configuration bit will be an EEPROM bit. When ROM devices complete testing, the EEPROM data memory code protect bit will be programmed to the same state as the program memory code protect bit. That is data EEPROM code protect is off, when program memory code protect is off, and data EEPROM code protect is on for all other selections.

In applications where the device is code protected and the data EEPROM needs to be programmed before the application can be released, the data EEPROM memory must have the entire data EEPROM memory erased. The device programming specification details the steps to do this. Microchip device programmers implement the specified sequence. Once this sequence is complete, the Data EEPROM memory code protect is disabled. This allows the desired data to be programmed into the device. After programming the data EEPROM memory array, the data EEPROM memory code protect configuration bit should be programmed as desired.

27.4 ID Locations

Four memory locations (2000h - 2003h) are designated as ID locations where the user can store checksum or other code-identification numbers. These locations are not accessible during normal execution but are readable and writable during program/verify. It is recommended that only the 4 least significant bits of the ID location are used.

27.5 Design Tips

Question 1: ***I have a JW device and I can no longer program it (reads scrambled data or all '0's). What's wrong with the device?***

Answer 1:

Nothing. You probably code protected the device. If this is the case, the device is no longer usable. See Subsection **27.3 "Program Verification/Code Protection"** for more details.

Question 2: ***In converting from a PIC16C74 to a PIC16C74A, my application no longer works.***

Answer 2:

1. Did you re-assemble the source file specifying the PIC16C74A in the INCLUDE file and LIST directives. The use of the CONFIG directive is highly recommended.
2. On the device programmer, did you specify the PIC16C74A, and were all the configuration bits as desired?

Question 3: ***When I erase the device, the program memory is blank but the configuration word is not yet erased.***

Answer 3:

That is by design. Also remember that Microchip does not recommend code protecting windowed devices.

27.6 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to Configuration Word are:

Title	Application Note #
No related Application Notes at this time.	

27.7 Revision History

Revision A

This is the initial released revision of the Configuration Word description.

Section 28. In-Circuit Serial Programming™ (ICSP™)

HIGHLIGHTS

This section of the manual contains the following major topics:

28.1 Introduction

All midrange devices can be In-Circuit Serial Programmed (ICSP™) 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.

In-Circuit Serial Programming (ICSP™) is a great way to reduce your inventory overhead and time-to-market for your product. By assembling your product with a blank Microchip microcontroller (MCU), you can stock one design. When an order has been placed, these units can be programmed with the latest revision of firmware, tested, and shipped in a very short time. This method also reduces scrapped inventory due to old firmware revisions. This type of manufacturing system can also facilitate quick turnarounds on custom orders for your product.

Most people would think to use ICSP with PICmicro™ OTP MCUs only on an assembly line where the device is programmed once. However, there is a method by which an OTP device can be programmed several times depending on the size of the firmware. This method, explained later, provides a way to field upgrade your firmware in a way similar to EEPROM- or Flash-based devices.

28.2 Entering In-Circuit Serial Programming Mode

The device is placed into a program/verify mode by holding the RB6 and RB7 pins low while raising the $\overline{MCLR}$ (V_{PP}) pin from V_{IL} to V_{IHH} (see programming specification) and having V_{DD} at the programming voltage. RB6 becomes the programming clock and RB7 becomes the programming data. Both RB6 and RB7 are Schmitt Trigger inputs in this mode, and when RB7 is driving data it is a CMOS output driver.

After reset, to place the device into programming/verify mode, the program counter (PC) is at location 00h. A 6-bit command is then supplied to the device. Some commands then specify that 14-bits of program data are then supplied to or read from the device, depending on if the command was a load or a read. For complete details of serial programming, please refer to the device specific Programming Specifications.

During the In-Circuit Serial Programming Mode, the WDT circuitry is disabled from generating a device reset.

28.3 Application Circuit

The application circuit must be designed to allow all the programming signals to be directly connected to the PICmicro MCU. Figure 28-1 shows a typical circuit that is a starting point for when designing with ICSP. The application must compensate for the following issues:

- Isolation of the $\overline{MCLR}$/VPP pin from the rest of the circuit
- Loading of pins RB6 and RB7
- Capacitance on each of the VDD, $\overline{MCLR}$/VPP, RB6, and RB7 pins
- Minimum and maximum operating voltage for VDD
- PICmicro Oscillator
- Interface to the programmer

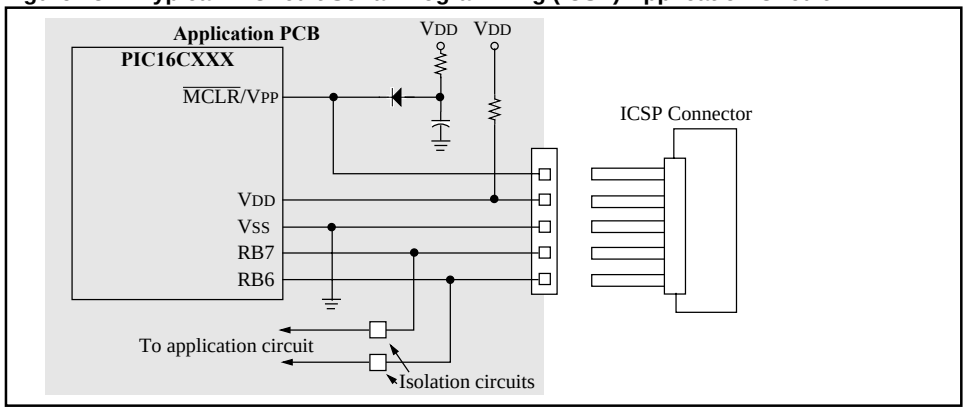
The $\overline{MCLR}$/VPP pin is normally connected to an RC circuit. The pull-up resistor is tied to VDD and a capacitor is tied to ground. This circuit can affect the operation of ICSP depending on the size of the capacitor since the VPP voltage must be isolated from the rest of the circuit (in most cases a resistor is not capable of isolating the circuit). It is, therefore, recommended that the circuit in Figure 28-1 be used when an RC is connected to $\overline{MCLR}$/VPP. The diode should be a Schottky-type device. Another issue with $\overline{MCLR}$/VPP is that when the PICmicro device is programmed, this pin is driven to approximately 13V and also to ground. Therefore, the application circuit must be isolated from this voltage provided by the programmer.

Pins RB6 and RB7 are used by the PICmicro for serial programming. RB6 is the clock line and RB7 is the data line. RB6 is driven by the programmer. RB7 is a bi-directional pin that is driven by the programmer when programming, and driven by the PICmicro when verifying. These pins must be isolated from the rest of the application circuit so as not to affect the signals during programming. You must take into consideration the output impedance of the programmer when isolating RB6 and RB7 from the rest of the circuit. This isolation circuit must account for RB6 being an input on the PICmicro, and for RB7 being bi-directional (can be driven by both the PICmicro and the programmer). For instance, PRO MATE® II has an output impedance of 1kΩ. If the design permits, these pins should not be used by the application. This is not the case with most applications so it is recommended that the designer evaluate whether these signals need to be buffered. As a designer, you must consider what type of circuitry is connected to RB6 and RB7 and then make a decision on how to isolate these pins. Figure 28-1 does not show any circuitry to isolate RB6 and RB7 on the application circuit because this is very application dependent.

To simplify this interface the optimal usage of these I/O in the application are (in order):

1. Do not use RB6/RB7 so they are dedicated to ICSP.
2. Use these pins as outputs with minimal loading on signal line.
3. Isolation circuitry so that these signals can be driven to the ICSP specifications.

Figure 28-1: Typical In-Circuit Serial Programming (ICSP) Application Circuit



The total capacitance on the programming pins affects the rise rates of these signals as they are driven out of the programmer. Typical circuits use several hundred microfarads of capacitance on VDD which helps to dampen noise and ripple. However, this capacitance requires a fairly strong driver in the programmer to meet the rise rate timings for VDD. Most programmers are designed to simply program the PICmicro itself and don't have strong enough drivers to power the application circuit. One solution is to use a driver board between the programmer and the application circuit. The driver board requires a separate power supply that is capable of driving the VPP and VDD pins with the correct rise rates and should also provide enough current to power the application circuit. RB6 and RB7 are not buffered on this schematic but may require buffering depending upon the application. A sample driver board schematic is shown in Figure 28-2.

Note:	The driver board design MUST be tested in the user's application to determine the effects of the application circuit on the programming signals timing. Changes may be required if the application places a significant load on VDD, VPP, RB6 OR RB7.

The Microchip programming specification states that the device should be programmed at 5V. Special considerations must be made if your application circuit operates at 3V only. These considerations may include totally isolating the PICmicro during programming. The other issue is that the device must be verified at the minimum and maximum voltages at which the application circuit will be operating. For instance, a battery operated system may operate from three 1.5V cells giving an operating voltage range of 2.7V to 4.5V. The programmer must program the device at 5V and must verify the program memory contents at both 2.7V and 4.5V to ensure that proper programming margins have been achieved. This ensures the PICmicro operation over the voltage range of the system.

The final issue deals with the oscillator circuit on the application board. The voltage on $\overline{\text{MCLR}}$/VPP must rise to the specified program mode entry voltage before the device executes any code. The crystal modes available on the PICmicro are not affected by this issue because the Oscillator Start-up Timer waits for 1024 oscillations before any code is executed. However, RC oscillators do not require any start-up time and, therefore, the Oscillator Start-up Timer is not used. The programmer must drive $\overline{\text{MCLR}}$/VPP to the program mode entry voltage before the RC oscillator toggles four times. If the RC oscillator toggles four or more times, the program counter will be incremented to some value X. Now when the device enters programming mode, the program counter will not be zero and the programmer will start programming your code at an offset of X. There are several alternatives that can compensate for a slow rise rate on $\overline{\text{MCLR}}$/VPP. The first method would be to not populate the R, program the device, and then insert the R. The other method would be to have the programming interface drive the OSC1 pin of the PICmicro to ground while programming. This will prevent any oscillations from occurring during programming.

Now all that is left is how to connect the application circuit to the programmer. This depends a lot on the programming environment and will be discussed in that section.

28.4 Programmer

The second consideration is the programmer. PIC16CXXX MCUs only use serial programming and therefore all programmers supporting these devices will support ICSP. One issue with the programmer is the drive capability. As discussed before, it must be able to provide the specified rise rates on the ICSP signals and also provide enough current to power the application circuit. Figure 28-2 shows an example driver board. This driver schematic does not show any buffer circuitry for RB6 and RB7. It is recommended that an evaluation be performed to determine if buffering is required. Another issue with the programmer is what VDD levels are used to verify the memory contents of the PICmicro. For instance, the PRO MATE II verifies program memory at the minimum and maximum VDD levels for the specified device and is therefore considered a production quality programmer. On the other hand, the PICSTART® Plus only verifies at 5V and is for prototyping use only. The Microchip programming specifications state that the program memory contents should be verified at both the minimum and maximum VDD levels that the application circuit will be operating. This implies that the application circuit must be able to handle the varying VDD voltages.

There are also several third party programmers that are available. You should select a programmer based on the features it has and how it fits into your programming environment. The *Microchip Development Systems Ordering Guide* (DS30177) provides detailed information on all our development tools. The *Microchip Third Party Guide* (DS00104) provides information on all of our third party tool developers. Please consult these two references when selecting a programmer. Many options exist including serial or parallel PC host connection, stand-alone operation, and single or gang programmers. Some of the third party developers include Advanced Transdata Corporation, BP Microsystems, Data I/O, Emulation Technology and Logical Devices.

28.5 Programming Environment

The programming environment will affect the type of programmer used, the programmer cable length, and the application circuit interface. Some programmers are well suited for a manual assembly line while others are desirable for an automated assembly line. You may want to choose a gang programmer to program multiple systems at a time.

The physical distance between the programmer and the application circuit affects the load capacitance on each of the programming signals. This will directly affect the drive strength needed to provide the correct signal rise rates and current. This programming cable must also be as short as possible and properly terminated and shielded, or the programming signals may be corrupted by ringing or noise.

Finally, the application circuit interface to the programmer depends on the size constraints of the application circuit itself and the assembly line. A simple header can be used to interface the application circuit to the programmer. This might be more desirable for a manual assembly line where a technician plugs the programmer cable into the board. A different method is the use of spring loaded test pins (commonly referred to as pogo pins). The application circuit has pads on the board for each of the programming signals. Then there is a fixture that has pogo pins in the same configuration as the pads on the board. The application circuit or fixture is moved into position such that the pogo pins come into contact with the board. This method might be more suitable for an automated assembly line.

After taking into consideration the issues with the application circuit, the programmer, and the programming environment, anyone can build a high quality, reliable manufacturing line based on ICSP.

28.6 Other Benefits

ICSP provides other benefits, such as calibration and serialization. If program memory permits, it would be cheaper and more reliable to store calibration constants in program memory instead of using an external serial EEPROM. For example, if your system has a thermistor which can vary from one system to another, storing some calibration information in a table format allows the microcontroller to compensate (in software) for external component tolerances. System cost can be reduced without affecting the required performance of the system by using software calibration techniques. But how does this relate to ICSP? The PICmicro has already been programmed with firmware that performs a calibration cycle. The calibration data is transferred to a calibration fixture. When all calibration data has been transferred, the fixture places the PICmicro in programming mode and programs the PICmicro with the calibration data. Application note *AN656, In-Circuit Serial Programming of Calibration Parameters Using a PICmicro Microcontroller*, shows exactly how to implement this type of calibration data programming.

The other benefit of ICSP is serialization. Each individual system can be programmed with a unique or random serial number. One such application of a unique serial number would be for security systems. A typical system might use DIP switches to set the serial number. Instead, this number can be burned into program memory, thus reducing the overall system cost and lowering the risk of tampering.

28.7 Field Programming of PICmicro OTP MCUs

An OTP device is not normally capable of being reprogrammed, but the PICmicro architecture gives you this flexibility provided the size of your firmware is at least half that of the desired device and the device is not code protected. If your target device does not have enough program memory, Microchip provides a wide spectrum of devices from 0.5K to 8K program memory with the same set of peripheral features that will help meet the criteria.

The PIC16CXXX microcontrollers have two vectors, reset and interrupt, at locations 0x0000 and 0x0004. When the PICmicro encounters a reset or interrupt condition, the code located at one of these two locations in program memory is executed. The first listing of Example 28-2 shows the code that is first programmed into the PICmicro. The second listing of Example 28-2 shows the code that is programmed into the PICmicro for the second time.

Example 28-2 shows that to program the PICmicro a second time the memory location 0x0000, originally `goto Main` (0x2808), is reprogrammed to all 0's which happens to be a `NOP` instruction. This location cannot be reprogrammed to the new opcode (0x2860) because the bits that are 0's cannot be reprogrammed to 1's, only bits that are 1's can be reprogrammed to 0's. The next memory location 0x0001 was originally blank (all 1's) and now becomes a `goto Main` (0x2860). When a reset condition occurs, the PICmicro executes the instruction at location 0x0000 which is the `NOP`, a completely benign instruction, and then executes the `goto Main` to start the execution of code. The example also shows that all program memory locations after 0x005A are blank in the original program so that the second time the PICmicro is programmed, the revised code can be programmed at these locations. The same descriptions can be given for the interrupt vector at location 0x0004.

This method changes slightly for PICmicros with >2K words of program memory. Each of the `goto Main` and `goto ISR` instructions are replaced by the following code segment is Example 28-1 due to paging on devices with >2K words of program memory.

Example 28-1: Crossing Program Memory Pages

```
movlw <page>            movlw <page>
movwf PCLATH            movwf PCLATH
goto  Main              goto  ISR
```

Now your one-time programmable PICmicro is exhibiting EEPROM- or Flash-like qualities.

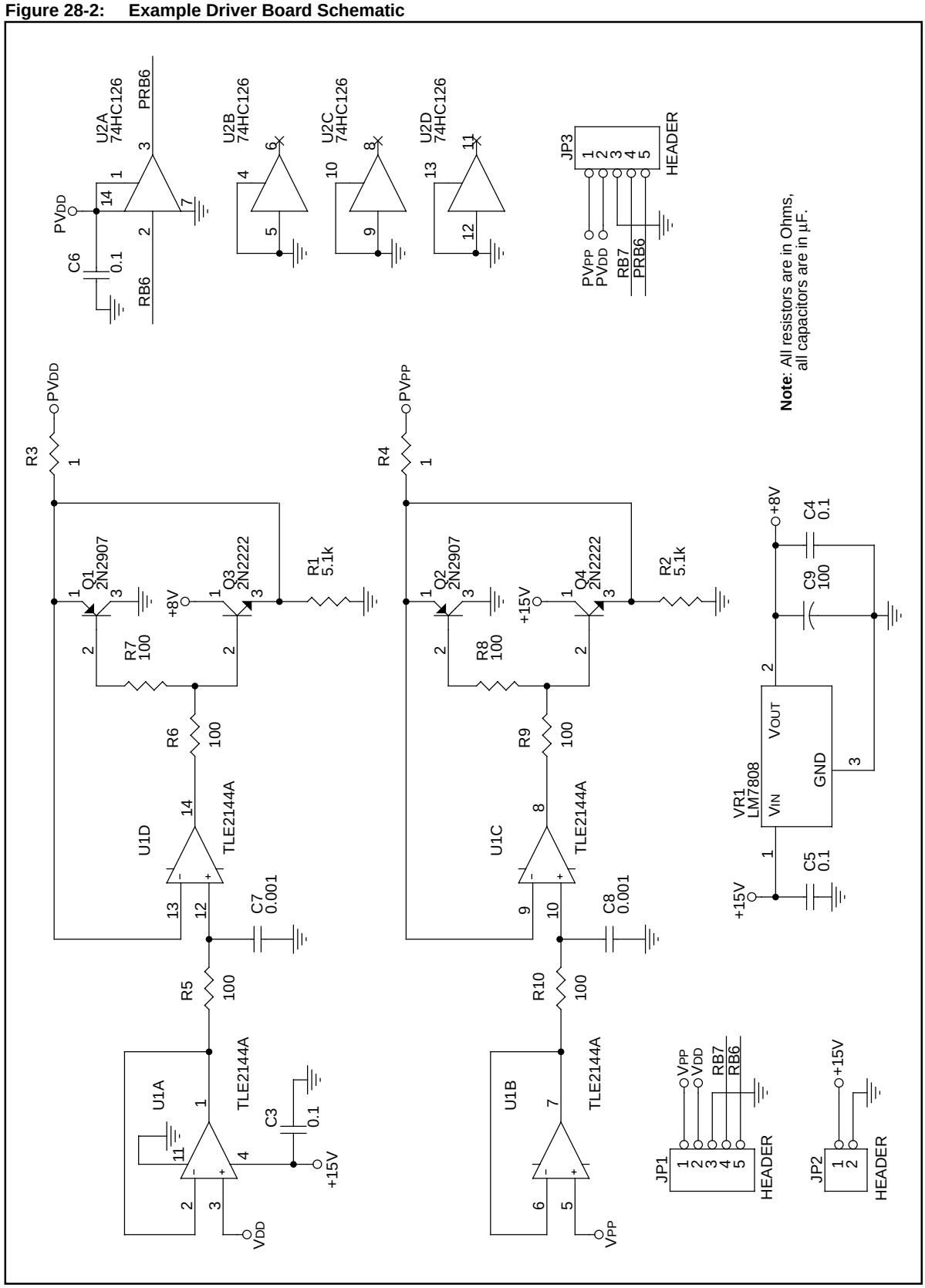
Example 28-2: Programming Cycle Listing Files

```
        First Program Cycle                                  Second Program Cycle
_____________________________________________________________________________________________
Prog    Opcode    Assembly                        |  Prog   Opcode  Assembly
Mem               Instruction                     |  Mem            Instruction
-------------------------------------------------------------------------------------------
0000    2808      goto  Main        ;Main loop    |  0000   0000    nop
0001    3FFF      <blank>           ; at 0x0008   |  0001   2860    goto  Main; Main now
0002    3FFF      <blank>                         |  0002   3FFF    <blank>   ;   at 0x0060
0003    3FFF      <blank>                         |  0003   3FFF    <blank>
0004    2848      goto  ISR         ; ISR at      |  0004   0000    nop
0005    3FFF      <blank>           ;   0x0048    |  0005   28A8    goto  ISR ; ISR now at
0006    3FFF      <blank>                         |  0006   3FFF    <blank>   ;   0x00A8
0007    3FFF      <blank>                         |  0007   3FFF    <blank>
0008    1683      bsf   STATUS,RP0                |  0008   1683    bsf   STATUS,RP0
0009    3007      movlw 0x07                      |  0009   3007    movlw 0x07
000A    009F      movwf ADCON1                    |  000A   009F    movwf ADCON1
 .                                                |   .
 .                                                |   .
 .                                                |   .
0048    1C0C      btfss PIR1,RBIF                 |  0048   1C0C    btfss PIR1,RBIF
0049    284E      goto  EndISR                    |  0049   284E    goto  EndISR
004A    1806      btfsc PORTB,0                   |  004A   1806    btfsc PORTB,0
 .                                                |   .
 .                                                |   .
 .                                                |   .
0060    3FFF      <blank>                         |  0060   1683    bsf   STATUS,RP0
0061    3FFF      <blank>                         |  0061   3005    movlw 0x05
0062    3FFF      <blank>                         |  0062   009F    movwf ADCON1
 .                                                |   .
 .                                                |   .
 .                                                |   .
00A8    3FFF      <blank>                         |  00A8   1C0C    btfss PIR1,RBIF
00A9    3FFF      <blank>                         |  00A9   28AE    goto  EndISR
00AA    3FFF      <blank>                         |  00AA   1806    btfsc PORTB,0
 .                                                |   .
 .                                                |   .
 .                                                |   .
-------------------------------------------------------------------------------------------
```

28.8 Field Programming of FLASH PICmicros

With the ICSP interface circuitry already in place, FLASH-based PICmicros can be easily reprogrammed in the field. These FLASH devices allow you to reprogram them even if they are code protected. A portable ICSP programming station might consist of a laptop computer and programmer. The technician plugs the ICSP interface cable into the application circuit and downloads the new firmware into the PICmicro. The next thing you know the system is up and running without those annoying “bugs.” Another instance would be that you want to add an additional feature to your system. All of your current inventory can be converted to the new firmware and field upgrades can be performed to bring your installed base of systems up to the latest revision of firmware.

Figure 28-2: Example Driver Board Schematic



28.9 Design Tips

Question 1: ***When I try to do ICSP, the entire program is shifted (offset) in the device program memory.***

Answer 1:

If the $\overline{\text{MCLR}}$ pin does not rise fast enough, while the device's voltage is in the valid operating range, the internal Program Counter (PC) can increment. This means that the PC is no longer pointing to the address that you expected to be at. The exact location depends on the number of device clocks that occurred in the valid operating region of the device.

Question 2: ***I am using a PRO MATE II with a socket that I designed to bring the programming signal to my application board. Sometimes when I try to do ICSP, the program memory is programmed wrong.***

Answer 2:

The voltages / timings may be violated at the device. This could be due to the:

- Application board circuitry
- Cable length from programmer to target
- Large capacitance on V_{DD} which affects levels / timings

28.10 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to In-Circuit Serial Programming are:

Title	Application Note #
In-Circuit Serial Programming of Calibration Parameters using a PICmicro	AN656
In-Circuit Serial Programming Guide	DS30277

28.11 Revision History

Revision A

This is the initial released revision of the In-Circuit Serial Programming description.

Section 29. Instruction Set

HIGHLIGHTS

This section of the manual contains the following major topics:

29.1 Introduction

Each midrange instruction is a 14-bit word divided into an OPCODE which specifies the instruction type and one or more operands which further specify the operation of the instruction. The midrange Instruction Set Summary in Table 29-1 lists the instructions recognized by the MPASM assembler. The instruction set is highly orthogonal and is grouped into three basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal and control** operations

Table 29-2 gives the opcode field 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' is zero, the result is placed in the W register. If 'd' is one, the result is placed in the file register specified in the instruction.

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

For **literal and control** operations, 'k' represents an eight or eleven bit constant or literal value.

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

Table 29-1: Midrange Instruction Set

Mnemonic, Operands		Description	Cycles	14-Bit Instruction Word MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: When an I/O register is modified as a function of itself (e.g., `MOVF PORTB, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

29.2 Instruction Formats

Figure 29-1 shows the three general formats that the instructions can have. As can be seen from the general format of the instructions, the opcode portion of the instruction word varies from 3-bits to 6-bits of information. This is what allows the midrange instruction set to have 35 instructions.

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

Note 2: To maintain upward compatibility with future midrange products, do not use the `OPTION` and `TRIS` instructions.

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

0xhh

where h signifies a hexadecimal digit.

To represent a binary number:

`00000100b`

where b is a binary string identifier.

Figure 29-1: General Format for Instructions

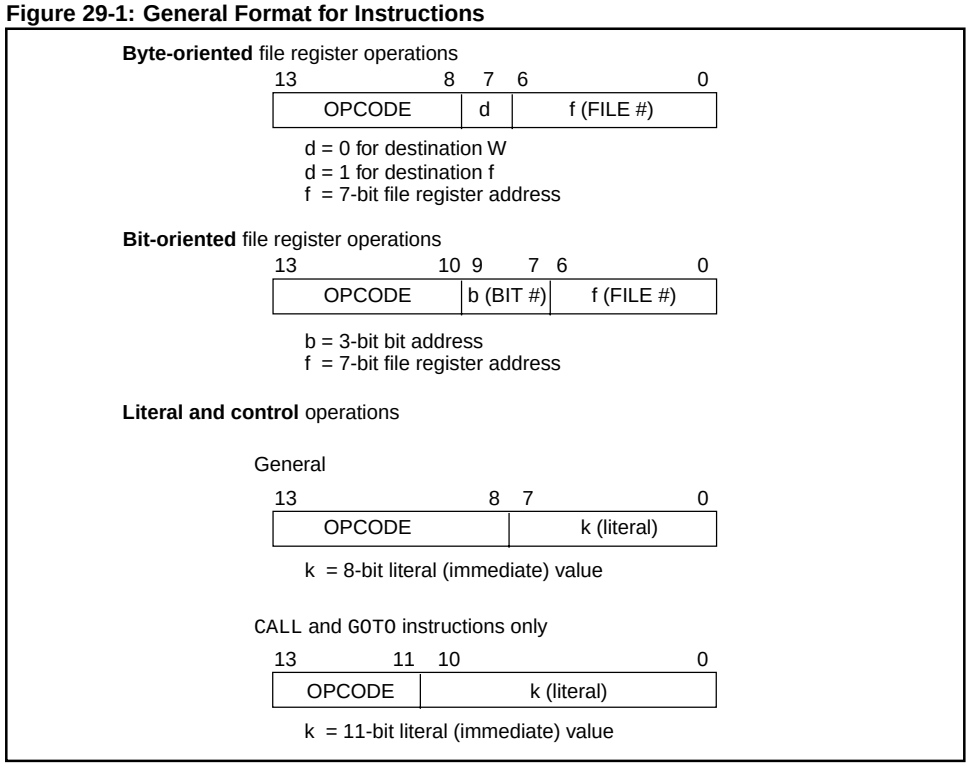


Table 29-2: Instruction Description Conventions

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register (0 to 7)
k	Literal field, constant data or label (may be either an 8-bit or an 11-bit value)
x	Don't care (0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f.
dest	Destination either the W register or the specified register file location
label	Label name
TOS	Top of Stack
PC	Program Counter
PCLATH	Program Counter High Latch
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer
$\overline{TO}$	Time-out bit
$\overline{PD}$	Power-down bit
[ ]	Optional
( )	Contents
$\rightarrow$	Assigned to
< >	Register bit field
$\in$	In the set of
italics	User defined term (font is courier)

29.3 Special Function Registers as Source/Destination

The Section 29. Instruction Set's orthogonal instruction set allows read and write of all file registers, including special function registers. Some special situations the user should be aware of are explained in the following subsections:

29.3.1 STATUS Register as Destination

If an instruction writes to the STATUS register, 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 STATUS` will clear register STATUS, and then set the Z bit leaving `0000 0100`b in the register.

29.3.2 PCL as Source or Destination

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

Read PC:	PCL → dest; PCLATH does not change;
Write PCL:	PCLATH → PCH; 8-bit destination value → PCL
Read-Modify-Write:	PCL→ ALU operand PCLATH → PCH; 8-bit result → PCL

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

29.3.3 Bit Manipulation

All bit manipulation instructions will first read the entire register, operate on the selected bit and then write the result back (read-modify-write (R-M-W)) the specified register. 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 with executing R-M-W instructions on registers which contain these bits.

29.4 Q Cycle Activity

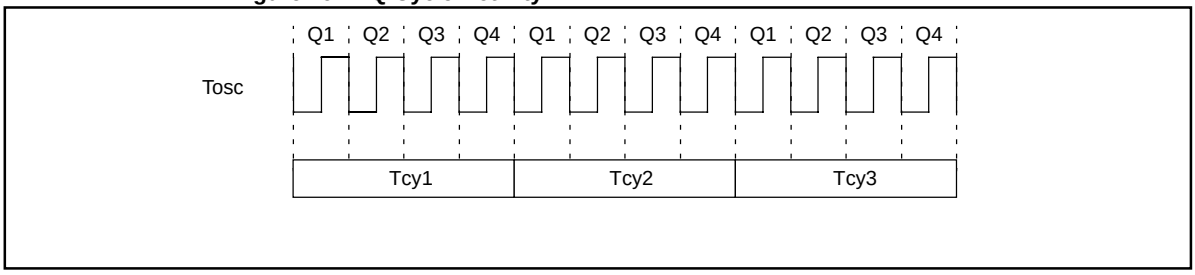
Each instruction cycle (Tcy) is comprised of four Q cycles (Q1-Q4). The Q cycle is the same as the device oscillator cycle (Tosc). The Q cycles provide the timing/designation for the Decode, Read, Process Data, Write etc., of each instruction cycle. The following diagram shows the relationship of the 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 29-2: Q Cycle Activity



29.5 Instruction Descriptions

ADDLW Add Literal and W

Syntax: [*label*] ADDLW k

Operands: $0 \le k \le 255$

Operation: $(W) + k \rightarrow W$

Status Affected: C, DC, Z

Encoding:

11	111x	kkkk	kkkk

Description: The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W register

Example1

```
ADDLW   0x15
```

Before Instruction
W = 0x10
After Instruction
W = 0x25

Example 2

```
ADDLW   MYREG
```

Before Instruction
W = 0x10
Address of MYREG † = 0x37
† MYREG is a symbol for a data memory location
After Instruction
W = 0x47

Example 3

```
ADDLW   HIGH (LU_TABLE)
```

Before Instruction
W = 0x10
Address of LU_TABLE † = 0x9375
† LU_TABLE is a label for an address in program memory
After Instruction
W = 0xA3

Example 4

```
ADDLW   MYREG
```

Before Instruction
W = 0x10
Address of PCL † = 0x02
† PCL is the symbol for the Program Counter low byte location
After Instruction
W = 0x12

ADDWF Add W and f

Syntax: [*label*] ADDWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (W) + (f) → destination

Status Affected: C, DC, Z

Encoding:

00	0111	dfff	ffff

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

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1

```
ADDWF   FSR, 0
```

Before Instruction
W = 0x17
FSR = 0xC2
After Instruction
W = 0xD9
FSR = 0xC2

Example 2

```
ADDWF   INDF, 1
```

Before Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x20
After Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x37

Example 3

```
ADDWF   PCL
```

Case 1: Before Instruction
W = 0x10
PCL = 0x37
C = x
After Instruction
PCL = 0x47
C = 0

Case 2: Before Instruction
W = 0x10
PCL = 0xF7
PCH = 0x08
C = x
After Instruction
PCL = 0x07
PCH = 0x08
C = 1

ANDLW — And Literal with W

Syntax: [*label*] ANDLW k

Operands: $0 \le k \le 255$

Operation: (W).AND. (k) → W

Status Affected: Z

Encoding:

11	1001	kkkk	kkkk

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

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W register

Example 1

```
ANDLW   0x5F
```

Before Instruction
W = 0xA3

After Instruction
W = 0x03

```
; 0101 1111  (0x5F)
; 1010 0011  (0xA3)
;----------  ------
; 0000 0011  (0x03)
```

Example 2

```
ANDLW   MYREG
```

Before Instruction
W = 0xA3
Address of MYREG † = 0x37
† MYREG is a symbol for a data memory location
After Instruction
W = 0x23

Example 3

```
ANDLW   HIGH (LU_TABLE)
```

Before Instruction
W = 0xA3
Address of LU_TABLE † = 0x9375
† LU_TABLE is a label for an address in program memory
After Instruction
W = 0x83

ANDWF AND W with f

Syntax: [*label*] ANDWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (W).AND. (f) → destination

Status Affected: Z

Encoding:

00	0101	dfff	ffff

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

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1

```
ANDWF   FSR, 1
```

Before Instruction
W = 0x17
FSR = 0xC2
After Instruction
W = 0x17
FSR = 0x02

```
; 0001 0111  (0x17)
; 1100 0010  (0xC2)
;----------  ------
; 0000 0010  (0x02)
```

Example 2

```
ANDWF   FSR, 0
```

Before Instruction
W = 0x17
FSR = 0xC2
After Instruction
W = 0x02
FSR = 0xC2

```
; 0001 0111  (0x17)
; 1100 0010  (0xC2)
;----------  ------
; 0000 0010  (0x02)
```

Example 3

```
ANDWF   INDF, 1
```

Before Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x5A
After Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x15

BCF

Bit Clear f

Syntax: [*label*] BCF f,b

Operands: $0 \le f \le 127$
$0 \le b \le 7$

Operation: $0 \rightarrow f<b>$

Status Affected: None

Encoding:

01	00bb	bfff	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 1

```
BCF     FLAG_REG, 7
```

Before Instruction
FLAG_REG = 0xC7 ; **1**100 0111

After Instruction
FLAG_REG = 0x47 ; **0**100 0111

Example 2

```
BCF     INDF, 3
```

Before Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x2F

After Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x27

BSF — Bit Set f

Syntax: [*label*] BSF f,b

Operands: $0 \le f \le 127$
$0 \le b \le 7$

Operation: $1 \rightarrow f<b>$

Status Affected: None

Encoding:

01	01bb	bfff	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 1

```
BSF     FLAG_REG, 7
```

Before Instruction
FLAG_REG =0x0A ; **0**000 1010

After Instruction
FLAG_REG =0x8A ; **1**000 1010

Example 2

```
BSF     INDF, 3
```

Before Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x20

After Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x28

BTFSC Bit Test, Skip if Clear

Syntax: [*label*] BTFSC f,b

Operands: 0 ≤ f ≤ 127
0 ≤ b ≤ 7

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

Status Affected: None

Encoding:

01	10bb	bfff	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 2 cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	No operation

If skip (2nd cycle):

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example 1

```
HERE    BTFSC  FLAG, 4
FALSE   GOTO   PROCESS_CODE
TRUE    •
        •
        •
```

Case 1: Before Instruction
PC = addressHERE
FLAG= xxx0 xxxx
After Instruction
Since FLAG<4>= 0,
PC = addressTRUE

Case 2: Before Instruction
PC = addressHERE
FLAG= xxx1 xxxx
After Instruction
Since FLAG<4>=1,
PC = addressFALSE

BTFSS Bit Test f, 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:

01	11bb	bfff	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 2 cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	No operation

If skip (2nd cycle):

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example 1

```
HERE    BTFSS  FLAG, 4
FALSE   GOTO   PROCESS_CODE
TRUE    •
        •
        •
```

Case 1: Before Instruction
PC = address HERE
FLAG= xxx0 xxxx
After Instruction
Since FLAG<4>= 0,
PC = address FALSE

Case 2: Before Instruction
PC = address HERE
FLAG= xxx1 xxxx
After Instruction
Since FLAG<4>=1,
PC = address TRUE

CALL — Call Subroutine

Syntax: [*label*] CALL k

Operands: $0 \le k \le 2047$

Operation: (PC)+ 1→ TOS,
k → PC<10:0>,
(PCLATH<4:3>) → PC<12:11>

Status Affected: None

Encoding:

10	0kkk	kkkk	kkkk

Description: Call Subroutine. First, the 13-bit return address (PC+1) is pushed onto the stack. The eleven bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH<4:3>. CALL is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

1st cycle:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	No operation

2nd cycle:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example 1

```
HERE    CALL   THERE
```

Before Instruction
PC = Address HERE
After Instruction
TOS = Address HERE+1
PC = Address THERE

CLRF Clear f

Syntax: [*label*] CLRF f

Operands: $0 \le f \le 127$

Operation: $00h \rightarrow f$
$1 \rightarrow Z$

Status Affected: Z

Encoding:

00	0001	1fff	ffff

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

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write register 'f'

Example 1

```
CLRF    FLAG_REG
```

Before Instruction
FLAG_REG=0x5A
After Instruction
FLAG_REG=0x00
Z = 1

Example 2

```
CLRF    INDF
```

Before Instruction
FSR = 0xC2
Contents of Address (FSR)=0xAA
After Instruction
FSR = 0xC2
Contents of Address (FSR)=0x00
Z = 1

CLRW Clear W

Syntax: [*label*] CLRW

Operands: None

Operation: 00h → W
1 → Z

Status Affected: Z

Encoding:

00	0001	0xxx	xxxx

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

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write register 'W'

Example 1

```
CLRW
```

Before Instruction
W = 0x5A
After Instruction
W = 0x00
Z = 1

CLRWDT — Clear Watchdog Timer

Syntax: [*label*] CLRWDT

Operands: None

Operation: 00h → WDT
0 → WDT prescaler count,
1 → $\overline{TO}$
1 → $\overline{PD}$

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

Encoding:

00	0000	0110	0100

Description: CLRWDT instruction clears the Watchdog Timer. It also clears the prescaler count 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	Clear WDT Counter

Example 1

```
CLRWDT
```

Before Instruction
WDT counter= x
WDT prescaler =1:128
After Instruction
WDT counter=0x00
WDT prescaler count=0
$\overline{TO}$ = 1
$\overline{PD}$ = 1
WDT prescaler =1:128

Note: The CLRWDT instruction does not affect the assignment of the WDT prescaler.

COMF Complement f

Syntax: [*label*] COMF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: $(\bar{f}) \rightarrow$ destination

Status Affected: Z

Encoding:

00	1001	dfff	ffff

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

```
COMF    REG1, 0
```

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

Example 2

```
COMF    INDF, 1
```

Before Instruction
FSR = 0xC2
Contents of Address (FSR)=0xAA
After Instruction
FSR = 0xC2
Contents of Address (FSR)=0x55

Example 3

```
COMF    REG1, 1
```

Before Instruction
REG1= 0xFF
After Instruction
REG1= 0x00
Z = 1

DECF Decrement f

Syntax: [*label*] DECF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - 1 → destination

Status Affected: Z

Encoding:

00	0011	dfff	ffff

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

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1

```
DECF    CNT, 1
```

Before Instruction
CNT = 0x01
Z = 0
After Instruction
CNT = 0x00
Z = 1

Example 2

```
DECF    INDF, 1
```

Before Instruction
FSR = 0xC2
Contents of Address (FSR) = 0x01
Z = 0
After Instruction
FSR = 0xC2
Contents of Address (FSR) = 0x00
Z = 1

Example 3

```
DECF    CNT, 0
```

Before Instruction
CNT = 0x10
W = x
Z = 0
After Instruction
CNT = 0x10
W = 0x0F
Z = 0

DECFSZ

Decrement f, Skip if 0

Syntax: [*label*] DECFSZ f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - 1 → destination; skip if result = 0

Status Affected: None

Encoding:

00	1011	dfff	ffff

Description: The contents of register 'f' are decremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
If the result is 0, then the next instruction (fetched during the current instruction execution) is discarded and a NOP is executed instead, making this a 2 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 (2nd cycle):

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example

```
HERE     DECFSZ   CNT, 1
         GOTO     LOOP
CONTINUE •
         •
         •
```

Case 1: Before Instruction
PC = address HERE
CNT = 0x01
After Instruction
CNT = 0x00
PC = address CONTINUE

Case 2: Before Instruction
PC = address HERE
CNT = 0x02
After Instruction
CNT = 0x01
PC = address HERE + 1

GOTO — Unconditional Branch

Syntax: [*label*] GOTO k

Operands: $0 \le k \le 2047$

Operation: $k \rightarrow PC<10:0>$
$PCLATH<4:3> \rightarrow PC<12:11>$

Status Affected: None

Encoding:

10	1kkk	kkkk	kkkk

Description: GOTO is an unconditional branch. The eleven bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

1st cycle:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>	Process data	No operation

2nd cycle:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example

```
GOTO THERE
```

After Instruction

```
PC =AddressTHERE
```

INCF Increment f

Syntax: [*label*] INCF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) + 1 → destination

Status Affected: Z

Encoding:

00	1010	dfff	ffff

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

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1
```
INCF    CNT, 1
```
Before Instruction
CNT = 0xFF
Z = 0
After Instruction
CNT = 0x00
Z = 1

Example 2
```
INCF    INDF, 1
```
Before Instruction
FSR = 0xC2
Contents of Address (FSR) = 0xFF
Z = 0
After Instruction
FSR = 0xC2
Contents of Address (FSR) = 0x00
Z = 1

Example 3
```
INCF    CNT, 0
```
Before Instruction
CNT = 0x10
W = x
Z = 0
After Instruction
CNT = 0x10
W = 0x11
Z = 0

INCFSZ — Increment f, Skip if 0

Syntax: [*label*] INCFSZ f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) + 1 → destination, skip if result = 0

Status Affected: None

Encoding:

00	1111	dfff	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
If the result is 0, then the next instruction (fetched during the current instruction execution) is discarded and a NOP is executed instead, making this a 2 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 (2nd cycle):

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example

```
HERE     INCFSZ   CNT, 1
         GOTO     LOOP
CONTINUE •
         •
         •
```

Case 1: Before Instruction
PC = address HERE
CNT = 0xFF
After Instruction
CNT = 0x00
PC = address CONTINUE

Case 2: Before Instruction
PC = address HERE
CNT = 0x00
After Instruction
CNT = 0x01
PC = address HERE + 1

IORLW Inclusive OR Literal with W

Syntax: [*label*] IORLW k

Operands: $0 \le k \le 255$

Operation: (W).OR. k → W

Status Affected: Z

Encoding:

11	1000	kkkk	kkkk

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

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W register

Example 1

```
IORLW   0x35
```

Before Instruction
W = 0x9A
After Instruction
W = 0xBF
Z = 0

Example 2

```
IORLW   MYREG
```

Before Instruction
W = 0x9A
Address of MYREG † = 0x37
† MYREG is a symbol for a data memory location
After Instruction
W = 0x9F
Z = 0

Example 3

```
IORLW   HIGH (LU_TABLE)
```

Before Instruction
W = 0x9A
Address of LU_TABLE † = 0x9375
† LU_TABLE is a label for an address in program memory
After Instruction
W = 0x9B
Z = 0

Example 4

```
IORLW   0x00
```

Before Instruction
W = 0x00
After Instruction
W = 0x00
Z = 1

IORWF — Inclusive OR W with f

Syntax: [*label*] IORWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (W).OR. (f) → destination

Status Affected: $\overline{Z}$

Encoding:

00	0100	dfff	ffff

Description: Inclusive OR the W register with register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1

```
IORWF   RESULT, 0
```

Before Instruction
RESULT=0x13
W = 0x91
After Instruction
RESULT=0x13
W = 0x93
Z = 0

Example 2

```
IORWF   INDF, 1
```

Before Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x30
After Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x37
Z = 0

Example 3

```
IORWF   RESULT, 1
```

Case 1: Before Instruction
RESULT=0x13
W = 0x91
After Instruction
RESULT=0x93
W = 0x91
Z = 0

Case 2: Before Instruction
RESULT=0x00
W = 0x00
After Instruction
RESULT=0x00
W = 0x00
Z = 1

MOVLW Move Literal to W

Syntax: [*label*] MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow W$

Status Affected: None

Encoding:

11	00xx	kkkk	kkkk

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

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W register

Example 1

```
MOVLW   0x5A
```

After Instruction
W = 0x5A

Example 2

```
MOVLW   MYREG
```

Before Instruction
W = 0x10
Address of MYREG † = 0x37
† MYREG is a symbol for a data memory location
After Instruction
W = 0x37

Example 3

```
MOVLW   HIGH (LU_TABLE)
```

Before Instruction
W = 0x10
Address of LU_TABLE † = 0x9375
† LU_TABLE is a label for an address in program memory
After Instruction
W = 0x93

MOVF Move f

Syntax: [*label*] MOVF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) → destination

Status Affected: Z

Encoding:

00	1000	dfff	ffff

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

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1
```
MOVF    FSR, 0
```
Before Instruction
W = 0x00
FSR = 0xC2
After Instruction
W = 0xC2
Z = 0

Example 2
```
MOVF    INDF, 0
```
Before Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x00
After Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x00
Z = 1

Example 3
```
MOVF    FSR, 1
```
Case 1: Before Instruction
FSR = 0x43
After Instruction
FSR = 0x43
Z = 0

Case 2: Before Instruction
FSR = 0x00
After Instruction
FSR = 0x00
Z = 1

MOVWF Move W to f

Syntax: [*label*] MOVWF f

Operands: $0 \le f \le 127$

Operation: $(W) \rightarrow f$

Status Affected: None

Encoding:

00	0000	1fff	ffff

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

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	Write register 'f'

Example 1

```
MOVWF   OPTION_REG
```

Before Instruction
OPTION_REG=0xFF
W = 0x4F
After Instruction
OPTION_REG=0x4F
W = 0x4F

Example 2

```
MOVWF   INDF
```

Before Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x00
After Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x17

NOP — No Operation

Syntax: [*label*] NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding:

00	0000	0xx0	0000

Description: No operation.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

Example :

```
HERE    NOP
```

Before Instruction
PC = address HERE
After Instruction
PC = address HERE + 1

OPTION — Load Option Register

Syntax:	[*label*] OPTION
Operands:	None
Operation:	(W) → OPTION
Status Affected:	None
Encoding:	00 | 0000 | 0110 | 0010
Description:	The contents of the W register are loaded in the OPTION register. This instruction is supported for code compatibility with PIC16C5X products. Since OPTION is a readable/writable register, the user can directly address it.
Words:	1
Cycles:	1

To maintain upward compatibility with future PIC16CXX products, do not use this instruction.

RETFIE Return from Interrupt

Syntax: [*label*] RETFIE

Operands: None

Operation: TOS → PC,
1 → GIE

Status Affected: None

Encoding:

00	0000	0000	1001

Description: Return from Interrupt. The 13-bit address at the Top of Stack (TOS) is loaded in the PC. The Global Interrupt Enable bit, GIE (INTCON<7>), is automatically set, enabling Interrupts. This is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

1st cycle:

Q1	Q2	Q3	Q4
Decode	No operation	Process data	No operation

2nd cycle:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example

```
RETFIE
```

After Instruction

```
PC  =  TOS
GIE =  1
```

RETLW — Return with Literal in W

Syntax: [*label*] RETLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow W$;
$TOS \rightarrow PC$

Status Affected: None

Encoding:

11	01xx	kkkk	kkkk

Description: The W register is loaded with the eight bit literal 'k'. The program counter is loaded 13-bit address at the Top of Stack (the return address). This is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

1st cycle:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W register

2nd cycle:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example

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

Before Instruction
W = 0x07
After Instruction
W = value of k8
PC = TOS = Address `Here + 1`

RETURN — Return from Subroutine

Syntax: [*label*] RETURN

Operands: None

Operation: TOS → PC

Status Affected: None

Encoding:

00	0000	0000	1000

Description: Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

1st cycle:

Q1	Q2	Q3	Q4
Decode	No operation	Process data	No operation

2nd cycle:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example

```
HERE    RETURN
```

After Instruction

PC = TOS

RLF — Rotate Left f through Carry

Syntax: [*label*] RLF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

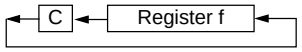
Operation: See description below

Status Affected: C

Encoding:

00	1101	dfff	ffff

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is stored back in register 'f'.



Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1

```
RLF     REG1,0
```

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

Example 2

```
RLF     INDF, 1
```

Case 1: Before Instruction
```
W   =  xxxx xxxx
FSR =  0xC2
Contents of Address (FSR) = 0011 1010
C   =  1
```
After Instruction
```
W   =  0x17
FSR =  0xC2
Contents of Address (FSR) = 0111 0101
C   =  0
```

Case 2: Before Instruction
```
W   =  xxxx xxxx
FSR =  0xC2
Contents of Address (FSR) = 1011 1001
C   =  0
```
After Instruction
```
W   =  0x17
FSR =  0xC2
Contents of Address (FSR) = 0111 0010
C   =  1
```

RRF — Rotate Right f through Carry

Syntax: [*label*] RRF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

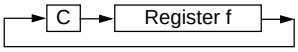
Operation: See description below

Status Affected: C

Encoding:

00	1100	dfff	ffff

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.



Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1

```
RRF          REG1,0
```

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

Example 2

```
RRF     INDF, 1
```

Case 1: Before Instruction
W = xxxx xxxx
FSR = 0xC2
Contents of Address (FSR) = 0011 1010
C = 1
After Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 1001 1101
C = 0

Case 2: Before Instruction
W = xxxx xxxx
FSR = 0xC2
Contents of Address (FSR) = 0011 1001
C = 0
After Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0001 1100
C = 1

SLEEP

Syntax: [*label*] SLEEP

Operands: None

Operation: 00h → WDT,
0 → WDT prescaler count,
1 → $\overline{TO}$,
0 → $\overline{PD}$

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

Encoding:

00	0000	0110	0011

Description: The power-down status bit, $\overline{PD}$ is cleared. Time-out status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler count 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	No operation	Go to sleep

Example: SLEEP

Note: The SLEEP instruction does not affect the assignment of the WDT prescaler

SUBLW Subtract W from Literal

Syntax: [*label*] SUBLW k

Operands: $0 \le k \le 255$

Operation: $k - (W) \rightarrow W$

Status Affected: C, DC, Z

Encoding:

11	110x	kkkk	kkkk

Description: The W register is subtracted (2's complement method) from the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W register

Example 1:
```
SUBLW   0x02
```
Case 1: Before Instruction
```
W = 0x01
C = x
Z = x
```
After Instruction
```
W = 0x01
C = 1       ; result is positive
Z = 0
```
Case 2: Before Instruction
```
W = 0x02
C = x
Z = x
```
After Instruction
```
W = 0x00
C = 1       ; result is zero
Z = 1
```
Case 3: Before Instruction
```
W = 0x03
C = x
Z = x
```
After Instruction
```
W = 0xFF
C = 0       ; result is negative
Z = 0
```

Example 2
```
SUBLW   MYREG
```
Before Instruction
```
W = 0x10
```
Address of MYREG † = 0x37

† MYREG is a symbol for a data memory location

After Instruction
```
W = 0x27
C = 1       ; result is positive
```

SUBWF — Subtract W from f

Syntax: [*label*] SUBWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - (W) → destination

Status Affected: C, DC, Z

Encoding:

00	0010	dfff	ffff

Description: Subtract (2's complement method) W register from register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1:
```
SUBWF   REG1,1
```
Case 1: Before Instruction
```
REG1= 3
W   = 2
C   = x
Z   = x
```
After Instruction
```
REG1= 1
W   = 2
C   = 1          ; result is positive
Z   = 0
```
Case 2: Before Instruction
```
REG1= 2
W   = 2
C   = x
Z   = x
```
After Instruction
```
REG1= 0
W   = 2
C   = 1          ; result is zero
Z   = 1
```
Case 3: Before Instruction
```
REG1= 1
W   = 2
C   = x
Z   = x
```
After Instruction
```
REG1= 0xFF
W   = 2
C   = 0          ; result is negative
Z   = 0
```

SWAPF — Swap Nibbles in f

Syntax: [*label*] SWAPF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f<3:0>) → destination<7:4>,
(f<7:4>) → destination<3:0>

Status Affected: None

Encoding:

00	1110	dfff	ffff

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

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1

```
SWAPF   REG, 0
```

Before Instruction

REG1= 0xA5

After Instruction

REG1= 0xA5
W = 0x5A

Example 2

```
SWAPF   INDF, 1
```

Before Instruction

W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x20

After Instruction

W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x02

Example 3

```
SWAPF   REG, 1
```

Before Instruction

REG1= 0xA5

After Instruction

REG1= 0x5A

TRIS — Load TRIS Register

Syntax:	[*label*] TRIS f
Operands:	$5 \le f \le 7$
Operation:	(W) $\rightarrow$ TRIS register f;
Status Affected:	None
Encoding:	00 | 0000 | 0110 | 0fff
Description:	The instruction is supported for code compatibility with the PIC16C5X products. Since TRIS registers are readable and writable, the user can directly address them.
Words:	1
Cycles:	1

Example

To maintain upward compatibility with future PIC16CXX products, do not use this instruction.

XORLW Exclusive OR Literal with W

Syntax: [*label*] XORLW k

Operands: $0 \le k \le 255$

Operation: (W).XOR. k → W

Status Affected: Z

Encoding:

11	1010	kkkk	kkkk

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

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W register

Example 1

```
XORLW   0xAF                ; 1010 1111   (0xAF)
Before Instruction          ; 1011 0101   (0xB5)
    W   =   0xB5            ; ---------   ------
After Instruction           ; 0001 1010   (0x1A)
    W   =   0x1A
    Z   =   0
```

Example 2

```
XORLW   MYREG
```

Before Instruction
W = 0xAF
Address of MYREG † = 0x37
† MYREG is a symbol for a data memory location
After Instruction
W = 0x18
Z = 0

Example 3

```
XORLW   HIGH (LU_TABLE)
```

Before Instruction
W = 0xAF
Address of LU_TABLE † = 0x9375
† LU_TABLE is a label for an address in program memory
After Instruction
W = 0x3C
Z = 0

XORWF — Exclusive OR W with f

Syntax: [*label*] XORWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (W).XOR. (f) → destination

Status Affected: Z

Encoding:

00	0110	dfff	ffff

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

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1

```
XORWF    REG, 1                 ; 1010 1111   (0xAF)
Before Instruction              ; 1011 0101   (0xB5)
        REG= 0xAF               ; ---------   ------
        W  = 0xB5               ; 0001 1010   (0x1A)
After Instruction
        REG= 0x1A
        W  = 0xB5
```

Example 2

```
XORWF    REG, 0                 ; 1010 1111   (0xAF)
Before Instruction              ; 1011 0101   (0xB5)
        REG= 0xAF               ; ---------   ------
        W  = 0xB5               ; 0001 1010   (0x1A)
After Instruction
        REG= 0xAF
        W  = 0x1A
```

Example 3

```
XORWF   INDF, 1
```

Before Instruction
W = 0xB5
FSR = 0xC2
Contents of Address (FSR) = 0xAF

After Instruction
W = 0xB5
FSR = 0xC2
Contents of Address (FSR) = 0x1A

29.6 Design Tips

Question 1: ***How can I modify the value of W directly? I want to decrement W.***

Answer 1:

There are a few possibilities, two are:

1. For the midrange devices, there are several instructions that work with a literal and W. For instance, if it were desired to decrement W, this can be done with an `ADDLW 0xFF`. (the 0x prefix denotes hex to the assembler)
2. Notice that all of the instructions can modify a value right where it sits in the file register. This means you can decrement it right where it is. You do not even need to move it to W. If you want to decrement it AND move it somewhere else, then you make W the DESTINATION of the decrement (`DECF register,W`) then put it where you want it. It is the same number of instructions as a straight move, but it gets decremented along the way.

Question 2: ***Is there any danger in using the `TRIS` instruction for the PIC16CXXX since there is a warning in the Data book suggesting it not be used?***

Answer 2:

For code compatibility and upgrades to later parts, the use of the TRIS instruction is not recommended. You should note the `TRIS` instruction is limited to ports A, B and C. Future devices may not support these instructions.

Question 3: ***Do I have to switch to Bank1 of data memory before using the `TRIS` instruction (for parts with TRIS registers in the memory map)?***

Answer 3:

No. The `TRIS` instruction is Bank independent. Again the use of the `TRIS` instruction is not recommended.

Question 4: ***I have seen references to "Read-Modify-Write" instructions in your data sheet, but I do not know what that is. Can you explain what it is and why I need to know this?***

Answer 4:

An easy example of a Read-Modify-Write (R-M-W) instruction is the bit clear instruction `BCF`. You might think that the processor just clears the bit, which on a port output pin would clear the pin. What actually happens is the whole port (or register) is first read, THEN the bit is cleared, then the new modified value is written back to the port (or register). Actually, any instruction that depends on a value currently in the register is going to be a Read-Modify-Write instruction. This includes `ADDWF, SUBWF, BCF, BSF, INCF, XORWF`, etc... Instructions that do not depend on the current register value, like `MOVWF, CLRF`, and so on are not R-M-W instructions.

One situation where you would want to consider the affects of a R-M-W instruction is a port that is continuously changed from input to output and back. For example, say you have TRISB set to all outputs, and write all ones to the PORTB register, all of the PORTB pins will go high. Now, say you turn pin RB3 into an input, which happens to go low. A `BCF PORTB,6` is then executed to drive pin RB6 low. If you then turn RB3 back into an output, it will now drive low, even though the last value you put there was a one. What happened was that the `BCF` of the other pin (RB6) caused the whole port to be read, including the zero on RB3 when it was an input. Then, bit 6 was changed as requested, but since RB3 was read as a zero, zero will also be placed back into that port latch, overwriting the one that was there before. When the pin is turned back into an output, the new value was reflected.

Question 5: ***When I perform a BCF other pins get cleared in the port. Why?***

Answer 5:

There are a few possibilities, two are:

1. Another case where a R-M-W instruction may seem to change other pin values unexpectedly can be illustrated as follows: Suppose you make PORTC all outputs and drive the pins low. On each of the port pins is an LED connected to ground, such that a high output lights it. Across each LED is a 100 µF capacitor. Let's also suppose that the processor is running very fast, say 20 MHz. Now if you go down the port setting each pin in order; `BSF PORTC,0` then BSF PORTC,1 then `BSF PORTC,2` and so on, you may see that only the last pin was set, and only the last LED actually turns on. This is because the capacitors take a while to charge. As each pin was set, the pin before it was not charged yet and so was read as a zero. This zero is written back out to the port latch (R-M-W, remember) which clears the bit you just tried to set the instruction before. This is usually only a concern at high speeds and for successive port operations, but it can happen so take it into consideration.
2. If this is on a PIC16C7X device, you may not have configured the I/O pins properly in the ADCON1 register. If a pin is configured for analog input, any read of that pin will read a zero, regardless of the voltage on the pin. This is an exception to the normal rule that the pin state is always read. You can still configure an analog pin as an output in the TRIS register, and drive the pin high or low by writing to it, but you will always read a zero. Therefore if you execute a Read-Modify-Write instruction (see previous question) all analog pins are read as zero, and those not directly modified by the instruction will be written back to the port latch as zero. A pin configured as analog is expected to have values that may be neither high nor low to a digital pin, or floating. Floating inputs on digital pins are a no-no, and can lead to high current draw in the input buffer, so the input buffer is disabled.

29.7 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the instruction set are:

Currently No related Application Notes

29.8 Revision History

Revision A

This is the initial released revision of the Instruction Set description.

Section 30. Electrical Specifications

HIGHLIGHTS

30.1 Introduction

This section is intended to show you the electrical specifications that may be specified in a particular device data sheet and what is meant by the specification. This section is **NOT** intended to give the values of these specifications. For the device specific values you **must** refer to the device's data sheet. All values show in this section should be considered as Example Values.

In the description of the device and the functional modules (previous sections), there have been references to electrical specification parameters. These references have been hyperlinked in the electronic version to aid in the use of this manual.

Note: Before starting any design, Microchip HIGHLY recommends that you acquire the most recent copy of the device data sheet and review the electrical specifications to ensure that they will meet your requirements.

Throughout this section, certain terms will be used. Table 30-1 shows the conventions that will be used.

Table 30-1: Term Conventions

Term	Description
PIC16**C**XXX	For devices tested to standard voltage range
PIC16**LC**XXX	For devices tested to extended voltage range
PIC16**F**XXX	For devices tested to standard voltage range
PIC16**LF**XXX	For devices tested to extended voltage range
PIC16**CR**XXX	For devices tested to standard voltage range
PIC16**LCR**XXX	For devices tested to extended voltage range
PIC16XXXX-**04**	For devices that have been tested up to 4 MHz operation
PIC16XXXX-**08**	For devices that have been tested up to 8 MHz operation
PIC16XXXX-**10**	For devices that have been tested up to 10 MHz operation
PIC16XXXX-**20**	For devices that have been tested up to 20 MHz operation
LP osc	For devices configured with the LP device oscillator selected
XT osc	For devices configured with the XT device oscillator selected
HS osc	For devices configured with the HS device oscillator selected
RC osc	For devices configured with the RC device oscillator selected
Commercial	For devices with the commercial temperature range grading ($0^{\circ}C \le T_A \le +70^{\circ}C$)
Industrial	For devices with the industrial temperature range grading ($-40^{\circ}C \le T_A \le +85^{\circ}C$)
Extended	For devices with the extended temperature range grading ($-40^{\circ}C \le T_A \le +125^{\circ}C$)

30.2 Absolute Maximums

The Absolute Maximum Ratings specify the worst case conditions that can be applied to the device. These ratings are not meant as operational specifications, and stresses above the listed values may cause damage to the device. Specifications are not always stand-alone, that is, the specification may have other requirements as well.

An example of this is the "maximum current sourced/sunk by any I/O pin". The number of I/O pins that can be sinking/sourcing current, at any one time, is dependent upon the maximum current sunk/source by the port(s) (combined) and the maximum current into the VDD pin or out of the VSS pin. In this example, the physical reason is the Power and Ground bus width to the I/O ports and internal logic. If these specifications are exceeded, then electromigration may occur on these Power and Ground buses. Over time electromigration would cause these buses to open (be disconnected from the pin), and therefore cause the logic attached to these buses to stop operating. So exceeding the absolute specifications may cause device reliability issues.

Input Clamp Current is defined as the current through the diode to VSS/VDD if pin voltage exceeds specification.

Example Absolute Maximum Ratings†

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

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

Voltage on any pin with respect to VSS (except VDD, $\overline{MCLR}$, and RA4) ..... -0.3V to (VDD + 0.3V)

Voltage on VDD with respect to VSS ..... -0.3 to +7.5V

Voltage on $\overline{MCLR}$ with respect to VSS **(2)** ..... 0 to +14V

Voltage on RA4 with respect to Vss ..... 0 to +14V

Total power dissipation **(1)** 1.0W

Maximum current out of VSS pin ..... 300 mA

Maximum current into VDD pin ..... 250 mA

Input clamp current, IIK (VI < 0 or VI > VDD) ..... ± 20 mA

Output clamp current, IOK (VO < 0 or VO > VDD) ..... ± 20 mA

Maximum output current sunk by any I/O pin ..... 25 mA

Maximum output current sourced by any I/O pin ..... 25 mA

Maximum current sunk by PORTA, PORTB, and PORTE (combined) ..... 200 mA

Maximum current sourced by PORTA, PORTB, and PORTE (combined) ..... 200 mA

Maximum current sunk by PORTC and PORTD (combined) ..... 200 mA

Maximum current sourced by PORTC and PORTD (combined) ..... 200 mA

Maximum current sourced by PORTC and PORTD (combined) ..... 200 mA

Maximum current sourced by PORTF and PORTG (combined) ..... 100 mA

Maximum current sourced by PORTF and PORTG (combined) ..... 100 mA

Note 1: Power dissipation is calculated as follows:
Pdis = VDD x {IDD - ∑ IOH} + ∑ {(VDD - VOH) x IOH} + ∑(VOl x IOL)

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

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

30.3 Device Selection Table

This table in the Device Data Sheet is intended to assist you in determining which oscillators are tested for which devices, and some of the specifications that are tested. Any oscillator may be selected at time of programming, but only the specified oscillator is tested by Microchip.

Since the RC and XT oscillators are only rated to 4 MHz, they are only tested on the -04 (4 MHz) devices.

PICmicros rated for 10 MHz or 20 MHz are only tested in HS mode. In Table 30-2 the IPD is grayed out for the HS mode since there is not an IPD test point within the voltage range of the HS oscillator. The value shown is a typical value from characterization.

Battery applications usually require an extended voltage range. Devices marked **LC** have an extended voltage range and have the RC, XT, and LP oscillators tested.

Windowed devices are superset devices and have had the oscillators tested to all the specification ranges of the -04, -20, and LC devices. The temperature range that the device is tested to should be considered commercial, though at a later time they may be tested to industrial or extended temperature levels.

Table 30-2: Example Cross Reference of Device Specifications for Oscillator Configurations and Frequencies of Operation (Commercial Devices)

OSC	PIC16CXXX-04	PIC16CXXX-10	PIC16CXXX-20	PIC16LCXXX-04	Windowed Devices
RC	VDD: 4.0V to 6.0V IDD: 5 mA max. at 5.5V IPD: 16 µA max. at 4V Freq: 4 MHz max.	VDD: 4.5V to 5.5V IDD: 2.7 mA typ. at 5.5V IPD: 1.5 µA typ. at 4V Freq: 4 MHz max.	VDD: 4.5V to 5.5V IDD: 2.7 mA typ. at 5.5V IPD: 1.5 µA typ. at 4V Freq: 4 MHz max.	VDD: 2.5V to 6.0V IDD: 3.8 mA max. at 3.0V IPD: 5 µA max. at 3V Freq: 4 MHz max.	VDD: 2.5V to 6.0V IDD: 3.8 mA max. at 5.5V IPD: 16 µA max. at 4V Freq: 4 MHz max.
XT	VDD: 4.0V to 6.0V IDD: 5 mA max. at 5.5V IPD: 16 µA max. at 4V Freq: 4 MHz max.	VDD: 4.5V to 5.5V IDD: 2.7 mA typ. at 5.5V IPD: 1.5 µA typ. at 4V Freq: 4 MHz max.	VDD: 4.5V to 5.5V IDD: 2.7 mA typ. at 5.5V IPD: 1.5 µA typ. at 4V Freq: 4 MHz max.	VDD: 2.5V to 6.0V IDD: 3.8 mA max. at 3.0V IPD: 5 µA max. at 3V Freq: 4 MHz max.	VDD: 2.5V to 6.0V IDD: 3.8 mA max. at 5.5V IPD: 16 µA max. at 4V Freq: 4 MHz max.
HS	VDD: 4.5V to 5.5V IDD: 13.5 mA typ. at 5.5V IPD: 1.5 µA typ. at 4.5V Freq: 4 MHz max.	VDD: 4.5V to 5.5V IDD: 10 mA max. at 5.5V IPD: 1.5 µA typ. at 4.5V Freq: 10 MHz max.	VDD: 4.5V to 5.5V IDD: 20 mA max. at 5.5V IPD: 1.5 µA typ. at 4.5V Freq: 20 MHz max.	Not recommended for use in HS mode	VDD: 4.5V to 5.5V IDD: 20 mA max. at 5.5V IPD: 1.5 µA typ. at 4.5V Freq: 20 MHz max.
LP	VDD: 4.0V to 6.0V IDD: 52.5 µA typ. at 32 kHz, 4.0V IPD: 0.9 µA typ. at 4.0V Freq: 200 kHz max.	Not recommended for use in LP mode	Not recommended for use in LP mode	VDD: 2.5V to 6.0V IDD: 48 µA max. at 32 kHz, 3.0V IPD: 5.0 µA max. at 3.0V Freq: 200 kHz max.	VDD: 2.5V to 6.0V IDD: 48 µA max. at 32 kHz, 3.0V IPD: 5.0 µA max. at 3.0V Freq: 200 kHz max.

The shaded sections indicate oscillator selections which are tested for functionality, but not for MIN/MAX specifications.
It is recommended that the user select the device type that ensures the specifications required.

Note: Devices that are marked with Engineering Sample (ENG SMP) are tested to the current engineering test program at time of the device testing. There is no implied warranty that these devices have been tested to any or all specifications in the Device Data Sheet.

30.4 Device Voltage Specifications

These specifications relate to the device VDD and the device power-up and function.

Supply Voltage is the voltage level that must be applied to the device for the proper functional operation.

Ram Data Retention Voltage is the level that the device voltage may be at and still retain the data value.

VDD Start Voltage to ensure the internal Power-on Reset signal, is the level that VDD must start from to ensure that the POR circuitry will operate properly.

VDD Rise Rate to ensure internal Power-on Reset signal, is the minimum slope that VDD must rise at to cause the POR circuitry to trip.

Brown-out Reset Voltage is the voltage range where the brown-out circuitry may trip. When the BOR circuitry trips, the device will either be in brown-out reset, or just came out of brown-out reset.

Table 30-3: Example DC Characteristics

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C for commercial and -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
	VDD	**Supply Voltage**					
D001		PIC16**C**XXX	4.0	—	6.0	V	XT, RC and LP osc mode
		PIC16**LC**XXX	2.5	—	6.0	V	
D001A		PIC16**C**XXX	4.5	—	5.5	V	HS osc mode
D002	VDR	**RAM Data Retention Voltage(1)**	1.5	—	—	V	
D003	VPOR	**VDD Start Voltage** to ensure internal Power-on Reset signal	—	VSS	—	V	See section on Power-on Reset for details
D004	SVDD	**VDD Rise Rate** to ensure internal Power-on Reset signal	0.05	—	—	V/ms	See section on Power-on Reset for details
	VBOR	**Brown-out Reset Voltage**					
D005			3.7	4.0	4.3	V	BODEN bit in Configuration Word enabled
D005A			3.7	4.0	4.4	V	Extended Temperature Range Devices Only

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which VDD can be lowered in SLEEP mode without losing RAM data.

30.5 Device Current Specifications

IDD is the current (I) that the device consumes when the device is in operating mode. This test is taken with all I/O as inputs, either pulled high or low. That is, there are no floating inputs, nor are any pins driving an output (with a load).

IPD is the current (I) that the device consumes when the device is in sleep mode (power-down), referred to as power-down current. These tests are taken with all I/O as inputs, either pulled high or low. That is, there are no floating inputs, nor are any pins driving an output (with a load), weak pull-ups are disabled.

A device may have certain features and modules that can operate while the device is in sleep mode. Some on these modules are:

- Watchdog Timer (WDT)
- Brown-out Reset (BOR) circuitry
- Timer1
- Analog to Digital converter
- LCD module
- Comparators
- Voltage Reference

When all features are disabled, the device will consume the lowest possible current (the leakage current). If any of these features are operating while the device is in sleep, a higher current will occur. The difference between the lowest power mode (everything off) at only that one feature enabled (such as the WDT) is what we call the **Module Differential Current**. If more then one feature is enabled then the expected current can easily be calculated as: the base current (everything disabled and in sleep mode) plus all Module Differential Currents (delta currents). Example 30-1 shows an example of calculating the typical currents for a device at 5V, with the WDT and Timer1 oscillator enabled.

Example 30-1: IPD Calculations with WDT and Timer1 Oscillator Enabled (@ 5V)

Base Current	14 nA	; Device leakage current
WDT Delta Current	14 µA	; 14 µA - 14 nA = 14 µA
Timer1 Delta Current	22 µA	; 22 µA - 14 nA = 22 µA
Total Sleep Current	36 µA	;

Table 30-4: Example DC Characteristics

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C for commercial and -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
	IDD	**Supply Current(2,4,5)**					XT, RC osc configuration (PIC16**C**XXX-**04**)
D010			—	2.7	5	mA	FOSC = 4 MHz, VDD = 5.5V
			—	2.0	3.8	mA	FOSC = 4 MHz, VDD = 3.0V
D010A			—	22.5	48	µA	LP osc configuration FOSC = 32 kHz, VDD = 3.0V, WDT disabled
D010C			—	7.7	5	mA	INTRC osc configuration, Fosc = 4 MHz, VDD = 5.5V
D013			—	13.5	30	mA	HS osc configuration (PIC16**C**XXX-**20**) Fosc = 20 MHz, VDD = 5.5V
	IPD	**Power-down Current(3,5)**					
D020			—	10.5	42	µA	VDD = 4.0V, WDT enabled, -40°C to +85°C
			—	7.5	30	µA	VDD = 3.0V, WDT enabled, -40°C to +85°C
			—	1.5	21	µA	VDD = 4.0V, WDT disabled, -0°C to +70°C
D021			—	0.9	13.5	µA	VDD = 3.0V, WDT disabled, 0°C to +70°C
			—	1.5	24	µA	VDD = 4.0V, WDT disabled, -40°C to +85°C
D021A			—	0.9	18	µA	VDD = 3.0V, WDT disabled, -40°C to +85°C
D021B			—	1.5	—	µA	VDD = 4.0V, WDT disabled, -40°C to +125°C

* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Not Applicable.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail to rail; all I/O pins tristated, pulled to VDD
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

3: The power-down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, with all I/O pins in hi-impedance state and tied to VDD and VSS.

4: For RC osc configuration, current through Rext is not included. The current through the resistor can be estimated by the formula Ir = VDD/2Rext (mA) with Rext in kOhm.

5: Timer1 oscillator (when enabled) adds approximately 20 µA to the specification. This value is from characterization and is for design guidance only. This is not tested.

Table 30-5: Example DC Characteristics

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature 0˚C ≤ TA ≤ +70˚C for commercial, -40˚C≤ TA ≤ +85˚C for industrial and -40˚C ≤ TA ≤ +125˚C for extended				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
		Module Differential Current (5)					
D022	ΔIWDT	Watchdog Timer	—	6.0	20	µA	VDD = 4.0V
			—	—	25	µA	-40°C to +125°C
D022A	ΔIBOR	Brown-out Reset	—	350	425	µA	BODEN bit is clear, VDD = 5.0V
D023	ΔICOMP	Comparator (per Comparator)	—	85	100	µA	VDD = 4.0V
D023A	ΔIVREF	Voltage Reference	—	94	300	µA	VDD = 4.0V
D024	ΔILCDRC	LCD internal RC osc enabled	—	6.0	20	µA	VDD = 3.0V
D024A	ΔILCDVG	LCD voltage generation	—	TBD	TBD	µA	VDD = 3.0V
D025	ΔIT1OSC	Timer1 oscillator	—	3.1	6.5	µA	VDD = 3.0V
D026	ΔIAD	A/D Converter	—	1.0	—	µA	A/D on, not converting
D027	ΔISAD	Slope A/D (Total)	—	165 *	250 *	µA	REFOFF = 0
D027A	ΔISADVR	Slope A/D Bandgap Voltage Reference	—	20 *	30 *	µA	REFOFF = 0
D027B	ΔISADCDAC	Slope A/D Programmable Current Source	—	50 *	70 *	µA	ADCON1<7:4> = 1111b
D027C	ΔISADSREF	Slope A/D Reference Voltage Divider	—	55 *	85 *	µA	ADOFF = 0
D027D	ΔISADCMP	Slope A/D Comparator	—	40 *	65 *	µA	ADOFF = 0

† Data in “Typ” column is at 5V, 25˚C unless otherwise stated. These parameters are for design guidance only and are not tested.

30.6 Input Threshold Levels

The **Input Low Voltage** (V_{IL}) is the voltage level that will be read as a logic '0'. An input may not read a '0' at a voltage level above this. All designs should be to the specification since device to device (and to a much lesser extent pin to pin) variations will cause this level to vary.

The **Input High Voltage** (V_{IH}) is the voltage level that will be read as a logic '1'. An input may read a '1' at a voltage level below this. All designs should be to the specification since device to device (and to a much lesser extent pin to pin) variations will cause this level to vary.

The I/O pins with TTL levels are shown with two specifications. One is the industry standard TTL specification, which is specified for the voltage range of 4.5V to 5.5V. The other is a specification that operates over the entire voltage range of the device. The better of these two specifications may be used in the design.

Table 30-6: Example DC Characteristics

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C for commercial, -40°C ≤ TA ≤ +85°C for industrial and -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC spec Table 30-3.				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
	VIL	**Input Low Voltage**					
		I/O ports:					
D030		with TTL buffer	VSS	—	0.15VDD	V	For entire VDD range (4)
D030A			—	—	0.8	V	4.5V ≤ VDD ≤ 5.5V (4)
D031		with Schmitt Trigger buffer	VSS	—	0.2VDD	V	For entire VDD range
D032		$\overline{MCLR}$, OSC1 (RC mode)	VSS	—	0.2VDD	V	
D033		OSC1 (XT, HS and LP modes)(1)	VSS	—	0.3VDD	V	
	VIH	**Input High Voltage**					
		I/O ports:					
D040		with TTL buffer	0.25VDD + 0.8V	—	VDD	V	For entire VDD range (4)
D040A			2.0	—	VDD	V	4.5V ≤ VDD ≤ 5.5V (4)
D041		with Schmitt Trigger buffer	0.8VDD	—	VDD	V	For entire VDD range
D042		$\overline{MCLR}$	0.8VDD	—	VDD	V	
D042A		OSC1 (XT, HS and LP modes)(1)	0.7VDD	—	VDD	V	
D043		OSC1 (RC mode)	0.9VDD	—	VDD	V	
D050	VHYS	Hysteresis of Schmitt Trigger Inputs	TBD	—	—	V	

† 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: In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PICmicro be driven with an external clock while in RC mode.
2: Not Applicable.
3: Not Applicable.
4: The better of the two specifications may be used. For VIL this would be the higher voltage and for VIH this would be the lower voltage.

30.7 I/O Current Specifications

The PORT/GIO **Weak Pull-up Current** is the additional current that the device will draw when the weak pull-ups are enabled.

Leakage Currents are the currents that the device consumes, since the devices are manufactured in the real world and do not adhere to their ideal characteristics. Ideally there should be no current on an input, but due to the real world there is always some parasitic path that consumes negligible current.

Table 30-7: Example DC Characteristics

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C for commercial, -40°C ≤ TA ≤ +85°C for industrial and -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC spec Table 30-3.				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
	IIL	**Input Leakage Current(2,3)**					
D060		I/O ports	—	—	±1	µA	VSS ≤ VPIN ≤ VDD, Pin at hi-impedance
D060A		CDAC	—	—	±1	µA	VSS ≤ VPIN ≤ VDD, Pin at hi-impedance
D061		$\overline{MCLR}$	—	—	±5	µA	VSS ≤ VPIN ≤ VDD
D063			—	—	±5	µA	VSS ≤ VPIN ≤ VDD, XT, HS and LP osc modes
	IPU	**Weak Pull-up Current**					
D070	IPURB	PORTB weak pull-up current	50	250	400	µA	VDD = 5V, VPIN = VSS
D070A	IPUGIO	GIO weak pull-up current	50	250	400	µA	VDD = 5V, VPIN = VSS
		Programmable Current Source (Slope A/D devices)					CDAC pin = 0V
D160		Output Current	18.75	33.75	48.75	µA	ADCON1<7:4> = 1111b (full-scale)
D160A			1.25	2.25	3.25	µA	ADCON1<7:4> = 0001b (1 LSB)
D160B			-0.5	0	0.5	µA	ADCON1<7:4> = 0000b (zero-scale)

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

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

3: Negative current is defined as current sourced by the pin.

30.8 Output Drive Levels

The **Output Low Voltage** (V_{OL}) of an I/O pin depends on the external connections to that I/O. If an I/O pin is shorted to V_{DD}, no matter the drive capability of the I/O pin, a low level would not be reached (and the device would consume excessive drive current). The V_{OL} is the output voltage that the I/O pin will drive, given the I/O does not need to sink more then the I_{OL} current (at the specified device voltage) as specified in the conditions portion of the specification.

The **Output High Voltage** (V_{OH}) of an I/O pin depends on the external connections to that I/O. If an I/O pin is shorted to V_{SS}, no matter the drive capability of the I/O pin, a high level would not be reached (and the device would consume excessive drive current). The V_{OH} is the output voltage that the I/O pin will drive, given the I/O does not need to source more then the I_{OH} current (at the specified device voltage) as specified in the conditions portion of the specification.

Table 30-8: Example DC Characteristics

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C for commercial, -40°C ≤ TA ≤ +85°C for industrial and -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC spec Table 30-3.				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
	V_{OL}	**Output Low Voltage**					
D080		I/O ports	—	—	0.6	V	I_{OL} = 8.5 mA, V_{DD} = 4.5V, -40°C to +85°C
D080A			—	—	0.6	V	I_{OL} = 7.0 mA, V_{DD} = 4.5V, -40°C to +125°C
D083		OSC2/CLKOUT (RC mode)	—	—	0.6	V	I_{OL} = 1.6 mA, V_{DD} = 4.5V, -40°C to +85°C
D083A			—	—	0.6	V	I_{OL} = 1.2 mA, V_{DD} = 4.5V, -40°C to +125°C
	V_{OH}	**Output High Voltage(3)**					
D090		I/O ports	V_{DD} - 0.7	—	—	V	I_{OH} = -3.0 mA, V_{DD} = 4.5V, -40°C to +85°C
D090A			V_{DD} - 0.7	—	—	V	I_{OH} = -2.5 mA, V_{DD} = 4.5V, -40°C to +125°C
D092		OSC2/CLKOUT (RC mode)	V_{DD} - 0.7	—	—	V	I_{OH} = -1.3 mA, V_{DD} = 4.5V, -40°C to +85°C
D092A			V_{DD} - 0.7	—	—	V	I_{OH} = -1.0 mA, V_{DD} = 4.5V, -40°C to +125°C
D150	V_{OD}	**Open-drain High Voltage**	—	—	12	V	RA4 pin
		Programmable Current Source					
D170	V_{PCS}	Output Voltage Range	Vss	—	V_{DD} – 1.4	V	CDAC pin
D171	SN_{PCS}	Output Voltage Sensitivity	– 0.1	–0.01	—	%/V	Vss ≤ V_{CDAC} ≤ V_{DD} – 1.4
		Bandgap Reference					
D180	V_{BGR}	Output Voltage Range	1.14	1.19	1.24	V	on AN0 pin when AMUXOE =1 and ADCS3:ADSC0 = 0100b

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

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

3: Negative current is defined as current sourced by the pin.

30.9 I/O Capacitive Loading

These specifications indicate the conditions that the I/O pins have on them from the device tester. These loadings effect the specifications for the timing specifications. If the loading in you application are different, then you will need to determine how this will effect the characteristic of the device in your system. Capacitances less then these specifications should not have effects on a system.

Table 30-9: Example DC Characteristics

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C for commercial -40°C ≤ TA ≤ +85°C for industrial and -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC spec Table 30-3.				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
		Capacitive Loading Specs on Output Pins					
D100	COSC2	OSC2 pin	—	—	15	pF	In XT, HS and LP modes when external clock is used to drive OSC1.
D101	CIO	All I/O pins and OSC2 (in RC mode)	—	—	50	pF	To meet the Timing Specifications of the Device
D102	CB	SCL, SDA	—	—	400	pF	In I^2C mode

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

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

3: Negative current is defined as current sourced by the pin.

30.10 Data EEPROM / Flash

Table 30-10: Example Data EEPROM / Flash Characteristics

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C for commercial, -40°C ≤ TA ≤ +85°C for industrial and -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC spec Table 30-3.				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
		Data EEPROM Memory					
D120	ED	Endurance	1M	10M	—	E/W	25°C at 5V
D121	VDRW	VDD for read/write	VMIN	—	6.0	V	VMIN = Minimum operating voltage
D122	TDEW	Erase/Write cycle time	—	—	10	ms	
		Program Flash Memory					
D130	EP	Endurance	100	1000	—	E/W	
D131	VPR	VDD for read	VMIN	—	6.0	V	VMIN = Minimum operating voltage
D132	VPEW	VDD for erase/write	4.5	—	5.5	V	
D133	TPEW	Erase/Write cycle time	—	—	10	ms	

† Data in “Typ” column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

30.11 LCD

Table 30-11: Example LCD Module Electrical Characteristics

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C for commercial, -40°C ≤ TA ≤ +85°C for industrial and -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC spec Table 30-3.				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
D200	VLCD3	LCD Voltage on pin VLCD3	VDD - 0.3	—	Vss + 7.0	V	
D201	VLCD2	LCD Voltage on pin VLCD2	—	—	VLCD3	V	
D202	VLCD1	LCD Voltage on pin VLCD1	—	—	VDD	V	
D210	RCOM	Com Output Source Impedance	—	—	1k	Ω	COM outputs
D211	RSEG	Seg Output Source Impedance	—	—	10k	Ω	SEG outputs
D220	VOH	Output High Voltage	Max (VLCDN) - 0.1	—	Max (VLCDN)	V	COM outputs IOH = 25 µA SEG outputs IOH = 3 µA
D221	VOL	Output Low Voltage	Min (VLCDN)	—	Min (VLCDN) + 0.1	V	COM outputs IOL = 25 µA SEG outputs IOL = 3 µA

† 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: 0 ohm source impedance at VLCD.

Table 30-12: Example VLCD Charge Pump Electrical Characteristics

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C for commercial, -40°C ≤ TA ≤ +85°C for industrial and -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC spec Table 30-3.				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
D250	IVADJ	VLCDADJ Regulated Current Output	—	10	—	µA	
D251	Ivr	VLCDADJ Current Consumption	—	—	20	µA	
D252	$\frac{\Delta I_{VADJ}}{\Delta V_{DD}}$	VLCDADJ Current VDD Rejection	—	—	0.1/1	µA/V	
D253	$\frac{\Delta I_{VADJ}}{\Delta T}$	VLCDADJ Current Variation With Temperature	—	—	0.1/70	µA/°C	
D260(1)	RVADJ	VLCDADJ External Resistor	100	—	230	kΩ	
D265	VVADJ	VLCDADJ Voltage Limits	1.0	—	2.3	V	
D271(1)	CECPC	External Charge Pump Capacitance	—	0.5	—	µF	

Note 1: For design guidance only.

30.12 Comparators and Voltage Reference

Table 30-13: Example Comparator Characteristics

DC CHARACTERISTICS				Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C for commercial, -40°C ≤ TA ≤ +85°C for industrial and -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC spec Table 30-3.				
Param No.	**Symbol**	**Characteristics**		**Min**	**Typ**	**Max**	**Units**	**Comments**
D300	VIOFF	Input offset voltage		—	± 5.0	± 10	mV	
D301	VICM	Input common mode voltage		0	—	VDD - 1.5	V	
D302	CMRR	Common Mode Rejection Ratio		35	70	—	db	
300	TRESP	Response Time(1)	PIC16**C**XXX	—	150	400	ns	
300A			PIC16**LC**XXX	—	210	600	ns	
301	TMC2OV	Comparator Mode Change to Output Valid		—	—	10	µs	

Note 1: Response time measured with one comparator input at (VDD - 1.5)/2 while the other input transitions from VSS to VDD.

Table 30-14: Example Voltage Reference Characteristics

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C for commercial, -40°C ≤ TA ≤ +85°C for industrial and -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC spec Table 30-3.				
Param No.	**Symbol**	**Characteristics**	**Min**	**Typ**	**Max**	**Units**	**Comments**
D310	VRES	Resolution	VDD/32	—	VDD/24	V	
D311	VRAA	Absolute Accuracy	— —	— —	1/4 1/2	LSb LSb	Low Range (VRR = 1) High Range (VRR = 0)
D312	VRUR	Unit Resistor Value (R)	—	2k	—	Ω	
310	TSET	Settling Time(1)	—	—	10	µs	

Note 1: Settling time measured while VRR = 1 and VR3:VR0 transitions from 0000 to 1111.

30.13 Timing Parameter Symbology

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

1. TppS2ppS
2. TppS
3. TCC:ST (I^2C specifications only)
4. Ts (I^2C specifications only)

T			
F	Frequency	T	Time

Lowercase letters (pp) and their meanings:

pp			
cc	CCP1	osc	OSC1
ck	CLKOUT	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ or $\overline{WR}$
di	SDI	sc	SCK
do	SDO	ss	$\overline{SS}$
dt	Data in	t0	T0CKI
io	I/O port	t1	T1CKI
mc	$\overline{MCLR}$	wr	$\overline{WR}$

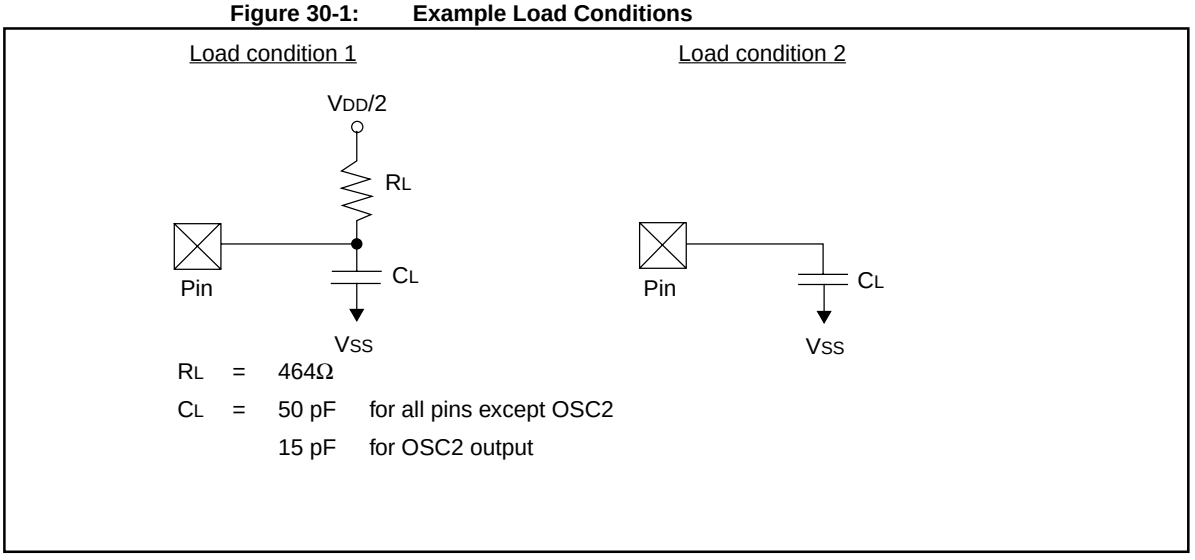
Uppercase letters and their meanings:

S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (Hi-impedance)	V	Valid
L	Low	Z	Hi-impedance
I^2C only			
AA	output access	High	High
BUF	Bus free	Low	Low

TCC:ST (I^2C specifications only)

CC			
HD	Hold	SU	Setup
ST			
DAT	DATA input hold	STO	STOP condition
STA	START condition		

Figure 30-1: Example Load Conditions



30.14 Example External Clock Timing Waveforms and Requirements

Figure 30-2: Example External Clock Timing Waveforms

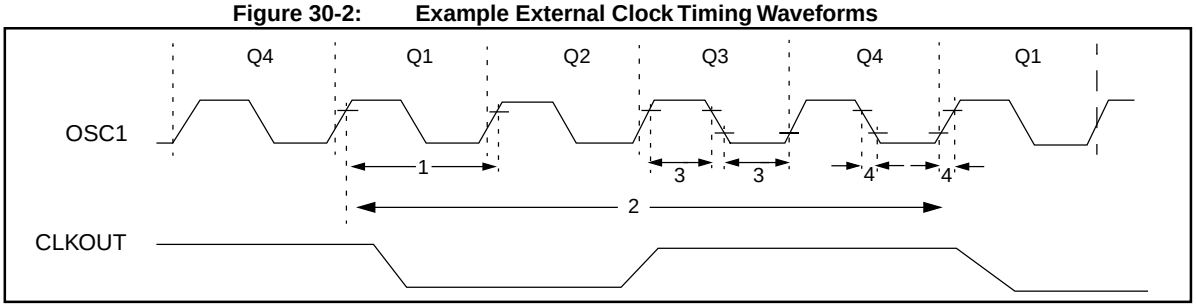


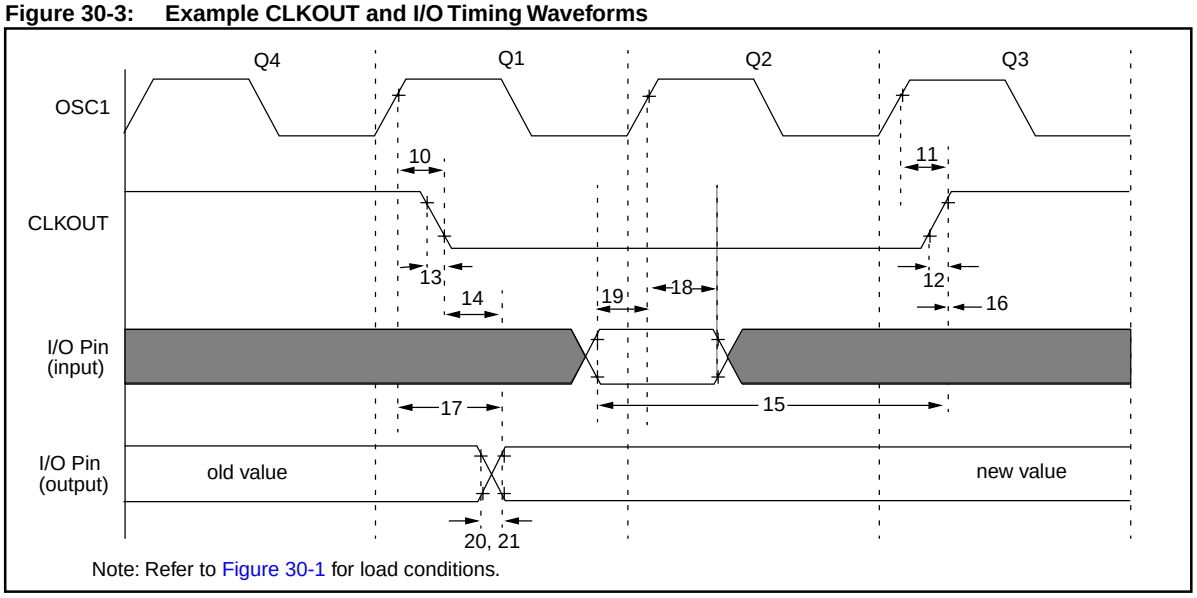
Table 30-15: Example External Clock Timing Requirements

Param. No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions	
1A	Fosc	External CLKIN Frequency(1)	DC	—	4	MHz	XT and RC osc	PIC16**C**XXX-**04**
								PIC16**LC**XXX-**04**
			DC	—	10	MHz	HS osc	PIC16**C**XXX-**10**
			DC	—	20	MHz		PIC16**C**XXX-**20**
			DC	—	200	kHz	LP osc	PIC16**LC**XXX-**04**
		Oscillator Frequency(1)	DC	—	4	MHz	RC osc	PIC16**C**XXX-**04**
								PIC16**LC**XXX-**04**
			0.1	—	4	MHz	XT osc	PIC16**C**XXX-**04**
								PIC16**LC**XXX-**04**
			4	—	10	MHz	HS osc	PIC16**C**XXX-**10**
			4	—	20	MHz		PIC16**C**XXX-**20**
			5	—	200	kHz	LP osc mode	PIC16**LC**XXX-**04**
1	Tosc	External CLKIN Period(1)	250	—	—	ns	XT and RC osc	PIC16**C**XXX-**04**
								PIC16**LC**XXX-**04**
			100	—	—	ns	HS osc	PIC16**C**XXX-**10**
			50	—	—	ns		PIC16**C**XXX-**20**
			5	—	—	µs	LP osc	PIC16**LC**XXX-**04**
		Oscillator Period(1)	250	—	—	ns	RC osc	PIC16**C**XXX-**04**
								PIC16**LC**XXX-**04**
			250	—	10,000	ns	XT osc	PIC16**C**XXX-**04**
								PIC16**LC**XXX-**04**
			100	—	250	ns	HS osc	PIC16**C**XXX-**10**
			50	—	250	ns		PIC16**C**XXX-**20**
			5	—	—	µs	LP osc	PIC16**LC**XXX-**04**
2	TCY	Instruction Cycle Time(1)	200	—	DC	ns	TCY = 4/FOSC	
3	TosL, TosH	External Clock in (OSC1) High or Low Time	50	—	—	ns	XT osc	PIC16**C**XXX-**04**
			60	—	—	ns	XT osc	PIC16**LC**XXX-**04**
			2.5	—	—	µs	LP osc	PIC16**LC**XXX-**04**
			15	—	—	ns	HS osc	PIC16**C**XXX-**20**
4	TosR, TosF	External Clock in (OSC1) Rise or Fall Time	—	—	25	ns	XT osc	PIC16**C**XXX-**04**
			—	—	50	ns	LP osc	PIC16**LC**XXX-**04**
			—	—	15	ns	HS osc	PIC16**C**XXX-**20**

† 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: 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 30-3: Example CLKOUT and I/O Timing Waveforms



Note: Refer to Figure 30-1 for load conditions.

Table 30-16: Example CLKOUT and I/O Timing Requirements

Param. No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
10	TosH2ckL	OSC1↑ to CLKOUT↓		—	75	200	ns	(1)
11	TosH2ckH	OSC1↑ to CLKOUT↑		—	75	200	ns	(1)
12	TckR	CLKOUT rise time		—	35	100	ns	(1)
13	TckF	CLKOUT fall time		—	35	100	ns	(1)
14	TckL2ioV	CLKOUT ↓ to Port out valid		—	—	0.5TCY + 20	ns	(1)
15	TioV2ckH	Port in valid before CLKOUT ↑		0.25TCY + 25	—	—	ns	(1)
16	TckH2ioI	Port in hold after CLKOUT ↑		0	—	—	ns	(1)
17	TosH2ioV	OSC1↑ (Q1 cycle) to Port out valid		—	50	150	ns	
18	TosH2ioI	OSC1↑ (Q2 cycle) to Port input invalid (I/O in hold time)	PIC16**C**XXX	100	—	—	ns	
18A			PIC16**LC**XXX	200	—	—	ns	
19	TioV2osH	Port input valid to OSC1↑ (I/O in setup time)		0	—	—	ns	
20	TioR	Port output rise time	PIC16**C**XXX	—	10	25	ns	
20A			PIC16**LC**XXX	—	—	60	ns	
21	TioF	Port output fall time	PIC16**C**XXX	—	10	25	ns	
21A			PIC16**LC**XXX	—	—	60	ns	
22††	Tinp	INT pin high or low time		TCY	—	—	ns	
23††	Trbp	RB7:RB4 change INT high or low time		TCY	—	—	ns	
24††	Trcp	RC7:RC4 change INT high or low time		20			ns	

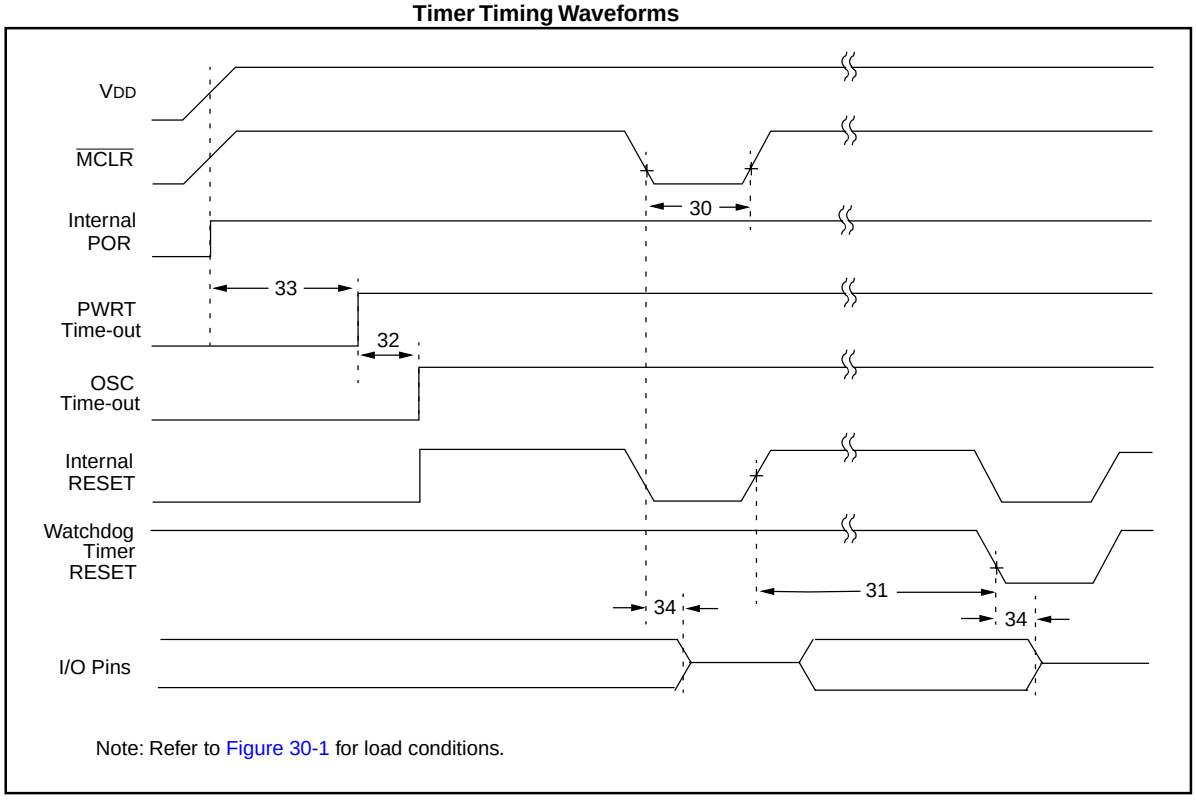
† Data in "Typ" column is at 5V, 25˚C unless otherwise stated. These parameters are for design guidance only and are not tested.

†† These parameters are asynchronous events not related to any internal clock edges.

Note 1: Measurements are taken in RC Mode where CLKOUT output is 4 x TOSC.

30.15 Example Power-up and Reset Timing Waveforms and Requirements

Figure 30-4: Example Reset, Watchdog Timer, Oscillator Start-up Timer and Power-up Timer Timing Waveforms



Note: Refer to Figure 30-1 for load conditions.

Figure 30-5: Brown-out Reset Timing

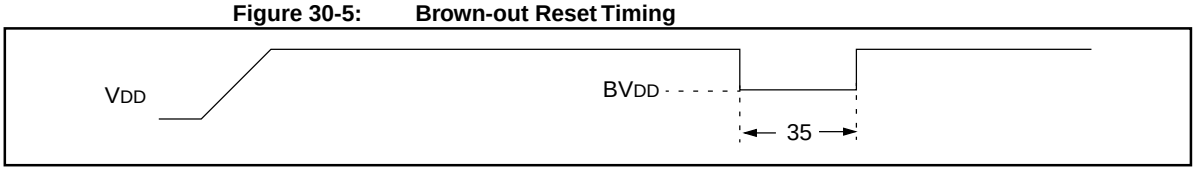


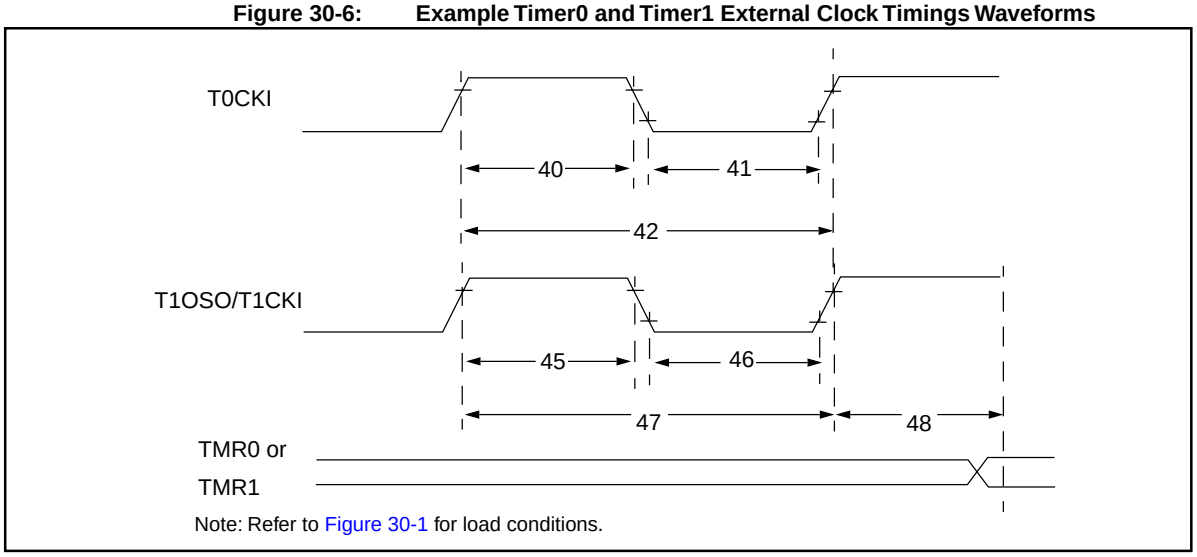
Table 30-17: Example Reset, Watchdog Timer, Oscillator Start-up Timer, Brown-out Reset, and Power-up Timer Requirements

Param. No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
30	TmcL	MCLR Pulse Width (low)	2	—	—	µs	VDD = 5V, -40˚C to +125˚C
31	Twdt	Watchdog Timer Time-out Period (No Prescaler)	7	18	33	ms	VDD = 5V, -40˚C to +125˚C
32	Tost	Oscillation Start-up Timer Period	—	1024TOSC	—	—	TOSC = OSC1 period
33	Tpwrt	Power up Timer Period	28	72	132	ms	VDD = 5V, -40˚C to +125˚C
34	TIOZ	I/O Hi-impedance from MCLR Low or Watchdog Timer Reset	—	—	2.1	µs	
35	TBOR	Brown-out Reset Pulse Width	100	—	—	µs	VDD ≤ BVDD (See D005)

† Data in "Typ" column is at 5V, 25˚C unless otherwise stated. These parameters are for design guidance only and are not tested.

30.16 Example Timer0 and Timer1 Timing Waveforms and Requirements

Figure 30-6: Example Timer0 and Timer1 External Clock Timings Waveforms



Note: Refer to Figure 30-1 for load conditions.

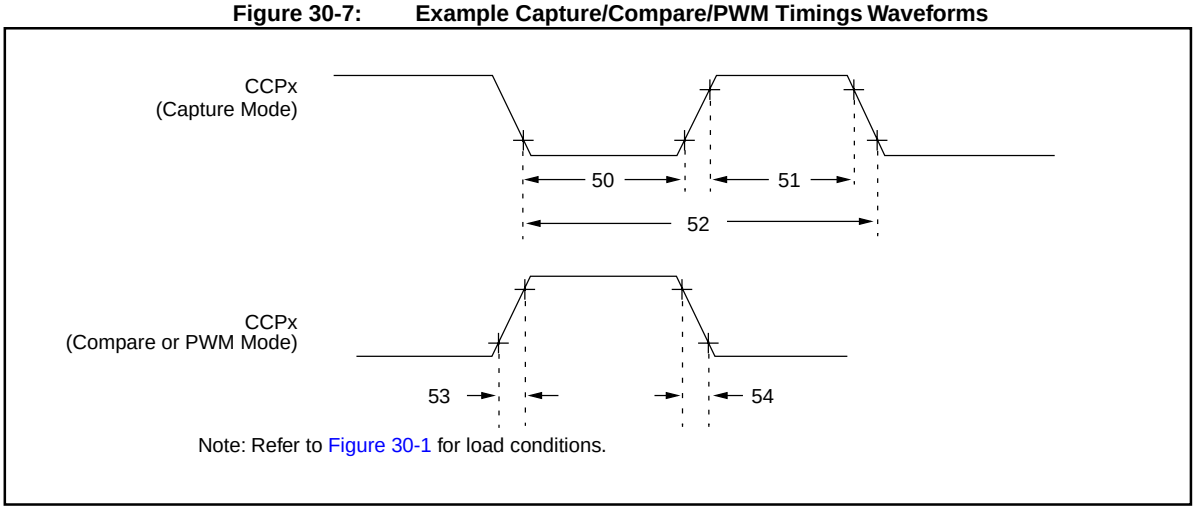
Table 30-18: Example Timer0 and Timer1 External Clock Requirements

Param No.	Symbol	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 µS OR TCY + 40 / N	—	—	ns	N = prescale value (1, 2, 4,..., 256)
45	Tt1H	T1CKI High Time	Synchronous, no prescaler		0.5TCY + 20	—	—	ns	
			Synchronous, with prescaler	PIC16**C**XXX	15	—	—	ns	
				PIC16**LC**XXX	25	—	—	ns	
			Asynchronous	PIC16**C**XXX	30	—	—	ns	
				PIC16**LC**XXX	50	—	—	ns	
46	Tt1L	T1CKI Low Time	Synchronous, no prescaler		0.5TCY + 20	—	—	ns	
			Synchronous, with prescaler	PIC16**C**XXX	15	—	—	ns	
				PIC16**LC**XXX	25	—	—	ns	
			Asynchronous	PIC16**C**XXX	2TCY	—	—	ns	
				PIC16**LC**XXX					
47	Tt1P	T1CKI input period	Synchronous		GREATER OF: 20 µS OR TCY + 40 / N	—	—	ns	N = prescale value (1, 2, 4, 8)
			Asynchronous		Greater of: 20µS or 4TCY	—	—	ns	
	Ft1	Timer1 oscillator input frequency range (oscillator enabled by setting the T1OSCEN bit)			DC	—	200	kHz	
48	Tcke2tmrI	Delay from external clock edge to timer increment			2Tosc	—	7Tosc	—	

† Data in "Typ" column is at 5V, 25˚C unless otherwise stated. These parameters are for design guidance only and are not tested.

30.17 Example CCP Timing Waveforms and Requirements

Figure 30-7: Example Capture/Compare/PWM Timings Waveforms



Note: Refer to Figure 30-1 for load conditions.

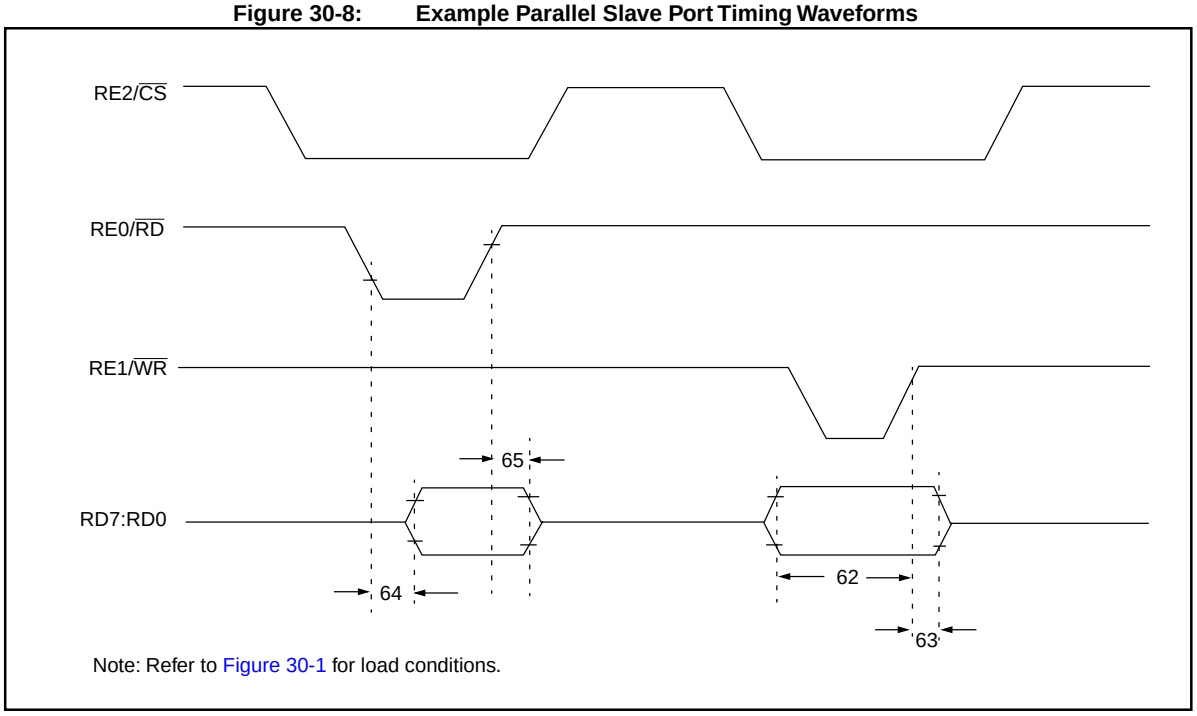
Table 30-19: Example Capture/Compare/PWM Requirements

Param. No.	Symbol	Characteristic			Min	Typ†	Max	Units	Conditions
50	TccL	CCPx input low time	No Prescaler		0.5TCY + 20	—	—	ns	
			With Prescaler	PIC16**C**XXX	10	—	—	ns	
				PIC16**LC**XXX	20	—	—	ns	
51	TccH	CCPx input high time	No Prescaler		0.5TCY + 20	—	—	ns	
			With Prescaler	PIC16**C**XXX	10	—	—	ns	
				PIC16**LC**XXX	20	—	—	ns	
52	TccP	CCPx input period			$\frac{3T_{CY} + 40}{N}$	—	—	ns	N = prescale value (1,4 or 16)
53	TccR	CCPx output fall time		PIC16**C**XXX	—	10	25	ns	
				PIC16**LC**XXX	—	25	45	ns	
54	TccF	CCPx output fall time		PIC16**C**XXX	—	10	25	ns	
				PIC16**LC**XXX	—	25	45	ns	

† Data in "Typ" column is at 5V, 25˚C unless otherwise stated. These parameters are for design guidance only and are not tested.

30.18 Example Parallel Slave Port (PSP) Timing Waveforms and Requirements

Figure 30-8: Example Parallel Slave Port Timing Waveforms



Note: Refer to Figure 30-1 for load conditions.

Table 30-20: Example Parallel Slave Port Requirements

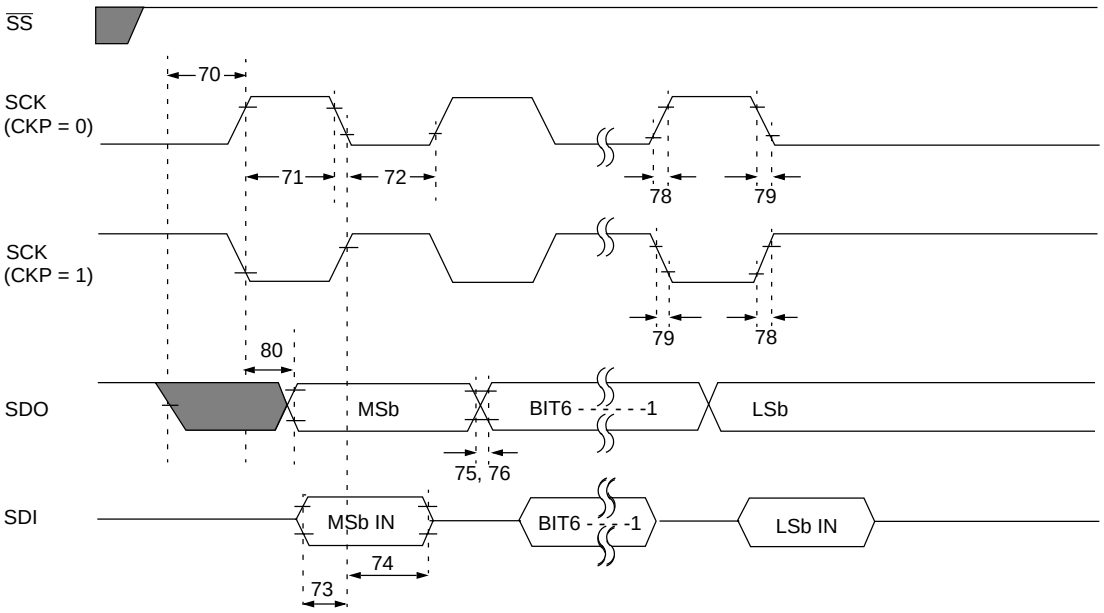
Param. No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
62	TdtV2wrH	Data in valid before $\overline{WR}\uparrow$ or $\overline{CS}\uparrow$ (setup time)		20	—	—	ns	
63	TwrH2dtI	$\overline{WR}\uparrow$ or $\overline{CS}\uparrow$ to data–in invalid (hold time)	PIC16**C**XXX	20	—	—	ns	
			PIC16**LC**XXX	35	—	—	ns	
64	TrdL2dtV	$\overline{RD}\downarrow$ and $\overline{CS}\downarrow$ to data–out valid		—	—	80	ns	
65	TrdH2dtI	$\overline{RD}\uparrow$ or $\overline{CS}\downarrow$ to data–out invalid		10	—	30	ns	
66	TibfINH	Inhibit of the IBF flag bit being cleared from $\overline{WR}\uparrow$ or $\overline{CS}\uparrow$		—	—	3Tcy§		

† Data in "Typ" column is at 5V, 25˚C unless otherwise stated. These parameters are for design guidance only and are not tested.

§ This specification ensured by design.

30.19 Example SSP and Master SSP SPI Mode Timing Waveforms and Requirements

Figure 30-9: Example SPI Master Mode Timing (CKE = 0)



Refer to Figure 30-1 for load conditions.

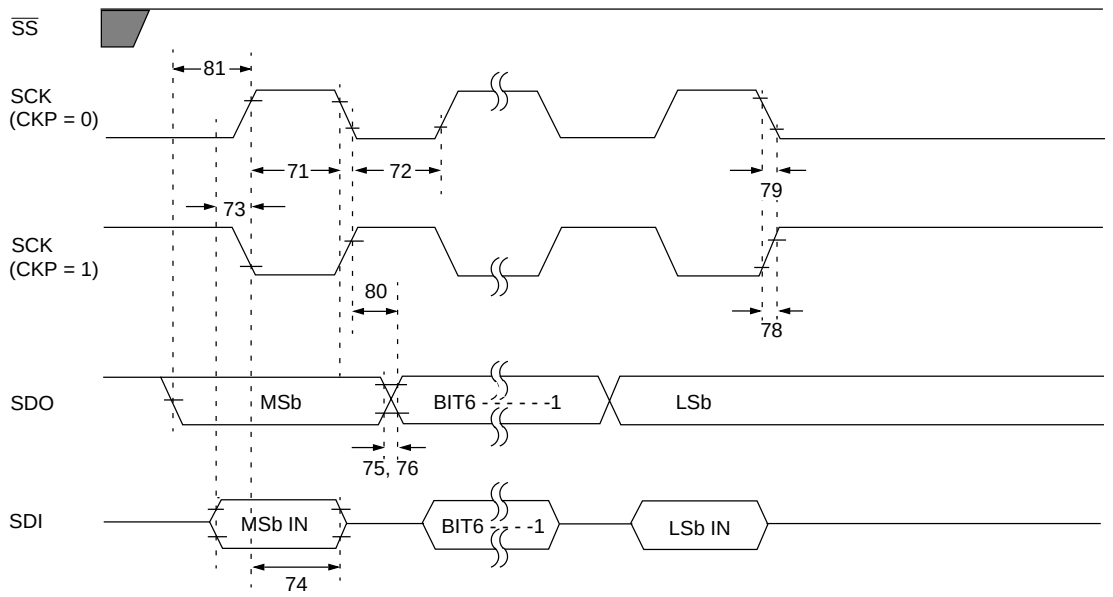
Table 30-21: Example SPI Mode Requirements (Master Mode, CKE = 0)

Param. No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SS}$↓ 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
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	PIC16**C**XXX	—	10	25	ns	
			PIC16**LC**XXX	—	20	45	ns	
76	TdoF	SDO data output fall time		—	10	25	ns	
78	TscR	SCK output rise time (master mode)	PIC16**C**XXX	—	10	25	ns	
			PIC16**LC**XXX	—	20	45	ns	
79	TscF	SCK output fall time (master mode)		—	10	25	ns	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	PIC16**C**XXX	—	—	50	ns	
			PIC16**LC**XXX	—	—	100	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: Specification 73A is only required if specifications 71A and 72A are used.

Figure 30-10: Example SPI Master Mode Timing (CKE = 1)



Refer to Figure 30-1 for load conditions.

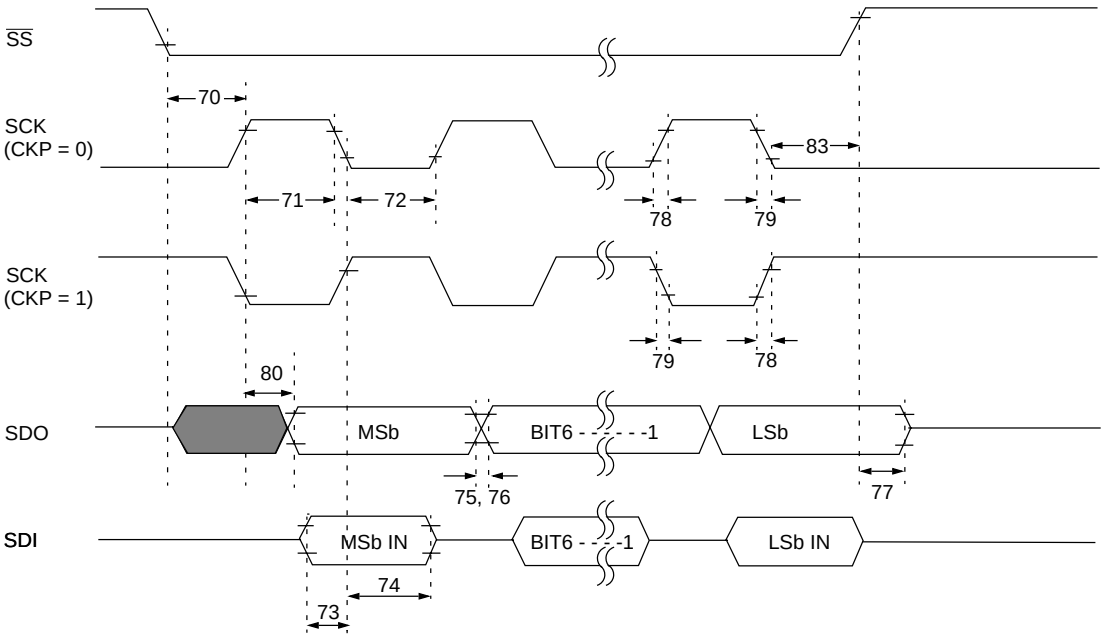
Table 30-22: Example SPI Mode Requirements (Master Mode, CKE = 1)

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	1.25TCY + 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	PIC16**C**XXX	—	10	25	ns	
			PIC16**LC**XXX		20	45	ns	
76	TdoF	SDO data output fall time		—	10	25	ns	
78	TscR	SCK output rise time (master mode)	PIC16**C**XXX	—	10	25	ns	
			PIC16**LC**XXX		20	45	ns	
79	TscF	SCK output fall time (master mode)		—	10	25	ns	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	PIC16**C**XXX	—	—	50	ns	
			PIC16**LC**XXX		—	100	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. These parameters are for design guidance only and are not tested.

Note 1: Specification 73A is only required if specifications 71A and 72A are used.

Figure 30-11: Example SPI Slave Mode Timing (CKE = 0)



Refer to Figure 30-1 for load conditions.

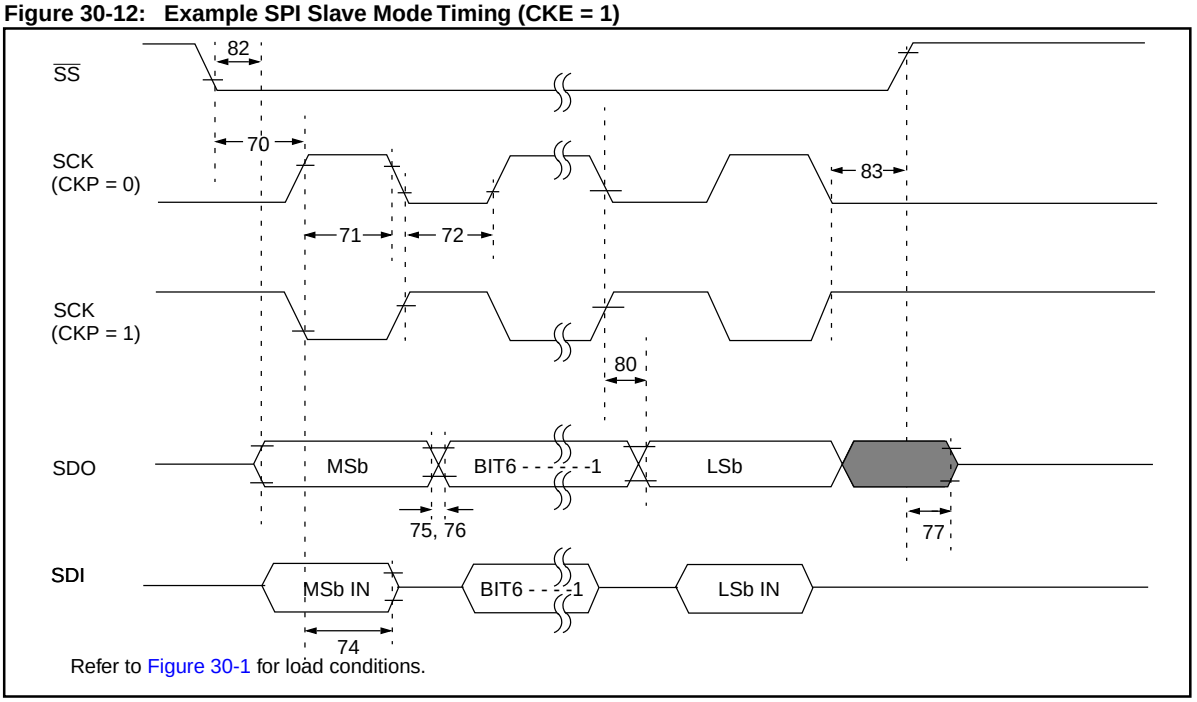
Table 30-23: Example SPI Mode Requirements (Slave Mode Timing (CKE = 0)

Param. No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SS}$↓ 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
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	PIC16**C**XXX	—	10	25	ns	
			PIC16**LC**XXX		20	45	ns	
76	TdoF	SDO data output fall time		—	10	25	ns	
77	TssH2doZ	$\overline{SS}$↑ to SDO output hi-impedance		10	—	50	ns	
78	TscR	SCK output rise time (master mode)	PIC16**C**XXX	—	10	25	ns	
			PIC16**LC**XXX		20	45	ns	
79	TscF	SCK output fall time (master mode)		—	10	25	ns	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	PIC16**C**XXX	—	—	50	ns	
			PIC16**LC**XXX		—	100	ns	
83	TscH2ssH, TscL2ssH	$\overline{SS}$ ↑ after SCK edge		1.5TCY + 40	—	—	ns	

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Specification 73A is only required if specifications 71A and 72A are used.

Figure 30-12: Example SPI Slave Mode Timing (CKE = 1)



Refer to Figure 30-1 for load conditions.

Table 30-24: Example SPI Slave Mode Mode Requirements (CKE = 1)

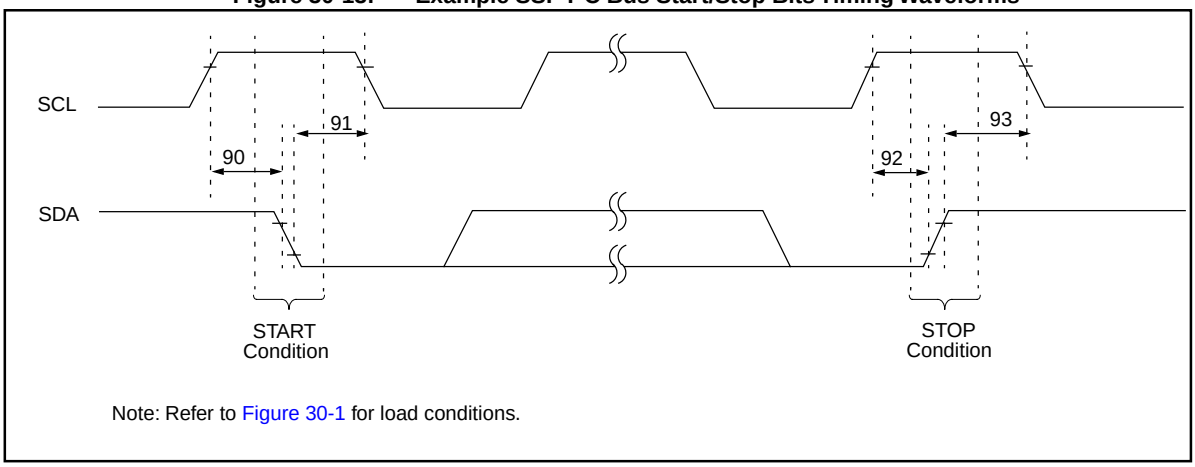
Param. No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SS}\downarrow$ to SCK$\downarrow$ or SCK$\uparrow$ input		TCY	—	—	ns	
71	TscH	SCK input high time (slave mode)	Continuous	1.25TCY + 30	—	—	ns	
71A			Single Byte	40	—	—	ns	Note 1
72	TscL	SCK input low time (slave mode)	Continuous	1.25TCY + 30	—	—	ns	
72A			Single Byte	40	—	—	ns	Note 1
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	PIC16**C**XXX	—	10	25	ns	
			PIC16**LC**XXX		20	45	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)	PIC16**C**XXX	—	10	25	ns	
			PIC16**LC**XXX	—	20	45	ns	
79	TscF	SCK output fall time (master mode)		—	10	25	ns	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	PIC16**C**XXX	—	—	50	ns	
			PIC16**LC**XXX	—	—	100	ns	
82	TssL2doV	SDO data output valid after $\overline{SS}\downarrow$ edge	PIC16**C**XXX	—	—	50	ns	
			PIC16**LC**XXX	—	—	100	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. These parameters are for design guidance only and are not tested.

Note 1: Specification 73A is only required if specifications 71A and 72A are used.

30.20 Example SSP I²C Mode Timing Waveforms and Requirements

Figure 30-13: Example SSP I²C Bus Start/Stop Bits Timing Waveforms

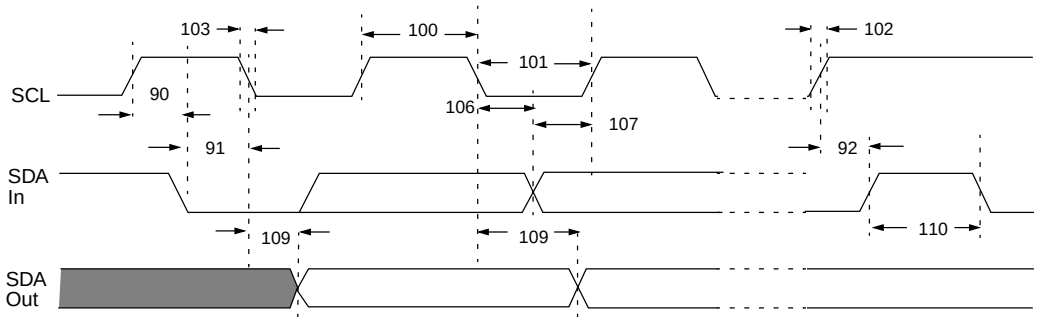


Note: Refer to Figure 30-1 for load conditions.

Table 30-25: Example SSP I²C Bus Start/Stop Bits Requirements

Param. No.	Symbol	Characteristic		Min	Typ	Max	Units	Conditions
90	TSU:STA	START condition	100 kHz mode	4700	—	—	ns	Only relevant for repeated START condition
		Setup time	400 kHz mode	600	—	—		
91	THD:STA	START condition	100 kHz mode	4000	—	—	ns	After this period the first clock pulse is generated
		Hold time	400 kHz mode	600	—	—		
92	TSU:STO	STOP condition	100 kHz mode	4700	—	—	ns	
		Setup time	400 kHz mode	600	—	—		
93	THD:STO	STOP condition	100 kHz mode	4000	—	—	ns	
		Hold time	400 kHz mode	600	—	—		

Figure 30-14: Example SSP I²C Bus Data Timing Waveforms



Note: Refer to Figure 30-1 for load conditions.

Table 30-26: Example SSP I^2C Bus Data Requirements

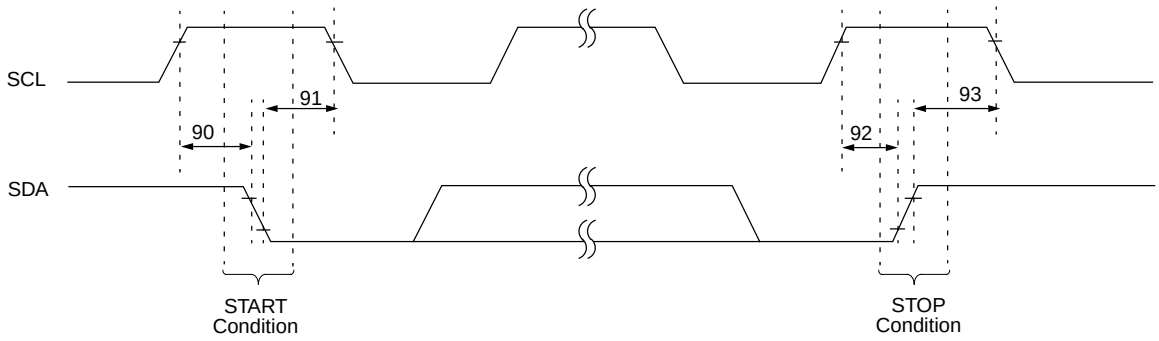
Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock high time	100 kHz mode	4.0	—	µs	PIC16CXXX must operate at a minimum of 1.5 MHz
			400 kHz mode	0.6	—	µs	PIC16CXXX must operate at a minimum of 10 MHz
			SSP Module	1.5TCY	—		
101	TLOW	Clock low time	100 kHz mode	4.7	—	µs	PIC16CXXX must operate at a minimum of 1.5 MHz
			400 kHz mode	1.3	—	µs	PIC16CXXX must operate at a minimum of 10 MHz
			SSP Module	1.5TCY	—		
102	TR	SDA and SCL rise time	100 kHz mode	—	1000	ns	
			400 kHz mode	20 + 0.1Cb	300	ns	Cb is specified to be from 10 to 400 pF
103	TF	SDA and SCL fall time	100 kHz mode	—	300	ns	
			400 kHz mode	20 + 0.1Cb	300	ns	Cb is specified to be from 10 to 400 pF
90	TSU:STA	START condition setup time	100 kHz mode	4.7	—	µs	Only relevant for repeated START condition
			400 kHz mode	0.6	—	µs	
91	THD:STA	START condition hold time	100 kHz mode	4.0	—	µs	After this period the first clock pulse is generated
			400 kHz mode	0.6	—	µs	
106	THD:DAT	Data input hold time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	µs	
107	TSU:DAT	Data input setup time	100 kHz mode	250	—	ns	Note 2
			400 kHz mode	100	—	ns	
92	TSU:STO	STOP condition setup time	100 kHz mode	4.7	—	µs	
			400 kHz mode	0.6	—	µs	
109	TAA	Output valid from clock	100 kHz mode	—	3500	ns	Note 1
			400 kHz mode	—	—	ns	
110	TBUF	Bus free time	100 kHz mode	4.7	—	µs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	µs	
D102	Cb	Bus capacitive loading		—	400	pF	

Note 1: As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of START or STOP conditions.

2: A fast-mode I^2C-bus device can be used in a standard-mode I^2C-bus system, but the requirement tsu;DAT ≥ 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.
TR max. + tsu;DAT = 1000 + 250 = 1250 ns (according to the standard-mode I^2C bus specification) before the SCL line is released.

30.21 Example Master SSP I²C Mode Timing Waveforms and Requirements

Figure 30-15: Example Master SSP I²C Bus Start/Stop Bits Timing Waveforms



Note: Refer to Figure 30-1 for load conditions.

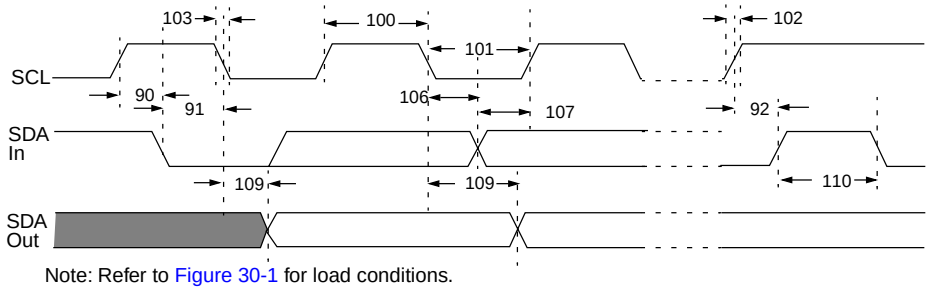
Table 30-27: Example Master SSP I²C Bus Start/Stop Bits Requirements

Param. No.	Symbol	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) §	—	—		

§ This specification ensured by design. For the value required by the I²C specification, please refer to Figure A-11 of the **"Appendix."**

Maximum pin capacitance = 10 pF for all I²C pins.

Figure 30-16: Example Master SSP I²C Bus Data Timing



Note: Refer to Figure 30-1 for load conditions.

Table 30-28: Example Master SSP I²C Bus Data Requirements

Param. No.	Symbol	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)	—	100	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)	TBD	—	ns	
107	TSU:DAT	Data input setup time	100 kHz mode	250	—	ns	Note 2
			400 kHz mode	100	—	ns	
			1 MHz mode (1)	TBD	—	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)	—	—	ns	
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)	TBD	—	ms	
D102 ‡	Cb	Bus capacitive loading		—	400	pF	

§ This specification ensured by design. For the value required by the I²C specification, please refer to Figure A-11 of the **"Appendix."**

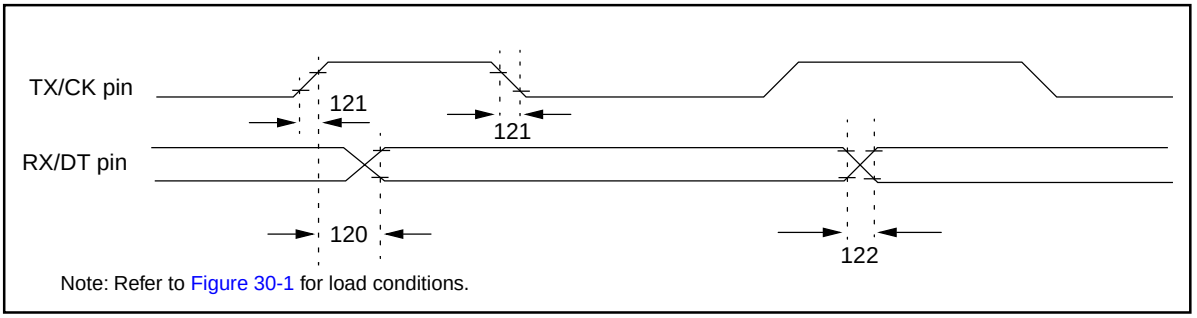
‡ These parameters are for design guidance only and are not tested, nor characterized.

Note 1: Maximum pin capacitance = 10 pF for all I²C pins.

2: A fast-mode I²C-bus device can be used in a standard-mode I²C-bus system, but 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.+ parameter 107 = 1000 + 250 = 1250 ns (for 100 kHz-mode) before the SCL line is released.

30.22 Example USART/SCI Timing Waveforms and Requirements

Figure 30-17: Example USART Synchronous Transmission (Master/Slave) Timing Waveforms



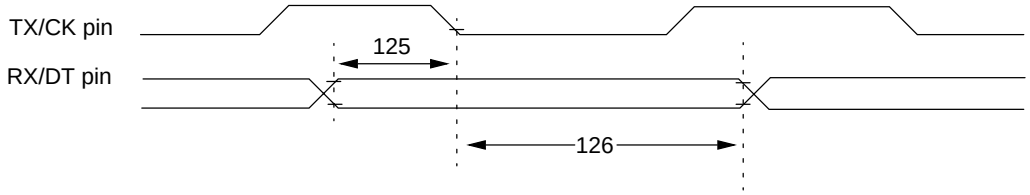
Note: Refer to Figure 30-1 for load conditions.

Table 30-29: Example USART Synchronous Transmission Requirements

Param. No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
120	TckH2dtV	SYNC XMIT (MASTER & SLAVE) Clock high to data out valid	PIC16**C**XXX	—	—	80	ns	
			PIC16**LC**XXX	—	—	100	ns	
121	Tckrf	Clock out rise time and fall time (Master Mode)	PIC16**C**XXX	—	—	45	ns	
			PIC16**LC**XXX	—	—	50	ns	
122	Tdtrf	Data out rise time and fall time	PIC16**C**XXX	—	—	45	ns	
			PIC16**LC**XXX	—	—	50	ns	

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Figure 30-18: Example USART Synchronous Receive (Master/Slave) Timing Waveforms



Note: Refer to Figure 30-1 for load conditions.

Table 30-2: Example USART Synchronous Receive Requirements

Param. No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
125	TdtV2ckl	SYNC RCV (MASTER & SLAVE) Data hold before CK ↓ (DT hold time)	15	—	—	ns	
126	TckL2dtl	Data hold after CK ↓ (DT hold time)	15	—	—	ns	

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

30.23 Example 8-bit A/D Timing Waveforms and Requirements

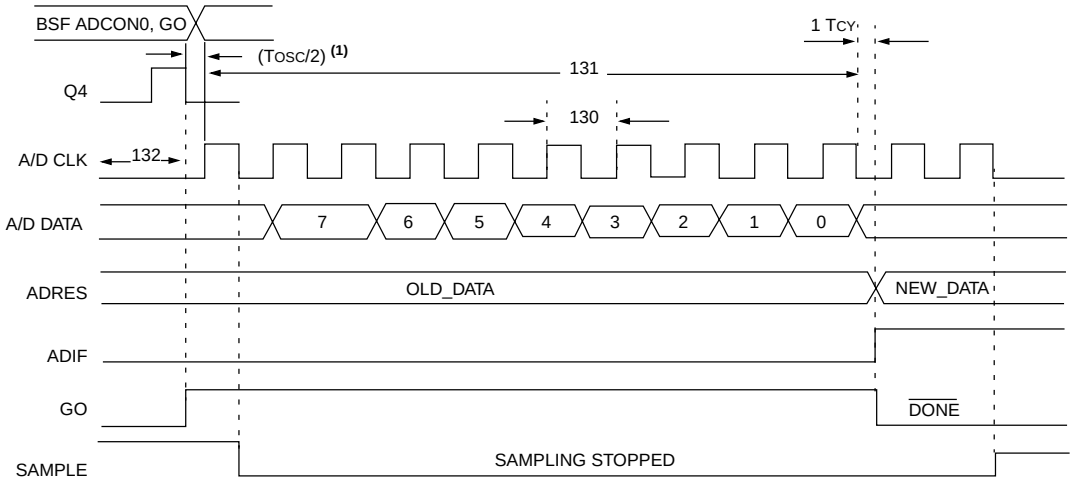
Table 30-30: Example 8-bit A/D Converter Characteristics

Param No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
A01	NR	Resolution		—	—	8-bits	bit	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A02	EABS	Total Absolute error		—	—	< ± 1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A03	EIL	Integral linearity error		—	—	< ± 1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A04	EDL	Differential linearity error		—	—	< ± 1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A05	EFS	Full scale error		—	—	< ± 1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A06	EOFF	Offset error		—	—	< ± 1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A10	—	Monotonicity		—	guaran-teed	—	—	VSS ≤ VAIN ≤ VREF
A20	VREF	Reference voltage		3.0V	—	VDD + 0.3	V	
A25	VAIN	Analog input voltage		VSS - 0.3	—	VREF + 0.3	V	
A30	ZAIN	Recommended impedance of analog voltage source		—	—	10.0	kΩ	
A40	IAD	A/D conversion current (VDD)	PIC16**C**XXX	—	180	—	µA	Average current consumption when A/D is on**(Note 1)**
			PIC16**LC**XXX	—	90	—	µA	
A50	IREF	VREF input current (Note 2)		10	—	1000	µA	During VAIN acquisition. Based on differential of VHOLD to VAIN to charge CHOLD See the **"8-bit A/D Converter"** section
				—	—	10	µA	During A/D Conversion cycle

† 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: 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. VREF current is from RA3 pin or VDD pin, whichever is selected as reference input.

Figure 30-19: Example 8-bit A/D Conversion Timing Waveforms



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 30-31: Example 8-bit A/D Conversion Requirements

Param No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
130	TAD	A/D clock period	PIC16**C**XXX	1.6	—	—	µs	TOSC based, VREF ≥ 3.0V
			PIC16**LC**XXX	2.0	—	—	µs	TOSC based, VREF full range
			PIC16**C**XXX	2.0	4.0	6.0	µs	A/D RC Mode
			PIC16**LC**XXX	3.0	6.0	9.0	µs	A/D RC Mode
131	TCNV	Conversion time (not including S/H time) (Note 1)		11	—	11	TAD	
132	TACQ	Acquisition time		Note 2 5	20 —	— —	µs µs	The minimum time is the amplifier settling time. This may be used if the "new" input voltage has not changed by more than 1 LSb (i.e., 20.0 mV @ 5.12V) from the last sampled voltage (as stated on CHOLD).
134	TGO	Q4 to A/D clock start		—	2TOSC §	—	—	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.
136	TAMP	Amplifier settling time (Note 2)		1	—	—	µs	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).
135	TSWC	Switching Time from convert → sample		1 §	—	1 §	TAD	

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

§ This specification ensured by design.

Note 1: ADRES register may be read on the following TCY cycle. See the **"8-bit A/D Converter"** section for minimum requirements.

30.24 Example 10-bit A/D Timing Waveforms and Requirements

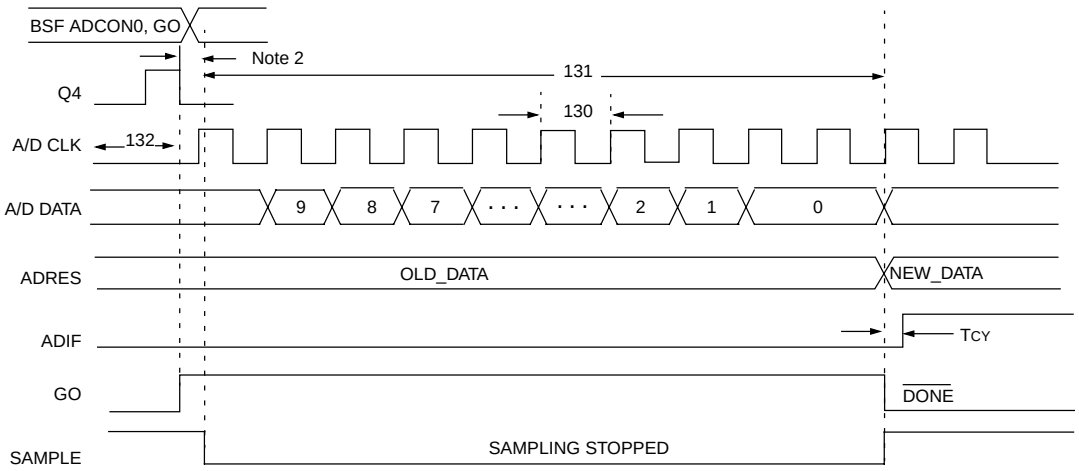
Table 30-32: Example 10-bit A/D Converter Characteristics

Param No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
A01	NR	Resolution		—	—	10	bit	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A02	EABS	Absolute error		—	—	<±1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A03	EIL	Integral linearity error		—	—	<±1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A04	EDL	Differential linearity error		—	—	<±1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A05	EFS	Full scale error		—	—	<±1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A06	EOFF	Offset error		—	—	<±1	LSb	VREF = VDD = 5.12V, VSS ≤ VAIN ≤ VREF
A10	—	Monotonicity		—	guaran-teed	—	—	VSS ≤ VAIN ≤ VREF
A20	VREF	Reference voltage (VREFH - VREFL)		0V	—	—	V	For no latch-up
A20A				3V	—	—	V	For 10-bit resolution
A21	VREFH	Reference voltage High		AVSS	—	AVDD + 0.3V	V	
A22	VREFL	Reference voltage Low		AVSS - 0.3V	—	AVDD	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)	PIC16**C**XXX	—	180	—	µA	Average current consumption when A/D is on. (Note 1)
			PIC16**LC**XXX	—	90	—	µA	
A50	IREF	VREF input current (Note 2)		10	—	1000	µA	During VAIN acquisition. Based on differential of VHOLD to VAIN. To charge CHOLD see the **"10-bit A/D Converter"** section.
				—	—	10	µA	During A/D conversion cycle

† 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: 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.
VREF current is from RG0 and RG1 pins or AVDD and AVSS pins, whichever is selected as reference input.

Figure 30-20: Example 10-bit A/D Conversion Timing Waveforms



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.
2: This is a minimal RC delay (typically 100 nS), which also disconnects the holding capacitor from the analog input.

Table 30-33: Example 10-bit A/D Conversion Requirements

Param No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
130	TAD	A/D clock period	PIC16**C**XXX	1.6	—	—	µs	TOSC based, VREF ≥ 3.0V
			PIC16**LC**XXX	3.0	—	—	µs	TOSC based, VREF full range
			PIC16**C**XXX	2.0	4.0	6.0	µs	A/D RC Mode
			PIC16**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 3)		15 10	— —	— —	µs µs	-40°C ≤ Temp ≤ 125°C 0°C ≤ Temp ≤ 125°C
136	TAMP	Amplifier settling time (Note 2)		1	—	—	µs	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).
135	TSWC	Switching Time from convert → sample		—	—	Note 4		

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.
§ This specification ensured by design.

Note 1: ADRES register may be read on the following TCY cycle.
2: See the **"10-bit A/D Converter"** section for minimum conditions when input voltage has changed more than 1 LSb.
3: The time for the holding capacitor to acquire the "New" input voltage when the voltage changes full scale after the conversion (AVDD to AVSS, or AVSS to AVDD). The source impedance (*RS*) on the input channels is 50 Ω.
4: On the next Q4 cycle of the device clock

30.25 Example Slope A/D Timing Waveforms and Requirements

Figure 30-21: Example Slope A/D Conversion Cycle

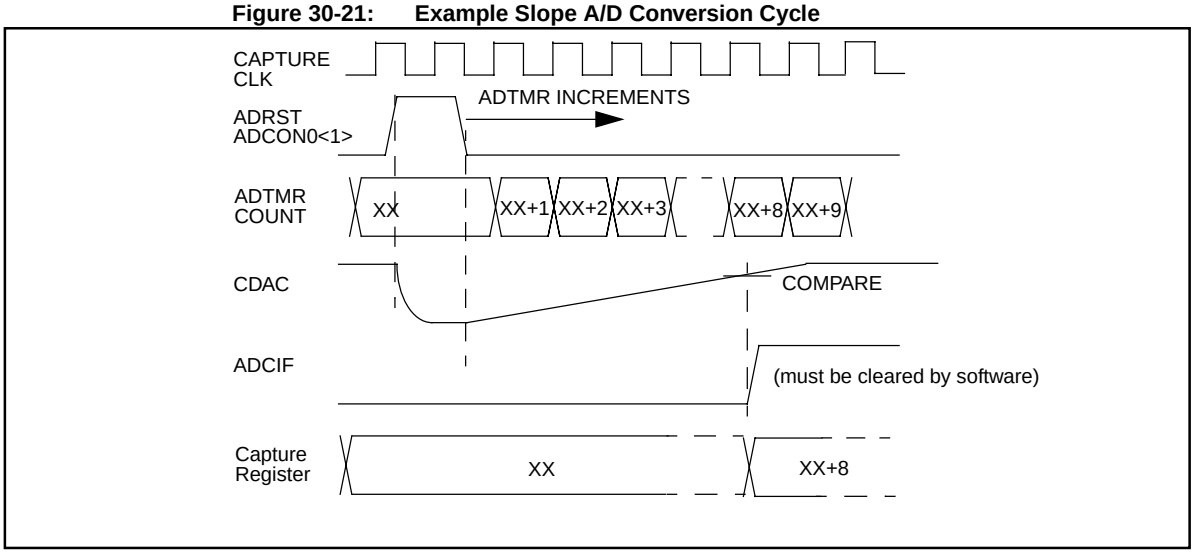


Table 30-34: Example Slope A/D Component Characteristics

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature 0°C ≤ TA ≤ +70°C for commercial, -40°C ≤ TA ≤ +85°C for industrial and -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC spec Table 30-3.				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
		Slope A/D Comparator					
A100	VAIN	Analog Input Voltage Range	VSS	—	VDD –1.4	V	
A101		Input Offset Voltage	–10	2	10	mV	Measured over common-mode range
A102	GDV	Differential Voltage Gain (Note 1)	—	100	—	dB	
A103	CMRR	Common Mode Rejection Ratio (Note 1)	—	80	—	dB	VDD = 5V, TA = 25°C, over common-mode range
A104	RRadc	Power Supply Rejection Ratio (Note 1)	—	70	—	dB	TA = 25°C, VDDmin ≤ VDD ≤ VDDmax
	TSET	**Turn-on Settling Time**					
140		Band Gap Reference (to < 0.1% (Note 1)	—	1	10	ms	REFOFF bit in SLPCON register 1 → 0
141		Programmable Current Source (to < 0.1%)	—	1	10	ms	Bias generator (reference) turn-on time (REFOFF 1 → 0) (reference start-up) (Note 1)
141A			—	1	10	µs	REFOFF = 0 (constant), ADCON1<7:4> 0000b → 1111b (reference already on and stable) (Note 3)
	TC	**Temperature Coefficient (Note 1)**					
A110	TCBGR	Band Gap Reference	— —	+50 –50	— —	ppm/°C	–40°C ≤ TA ≤ +25°C 25°C ≤ TA ≤ +85°C
A110A			— —	+20 –20	— —	ppm/°C	0°C ≤ TA ≤ +25°C 25°C ≤ TA ≤ +70°C
A111	TCPCS	Programmable Current Source	— —	+0.1 –0.1	— —	%/°C	–40°C ≤ TA ≤ +25°C 25°C ≤ TA ≤ +85°C
A112	TCkref	Slope Reference Divider	—	20	—	ppm/°C	–40°C ≤ TA ≤ +85°C
	CA	**Calibration Accuracy (Note3, 5)**					All parameters calibrated at VDD = 5V and TA = +25°C
A120	CABGR	Band Gap Reference	—	0.01	—	%	
A121	CASRV	Slope Reference Divider	—	0.02	—	%	
	SN	**Supply Sensitivity (Note 1)**					
A130	SNBGR	Band Gap Reference	—	0.04	—	%/V	From VDDmin to VDDmax
A131	SNPCS	Programmable Current Source	—	0.2	—	%/V	From VDDmin to VDDmax
A132	SNkref	Slope Reference Divider	—		—	%/V	From VDDmin to VDDmax
		Programmable Current Source					
A140	IRES	Resolution	1.25	2.25	3.25	µA	1 LSb
A141	EIL	Relative accuracy (linearity error)	–1/2		+1/2	LSb	CDAC = 0V

30.26 Example LCD Timing Waveforms and Requirements

Figure 30-22: Example LCD Voltage Waveform

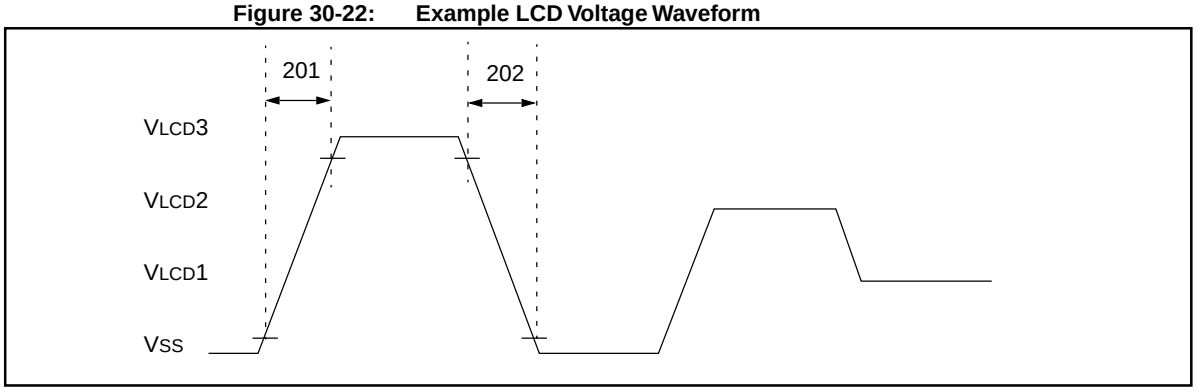


Table 30-35: Example LCD Module Timing Requirements

Param No.	Symbol	Characteristic	Min	Typ†	Max	Units	Conditions
200	FLCDRC	LCDRC Oscillator Frequency	—	14	22	kHz	VDD = 5V, -40˚C to +85˚C
201	TrLCD	Output Rise Time	—	—	200	µs	COM outputs Cload = 5,000 pF SEG outputs Cload = 500 pF VDD = 5.0V, T = 25°C
202	TfLCD	Output Fall Time (Note 1)	TrLCD - 0.05TrLCD	—	TrLCD + 0.05TrLCD	µs	COM outputs Cload = 5,000 pF SEG outputs Cload = 500 pF VDD = 5.0V, T = 25°C

† 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: 0Ω source impedance at VLCD.

30.27 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or High-End families), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to the Electrical Specifications are:

Title	Application Note #
No related Application Notes	

30.28 Revision History

Revision A

This is the initial released revision of the Electrical Specifications description.

Section 31. Device Characteristics

HIGHLIGHTS

31.1 Introduction

Microchip Technology Inc. provides characterization information on the devices that it manufactures. This information becomes available after the devices have undergone a complete characterization and the data has been analyzed. This data is taken on both device testers and on bench setups. The characterization data gives the designer a better understanding of the device characteristics, to better judge the acceptability of the device to the application.

31.2 Characterization vs. Electrical Specification

The difference between this information and the Electrical specifications can be classified as what the user should expect the devices to do vs. what Microchip tests the devices to. The characterization graphs and tables provided are for design guidance and are not tested or guaranteed.

There may be differences between what the characterization shows as the limits vs. that which is tested, as shown in the Electrical Specification section. This results from capabilities of the production tester equipment, plus whatever guard band that may be necessary.

31.3 DC and AC Characteristics Graphs and Tables

Each table gives specific information that may be useful design information. These values are taken under fixed circumstances. Measurements taken in your application may not lead to the same values if your circumstances are not the same.

In some graphs or tables the data presented are outside specified operating range (i.e., outside specified V_{DD} range). This is for information only and devices will operate properly only within the specified range.

Note:	The data presented in the device Data Sheet Characterization section is a statistical summary of data collected on units from different lots over a period of time and matrix samples. 'Typical' represents the mean of the distribution at, 25°C, while 'max' or 'min' represents (mean +3σ) and (mean -3σ) respectively where σ is standard deviation.

31.3.1 IPD vs. VDD

IPD is the current (I) that the device consumes when the device is in sleep mode (power-down), referred to as power-down current. These tests are taken with all I/O as inputs, either pulled high or low. That is, there are no floating inputs, nor are any pins driving an output (with a load).

The characterization shows graphs for both the Watchdog Timer (WDT) disabled and enabled. This is required since the WDT requires an on-chip RC oscillator which consumes additional current.

Since the device may have certain features and modules that can operate while the device is in sleep mode. Some of these modules are:

- Watchdog Timer (WDT)
- Brown-out Reset (BOR) circuitry
- Timer1
- Analog to Digital converter
- LCD module
- Comparators
- Voltage Reference

If these features are operating while the device is in sleep mode, a higher current will be consumed. When all features are disabled, the device will consume the lowest possible current (the leakage current). If more then one feature is enabled then the expected current can easily be calculated as the base current (everything disabled and in sleep mode) plus all delta currents. Example 31-1 shows an example of calculating the typical currents for a device at 5V, with the WDT and Timer1 oscillator enabled.

Example 31-1: IPD Calculations with WDT and TIMER1 Oscillator Enabled (@ 5V)

Base Current	14 nA	; Device leakage current
WDT Delta Current	14 µA	; 14 µA - 14 nA = 14 µA
Timer1 Delta Current	22 µA	; 22 µA - 14 nA = 22 µA
Total Sleep Current	36 µA	;

Figure 31-1: Example Typical IPD vs. VDD (WDT Disabled, RC Mode)

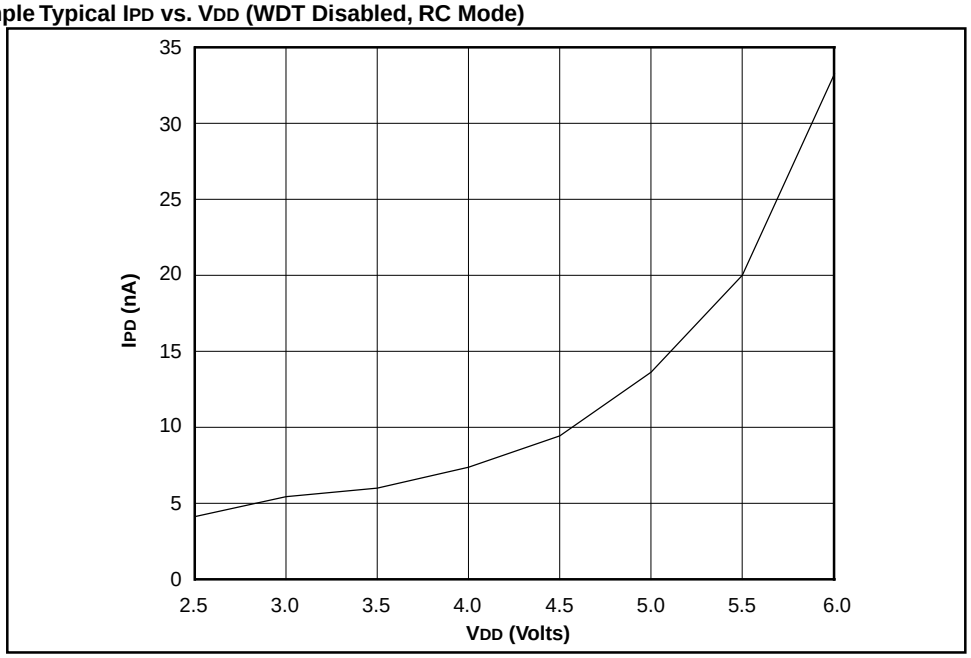


Figure 31-2: Example Maximum IPD vs. VDD (WDT Disabled, RC Mode)

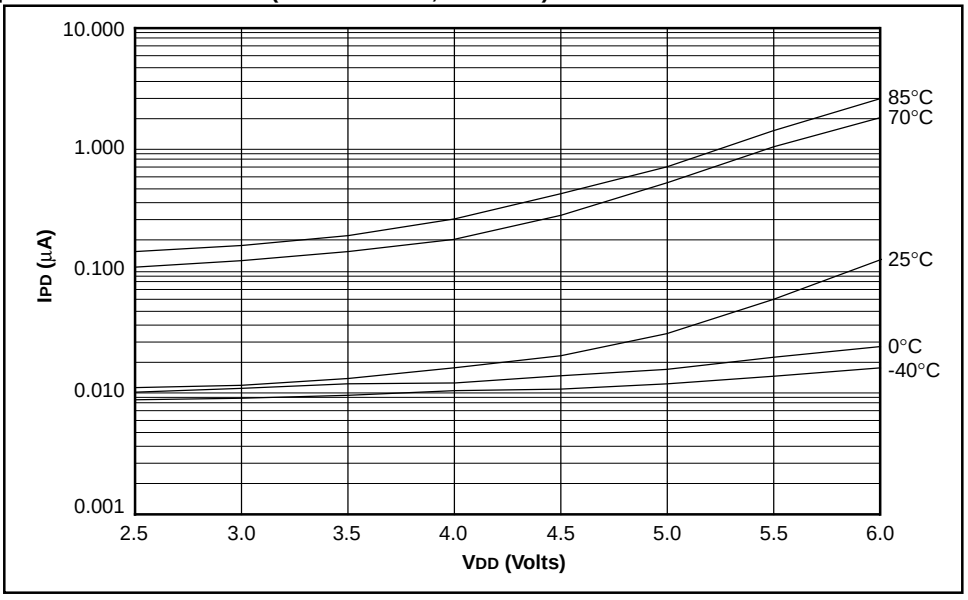


Figure 31-3: Example Typical I_{PD} vs. V_{DD} @ 25°C (WDT Enabled, RC Mode)

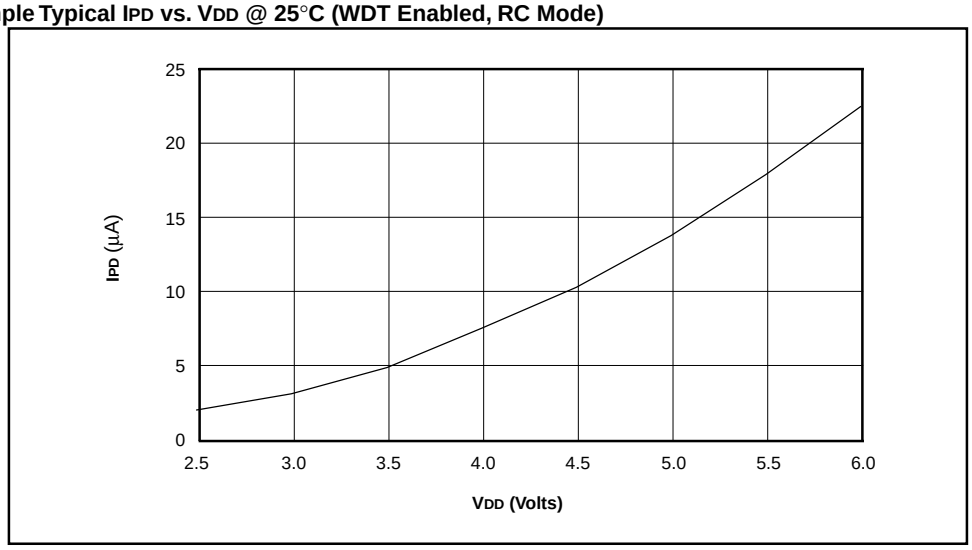


Figure 31-4: Example Maximum I_{PD} vs. V_{DD} (WDT Enabled, RC Mode)

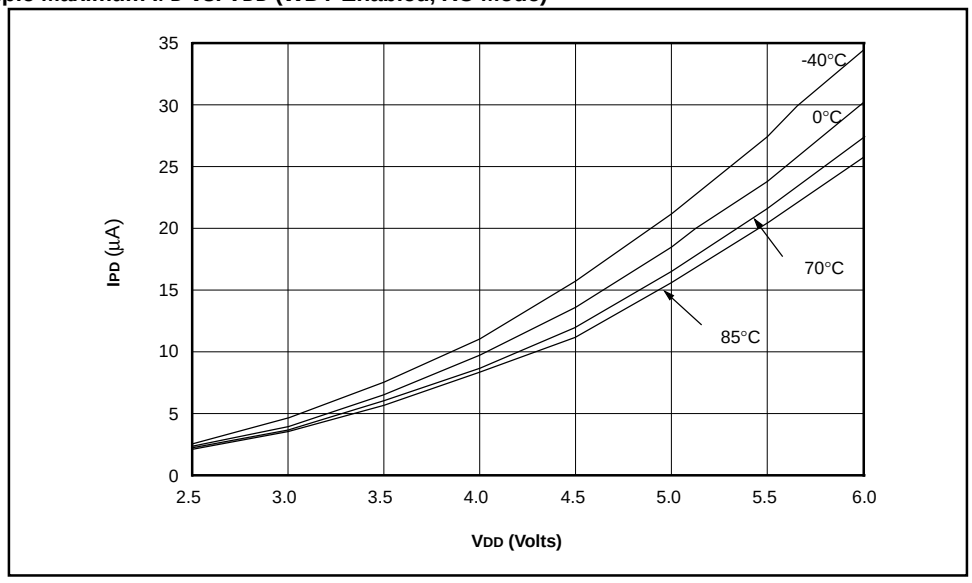
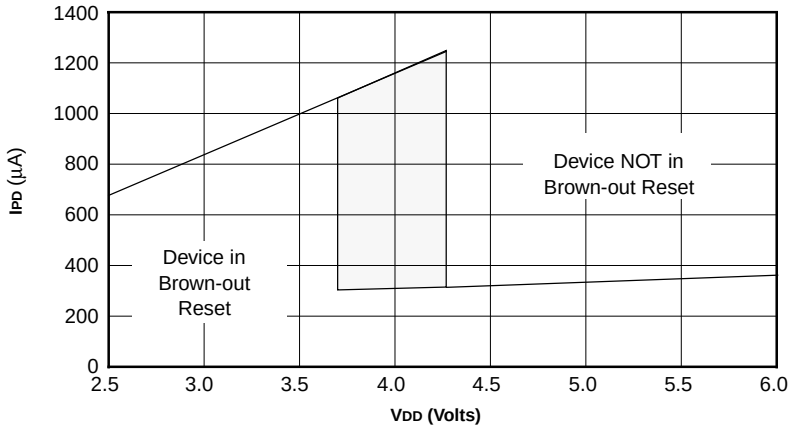
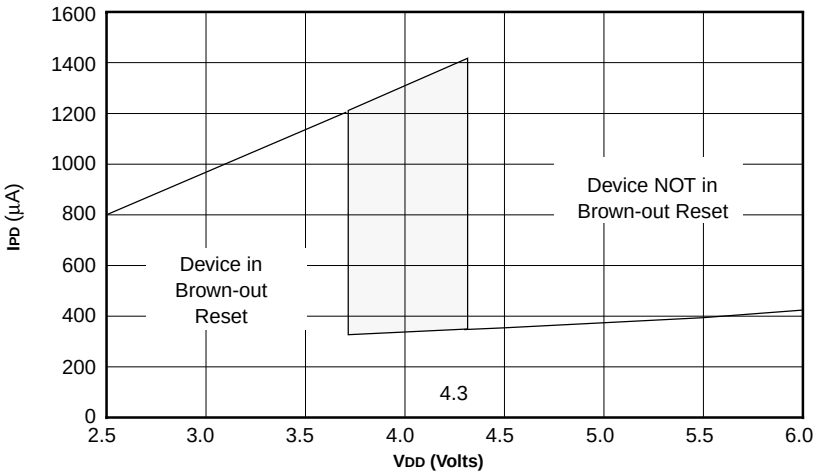


Figure 31-5: Example Typical IPD vs. VDD Brown-out Detect Enabled (RC Mode)



The shaded region represents the built-in hysteresis of the brown-out reset circuitry.

Figure 31-6: Example Maximum IPD vs. VDD Brown-out Detect Enabled (85°C to -40°C, RC Mode)



The shaded region represents the built-in hysteresis of the Brown-out Reset circuitry.

Figure 31-7: Example Typical IPD vs. Timer1 Enabled (32 kHz, RC0/RC1 = 33 pF/33 pF, RC Mode)

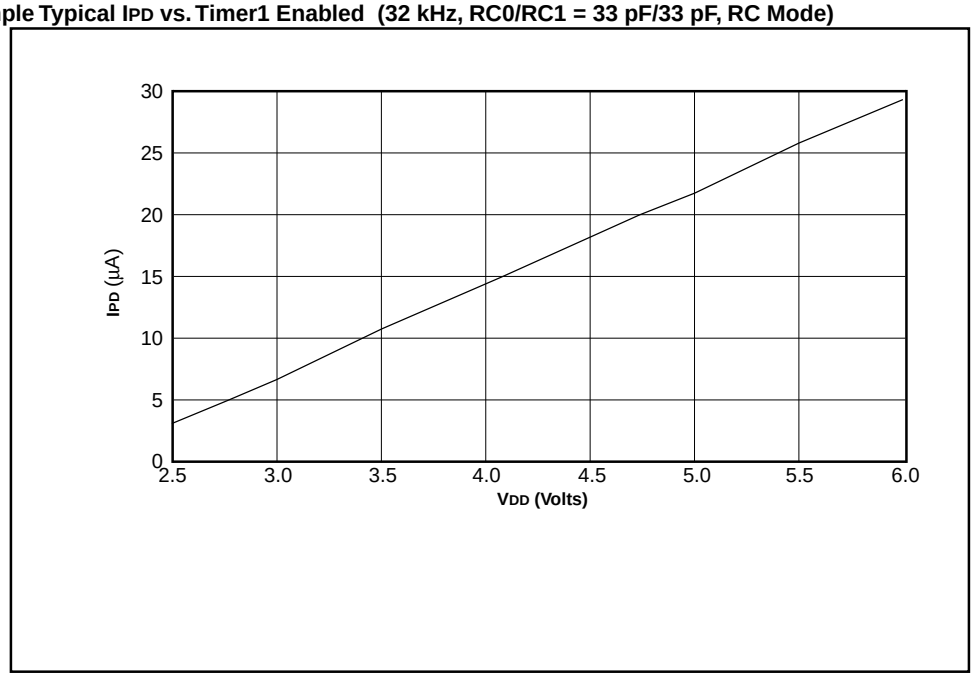
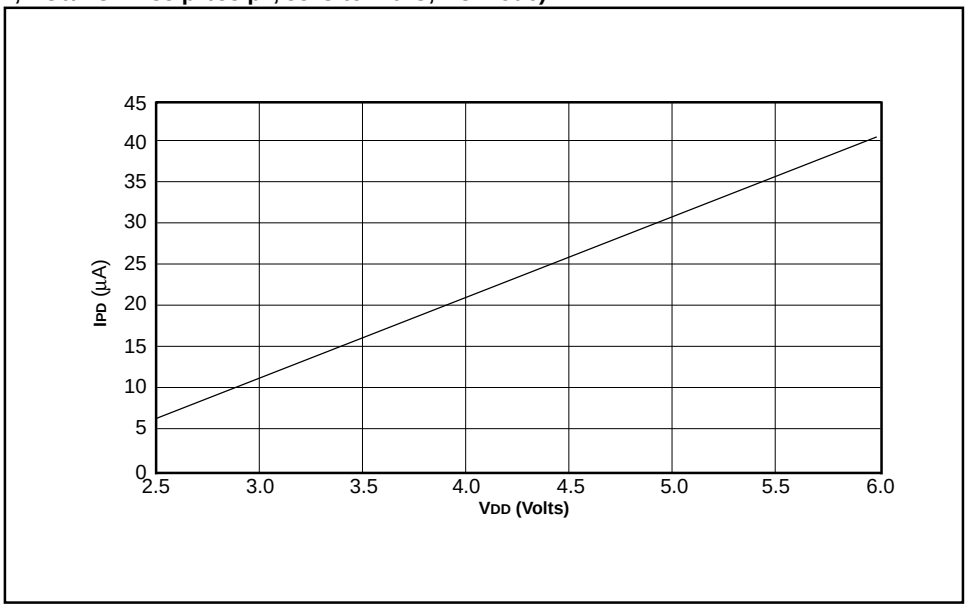


Figure 31-8: Example Maximum IPD vs. Timer1 Enabled (32 kHz, RC0/RC1 = 33 pF/33 pF, 85°C to -40°C, RC Mode)



31.3.2 IDD vs. Frequency

IDD is the current (I) that the device consumes when the device is in operating mode. This test is taken with all I/O as inputs, either pulled high or low. That is, there are no floating inputs, nor are any pins driving an output (with a load).

The IDD vs. Frequency charts measure the results on a Microchip automated bench setup, called the DCS (**D**ata **C**ollection **S**ystem). The DCS accurately reflects the device and specified component values, that is, it does not add stray capacitance or current.

31.3.2.1 RC Measurements

For the RC measurement, the DCS selects a resistor and capacitor value, and then varies the voltage over the specified range. As the voltage is changed, the frequency of operation changes. For a fixed RC, as VDD increases, the frequency increases. After the measurement, at this RC, has been taken, the RC value is changed and the measurements are taken again. Each point on the graph corresponds to a device voltage, resistor value (R), and capacitor value (C).

Figure 31-9: Example Typical IDD vs. Frequency (RC Mode @ 22 pF, 25°C)

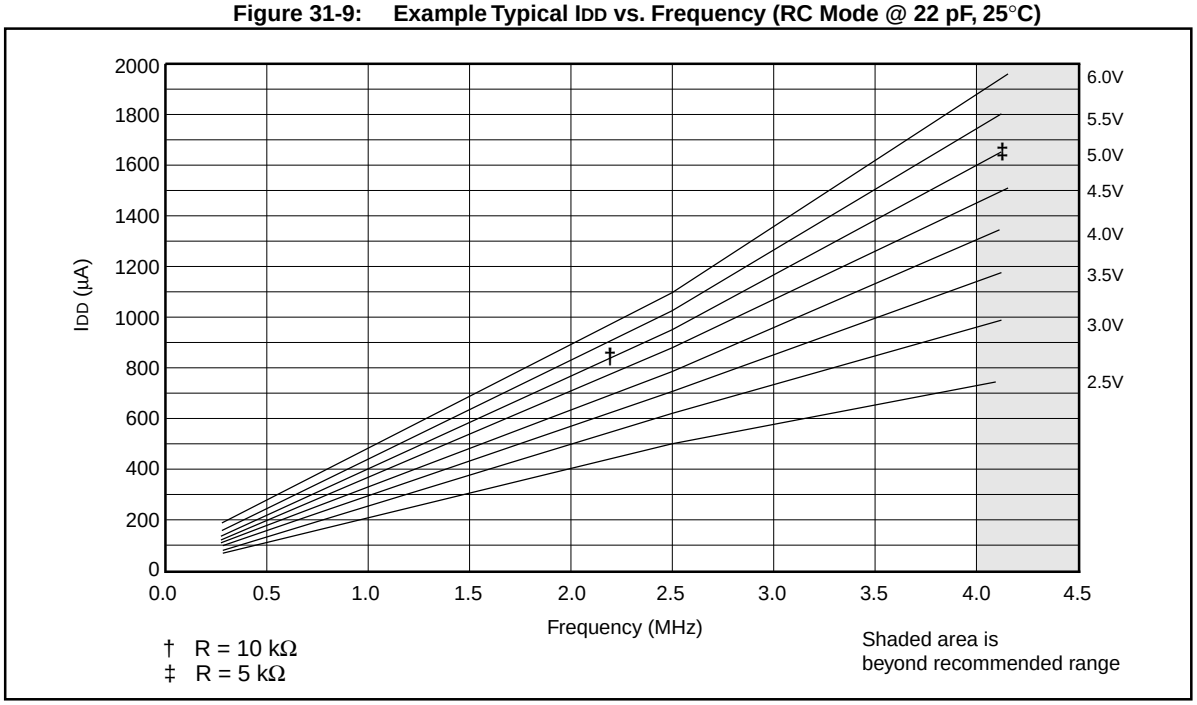


Figure 31-10: Example Maximum IDD vs. Frequency (RC Mode @ 22 pF, -40°C to 85°C)

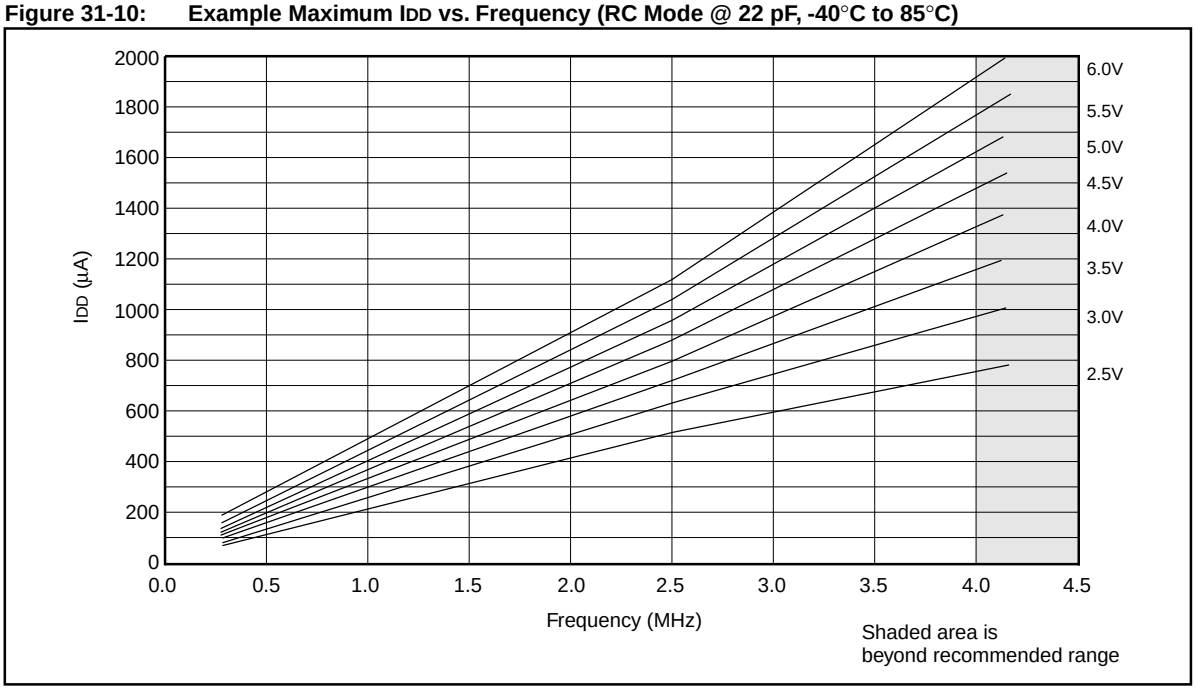


Figure 31-11: Example Typical IDD vs. Frequency (RC Mode @ 100 pF, 25°C)

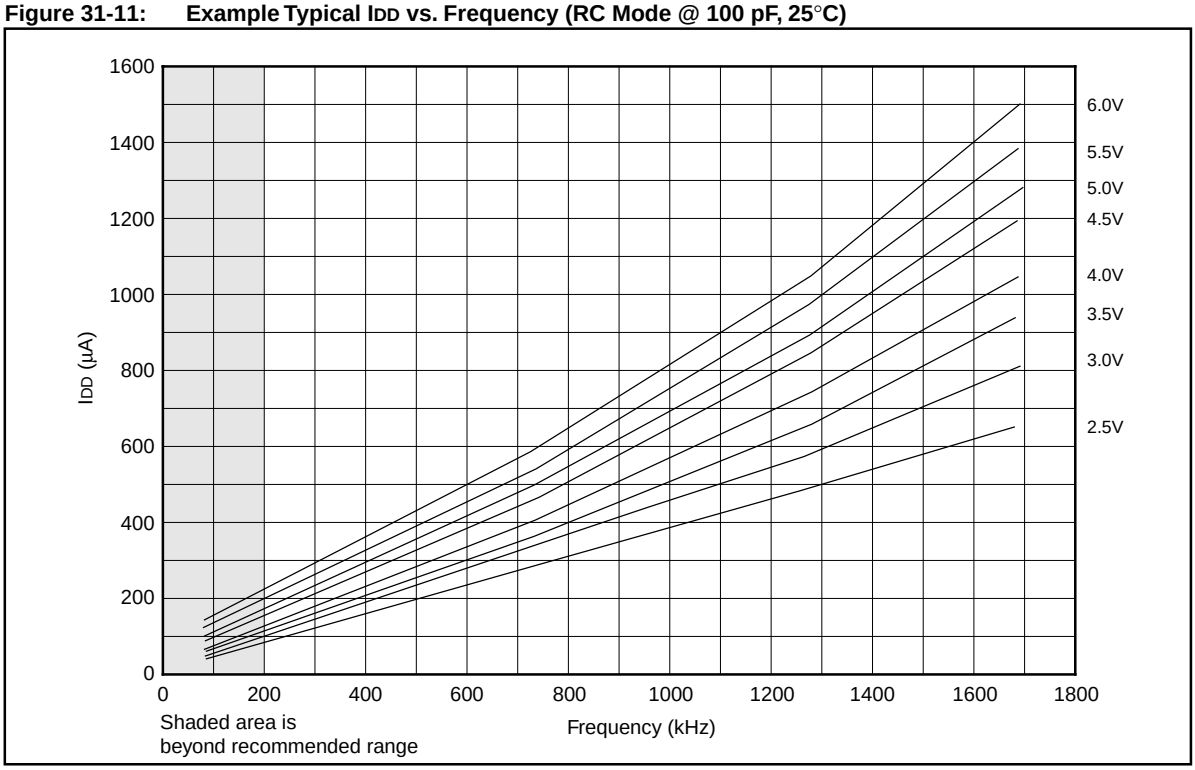


Figure 31-12: Example Maximum IDD vs. Frequency (RC Mode @ 100 pF, -40°C to 85°C)

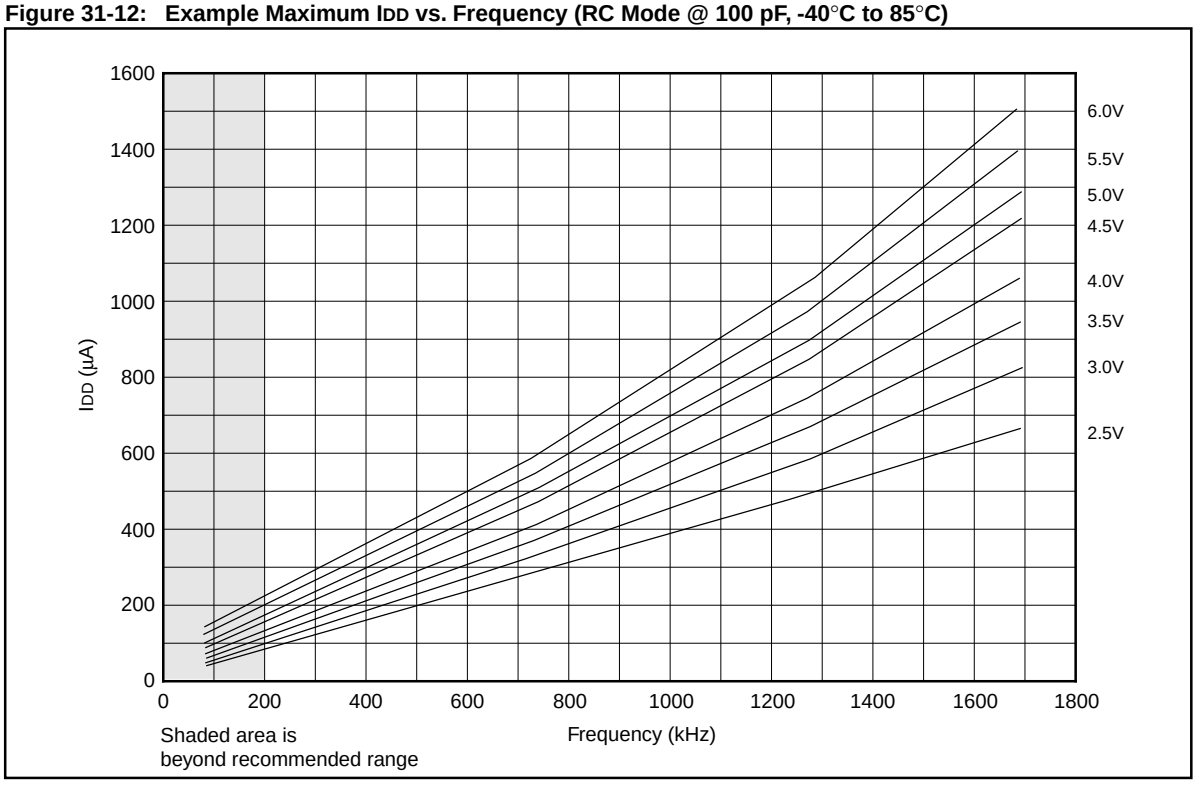


Figure 31-13: Example Typical IDD vs. Frequency (RC Mode @ 300 pF, 25°C)

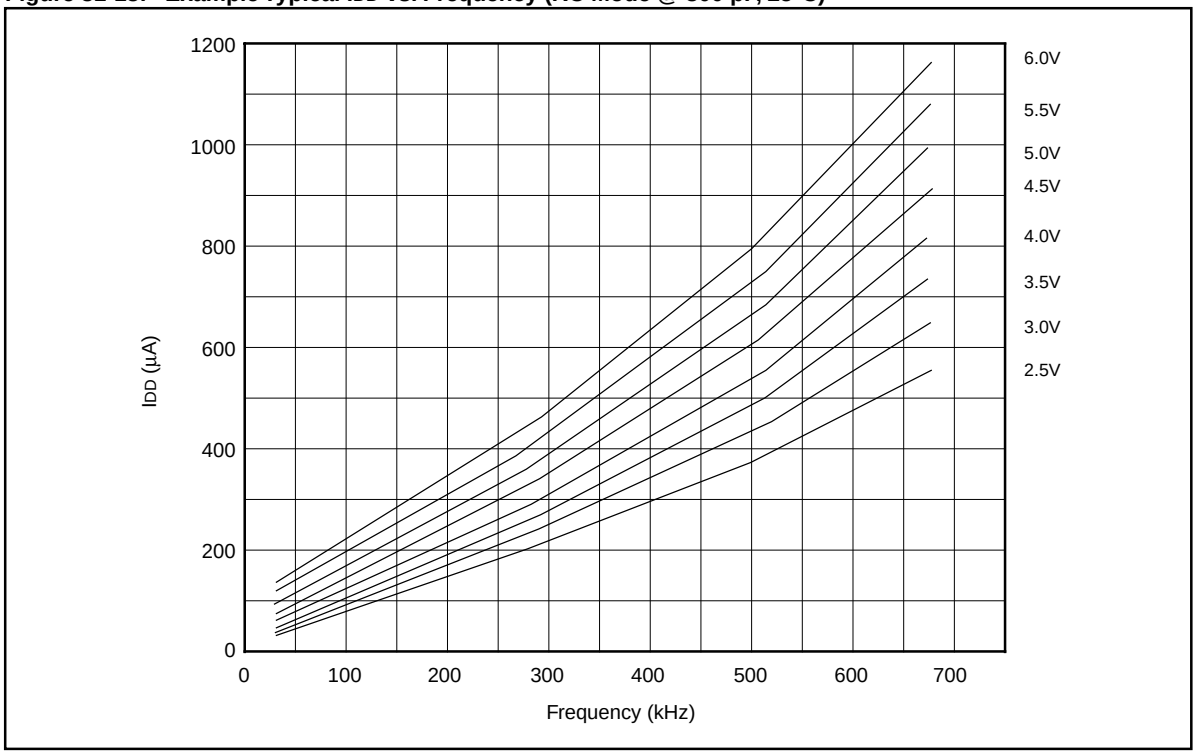


Figure 31-14: Example Maximum IDD vs. Frequency (RC Mode @ 300 pF, -40°C to 85°C)

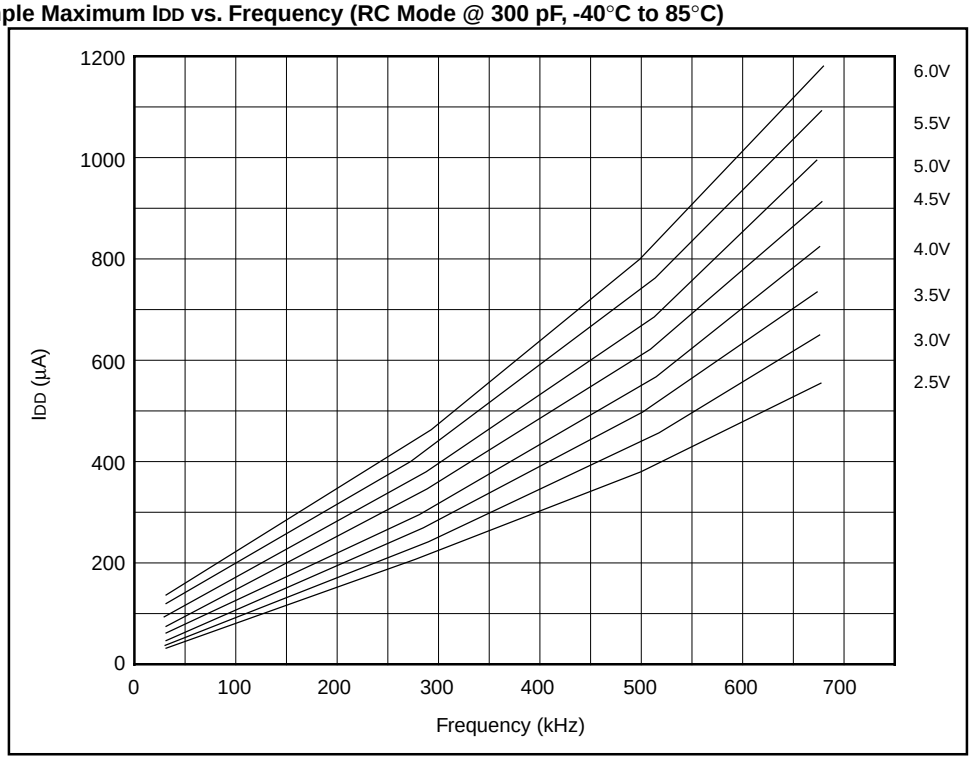
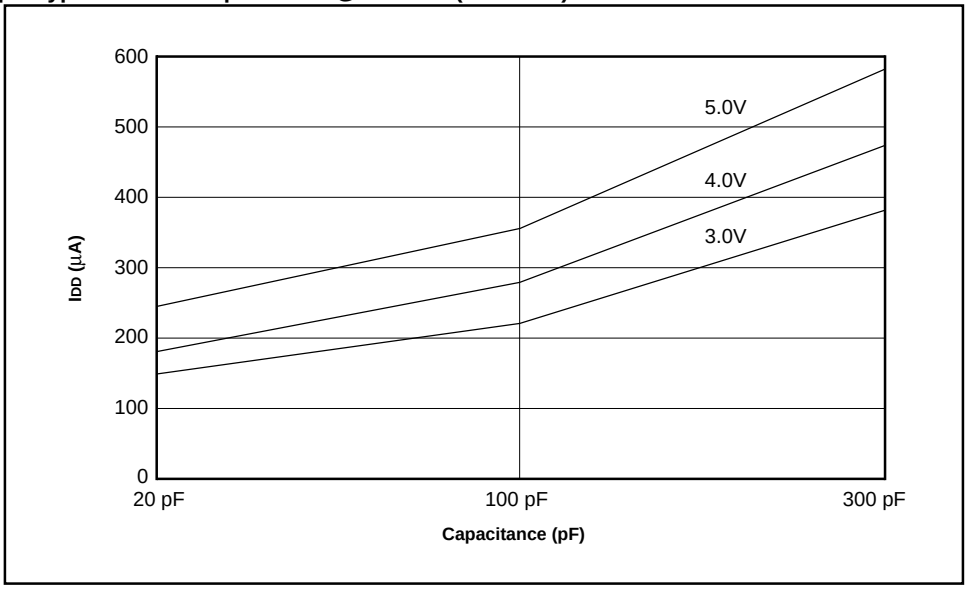


Figure 31-15: Example Typical IDD vs. Capacitance @ 500 kHz (RC Mode)



31.3.2.2 Crystal Oscillator Measurements

On the Data Collection System, there are several crystals. For this test a crystal is multiplexed into the device circuit, and the crystal's capacitance values can be varied. The capacitance and voltage values are varied to determine the best characteristics (current, oscillator waveform, and oscillator start-up), and then the currents are measured over voltage. The next crystal oscillator is then switched in and the procedure is repeated.

Figure 31-16: Example Typical IDD vs. Frequency (LP Mode, 25°C)

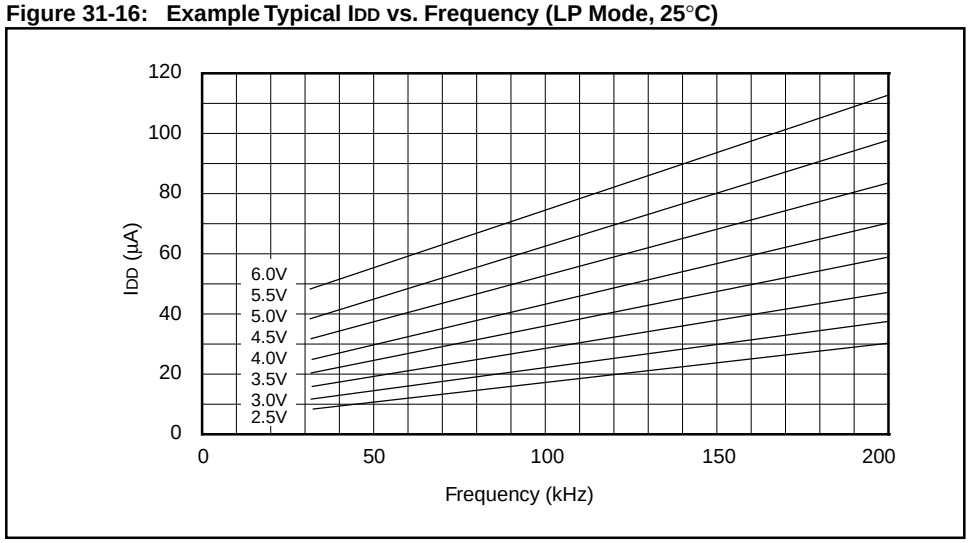


Figure 31-17: Example Maximum IDD vs. Frequency (LP Mode, 85°C to -40°C)

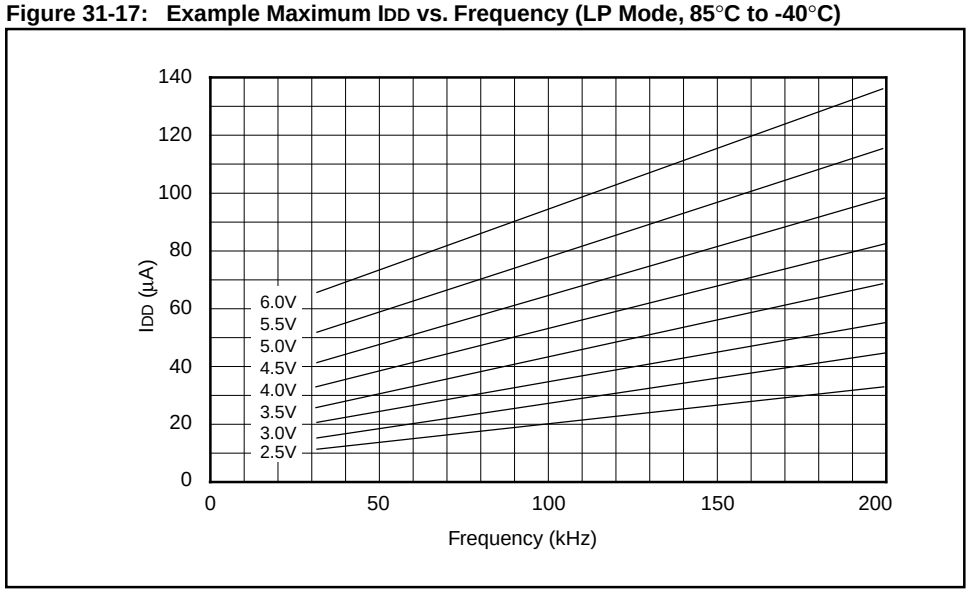


Figure 31-18: Example Typical IDD vs. Frequency (XT Mode, 25°C)

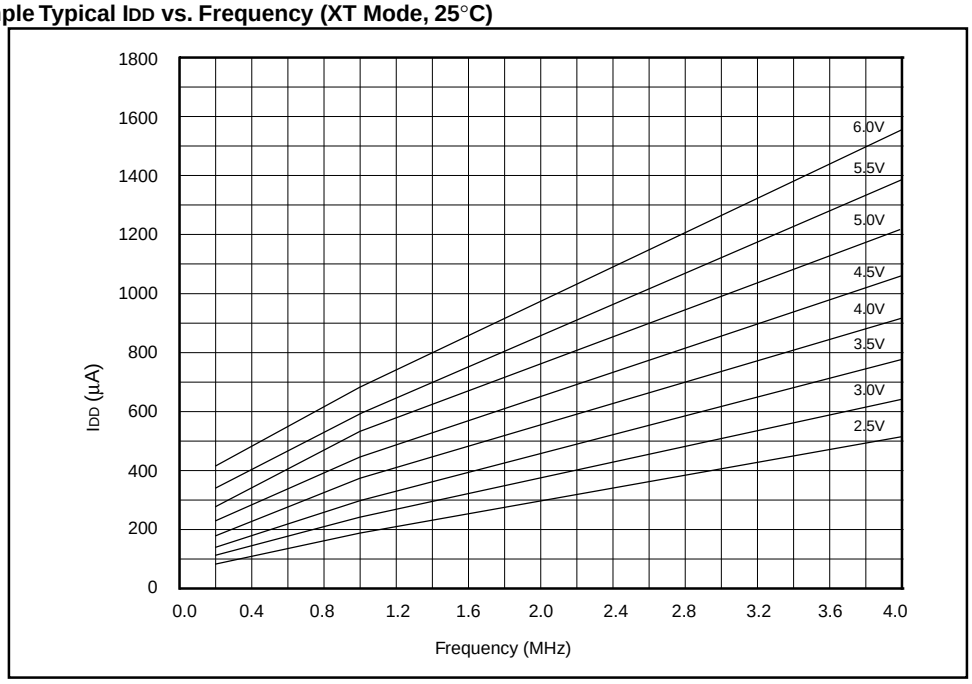


Figure 31-19: Example Maximum IDD vs. Frequency (XT Mode, -40°C to 85°C)

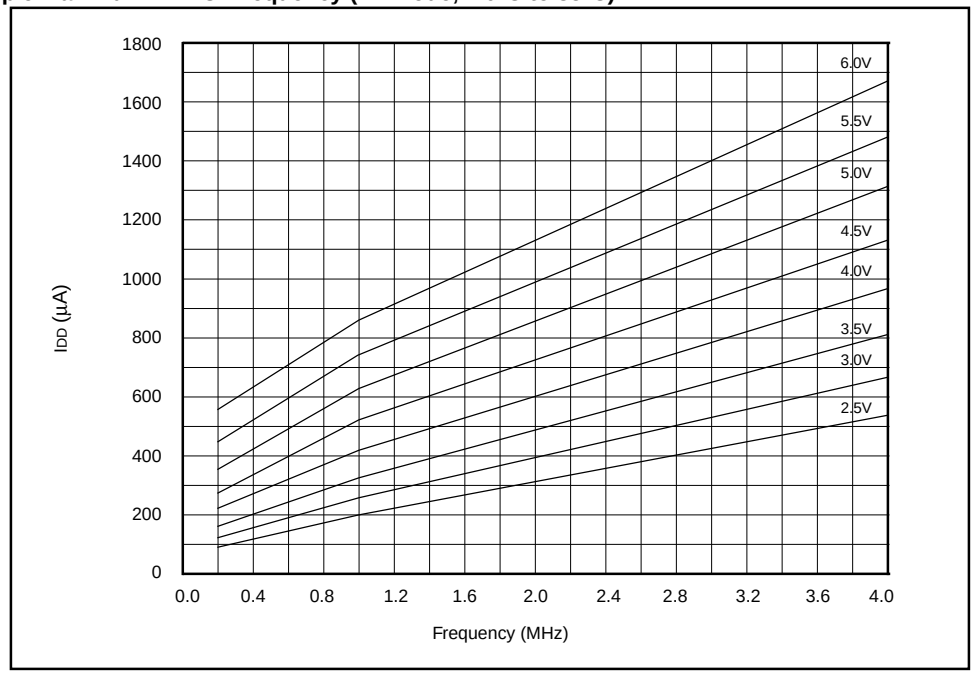


Figure 31-20: Example Typical IDD vs. Frequency (HS Mode, 25°C)

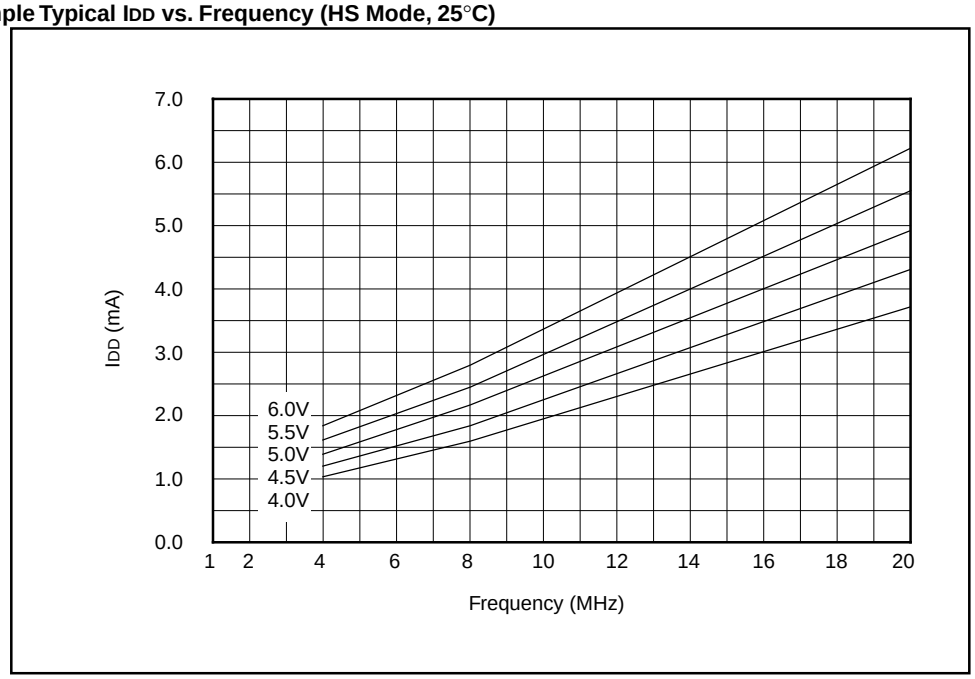
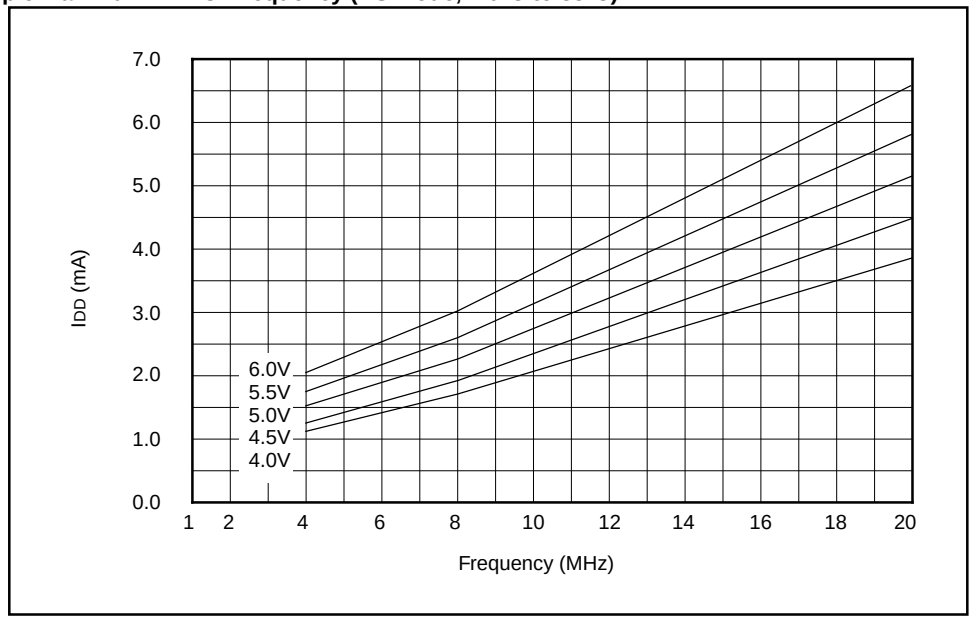


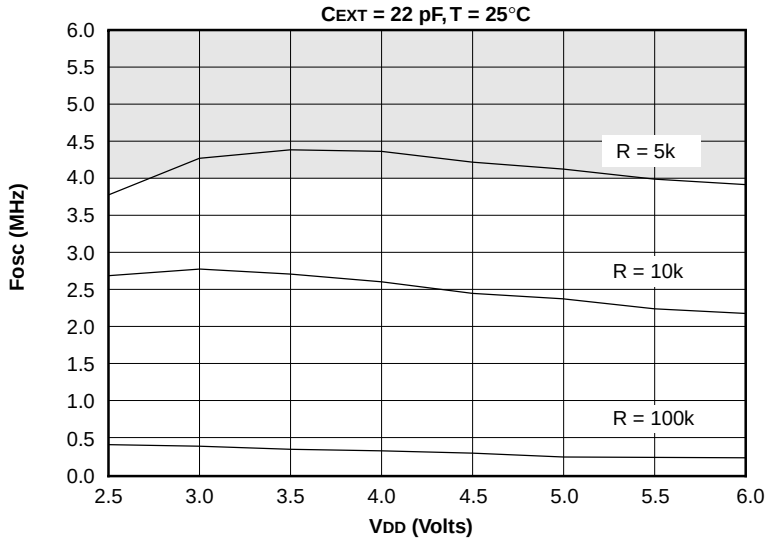
Figure 31-21: Example Maximum IDD vs. Frequency (HS Mode, -40°C to 85°C)



31.3.3 RC Oscillator Frequency

These tables show the effects of the RC oscillator frequency as the device voltage varies. In these measurements a capacitor and resistor value are selected and then the frequency of the RC is measured as the device voltage varies. The table shows the typical frequency for a R and C value at 5V, as well as the variation from this frequency that can be expected due to device processing.

Figure 31-22: Example Typical RC Oscillator Frequency vs. VDD



Shaded area is beyond recommended range.

Figure 31-23: Example Typical RC Oscillator Frequency vs. VDD

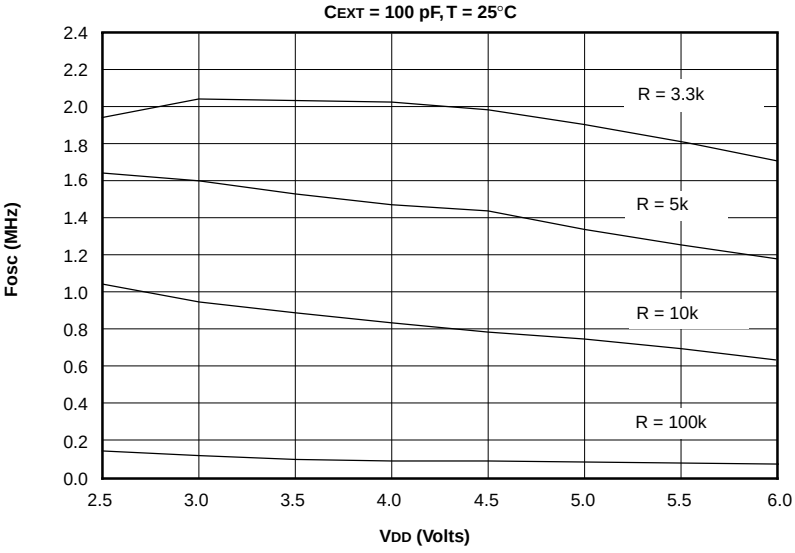


Figure 31-24: Example Typical RC Oscillator Frequency vs. VDD

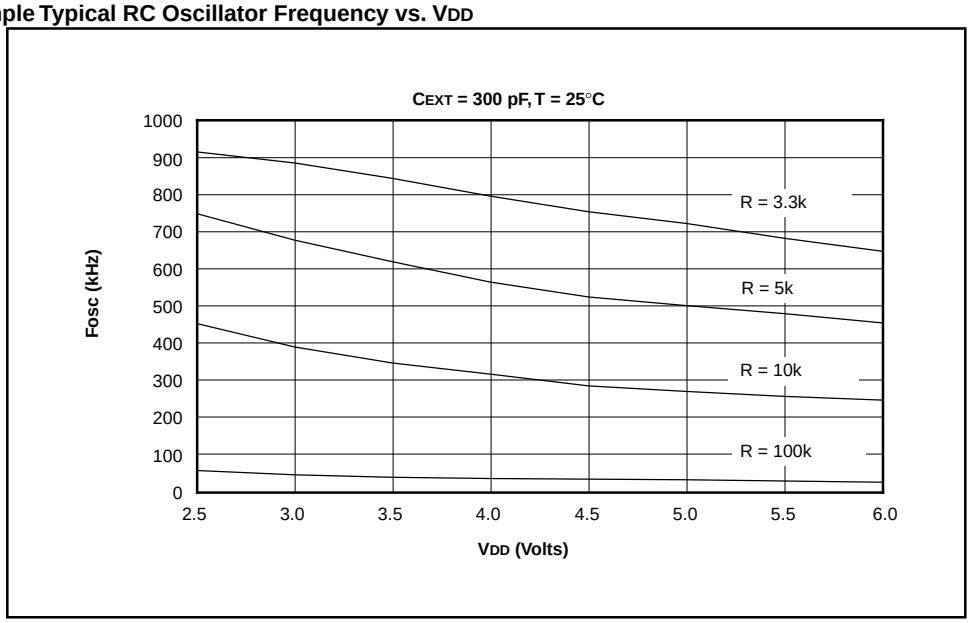


Table 31-1: Example RC Oscillator Frequencies

CEXT	REXT	Average Fosc @ 5V, 25°C	
22 pF	5k	4.12 MHz	± 1.4%
	10k	2.35 MHz	± 1.4%
	100k	268 kHz	± 1.1%
100 pF	3.3k	1.80 MHz	± 1.0%
	5k	1.27 MHz	± 1.0%
	10k	688 kHz	± 1.2%
	100k	77.2 kHz	± 1.0%
300 pF	3.3k	707 kHz	± 1.4%
	5k	501 kHz	± 1.2%
	10k	269 kHz	± 1.6%
	100k	28.3 kHz	± 1.1%

The percentage variation indicated here is part to part variation due to normal process distribution. The variation indicated is ±3 standard deviation from average value for VDD = 5V.

31.3.4 Oscillator Transconductance

Transconductance of the oscillator indicates the gain of the oscillator. As the transconductance increases, the gain of the oscillator circuit increases which causes the current consumption of the oscillator circuit to increase. Also as the transconductance increases the maximum frequency that the oscillator circuit can support also increases, or the start-up time of the oscillator decreases.

Figure 31-25: Example Transconductance (gm) of HS Oscillator vs. VDD

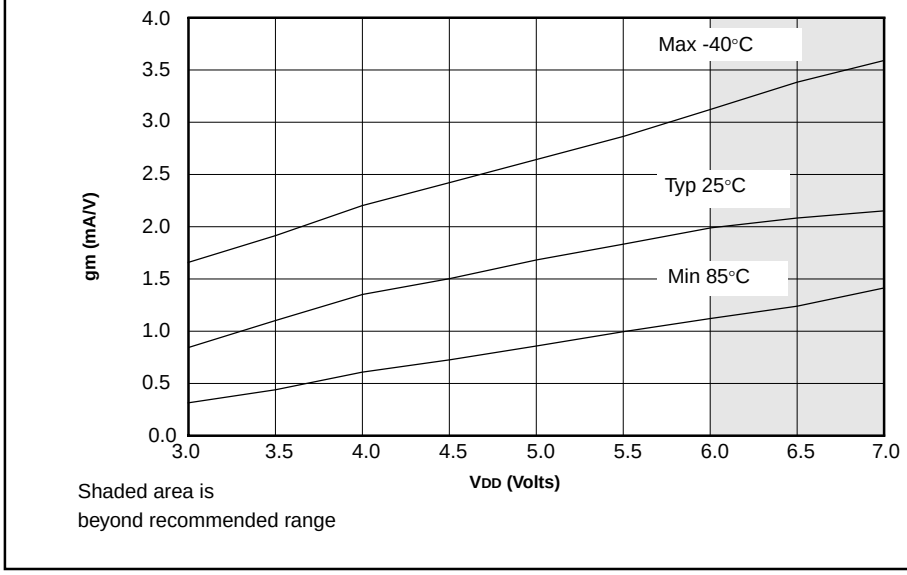


Figure 31-26: Example Transconductance (gm) of LP Oscillator vs. VDD

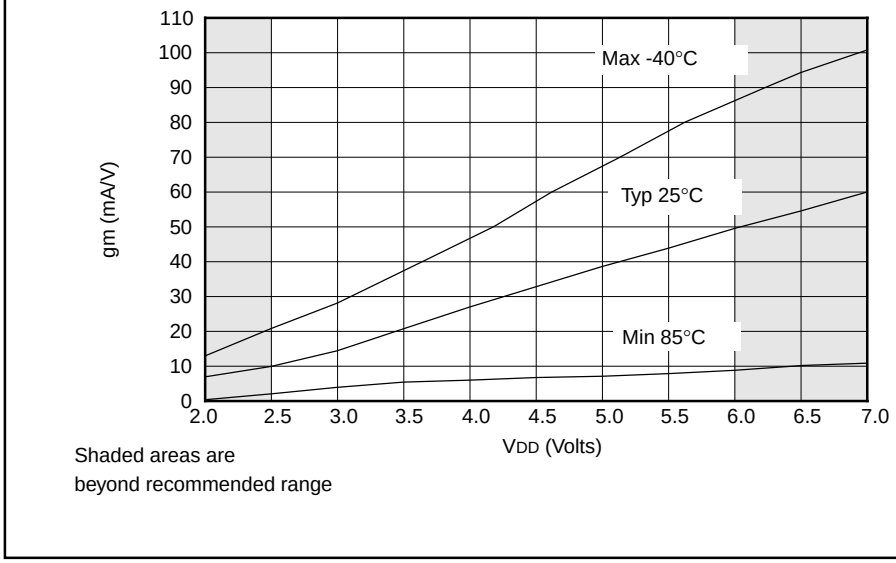
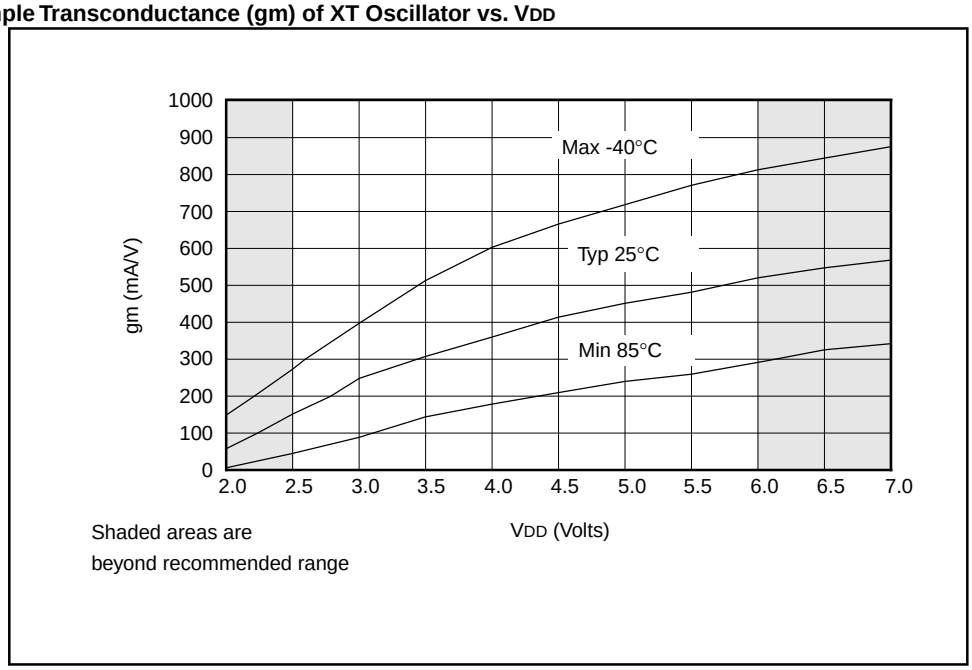


Figure 31-27: Example Transconductance (gm) of XT Oscillator vs. VDD



31.3.5 Crystal Start-up Time

These graphs show the start-up time that one should expect to see at the specified voltage level, for a given crystal/capacitor combination.

Figure 31-28: Example Typical XTAL Start-up Time vs. VDD (LP Mode, 25°C)

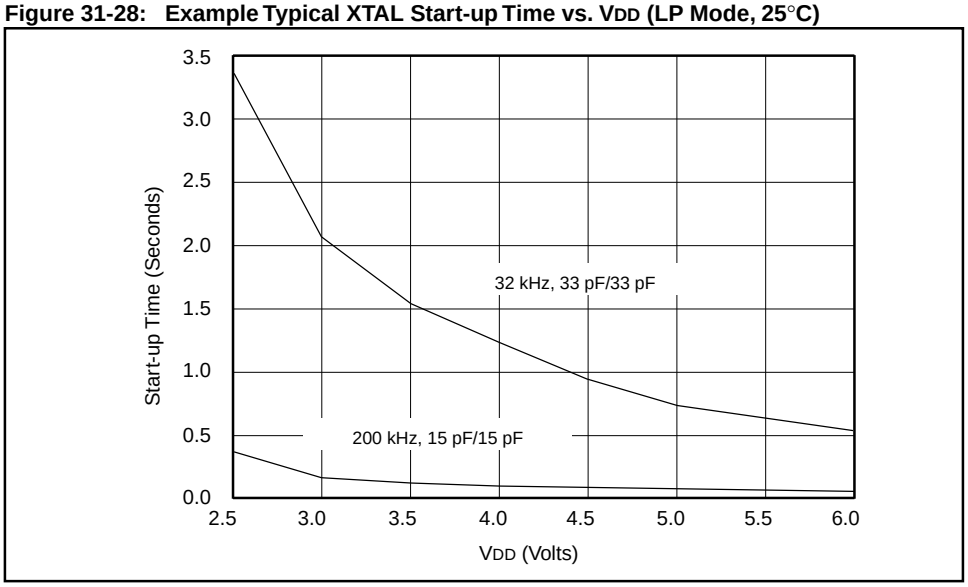


Figure 31-29: Example Typical XTAL Start-up Time vs. VDD (HS Mode, 25°C)

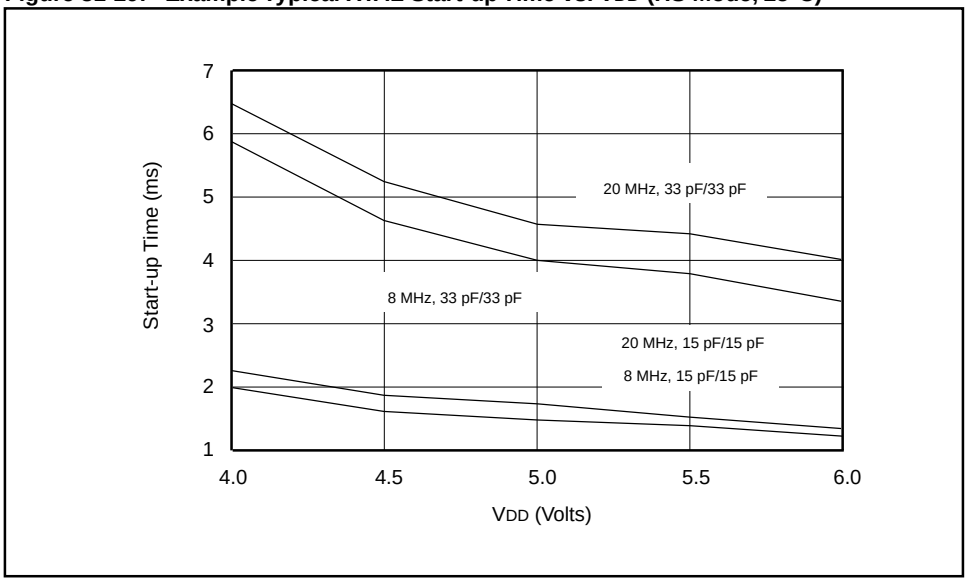
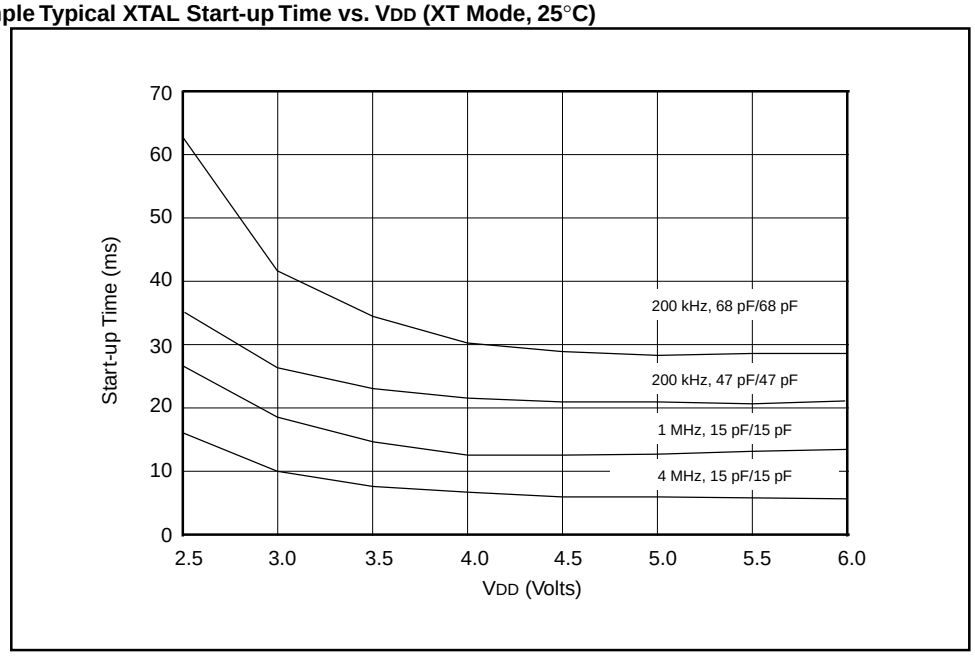


Figure 31-30: Example Typical XTAL Start-up Time vs. VDD (XT Mode, 25°C)



31.3.6 Tested Crystals and Their Capacitor Values

This table shows the crystal frequency and manufacturer that was used for every tests in this section, as well as the capacitor values/ranges that exhibited the best characteristics.

Table 31-2: Example Capacitor Selection for Crystal Oscillators

Osc Type	Crystal Frequency	Capacitor Range C1	Capacitor Range C2
LP	32 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	8 MHz	15-33 pF	15-33 pF
	20 MHz	15-33 pF	15-33 pF
Note: Higher capacitance increases the stability of the oscillator but also increases the start-up time. These values are for design guidance only. ***Rs*** may be required in HS mode as well as XT mode to avoid overdriving crystals with low drive level specification. Since each crystal has its own characteristics, the user should consult the crystal manufacturer for appropriate values of external components or verify oscillator performance.			
Crystals Used:			
32 kHz	Epson C-001R32.768K-A	± 20 PPM	
200 kHz	STD XTL 200.000KHz	± 20 PPM	
1 MHz	ECS ECS-10-13-1	± 50 PPM	
4 MHz	ECS ECS-40-20-1	± 50 PPM	
8 MHz	EPSON CA-301 8.000M-C	± 30 PPM	
20 MHz	EPSON CA-301 20.000M-C	± 30 PPM	

31.3.7 Example EPROM Memory Erase Times

The UV erase time of an EPROM cell depends on the geometry size of the EPROM cell and the manufacturing technology. Table 31-3 shows some of the expected erase times for each different device.

Table 31-3: Example of Typical EPROM Erase Time Recommendations

Example Device	Wavelength (Angstroms)	Intensity ($\mu W/cm^2$)	Distance from UV lamp (inches)	Typical Time (1) (minutes)
1	2537	12,000	1	15 - 20
2	2537	12,000	1	20
3	2537	12,000	1	40
4	2537	12,000	1	60

Note 1: If these criteria are not met, the erase times will be different.

Table 31-4: Refer to the device data sheet for the typical erase times for a device.

31.4 Revision History

Revision A

This is the initial released revision of the Device Characteristics description.

Section 32. Development Tools

HIGHLIGHTS

This section of the manual contains the following major topics:

32.1 Introduction

Microchip offers a wide range of tightly integrated development tools to ease the application development process. These can be broken down into the core development tools and the supplemental tools.

The core tools are as follows:

- MPLAB Integrated Development Environment, including full featured editor
- Language Products
  - MPASM Assembler
  - MPLAB-C C Compiler
- MPLAB-SIM Software Simulator
- Real-Time In-Circuit Emulators
  - PICMASTER/PICMASTER CE Emulator with Full Featured Trace and Breakpoint debug capabilities
  - ICEPIC Low-Cost Emulator with Breakpoint debug capabilities
- Device Programmers
  - PRO MATE® II Universal Programmer
  - PICSTART® Plus Entry-Level Prototype Programmer

Supplemental Tools:

- Other Software Programming Tools
  - fuzzyTECH®–MP Fuzzy logic development system
  - MP-Driveway Application Code Generator
- Development Boards
  - PICDEM-1 Low-Cost Demonstration Board
  - PICDEM-2 Low-Cost Demonstration Board
  - PICDEM-3 Low-Cost Demonstration Board
  - PICDEM-14A Low-Cost Demonstration Board

The minimum configuration of MPLAB, is the Integrated Development Environment (IDE), the assembler (MPASM), and the software simulator (MPLAB-SIM). Other tools are added to MPLAB as they are installed. This gives a common platform for the design activity, from the writing and assembling of the source code, through the simulation/emulation, to the programming of prototype devices.

Note:	The most current version may be downloaded from Microchip's web site or BBS for free.

In addition to Microchip, there are many third party vendors. Microchip's Third Party Handbook gives an overview of the manufactures and their tools.

32.2 The Integrated Development Environment (IDE)

The core set of development tools operate under the IDE umbrella, called MPLAB. This gives a consistent look and feel to all the development tools so that minimal learning of the new tool interface is required. The MPLAB IDE integrates all the following aspects of development:

- Source code editing
- Project management
- Machine code generation (from assembly or “C”)
- Device simulation
- Device emulation
- Device programming

MPLAB is a PC based Windows® 3.x application. It has been extensively tested using Windows 95 and recommended in either of these operating environments.

This comprehensive tool suite allows the complete development of a project without leaving the MPLAB environment.

Windows is a registered trademark of Microsoft Corporation.

32.2.1 MPLAB

The MPLAB IDE Software brings an ease of software development previously unseen in the 8-bit microcontroller market. MPLAB is a Windows based application that contains:

- A full featured editor
- Three operating modes
  - editor
  - emulator
  - simulator
- A project manager
- Customizable tool bar and key mapping
- A status bar with project information
- Extensive on-line help

MPLAB allows you to:

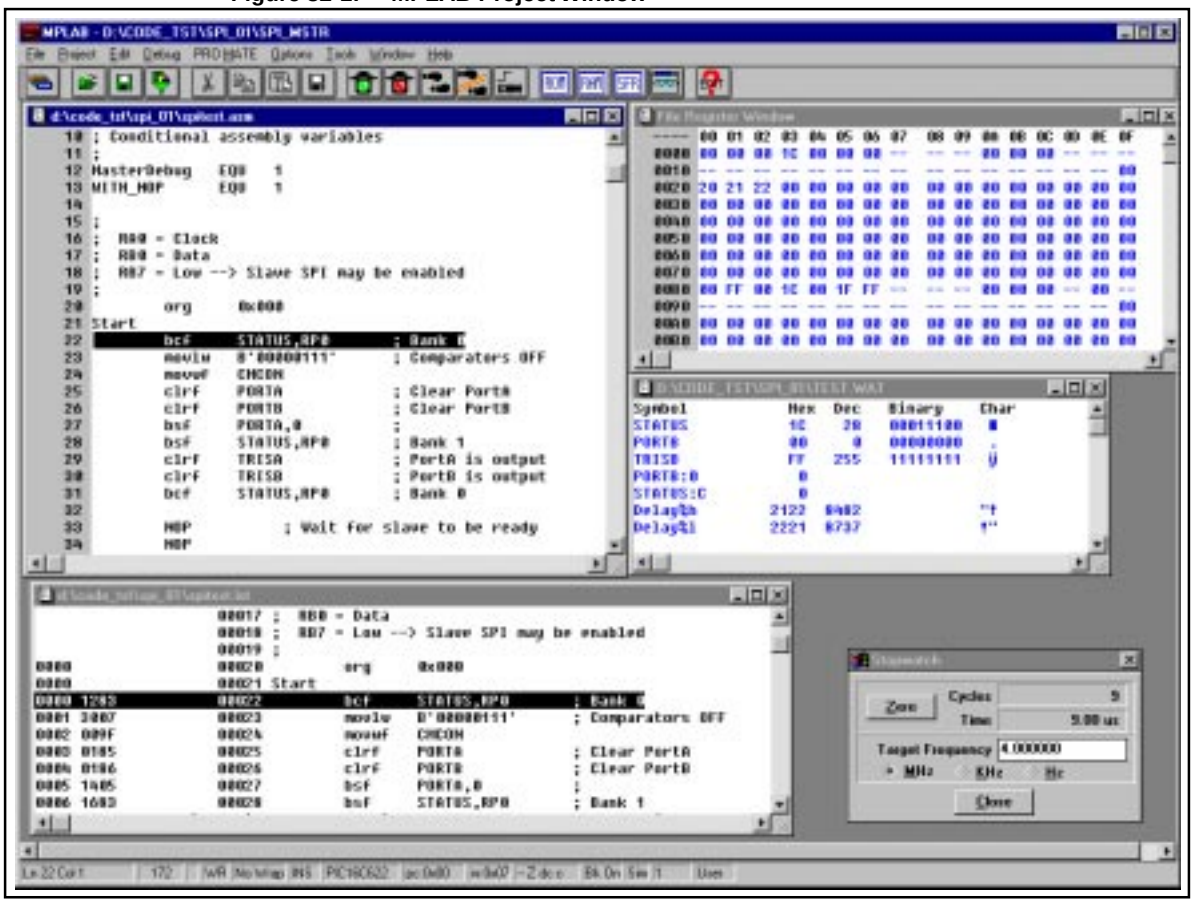
- Edit your source files. This includes:
  - MPASM assembly language
  - MPLAB-C 'C' language
- One touch assemble (or compile) and download to PIC16/17 tools (automatically updates all project information)
- Debug using:
  - source files
  - absolute listing file
  - program memory
- Run up to four emulators on the same PC
- Run or Single-step
  - program memory
  - source file
  - absolute listing

Microchip's simulator, MPLAB-SIM, operates under the same platform as the PICMASTER emulator. This allows the user to learn a single tool set which functions equivalently for both the simulator and the full featured emulator.

Figure 32-1 shows a typical MPLAB desktop in the middle of a project. Some of the highlights are:

- Tool bars, multiple choices and user configurable
- Status, mode information, and button help on footer bar
- Multiple windows, such as
  - Source code
  - Source listing (most useful for 'C' programs)
  - Register file window (RAM)
  - Watch windows (to look at specific register)
  - Stop watch window for time/cycle calculations
- Programmer support (in this case PRO MATE pull down menu)

Figure 32-1: MPLAB Project Window



32.3 MPLAB Software Language Support

To make the device operate as desired in the application, a software program needs to be written for the microcontroller. This software program needs to be written in one of the programming languages for the device. Currently MPLAB supports two of Microchip's language products:

- Microchip Assembler (MPASM)
- Microchip 'C' Compiler (MPLAB-C)
- Other language products that support Common Object Description (COD) may also work with MPLAB

32.3.1 Assembler (MPASM)

The MPASM Universal Macro Assembler is a PC-hosted symbolic assembler. It supports all Microchip microcontroller families.

MPASM offers full featured Macro capabilities, conditional assembly, and several source and listing formats. It generates various object code formats to support Microchip's development tools as well as third party programmers.

MPASM allow full symbolic debugging from the Microchip Universal Emulator System (PICMASTER).

MPASM has the following features to assist in developing software for specific use applications.

- Provides translation of Assembler source code to object code for all Microchip microcontrollers.
- Macro assembly capability.
- Produces all the files (Object, Listing, Symbol, and special) required for symbolic debug with Microchip's emulator systems.
- Supports Hex (default), Decimal and Octal source and listing formats.

MPASM provides a rich directive language to support programming of the PICmicro. Directives are helpful in making the development of your assemble source code shorter and more maintainable.

32.3.2 C Compiler (MPLAB-C)

The MPLAB-C is a complete 'C' compiler for Microchip's PICmicro family of microcontrollers. The compiler provides powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compiler provides symbol information that is compatible with the MPLAB IDE memory display, Watch windows, and File register windows.

32.3.3 MPLINK Linker

MPLINK is a linker for the Microchip C compiler, MPLAB-C, and the Microchip relocatable assembler, MPASM. MPLINK is introduced with MPLAB-C v2.00 and can only be used with these or later versions.

MPLINK allows you to produce modular, re-usable code with MPLAB-C and MPASM. Control over the linking process is accomplished through a linker “script” file and with command line options. MPLINK ensures that all symbolic references are resolved and that code and data fit into the available PICmicro device.

MPLINK combines multiple input object modules generated by MPLAB-C or MPASM, into a single executable file. The actual addresses of data and the location of functions will be assigned when MPLINK is executed. This means that you will instruct MPLINK to place code and data somewhere within the named regions of memory, not to specific physical locations.

Once the linker knows about the ROM and RAM memory regions available in the target PICmicro device and it analyzes all the input files, it will try to fit the application’s routines into ROM and assign it’s data variables into available RAM. If there is too much code or too many variables to fit, MPLINK will give an error message.

MPLINK also provides flexibility for specifying that certain blocks of data memory are re-usable, so that different routines (which never call each other and don’t depend on this data to be retained between execution) can share limited RAM space.

32.3.4 MPLIB Librarian

MPLIB is a librarian for use with COFF object modules created using either MPASM v2.0, MPASMWIN v2.0, or MPLAB-C v2.0 or later.

MPLIB manages the creation and modification of library files. A library file is a collection of object modules that are stored in a single file. There are several reasons for creating library files:

- Libraries make linking easier. Since library files can contain many object files, the name of a library file can be used instead of the names of many separate object when linking.
- Libraries help keep code small. Since a linker only uses the required object files contained in a library, not all object files which are contained in the library necessarily wind up in the linker’s output module.
- Libraries make projects more maintainable. If a library is included in a project, the addition or removal of calls to that library will not require a change to the link process.
- Libraries help convey the purpose of a group of object modules. Since libraries can group together several related object modules, the purpose of a library file is usually more understandable that the purpose of its individual object modules. For example, the purpose of a file named “math.lib” is more apparent that the purpose of 'power.o', 'ceiling.o', and 'floor.o'.

32.4 MPLAB-SIM Simulator Software

The software simulator is a no-cost tool with which to evaluate Microchip's products and designs. The use of the simulator greatly helps debug software, particularly algorithms. Depending on the complexity of a design project a time/cost benefit should be looked at comparing the simulator with an emulator.

For projects that have multiple engineers in the development, the simulator in conjunction with an emulator can keep costs down and will allow speedy debug of the tough problems.

MPLAB-SIM Simulator simulates the PICmicro series microcontrollers on an instruction level. On any given instruction, the user may examine or modify any of the data areas or provide external stimulus to any of the pins. The input/output radix can be set by the user and the execution can be performed in; single step, execute until break, or in a trace mode.

MPLAB-SIM supports symbolic debugging using MPLAB-C, and MPASM. The Software Simulator offers the low cost flexibility to develop and debug code outside of the laboratory environment making it an excellent multi-project software development tool.

32.5 MPLAB Emulator Hardware Support

Microchip offers two emulators, a high-end version (PICMASTER) and a low-cost version (ICEPIC). Both versions offer a very good price/feature value, and the selection of which emulator should depend on the feature set that you wish. For people looking at doing several projects with Microchip devices (or using the high-end devices) the use of PICMASTER may offset the additional investment, through time savings achieved with the sophisticated breakpoint and trace capabilities.

32.5.1 PICMASTER: High Performance Universal In-Circuit Emulator

The PICMASTER Universal In-Circuit Emulator provides the product development engineer with a complete microcontroller design tool set for all microcontrollers in the Baseline, Mid-Range, and High End families. PICMASTER operates in the MPLAB™ Integrated Development Environment (IDE), which allows editing, “make” and download, and source debugging from a single environment.

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

The PICMASTER Emulator System has been designed as a real-time emulation system with advanced features that are generally found on more expensive development tools.

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

32.5.2 ICEPIC: Low-Cost PIC16CXXX In-Circuit Emulator

ICEPIC is a low-cost in-circuit emulator solution for the Microchip Base-line and Mid-Range families of 8-bit OTP microcontrollers.

ICEPIC user interface operates on PC-compatible machines ranging from 286-AT® through Pentium™ based machines under Windows 3.x environment. ICEPIC features real-time emulation. ICEPIC is available under the MPLAB environment.

ICEPIC is designed by Neosoft Inc. and is manufactured under license by RF Solutions. Other emulator solutions may be available directly from RF solutions.

32.6 MPLAB Programmer Support

Microchip offers two levels of device programmer support. For most bench setups the PICSTART Plus is sufficient. When true system qualification is done, the PRO MATE II should be the minimum used, due to the validation of program memory at VDD min and VDD max for maximum reliability

32.6.1 PRO MATE® II: Universal Device Programmer

The PRO MATE II Universal Programmer is a full-featured programmer capable of operating in stand-alone mode as well as PC-hosted mode. PRO MATE II operates under MPLAB or as a DOS command driven program.

The PRO MATE II has programmable VDD and VPP supplies which allows it to verify programmed memory at VDD min and VDD max for maximum reliability. It has an LCD display for error messages, keys to enter commands and a modular detachable socket assembly to support various package types. In stand-alone mode the PRO MATE II can read, verify or program Baseline, Mid-Range, and High End devices. It can also set configuration and code-protect bits in this mode. The PRO MATE II programmer also supports Microchip's Serial EEPROM and KEELOQ® Security devices.

A separate In-Circuit Serial Programming (ICSP) module is available for volume programming in a manufacturing environment. See the Programming module documentation for specific application requirements.

32.6.2 PICSTART® Plus Low-Cost Development Kit

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

PICSTART Plus supports all Baseline, Mid-Range, and High End devices. For devices with up more than 40 pins an adapter socket is required. DIP packages are the form factor that are directly supported. Other package types may be supported with adapter sockets.

32.7 Supplemental Tools

Microchip endeavors to provide a broad range of solutions to our customers. Some of these products may fall outside the realm of the classic development tools and include more advanced topics such as high level languages, fuzzy logic, or visual programming aids. These tools are considered supplemental tools and may be available directly from Microchip or from another vendor. A comprehensive listing of alternate tool providers is contained in the Third Party Guide.

32.7.1 *fuzzy*TECH-MP Fuzzy Logic Development System

The *fuzzy*TECH-MP fuzzy logic development tool is available in two versions - a low cost introductory version, MP Explorer, for designers to gain a comprehensive working knowledge of fuzzy logic system design, and a full-featured version, *fuzzy*TECH-MP, for implementing more complex systems.

Both versions include Microchip's *fuzzy*LAB™ demonstration board for hands-on experience with fuzzy logic systems implementation.

32.7.2 MP-DriveWay™ – Application Code Generator

MP-DriveWay is an easy-to-use Windows-based Application Code Generator. With MP-DriveWay you can visually configure all the peripherals in a PIC16/17 device and, with a click of the mouse, generate all the initialization and many functional code modules in C language. The output is fully compatible with Microchip's MPLAB-C C compiler. The code produced is highly modular and allows easy integration of your own code.

32.7.3 Third Party Guide

Looking for something else? Microchip strongly encourages and supports it's Third Parties. Microchip publishes the "Third Party Guide". It is an extensive volume that provides:

- Company
- Product
- Contact Information
- Consultants

For over 100 companies and 200 products. These products include Emulators, Device Programmers, Gang Programmers, Language Products, and other tool solutions.

32.8 Development Boards

Development boards give a quick start on a circuit that demonstrates the capabilities of a particular device. The device program can then be modified for your own evaluation of the device functionality and operation.

32.8.1 PICDEM-1 Low-Cost PIC16/17 Demonstration Board

The PICDEM-1 is a simple board which demonstrates the capabilities of several of Microchip's microcontrollers. The microcontrollers supported are: PIC16C5X (PIC16C54 to PIC16C58A), PIC16C61, PIC16C62X, PIC16C71, PIC16C710, PIC16C711, PIC16C8X, PIC17C42A, PIC17C43 and PIC17C44. All necessary hardware and software is included to run basic demo programs. The users can program the sample microcontrollers provided with the PICDEM-1 board, on a PRO MATE II or PICSTART-Plus programmer, and easily test firmware. The user can also connect the PICDEM-1 board to the PICMASTER emulator and download the firmware to the emulator for testing. Additional prototype area is available to build additional hardware. Some of the features include an RS-232 interface, a potentiometer for simulated analog input, push-button switches and eight LEDs connected to PORTB.

32.8.2 PICDEM-2 Low-Cost PIC16CXXX Demonstration Board

The PICDEM-2 is a simple demonstration board that supports the PIC16C62, PIC16C63, PIC16C64, PIC16C65, PIC16C72, PIC16C73 and PIC16C74 microcontrollers. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM-2 board, on a PRO MATE II programmer or PICSTART-Plus, and easily test firmware. The PICMASTER emulator may also be used with the PICDEM-2 board to test firmware. Additional prototype area has been provided for additional hardware. 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.

32.8.3 PICDEM-3 Low-Cost PIC16CXXX Demonstration Board

The PICDEM-3 is a simple demonstration board that supports the PIC16C923 and PIC16C924 in the PLCC package. It will also support future 44-pin PLCC microcontrollers that have 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 board, on a PRO MATE II programmer or PICSTART Plus with an adapter socket, and easily test firmware. The PICMASTER emulator may also be used with the PICDEM-3 board to test firmware. Additional prototype area has been provided for adding hardware. Some of the features include an RS-232 interface, push-button switches, a potentiometer for simulated analog input, a thermistor and separate headers for connection to an external LCD module and a keypad. Also provided on the PICDEM-3 board is an LCD panel, with 4 commons and 12 segments, that is capable of displaying time, temperature and day of the week. The PICDEM-3 provides an additional RS-232 interface and Windows 3.1 software for showing the de-multiplexed LCD signals on a PC. A simple serial interface allows the user to construct a hardware de-multiplexer for the LCD signals.

32.8.4 PICDEM-14A Low-Cost PIC14C000 Demonstration Board

The PICDEM-14A demo board is a general purpose platform which is provided to help evaluate the PIC14C000 mixed signal microcontroller. The board runs a PIC14C000 measuring the voltage of a potentiometer and the on-chip temperature sensor. The voltages are then calibrated to the internal bandgap voltage reference. The voltage and temperature data are then transmitted to the RS-232 port. This data can be displayed using a terminal emulation program, such as Windows Terminal. This demo board also includes peripherals that allow users to display data on an LCD panel, read from and write to a serial EEPROM, and prototype custom circuitry to interface to the microcontroller.

32.9 Development Tools for Other Microchip Products

32.9.1 SEEVAL® Evaluation and Programming System

The SEEVAL Serial EEPROM Designer's Kit supports all Microchip 2-wire and 3-wire Serial EEPROMs. The kit includes everything necessary to read, write, erase or program special features of any Microchip SEEPROM product including Smart Serials™ and secure serials. The Total Endurance™ Disk is included to aid in trade-off analysis and reliability calculations. The total endurance kit can significantly reduce time-to-market and results in a more optimized system.

32.9.2 KEELOQ® Evaluation and Programming Tools

KEELOQ evaluation and programming tools supports Microchip's HCS Secure Data Products. The HCS evaluation kit includes an LCD display to show changing codes, a decoder to decode transmissions, and a programming interface to program test transmitters.

32.10 Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the Mid-Range MCU family (that is they may be written for the Base-Line, or the High-End), but the concepts are pertinent, and could be used (with modification and possible limitations). The current application notes related to Microchip's development tools are:

Title	Application Note #
Air Flow using Fuzzy Logic	AN600

32.11 Revision History

Revision A

This is the initial released revision of Microchip's development tools description.

Section 33. Code Development

HIGHLIGHTS

No material is available at this time. Please monitor the Microchip web site for the B revision of the Code Development section of the Mid-range Reference Manual.

33.1 Revision History

Revision A

This is the initial released revision for the Code Development with a PICmicro™ description.

Section 34. Appendix

HIGHLIGHTS

This section of the manual contains the following major topics:

I²C is a trademark of Philips Corporation.

APPENDIX A: I²C™ OVERVIEW

This section provides an overview of the Inter-Integrated Circuit (I²C™) bus, with Subsection **A.2 "Addressing I²C Devices"** discussing the operation of the SSP modules in I²C mode.

The I²C bus is a two-wire serial interface. The original specification, or standard mode, is for data transfers of up to 100 Kbps. An enhanced specification, or fast mode (400 Kbps) is supported. Standard and Fast mode devices will operate when attached to the same bus, if the bus operates at the speed of the slower device.

The I²C interface employs a comprehensive protocol to ensure reliable transmission and reception of data. When transmitting data, one device is the "master" which initiates transfer on the bus and generates the clock signals to permit that transfer, while the other device(s) acts as the "slave." All portions of the slave protocol are implemented in the SSP module's hardware, except general call support, while portions of the master protocol need to be addressed in the PIC16CXX software. The MSSP module supports the full implementation of the I²C master protocol, the general call address, and data transfers upto 1 Mbps. The 1 Mbps data transfers are supported by some of Microchips Serial EEPROMs. Table A-1 defines some of the I²C bus terminology.

In the I²C interface protocol each device has an address. When a master wishes to initiate a data transfer, it first transmits the address of the device that it wishes to "talk" to. All devices "listen" to see if this is their address. Within this address, a bit specifies if the master wishes to read-from/write-to the slave device. The master and slave are always in opposite modes (transmitter/receiver) of operation during a data transfer. That is they can be thought of as operating in either of these two relations:

- Master-transmitter and Slave-receiver
- Slave-transmitter and Master-receiver

In both cases the master generates the clock signal.

The output stages of the clock (SCL) and data (SDA) lines must have an open-drain or open-collector in order to perform the wired-AND function of the bus. External pull-up resistors are used to ensure a high level when no device is pulling the line down. The number of devices that may be attached to the I²C bus is limited only by the maximum bus loading specification of 400 pF and addressing capability.

A.1 Initiating and Terminating Data Transfer

During times of no data transfer (idle time), both the clock line (SCL) and the data line (SDA) are pulled high through the external pull-up resistors. The START and STOP conditions determine the start and stop of data transmission. The START condition is defined as a high to low transition of the SDA when the SCL is high. The STOP condition is defined as a low to high transition of the SDA when the SCL is high. Figure A-1 shows the START and STOP conditions. The master generates these conditions for starting and terminating data transfer. Due to the definition of the START and STOP conditions, when data is being transmitted, the SDA line can only change state when the SCL line is low.

Figure A-1: Start and Stop Conditions

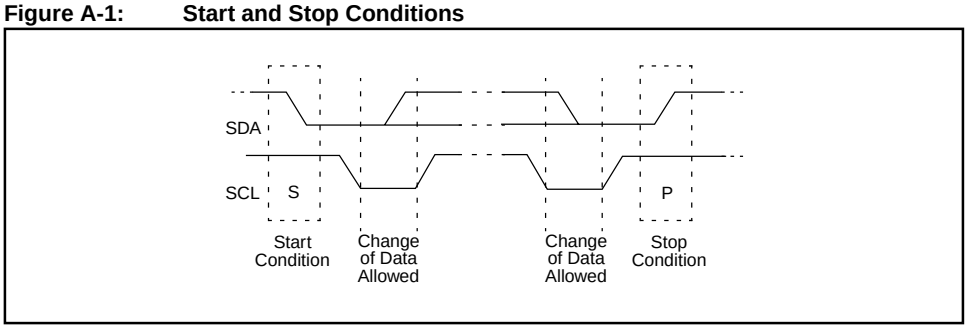


Table A-1: I^2C Bus Terminology

Term	Description
Transmitter	The device that sends the data to the bus.
Receiver	The device that receives the data from the bus.
Master	The device which initiates the transfer, generates the clock and terminates the transfer.
Slave	The device addressed by a master.
Multi-master	More than one master device in a system. These masters can attempt to control the bus at the same time without corrupting the message.
Arbitration	Procedure that ensures that only one of the master devices will control the bus. This ensure that the transfer data does not get corrupted.
Synchronization	Procedure where the clock signals of two or more devices are synchronized.

A.2 Addressing I²C Devices

There are two address formats. The simplest is the 7-bit address format with a R/$\overline{W}$ bit (Figure A-2). The more complex is the 10-bit address with a R/$\overline{W}$ bit (Figure A-3). For 10-bit address format, two bytes must be transmitted. The first five bits specify this to be a 10-bit address format. The 1st transmitted byte has 5-bits which specify a 10-bit address, the two MSbs of the address, and the R/$\overline{W}$ bit. The second byte is the remaining 8-bits of the address.

Figure A-2: 7-bit Address Format

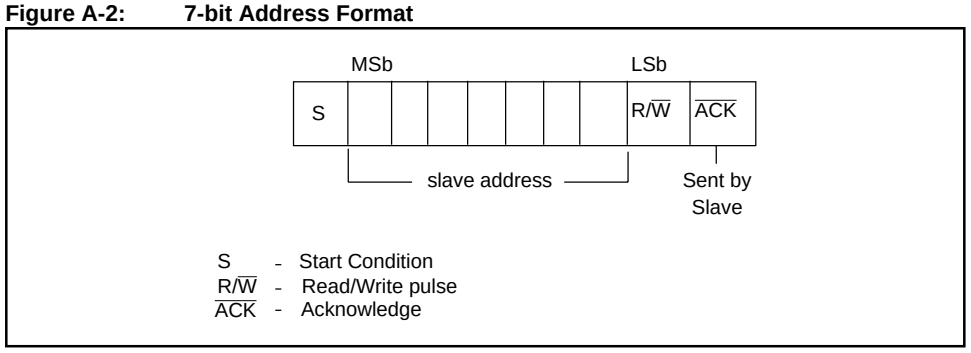
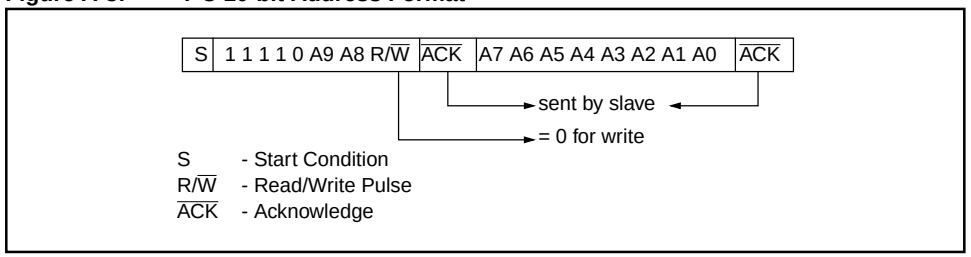


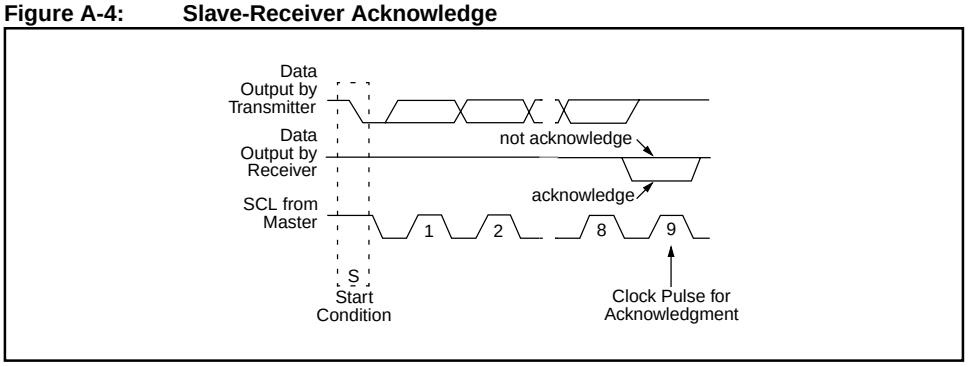
Figure A-3: I²C 10-bit Address Format



A.3 Transfer Acknowledge

All data must be transmitted per byte, with no limit to the number of bytes transmitted per data transfer. After each byte, the slave-receiver generates an acknowledge bit ($\overline{ACK}$) (Figure A-4). When a slave-receiver doesn't acknowledge the slave address or received data, the master must abort the transfer. The slave must leave SDA high so that the master can generate the STOP condition (Figure A-1).

Figure A-4: Slave-Receiver Acknowledge



If the master is receiving the data (master-receiver), it generates an acknowledge signal for each received byte of data, except for the last byte. To signal the end of data to the slave-transmitter, the master does not generate an acknowledge (not acknowledge). The slave then releases the SDA line so the master can generate the STOP condition. The master can also generate the STOP condition during the acknowledge pulse for valid termination of data transfer.

If the slave needs to delay the transmission of the next byte, holding the SCL line low will force the master into a wait state. Data transfer continues when the slave releases the SCL line. This allows the slave to move the received data or fetch the data it needs to transfer before allowing the clock to start. This wait state technique can also be implemented at the bit level, Figure A-5.

Figure A-5: Data Transfer Wait State

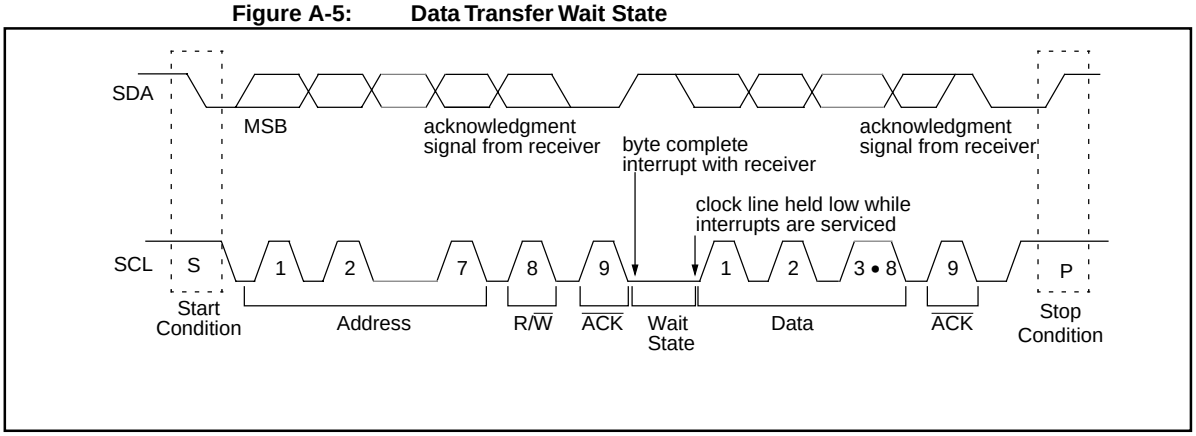


Figure A-6 and Figure A-7 show Master-transmitter and Master-receiver data transfer sequences.

Figure A-6: Master-Transmitter Sequence

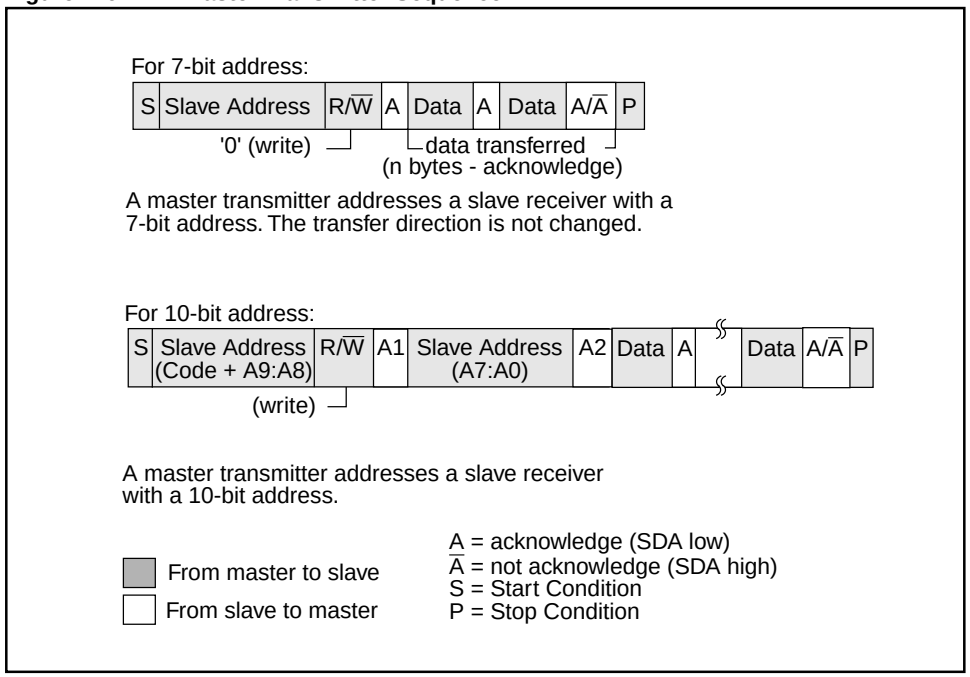
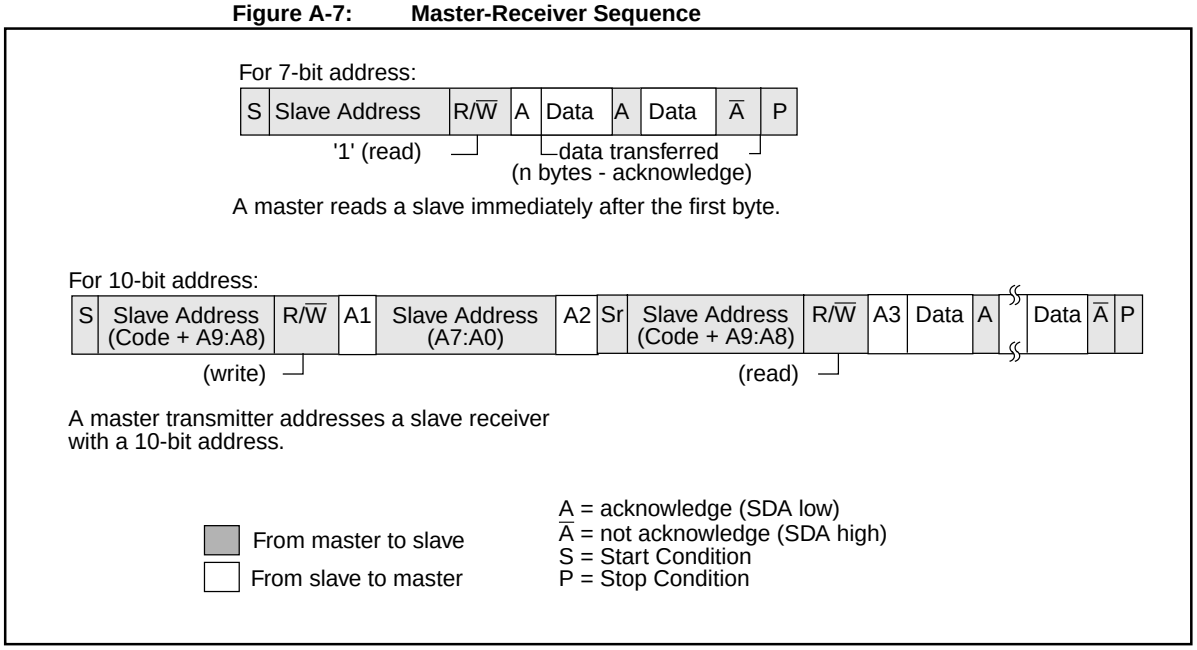
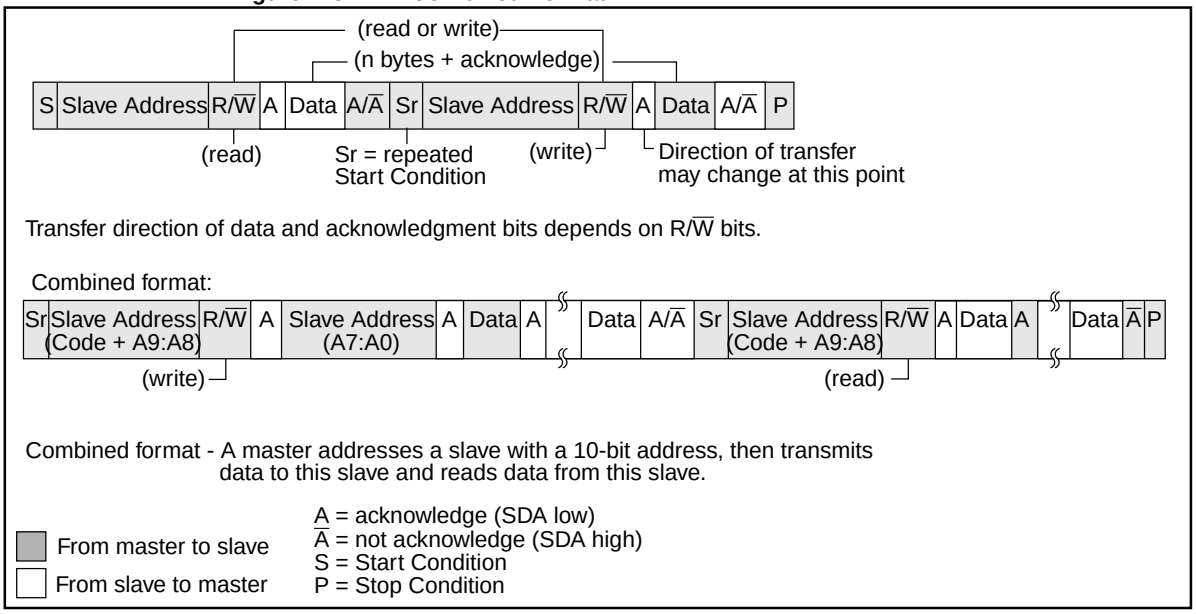


Figure A-7: Master-Receiver Sequence



When a master does not wish to relinquish the bus (which occurs by generating a STOP condition), a repeated START condition (Sr) must be generated. This condition is identical to the start condition (SDA goes high-to-low while SCL is high), but occurs after a data transfer acknowledge pulse (not the bus-free state). This allows a master to send “commands” to the slave and then receive the requested information or to address a different slave device. This sequence is shown in Figure A-8.

Figure A-8: Combined Format



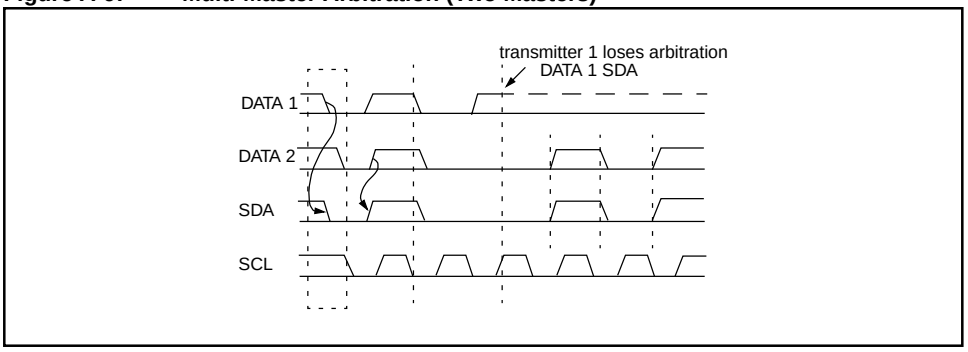
A.4 Multi-master

The I^2C protocol allows a system to have more than one master. This is called multi-master. When two or more masters try to transfer data at the same time, arbitration and synchronization occur.

A.4.1 Arbitration

Arbitration takes place on the SDA line, while the SCL line is high. The master which transmits a high when the other master transmits a low loses arbitration (Figure A-9), and turns off its data output stage. A master which lost arbitration can generate clock pulses until the end of the data byte where it lost arbitration. When the master devices are addressing the same device, arbitration continues into the data.

Figure A-9: Multi-Master Arbitration (Two Masters)



Masters that also incorporate the slave function, and have lost arbitration must immediately switch over to slave-receiver mode. This is because the winning master-transmitter may be addressing it.

Arbitration is not allowed between:

- A repeated START condition
- A STOP condition and a data bit
- A repeated START condition and a STOP condition

Care needs to be taken to ensure that these conditions do not occur.

A.4.2 Clock Synchronization

Clock synchronization occurs after the devices have started arbitration. This is performed using a wired-AND connection to the SCL line. A high to low transition on the SCL line causes the concerned devices to start counting off their low period. Once a device clock has gone low, it will hold the SCL line low until its SCL high state is reached. The low to high transition of this clock may not change the state of the SCL line, if another device clock is still within its low period. The SCL line is held low by the device with the longest low period. Devices with shorter low periods enter a high wait-state, until the SCL line comes high. When the SCL line comes high, all devices start counting off their high periods. The first device to complete its high period will pull the SCL line low. The SCL line high time is determined by the device with the shortest high period, Figure A-10.

Figure A-10: Clock Synchronization

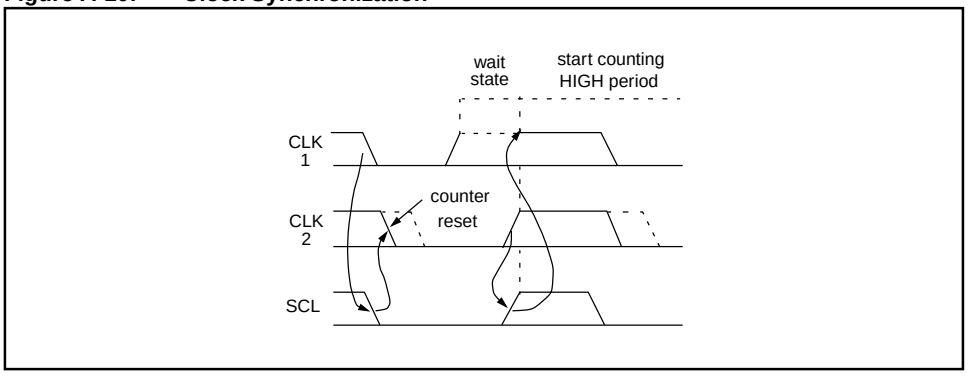


Figure A-11: I²C Bus Start/Stop Bits Timing Specification

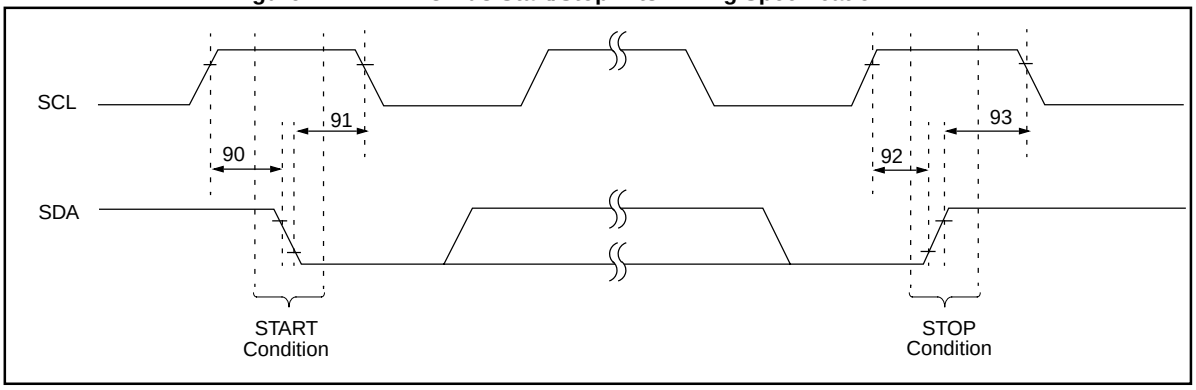


Table A-2: I²C Bus Start/Stop Bits Timing Specification

Microchip Parameter No.	Sym	Characteristic		Min	Typ	Max	Units	Conditions
90	TSU:STA	START condition Setup time	100 kHz mode	4700	—	—	ns	Only relevant for repeated START condition
			400 kHz mode	600	—	—		
91	THD:STA	START condition Hold time	100 kHz mode	4000	—	—	ns	After this period the first clock pulse is generated
			400 kHz mode	600	—	—		
92	TSU:STO	STOP condition Setup time	100 kHz mode	4700	—	—	ns	
			400 kHz mode	600	—	—		
93	THD:STO	STOP condition Hold time	100 kHz mode	4000	—	—	ns	
			400 kHz mode	600	—	—		

Figure A-12: I²C Bus Data Timing Specification

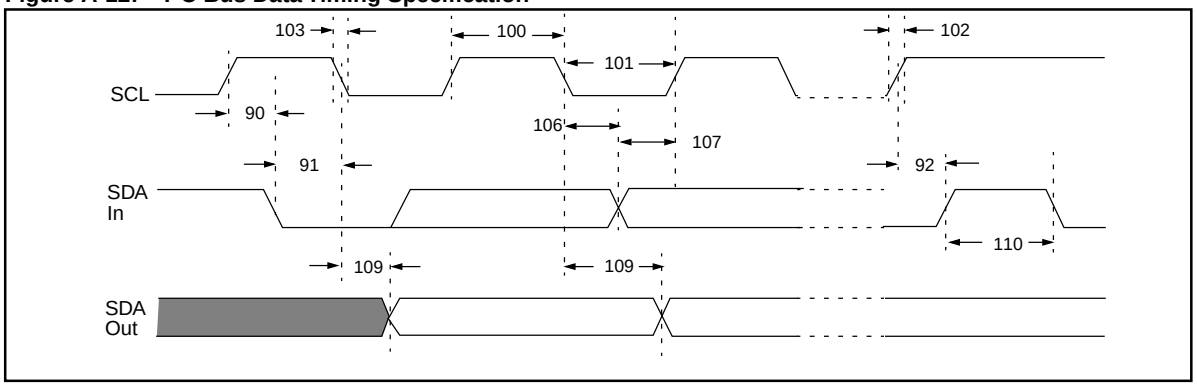


Table A-3: I²C Bus Data Timing Specification

Microchip Parameter No.	Sym	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock high time	100 kHz mode	4.0	—	µs	
			400 kHz mode	0.6	—	µs	
101	TLOW	Clock low time	100 kHz mode	4.7	—	µs	
			400 kHz mode	1.3	—	µs	
102	TR	SDA and SCL rise time	100 kHz mode	—	1000	ns	
			400 kHz mode	20 + 0.1Cb	300	ns	Cb is specified to be from 10 to 400 pF
103	TF	SDA and SCL fall time	100 kHz mode	—	300	ns	
			400 kHz mode	20 + 0.1Cb	300	ns	Cb is specified to be from 10 to 400 pF
90	TSU:STA	START condition setup time	100 kHz mode	4.7	—	µs	Only relevant for repeated START condition
			400 kHz mode	0.6	—	µs	
91	THD:STA	START condition hold time	100 kHz mode	4.0	—	µs	After this period the first clock pulse is generated
			400 kHz mode	0.6	—	µs	
106	THD:DAT	Data input hold time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	µs	
107	TSU:DAT	Data input setup time	100 kHz mode	250	—	ns	Note 2
			400 kHz mode	100	—	ns	
92	TSU:STO	STOP condition setup time	100 kHz mode	4.7	—	µs	
			400 kHz mode	0.6	—	µs	
109	TAA	Output valid from clock	100 kHz mode	—	3500	ns	Note 1
			400 kHz mode	—	1000	ns	
110	TBUF	Bus free time	100 kHz mode	4.7	—	µs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	µs	
D102	Cb	Bus capacitive loading		—	400	pF	

Note 1: As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of START or STOP conditions.

2: A fast-mode I²C-bus device can be used in a standard-mode I²C-bus system, but the requirement tsu;DAT ≥ 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
TR max.+tsu;DAT = 1000 + 250 = 1250 ns (according to the standard-mode I²C bus specification) before the SCL line is released.

APPENDIX B: LIST OF LCD GLASS MANUFACTURERS

AEG-MIS
3340 Peachtree Rd. NE Suite 500
Atlanta, GA 30326
TEL: 404-239-0277
FAX: 404-239-0383

All Shore INDS Inc.
1 Edgewater Plaza
Staten Island, NY 10305
TEL: 718-720-0018
FAX: 718-720-0225

Crystaloid
5282 Hudson Drive
Hudson, OH 44236-3769
TEL: 216-655-2429
FAX: 216-655-2176

DCI Inc.
14812 W. 117th St.
Olathe, KS 66062-9304
TEL: 913-782-5672
FAX: 913-782-5766

Excel Technology International Corporation
Unit 5, Bldg. 4, Stryker Lane
Belle Mead, NJ 08502
TEL: 908-874-4747
FAX: 908-874-3278

F-P Electronics/Mark IV Industries
6030 Ambler Drive
Mississauga, ON Canada L4W 2PI
TEL: 905-624-3020
FAX: 905-238-3141

Hunter Components
24800 Chagrin Blvd, Suite 101
Cleveland, OH 44122
TEL: 216-831-1464
FAX: 216-831-1463

Interstate Electronics Corp.
1001 E. Bull Rd.
Anaheim, CA 92805
TEL: 800-854-6979
FAX: 714-758-4111

Kent Display Systems
343 Portage Blvd.
Kent, OH 44240
TEL: 330-673-8784

LCD Planar Optics Corporation
2100-2 Artic Ave.
Bohemia, NY 11716
TEL: 516-567-4100
FAX: 516-567-8516

LXD Inc.
7650 First Place
Oakwood Village, OH 44146
TEL: 216-786-8700
FAX: 216-786-8711

Nippon Sheet Glass
Tomen America Inc.
1285 Avenue of the Americas
New York, NY 10019
TEL: 212-397-4600
FAX: 212-397-3351

OPTREX America
44160 Plymouth Oaks Blvd.
Plymouth, MI 48170
TEL: 313-416-8500
FAX: 313-416-8520

Phillips Components
LCD Business Unit
1273 Lyons Road, Bldg G
Dayton, OH 45459
TEL: 573-436-9500
FAX: 573-436-2230

Satori Electric
23717 Hawthorne Blvd. 3rd Floor
Torrance, CA 90505
TEL: 310-214-1791
FAX: 310-214-1721

Seiko Instruments USA Inc.
Electronic Components Division
2990 West Lomita Blvd.
Torrance, CA 90505
TEL: 213-517-7770
213-517-8113
FAX: 213-517-7792

Standish International
European Technical Center
Am Baümstuck II
65520 Bad Camberg/Erbach
Germany
TEL: 011 49 6434 3324
FAX: 011 49 6434 377238

Standish LCD
W7514 Highway V
Lake Mills, WI 53551
TEL: 414-648-1000
FAX: 414-648-1001

Truly Semiconductors Ltd. (USA)
2620 Concord Ave.
Suite 106
Alhambra, CA 91803
TEL: 818-284-3033
FAX: 818-284-6026

Truly Semiconductor Ltd.
2/F, Chung Shun Knitting Center
1-3 Wing Yip Street,
Kwai Chung, N.T., Hong Kong
TEL: 852 2487 9803
FAX: 852 2480 0126

Varitronix Limited Inc.
3250 Wilshire Blvd. Suite 1901
Los Angeles, CA 90010
TEL: 213-738-8700
FAX: 213-738-5340

Varitronix Limited Inc.
4/F, Liven House
61-63 King Yip Street
Kwun Tong, Kowloon
Hong Kong
TEL: 852 2389 4317
FAX: 852 2343 9555

Varitronix (France) S.A.R.L.
13/15 Chemin De Chilly
91160 Champlain
France
TEL:(33) 1 69 09 7070
FAX:(33) 1 69 09 0535

Varitronix Italia, S.R.L.
Via Bruno Buozzi 90
20099 Sesto San Giovanni
Milano, Italy
TEL:(39) 2 2622 2744
FAX:(39) 2 2622 2745

Varitronix (UK) Limited
Display House, 3 Milbanke Court
Milbanke Way, Bracknell
Berkshire RG12 1BR
United Kingdom
TEL:(44) 1344 30377
FAX(44) 1344 300099

Varitronix (Canada) Limited
18 Crown Steel Drive, Suite 101
Markham, Ontario
Canada L3R 9X8
TEL:(905) 415-0023
FAX:(905) 415-0094

Vikay America Inc.
195 W. Main St.
Avon, CT 06001-3685
TEL: 860-678-7600
FAX: 860-678-7625

APPENDIX C: DEVICE ENHANCEMENT

As the Midrange architecture matured, certain modules and features have been enhanced. They are:

1. The data memory map
2. The SSP module
3. The A/D module
4. Brown-out Reset added to the core
5. $\overline{\text{MCLR}}$ Filter
6. USART
7. Device Oscillator

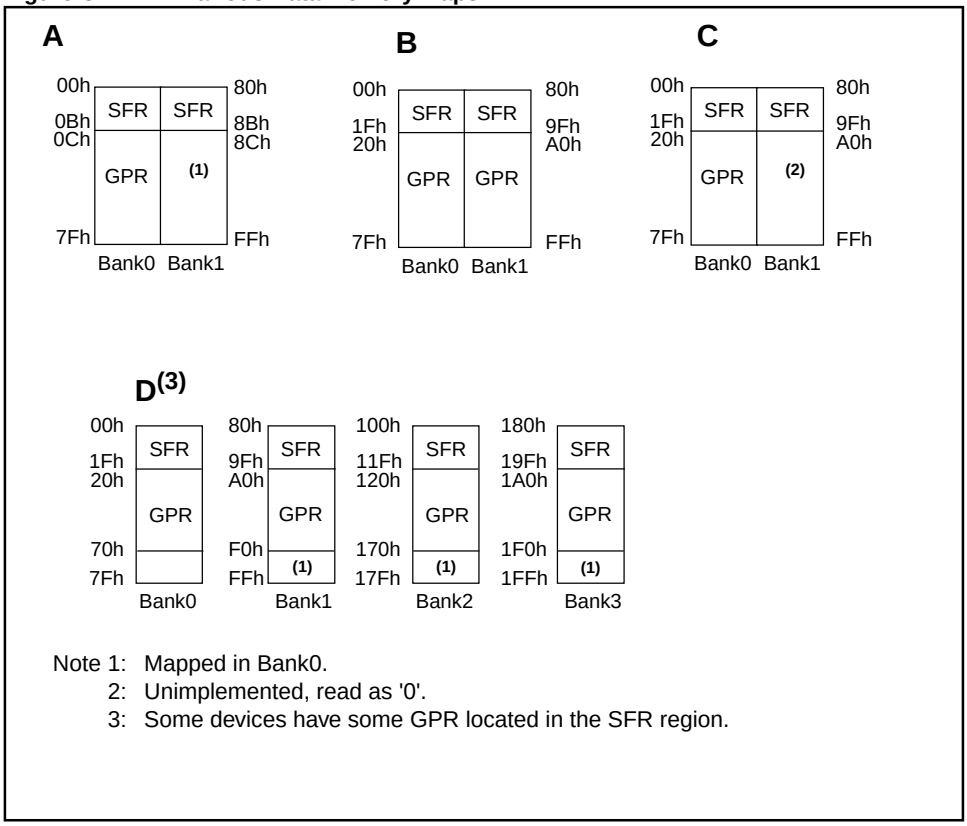
The following subsections discuss the implementations of these enhancements.

C.1 Data Memory Map

The Data Memory Map shows the location of the Special Function Registers (SFRs) and the General Purpose Registers (GPRs). SFRs provide controls and give status on the operation of the device, while the GPRs are the general purpose RAM.

Figure C-1 show the various memory maps that have been implemented in the midrange family. Memory Map **A** was implemented on the first midrange devices. They were 18/20-pin devices that had limited peripheral features. When the product roadmap dictated the requirement for devices with increased I/O, and a richer peripheral set, memory map **B** was implemented. Memory map **C** is actually a subset of memory map **B**, but context saving (due to an interrupt) requires additional software overhead. This is because there is no GPR in Bank1. To minimize the context saving software, memory map **D** was defined. A common RAM memory map will be used for all future devices. See the **"Memory Organization"** section for use and implementation of the Midrange PICmicro's memory.

Figure C-1: Various Data Memory Maps



Note 1: Mapped in Bank0.
2: Unimplemented, read as '0'.
3: Some devices have some GPR located in the SFR region.

C.2 SSP (Synchronous Serial Port) Module

The SSP module has two modes of operation;

- SPI (Serial Peripheral Interface)
- I^2C (Inter-Integrated Circuit).

There are now three different SSP modules that exist in Microchip's design library. The first SSP module (now called Basic SSP) implements two of the four SPI modes, and the I^2C module in slave mode. The second SSP module (called SSP) implements all four SPI modes, and the I^2C module in slave mode. The third SSP module (called Master SSP) implements all four SPI modes, and the I^2C module in master and slave modes. Table C-1 shows the devices that have an SSP module and denotes which version is implemented. As new devices are introduced, either the SSP module or Master SSP module will be implemented (that is, the Basic SSP module is being phased out). Only select devices will be introduced with the Master SSP module due to the size (silicon area => cost) difference in relation to the SSP module. If your application requires I^2C Master mode, then you should also check into Microchip's high-end family, PIC17CXXX.

Table C-1: Devices With an SSP module

Device	Synchronous Serial Port Version		
	SSP	Basic SSP	Master SSP (1)
PIC16C62	—	Yes	—
PIC16C62A	—	Yes	—
PIC16CR62	—	Yes	—
PIC16C63	—	Yes	—
PIC16CR63	—	Yes	—
PIC16C64	—	Yes	—
PIC16C64A	—	Yes	—
PIC16CR64	—	Yes	—
PIC16C65	—	Yes	—
PIC16C65A	—	Yes	—
PIC16CR65	—	Yes	—
PIC16C66	Yes	—	—
PIC16C67	Yes	—	—
PIC16C72	—	Yes	—
PIC16CR72	Yes	—	—
PIC16C73	—	Yes	—
PIC16C73A	—	Yes	—
PIC16C74	—	Yes	—
PIC16C74A	—	Yes	—
PIC16C76	Yes	—	—
PIC16C77	Yes	—	—
PIC16C923	Yes	—	—
PIC16C924	Yes	—	—
Future Devices with SSP module	See Device Data Sheet	—	See Device Data Sheet

Note 1: At present NO midrange devices are available with the Master SSP module. Please refer to Microchip's Web site or BBS for release of Product Briefs. You will be able to find out the details of features for new devices.

This module is available on Microchip's High End family (PIC17CXXX). Please refer to Microchip's Web site, BBS, Regional Sales Office, or Factory Representatives.

C.3 A/D (Analog-to-Digital) Module

There now exists several different versions of the A/D module in Microchip's design library. The first A/D module (now called Basic 8-bit A/D) is an 8-bit A/D with four input channels. The second A/D module (called 8-bit A/D) is an 8-bit A/D with up to 8 input channels. The Third A/D module (called 10-bit A/D) is a 10-bit A/D with up to16 input channels implemented. Table C-2 shows which devices have an A/D module, and the version implemented. As new devices are introduced, either the 8-bit A/D module or 10-bit A/D module will be implemented (that is the Basic 8-bit A/D module is being phased out). If your application requires the 10-bit A/D, you should refer to Microchip's High End Family (PIC17CXXX). This family currently has some devices that have this module implemented.

Table C-2: Devices With A/D modules

Device	8-bit A/D	Basic 8-bit A/D	10-bit A/D (1)	Slope A/D
PIC16C710	—	Yes	—	
PIC16C71	—	Yes	—	
PIC16C711	—	Yes	—	
PIC16C715	—	Yes	—	
PIC16C72	Yes	—	—	
PIC16CR72	Yes	—	—	—
PIC16C73	Yes	—	—	
PIC16C73A	Yes	—	—	
PIC16C74	Yes	—	—	
PIC16C74A	Yes	—	—	
PIC16C76	Yes	—	—	
PIC16C77	Yes	—	—	
PIC16C924	Yes	—	—	
PIC14C000	—	—	—	Yes
Future Devices with A/D module	See Device Data Sheet	See Device Data Sheet	See Device Data Sheet	See Device Data Sheet

Note 1: At present NO midrange devices are available with the 10-bit A/D module. Please refer to Microchip's Web site or BBS for release of Product Briefs. You will be able to find out the details of features for new devices.

This module is available on Microchip's High End family (PIC17CXXX). Please refer to Microchip's Web site, BBS, Regional Sales Office, or Factory Representatives.

C.4 Brown-out Reset

An internal Brown-out Reset (BOR) circuit was added as a special feature. This circuit will be added to most new devices. The exception will be for devices whose target market will require normal operation below the BOR trip point (handheld battery applications). Table C-3 shows the devices that evolved into having the BOR circuitry.

Table C-3: Devices That Were Revised to Include On-chip Brown-out Reset

Base Device No Brown-out Reset	Subsequent Device with Brown-out Reset
PIC16C62	PIC16C62A
PIC16C64	PIC16C64A
PIC16C65	PIC16C65A
PIC16C71	PIC16C711
PIC16C73	PIC16C73A
PIC16C74	PIC16C74A

C.5 Comparator

If a change in the CMCON register (C1OUT or C2OUT) should occur when a read operation is being executed (start of the Q2 cycle), then the CMIF interrupt flag bit may not get set.

C.6 $\overline{\text{MCLR}}$ Filter

The master clear ($\overline{\text{MCLR}}$) logic has had a filter added. This filter ignores short duration (glitch) low level pulses on the Master Clear pin. Table C-4 shows whether the device has the master clear filter.

Table C-4: Devices With Master Clear Filter

Device	Master Clear	
	No Filter (Fast Reset)	Filter
PIC16C61	Yes	—
PIC16C62	Yes	—
PIC16C62A	—	Yes
PIC16CR62	—	Yes
PIC16C63	—	Yes
PIC16CR63	—	Yes
PIC16C64	Yes	—
PIC16C64A	—	Yes
PIC16CR64	—	Yes
PIC16C65	Yes	—
PIC16C65A	—	Yes
PIC16CR65	—	Yes
PIC16C66	—	Yes
PIC16C67	—	Yes
PIC16C620	—	Yes
PIC16C621	—	Yes
PIC16C622	—	Yes
PIC16C710	—	Yes
PIC16C71	Yes	—
PIC16C711	—	Yes
PIC16C715	—	Yes
PIC16C72	—	Yes
PIC16CR72	—	Yes
PIC16C73	Yes	—
PIC16C73A	—	Yes
PIC16C74	Yes	—
PIC16C74A	—	Yes
PIC16C76	—	Yes
PIC16C77	—	Yes
PIC16C83	Yes	—
PIC16C84	Yes	—
PIC16F83	Yes	—
PIC16F84	Yes	—
PIC16C923	—	Yes
PIC16C924	—	Yes
All New Devices	—	Yes

C.7 USART

The original USART/SCI module that was offered on Midrange devices specified a “high speed” mode (when the BRGH control bit is set). Due to the design of the sampling circuitry, the operation of this mode was not as robust as desired. The sampling circuitry has been changed so that operation now meets Microchip’s requirements. The difference in the sampling is described in the **“USART”** section. Table C-5 shows which devices use the new and old sampling logic.

Table C-5: USART/SCI Sampling Logic

Device	Sampling Logic	
	Old	New
PIC16C63	Yes	—
PIC16CR63	Yes	—
PIC16C65	Yes	—
PIC16C65A	Yes	—
PIC16CR65	Yes	—
PIC16C66	—	Yes
PIC16C67	—	Yes
PIC16C73	Yes	—
PIC16C73A	Yes	—
PIC16C74	Yes	—
PIC16C74A	Yes	—
PIC16C76	—	Yes
PIC16C77	—	Yes
New Devices with USART/SCI module	—	Yes

C.8 Device Oscillator

A new mode has been added into the device oscillator which allows the device to operate from an internal RC. This is specified at time of device programming (configuration word). This mode will be included on many future devices. See the device data sheets configuration word to determine if the device supports this mode.

C.9 Parallel Slave Port

The control pins have changed from level sensitive to edge sensitive.

Table C-6: Parallel Slave Port Change Sensitivity

Device	Sensitivity	
	Level	Edge
PIC16C64	Yes	—
PIC16C64A	—	Yes
PIC16C65	Yes	—
PIC16C65A	—	Yes
PIC16C67	—	Yes
PIC16C74	Yes	—
PIC16C74A	—	Yes
PIC16C77	—	Yes
New Devices with Parallel Slave Port	—	Yes

APPENDIX D: REVISION HISTORY

Revision A

This is the initial released revision of the Reference Guide Appendix.

Section 35. Glossary

A

A/D

See Analog to Digital.

Acquisition Time (TACQ)

This is related to Analog to Digital (A/D) converters. This is the time that the A/D's holding capacitor acquires the analog input voltage level connected to it. When the GO bit is set, the analog input is disconnected from the holding capacitor and the A/D conversion is started.

ALU

Arithmetical Logical Unit. Device logic that is responsible for the mathematical (add, subtract, ...), logical (and, or, ...), and shifting operations.

Analog to Digital (A/D)

The conversion of an analog input voltage to a ratiometric digital equivalent value.

Assembly Language

A symbolic language that describes the binary machine code in a readable form.

B

Bank

This is a method of addressing Data Memory. Since midrange devices have 7-bits for direct addressing, instructions can address up to 128 bytes (including special function registers). To allow more data memory to be present on a device, data memory is partitioned into contiguous banks of 128 bytes each. To select the desired bank, the bank selection bits (RP1:RP0) need to be appropriately configured. Since there are presently 2 bank selection bits, 4 banks can be implemented.

Baud

Generally how the communication speed of serial ports is described. Equivalent to bits per second (bps).

BCD

See Binary Coded Decimal.

Binary Coded Decimal (BCD)

Each 4-bit nibble expresses a digit from 0-9. Usually two digits are packed to a byte giving a range of 0 - 99.

BOR

See Brown-out Reset.

Brown-out

A condition where the supply voltage of the device temporarily falls below the specified minimum operation point. This can occur when a load is switched on and causes the system/device voltage to drop.

Brown-out Reset (BOR)

Circuitry which will force the device to the reset state if the (device) voltage falls below a specified voltage level. Some devices have an internal BOR circuit, while other devices would require an external circuit to be created.

Bus width

This is the number of bits of information that the bus carries. For the Data Memory, the bus width is 8-bits. For the midrange devices the Program Memory bus width is 14-bits.

C

Capture

A function of the CCP module in which the value of a timer/counter is “captured”, into a holding register, when a predetermined event occurs.

CCP

Capture, Compare, Pulse Width Modulation (PWM). This module can be configured to operate as an input capture, or a timer compare, or a PWM output.

Common RAM

This is a region of the data memory RAM that is the same RAM location across all banks. This common RAM maybe implemented between addresses 70h -7Fh (inclusive). This common area is useful for the saving of required variables during context switching (such as during an interrupt).

Compare

A function of the CCP module in which the device will perform an action when a timer’s register value matches the value in the compare register.

Compare Register

A 16-bit register that contains a value that is compared to the 16-bit TMR1 register. The compare function triggers when the counter matches the contents of the compare register.

Capture Register

A 16-bit register that gets loaded with the value of the 16-bit TMR1 register when a capture event occurs.

Configuration Word

This is a location that specifies the characteristics that the device will have for operation (such as oscillator mode, WDT enable, start-up timer enables). These characteristics can be specified at time of device programming. For EPROM memory devices, as long as the bit is a '1', it may at a later time be programmed as a '0'. The device must be erased for a '0' to be returned to a '1'.

Conversion Time (Tconv)

This is related to Analog to Digital (A/D) converters. This is the time that the A/D converter requires to convert the analog voltage level on the holding capacitor to a digital value.

CPU

Central Processing Unit. Decodes the instructions, and determines the operands that are needed and the operations that need to be done. Arithmetic, logical, or shift operations will be passed to the ALU.

D

D/A

See Digital to analog

DAC

Digital to analog converter

Data Bus

The bus which is used to transfer data to and from the data memory.

Data EEPROM

Data Electrically Erasable Programmable Read Only Memory. This memory has the capability to be programmed and re-programmed by the CPU to ensure that in the case of a power loss critical values/variables are retained in the non-volatile memory.

Data Memory

The memory that is on the Data Bus. This memory is volatile (SRAM) and contains both the Special Function Registers and General Purpose Registers.

Direct Addressing

When the Data Memory Address is contained in the Instruction. The execution of this type of instruction will always access the data at the embedded address.

Digital to Analog

E

EEPROM

Electrically Erasable Programmable Read Only Memory. This memory has the capability to be programmed and erased in-circuit.

EPROM

Electrically Programmable Read Only Memory. This memory has the capability to be programmed in-circuit. Erasing requires that the program memory be exposed to UV light.

EXTRC

External Resistor-Capacitor (RC). Some devices have a device oscillator option that allows the clock to come from an external RC. This is the same as RC mode on some devices.

F

Flash Memory

This memory has the capability to be programmed and erased in-circuit. Program Memory technology that is almost functionally equivalent to Program EEPROM Memory.

Fosc

Frequency of the device oscillator.

G

GIO

General Input/Output

GPIO

General Purpose Input/Output

GPR

General Purpose Register (RAM). A portion of the data memory that can be used to store the program's dynamic variables.

H

Harvard Architecture

In this architecture the Program Memory and Data Memory buses are separated. This allows concurrent accesses to Data Memory and Program Memory, which increases the performance of the device.

Holding Capacitor

This is a capacitor in the Analog to Digital (A/D) module which "holds" to analog input level once the conversion is started. During acquisition, the holding capacitor is charged/discharged by the voltage level on the analog input pin. Once the conversion is started, the holding capacitor is disconnected from the analog input and "holds" this voltage for the A/D conversion.

HS

High Speed. One of the device oscillator modes. The oscillator circuit is tuned to support the high frequency operation. Used for operation from 4 MHz to 20 MHz.

I

I^2C

Inter-Integrated Circuit. This is a two wire communication interface. This feature is one of the modes of the SSP module.

Indirect Addressing

When the Data Memory Address is not contained in the Instruction. The instruction operates on the INDF address, which causes the Data Memory Address to be the value in the FSR register. The execution of the instruction will always access the data at the address pointed to by the FSR register.

Instruction Bus

The bus which is used to transfer instruction words from the program memory to the CPU.

Instruction Fetch

Due to the Harvard architecture, when one instruction is to be executed, the next location in program memory is “fetched” and ready to be decoded as soon as the currently executing instruction is completed.

Instruction cycle

The events for an instruction to execute. There are four events which can generally be described as: Decode, Read, Execute, and Write. Not all events will be done by all instructions. To see the operations during the instruction cycle, please look in the description of each instruction. Four external clocks (Tosc) make one instruction cycle (TCY).

Interrupt

A signal to the CPU that causes the program flow to be forced to the Interrupt Vector Address (04h in program memory). Before the program flow is changed, the contents of the Program Counter (PC) are forced onto the hardware stack, so that program execution may return to the interrupted point.

INTRC

Internal Resistor-Capacitor (RC). Some devices have a device oscillator option that allows the clock to come from an internal RC.

L

LCD

Liquid Crystal Display. Useful for giving visual status of a system. This may require the specification of custom LCD glass.

LED

Light Emitting Diode. Useful for giving visual status of a system.

Literal

This is a constant value that is embedded in an instruction word.

Long Word Instruction

An instruction word that embeds all the required information (opcode and data) into a single word. This ensures that every instruction is accessed and executed in a single instruction cycle.

LP

One of the device oscillator modes. Used for low frequency operation which allows the oscillator to be tuned for low power consumption. Operation is up to 200 kHz.

LSb

Least Significant Bit.

LSB

Least Significant Byte.

M

Machine cycle

This is a concept where the device clock is divided down to a unit time. For PICmicros this unit time is 4 times the device oscillator (4TOSC), also known as TCY.

MSb

Most Significant Bit.

MSB

Most Significant Byte.

N

Non-Return to Zero

Two level encoding used to transmit data over a communications medium. A bit value of '1' indicates a high voltage signal. A bit value of '0' indicates a low voltage signal. The data line defaults to a high level.

NRZ

See Non-Return to Zero

O

Opcode

The portion of the 14-bit instruction word that specifies the operation that needs to occur. The opcode is of variable length depending on the instruction that needs to be executed. The opcode varies from 4-bits to x-bits. The remainder of the instruction word contains program or data memory information.

Oscillator Start-up Timer (OST)

This timer counts 1024 crystal/resonator oscillator clock before releasing the internal reset signal.

OST

See Oscillator Start-up Timer.

P

Pages

Method of addressing the Program Memory. Midrange devices have 11-bit addressing for CALL and GOTO instructions, which gives these instructions a 2-Kword reach. To allow more program memory to be present on a device, program memory is partitioned into contiguous pages, where each page is 2-Kwords. To select the desired page, the page selection bits (PCLATCH<5:4>) need to be appropriately configured. Since there are presently 2 page selection bits, 4 pages can be implemented.

Parallel Slave Port (PSP)

A parallel communication port which is used to interface to a microprocessor's 8-bit data bus.

POP

A termed used to refer to the action of restoring information from a stack (software and/or hardware). See PUSH.

Postscaler

A circuit that slows the rate of the interrupt generation (or WDT reset) from a counter/timer by dividing it down.

Power-on Reset POR)

Circuitry which determines if the device voltage rose from a powered down level (0V). If the device voltage is rising from ground, a device reset occurs and the PWRT is started.

Power-up Timer (PWRT)

A timer which holds the internal reset signal low for a timed delay to allow the device voltage to reach the valid operating voltage range. Once the timer times out, the OST circuitry is enabled (for all crystal/resonator device oscillator modes).

Prescaler

A circuit that slows the rate of a clocking source to a counter/timer.

Program Bus

The bus which is used to transfer instruction words form the program memory to the CPU.

Program Counter

A register which specifies the address in program memory that is the next instruction to execute.

Program Memory

Any memory that is one the program memory bus. Static variables may be contained in program memory (such as tables).

PSP

See Parallel Slave Port.

Pulse Width Modulation (PWM)

A serial signal in which the information is contained in the width of a (high) pulse of a constant frequency signal. A PWM output, from the CCP module, of the same duty cycle requires no software overhead.

PUSH

A termed used to refer to the action of saving information onto a stack (software and/or hardware). See POP.

PWM

Pulse Width Modulation.

Q

Q-cycles

This is the same as a device oscillator cycle. There are 4 Q-cycles for each instruction cycle.

R

RC

Resistor-Capacitor. The default configuration for the device oscillator. This allows a "**R**eal-**C**heap" implementation for the device clock source. This clock source does not supply an accurate time-base. Operation to 4 MHz is supported. (See EXTRC).

Read-Modify-Write

This is where a register is read, then modified, and then written back to the original register. This may be done in one instruction cycle or multiple instruction cycles.

Register File

This is the Data Memory. Contains the SFRs and GPRs.

ROM

Read Only Memory. Memory that is fixed and cannot be modified.

S

Sampling Time

Sampling time is the complete time to get an A/D result. It includes the acquisition time and the conversion time.

Serial Peripheral Interface (SPI)

This is one of the modes of the SSP module. This is typically a 3-wire interface, with a data out line, a data in line, and a clock line. Since the clock is present, this is a synchronous interface.

SFR

Special Function Register. These registers contain the control bits and status information for the device.

Single cycle instruction

An instruction that executes in a “single” machine cycle (TCY).

Sleep

This is the low power mode of the device, where the device’s oscillator is disabled.This reduces the current the device consumes. Certain peripherals may be placed into modes where they continue to operate.

Special Function Registers (SFR)

These registers contain the control bits and status information for the device.

SPI

See Serial Peripheral Interface.

Stack

A portion of the CPU which retains the return address for program execution. The stack gets loaded with the value in the Program Counter when a CALL instruction is executed or an interrupt occurs.

T

TAD

In the A/D Converter, the time for a single bit of the analog voltage to be converted to a digital value.

TCY

The time for an instruction to complete. This time is equal to Fosc/4 and is divided into four Q-cycles.

Tosc

The time for the device oscillator to do a single period.

U

USART

Universal Synchronous Asynchronous Receiver Transmitter. This module can either operate as a full duplex asynchronous communications port, or a half duplex synchronous communications port. When operating in the asynchronous mode, this can be interfaced to a PC's serial port.

V

Voltage Reference (VREF)

A voltage level that can be used as a reference point for A/D conversions (AVDD and AVSS) or the trip point for comparators.

von Neumann Architecture

In this architecture the Program Memory and Data Memory are contained in the same area. This means that accesses to the program memory and data memory must occur sequentially, which affects the performance of the device.

W

W Register

See Working Register.

Watchdog Timer (WDT)

Used to increase the robustness of a design by recovering from software flows that were not expected in the design of the product or other system related issues. The Watchdog Timer causes a reset if it is not cleared prior to overflow. The clock source for a PICmicro is an on-chip RC oscillator which enhances system reliability.

WDT

Watchdog Timer.

Working Register (W)

Can also be thought of as the accumulator of the device. Also used as an operand in conjunction with the ALU during two operand instructions.

X

XT

One of the device oscillator modes. Used for operation from 100 kHz to 4 MHz.

35.1 Revision History

Revision A

This is the initial released revision of the Glossary.

WORLDWIDE SALES

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602-786-7200 Fax: 602-786-7277
Technical Support: 602 786-7627
Web: http://www.microchip.com

Atlanta
Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston
Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508-480-9990 Fax: 508-480-8575

Chicago
Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 972-991-7177 Fax: 972-991-8588

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

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

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 516-273-5305 Fax: 516-273-5335

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

Toronto
Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

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

India
Microchip Technology Inc.
India Liaison Office
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-0061 Fax: 91-80-229-0062

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Shanghai
Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hong Qiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700
Fax: 86 21-6275-5060

Singapore
Microchip Technology Taiwan
Singapore Branch
200 Middle Road
#07-02 Prime Centre
Singapore 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan, R.O.C
Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2-717-7175 Fax: 886-2-545-0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44-1189-21-5858 Fax: 44-1189-21-5835

France
Arizona Microchip Technology SARL
Zone Industrielle de la Bonde
2 Rue du Buisson aux Fraises
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany
Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 München, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-39-6899939 Fax: 39-39-6899883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa 222 Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

10/31/97

Printed in the USA, 12/97

Microchip Technology Incorporated • 2355 W. Chandler Blvd. • Chandler, AZ USA • 602-786-7200 • FAX: 602-899-9210