Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

44-PIN DEMO BOARD USER'S GUIDE

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

Printed on recycled paper.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona, Gresham, Oregon and Mountain View, California. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

44-PIN DEMO BOARD USER'S GUIDE

Table of Contents

Preface **1**
Introduction 1
Document Layout 1
Conventions Used in this Guide 2
Warranty Registration 3
Recommended Reading 3
The Microchip Web Site 4
Development Systems Customer Change Notification Service 4
Customer Support 5
Document Revision History 5

Chapter 1. 44-Pin Demo Board Overview
1.1 Introduction 7
1.2 Highlights 7
1.3 Devices Supported by the 44-Pin Demo Board 7
1.4 44-Pin Demo Board Overview 8
1.5 Running the Default Demonstration 8

Chapter 2. Mid-Range PIC® Microcontroller Architectural Overview
2.1 Introduction 9
2.2 Memory Organization 9
2.3 Instruction Formats 10
2.4 Assembler Basics 11

Chapter 3. 44-Pin Demo Board Lessons
3.1 Introduction 13
3.2 44-Pin Demo Board Lessons 13
3.2.1 Lesson 1: Hello World (Light a LED) 14
3.2.2 Blink (Delay Loop) 15
3.2.3 Lesson 3: Rotate (Move the LED) 17
3.2.4 Lesson 4: Analog-to-Digital 19
3.2.5 Lesson 5: Variable Speed Rotate 22
3.2.6 Lesson 6: Switch Debouncing 23
3.2.7 Lesson 7: Reversible Variable Speed Rotate 24
3.2.8 Lesson 8: Function Calls 26
3.2.9 Lesson 9: Timer0 26
3.2.10 Lesson 10: Interrupts 28
3.2.11 Lesson 11: Indirect Data Addressing 30
3.2.12 Lesson 12: Look-up Table (ROM Array) 32

Appendix A. Hardware Schematics
A.1 Introduction 35
Worldwide Sales and Service..38

Preface

NOTICE TO CUSTOMERS

All documentation becomes dated, and this manual is no exception. Microchip tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site (www.microchip.com) to obtain the latest documentation available.

Documents are identified with a "DS" number. This number is located on the bottom of each page, in front of the page number. The numbering convention for the DS number is "DSXXXXXA", where "XXXXX" is the document number and "A" is the revision level of the document.

For the most up-to-date information on development tools, see the MPLAB® IDE on-line help. Select the Help menu, and then Topics to open a list of available on-line help files.

INTRODUCTION

This chapter contains general information that will be useful to know before using the 44-Pin Demo Board. Items discussed in this chapter include:

- Document Layout
- Conventions Used in this Guide
- Warranty Registration
- Recommended Reading
- The Microchip Web Site
- Development Systems Customer Change Notification Service
- Customer Support
- Document Revision History

DOCUMENT LAYOUT

This document describes how to use the 44-Pin Demo Board as a development tool to emulate and debug firmware on a target board. The manual layout is as follows:

- **Chapter 1. "44-Pin Demo Board Overview"** – This chapter provides an overview of the 44-Pin Demo Board for Microchip's 44-pin Thin Quad Flatpack (TQFP) PIC® Microcontroller Units (MCU).
- **Chapter 2. "Mid-Range PIC® Microcontroller Architectural Overview"** – This chapter provides an overview of the mid-range PIC® microcontroller architecture.
- **Chapter 3. "44-Pin Demo Board Lessons"** – This chapter provides lessons that introduce mid-range PIC® MCU assembly instructions and cover basic 44-Pin Demo board features.
- **Appendix A. "Hardware Schematics"** – Illustrates the 44-Pin Demo Board hardware schematic diagram, PCB layout and Bill of Materials.

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

<table>
<tr><th>Description</th><th>Represents</th><th>Examples</th></tr>
<tr><td colspan="3">Arial font:</td></tr>
<tr><td rowspan="2">Italic characters</td><td>Referenced books</td><td>"MPLAB® IDE User's Guide"</td></tr>
<tr><td>Emphasized text</td><td>...is the only compiler...</td></tr>
<tr><td rowspan="3">Initial caps</td><td>A window</td><td>the Output window</td></tr>
<tr><td>A dialog</td><td>the Settings dialog</td></tr>
<tr><td>A menu selection</td><td>select Enable Programmer</td></tr>
<tr><td>Quotes</td><td>A field name in a window or dialog</td><td>"Save project before build"</td></tr>
<tr><td>Underlined, italic text with right angle bracket</td><td>A menu path</td><td><u>File>Save</u></td></tr>
<tr><td rowspan="2">Bold characters</td><td>A dialog button</td><td>Click OK</td></tr>
<tr><td>A tab</td><td>Click the Power tab</td></tr>
<tr><td>N'Rnnnn</td><td>A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.</td><td>4'b0010, 2'hF1</td></tr>
<tr><td>Text in angle brackets < ></td><td>A key on the keyboard</td><td>Press <Enter>, <F1></td></tr>
<tr><td colspan="3">Courier New font:</td></tr>
<tr><td rowspan="7">Plain Courier New</td><td>Sample source code</td><td><code>#define START</code></td></tr>
<tr><td>Filenames</td><td><code>autoexec.bat</code></td></tr>
<tr><td>File paths</td><td><code>c:\mcc18\h</code></td></tr>
<tr><td>Keywords</td><td><code>_asm, _endasm, static</code></td></tr>
<tr><td>Command-line options</td><td><code>-Opa+, -Opa-</code></td></tr>
<tr><td>Bit values</td><td><code>0, 1</code></td></tr>
<tr><td>Constants</td><td><code>0xFF, 'A'</code></td></tr>
<tr><td>Italic Courier New</td><td>A variable argument</td><td><code>file</code><code>.o</code>, where <code>file</code> can be any valid filename</td></tr>
<tr><td>Square brackets [ ]</td><td>Optional arguments</td><td><code>mcc18 [options] file [options]</code></td></tr>
<tr><td>Curly brackets and pipe character: { | }</td><td>Choice of mutually exclusive arguments; an OR selection</td><td><code>errorlevel {0|1}</code></td></tr>
<tr><td rowspan="2">Ellipses...</td><td>Replaces repeated text</td><td><code>var_name [, var_name...]</code></td></tr>
<tr><td>Represents code supplied by user</td><td><code>void main (void)
{ ...
}</code></td></tr>
</table>

WARRANTY REGISTRATION

Please complete the enclosed Warranty Registration Card and mail it promptly. Sending in the Warranty Registration Card entitles users to receive new product updates. Interim software releases are available at the Microchip web site.

RECOMMENDED READING

This user's guide describes how to use the 44-Pin Demo Board. Other useful documents are listed below. The following Microchip documents are available and recommended as supplemental reference resources.

PIC16F88X Data Sheet (DS41291)

Consult this document for information regarding the PIC16F88X 28/40/44-Pin Flash-Based, 8-Bit CMOS Microcontrollers with nanoWatt Technology device specification.

PICkit™ 2 Microcontroller Programmer User's Guide (DS51553)

Consult this document for instructions on how to use the PICkit 2 Microcontroller Programmer software and hardware.

MPLAB® ICD User's Guide (DS51184)

Consult this document for more information pertaining to the features and functions of the MPLAB In-Circuit Debugger (ICD) software.

MPLAB® IDE User's Guide (DS51519)

Consult this document for more information pertaining to the installation and features of the MPLAB Integrated Development Environment (IDE) Software.

Readme Files

For the latest information on using other tools, read the tool-specific Readme files in the Readmes subdirectory of the MPLAB IDE installation directory. The Readme files contain update information and known issues that may not be included in this user's guide.

THE MICROCHIP WEB SITE

Microchip provides online support via our web site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

DEVELOPMENT SYSTEMS CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com, click on Customer Change Notification and follow the registration instructions.

The Development Systems product group categories are:

- **Compilers** – The latest information on Microchip C compilers and other language tools. These include the MPLAB C18 and MPLAB C30 C compilers; MPASM™ and MPLAB ASM30 assemblers; MPLINK™ and MPLAB LINK30 object linkers; and MPLIB™ and MPLAB LIB30 object librarians.
- **Emulators** – The latest information on Microchip in-circuit emulators.This includes the MPLAB ICE 2000 and MPLAB ICE 4000.
- **In-Circuit Debuggers** – The latest information on the Microchip in-circuit debugger, MPLAB ICD 2.
- **MPLAB® IDE** – The latest information on Microchip MPLAB IDE, the Windows® Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE, MPLAB SIM simulator, MPLAB IDE Project Manager and general editing and debugging features.
- **Programmers** – The latest information on Microchip programmers. These include the MPLAB PM3 and PRO MATE® II device programmers and the PICSTART® Plus and PICkit™ 2 development programmers.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://support.microchip.com

DOCUMENT REVISION HISTORY

Revision A (August 2006)

- Initial release of this document.

Revision B (December 2006)

- Updated **Chapter 1. “PICkit™ 2 Overview”**.
- Added **Chapter 2. “Mid-Range PIC® Microcontroller Architectural Overview”**.
- Added **Chapter 3. “44-Pin Demo Board Lessons”**.
- Changed PICmicro® to PIC®.
- Changed PICkit® to PICkit™.
- Removed Development Systems Information Line from Customer Support bulleted list.
- Updated schematic in Appendix.

NOTES:

Chapter 1. 44-Pin Demo Board Overview

1.1 INTRODUCTION

The 44-Pin Demo Board is a small and simple demonstration PCB for Microchip's 44-pin Thin Quad Flatpack (TQFP) PIC® Microcontroller Units (MCU). It is populated with a PIC16F887 MCU, eight LEDs, push button and potentiometer. The demo board has several test points to access the I/O pins of the MCU and a surface mount prototyping area. The MCU can be programmed with the PICkit™ 2 Microcontroller Programmer or the MPLAB® ICD 2 using the RJ-11 to 6-pin inline adapter (AC164110).

1.2 HIGHLIGHTS

This chapter discusses:

- Devices supported by the 44-Pin Demo Board
- The 44-Pin Demo Board Overview
- Running the Default Demonstration

1.3 DEVICES SUPPORTED BY THE 44-PIN DEMO BOARD

The 44-Pin Demo Board can be used with virtually any 44-pin Thin Quad Flatpack (TQFP) PIC MCU. The assembled 44-Pin Demo Board is populated with a PIC16F887-I/PT microcontroller.

Additional 44-Pin Demo Boards can be ordered from Microchip Technology and distributors. Part number, DM164120-2, comes with one assembled and two blank 44-Pin Demo Boards. The blank demo board can be used for evaluating or prototyping circuits using any of the 44-pin devices listed below.

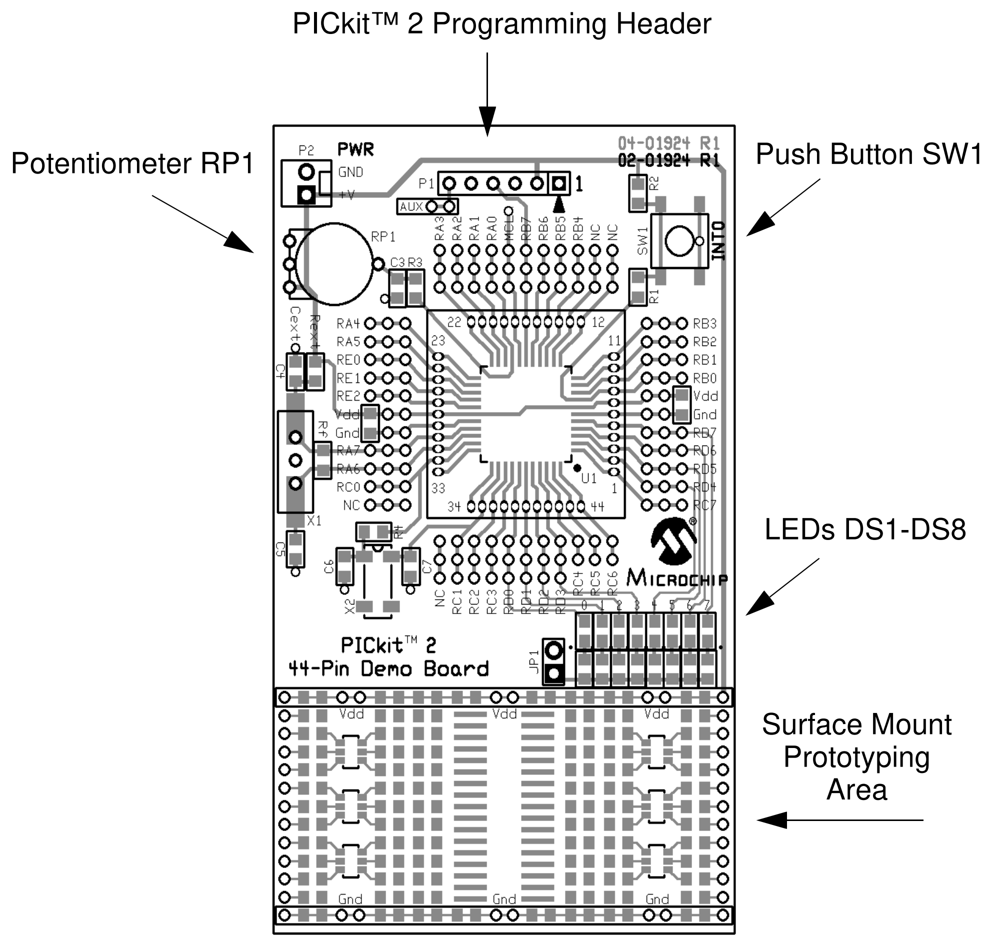
44-pin TQFP Flash Devices:

- PIC16F74
- PIC16F747
- PIC16F77
- PIC16F777
- PIC16F871
- PIC16F874A
- PIC16F877A
- PIC16F887
- PIC16F914
- PIC16F917
- PIC18F4220
- PIC18F4221
- PIC18F4320
- PIC18F4321
- PIC18F4331
- PIC18F4410
- PIC18F4420
- PIC18F4423
- PIC18F4431
- PIC18F4450
- PIC18F4455
- PIC18F4480
- PIC18F44J10
- PIC18F4510
- PIC18F4515
- PIC18F4520
- PIC18F4523
- PIC18F4525
- PIC18F4550
- PIC18F4580
- PIC18F4585
- PIC18F45J10
- PIC18F4610
- PIC18F4620
- PIC18F4680
- PIC18F4682
- PIC18F4685

1.4 44-PIN DEMO BOARD OVERVIEW

The 44-Pin Demo Board is populated with a PIC16F887 MCU (U1), eight LEDs (DS1-DS8), push button (SW1) and potentiometer (RP1). The board layout is shown in Figure 1-1. The demo board has several test points to access the I/O pins of the MCU and a surface mount prototyping area. The MCU can be programmed with the PICkit™ 2 Microcontroller Programmer from header P1.

FIGURE 1-1: 44-PIN DEMO BOARD



1.5 RUNNING THE DEFAULT DEMONSTRATION

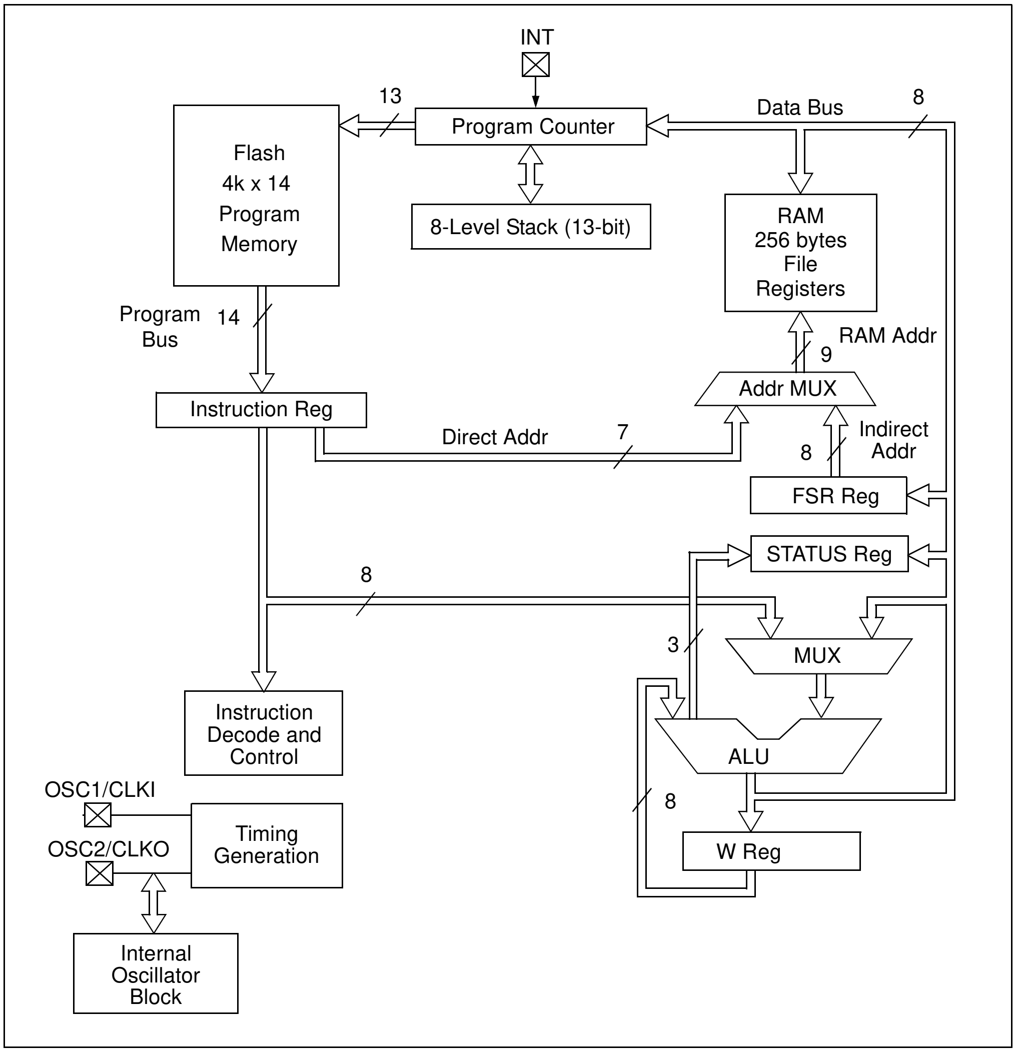
The assembled 44-Pin Demo Board comes preprogrammed with a demonstration program. To use this program, power the demo board (3.0-5.5 VDC) using a PICkit™ 2 Microcontroller Programmer, or a bench power supply connected to header P2. To use the PICkit™ 2 Microcontroller Programmer, connect it to a PC USB port using the USB cable. Start the PICkit™ 2 Microcontroller Programmer PC application and click on the target power box to apply power to the demo board. The demo program will blink the eight red lights in succession. Press the push button switch, labeled SW1, and the sequence of the lights will reverse. Rotate the potentiometer, RP1, and the light sequence will blink at a different rate.

Chapter 2. Mid-Range PIC® Microcontroller Architectural Overview

2.1 INTRODUCTION

This chapter provides a simple overview of the mid-range PIC® microcontroller architecture.

FIGURE 2-1: SIMPLIFIED MID-RANGE PIC® MICROCONTROLLER BLOCK DIAGRAM



2.2 MEMORY ORGANIZATION

PIC® microcontrollers are designed with separate program and data memory areas. This allows faster execution as the address and data busses are separate and do not have to do double duty.

Data Memory is held in **file registers**. Instructions referring to file registers use 7 bits, so only 128 file registers can be addressed. Multiple file registers are arranged into "banks". Two bits in the STATUS register, RP0 and RP1, allow accessing different banks. These two bits effectively become the top two bits of the file register address.

The additional banks may or may not be implemented, depending on the device.

Mid-range devices reserve the first 32 addresses of each bank for **Special Function Registers** (SFRs). SFRs are how the program interacts with the peripherals and some core features. The controls and data registers are memory mapped into the SFR space. Addresses above 0x20 to the end of each bank are **General Purpose Registers** (GPRs), where program variables may be stored.

Some frequently used registers may be accessed from any bank. For example, the STATUS register is always available no matter which bank is selected via the RP bits. The last 16 bytes (0x70-0x7F) of the GPRs may also be accessed from any bank.

Program Memory is accessed via a 13-bit Program Counter (PC). The lower 8 bits are accessible via SFR (PCL), and the upper 5 are at a PCLATH. See the PIC16F88X Data Sheet's (DS41291) section on PCL and PCLATH for more details on the PC. PCLATH becomes important when program memory size exceeds 1k instructions, and also for the table look-up in Lesson 12.

Mid-range PIC® MCUs may be clocked by a number of different devices. Unless otherwise noted, the lessons in this manual use the Internal Oscillator running at 4 MHz.

2.3 INSTRUCTION FORMATS

Most instructions follow one of three formats: Byte oriented instructions, Bit oriented instructions and Literal instructions.

Byte instructions contain a 7-bit data address, a destination bit, and a 6-bit op code. The data address plus the RP0 and RP1 bits create a 9-bit data memory address for one operand. The other operand is the Working register (called W or WREG). After the instruction executes, the destination bit (d) specifies whether the result will be stored in the `WREG` ('w') or back in the original file register ('f'). For example:

```
ADDWF   data,f
```

adds the contents of `WREG` and file register `data`, with the result going back into `data`.

Bit instructions operate on a specific bit within a file register. They contain 7 bits of data address, a 3-bit number and the remaining 4 bits are op code. These instructions may set or clear a specific bit within a file register. They may also be used to test a specific bit within a file register. For example:

```
BSF      STATUS,RP0
```

set the RP0 bit in the STATUS register.

Literal instructions contain the data operand within the instruction. The WREG becomes the other operand. `Calls` and `GOTO`'s use 11 bits as a literal address.

```
MOVLW   'A'
```

Moves the ASCII value of 'A' (0x41) into the WREG.

2.4 ASSEMBLER BASICS

Numbers in the Assembler

Unless otherwise specified, the assembler assumes any numeric constants in the program are hexadecimal (base 16). Binary (base 2), Octal (base 8), Decimal (base 10), and ASCII coding are also supported.

Hexadecimal	`12 or 0x12 or H'12'`
Decimal	`.12 or D'12'`
Octal	`O'12'`
Binary	`B'00010010'`
ASCII	`A'c' or 'c'`

`Org` (Origin)

`Org` tells the Assembler an address at which to start generating code. Normally we start coding at the Reset vector address '`0000`', but it could be anywhere. Baseline devices have a Reset vector at the last location in program memory, so it's good practice to have a `GOTO` instruction pointing to the beginning of the program.

`End`

`End` tells the assembler to stop assembling. There must be one at the end of the program. It does not necessarily have to be at the end of the file, but nothing after the end statement will be assembled.

Defining Data Memory Locations

There are three ways to name a location (see Example 2-1). All are equivalent in that the location name label will be substituted with the value assigned to it during assembly.

EXAMPLE 2-1: DEFINING DATA MEMORY

```
#define Length   0x20      ;c-like syntax

Length   equ     0x20      ;equate 0x20 with the symbol

   cblock        0x20      ;start a block of variables
Length                     ;this will be at address 0x20
Width                      ;this will be at address 0x21
Area:2                     ;this is 2 bytes long, starting at
                           ;address 0x22
Girth                      ;this will be at address 0x24
   endc
```

Note that if used as a literal, the label names will take on the value assigned. If used as an address operand in an instruction, the label names point to the contents of the file register with the address of the label's value.

Unless there is a reason to name a specific location address, the `cblock/endc` method is preferred. The advantage is that as variables come and go through the development process, the `cblock` keeps the block to a minimum. Using one of the other methods, you may have to go back and find an unused location.

NOTES:

Chapter 3. 44-Pin Demo Board Lessons

3.1 INTRODUCTION

The following lessons cover basic 44-Pin Demo Board features. Refer to applicable documents as needed. Any updates to the applicable documents are available on Microchip's web site.

The code and hex files may be installed from the PICkit™ 2 CD-ROM under path `Install /Lessons.`

3.2 44-PIN DEMO BOARD LESSONS

- Lesson 1: Hello World (Light a LED)
- Lesson 2: Blink (Delay Loop)
- Lesson 3: Rotate (Move the LED)
- Lesson 4: Analog-to-Digital
- Lesson 5: Variable Speed Rotate
- Lesson 6: Switch Debounce
- Lesson 7: Reversible Variable Speed Rotate
- Lesson 8: Function Calls
- Lesson 9: Timer0
- Lesson 10: Interrupts
- Lesson 11: Indirect Data Addressing
- Lesson 12: Look-up Table (ROM Array)

3.2.1 Lesson 1: Hello World (Light a LED)

The first lesson shows how to turn on a LED. This is the PIC® microcontroller version of "Hello World" and discusses the I/O pin structures.

New Instructions

BSF	Bit set
BCF	Bit clear

The LEDs are connected to I/O pins RD0 through RD7. When one of these I/O pins drives high, the LED turns on. The I/O pins can be configured for input or output. On start-up, the default is input. The TRIS Special Function Register bits use the convention of '0' for output and '1' for input. We want digital output so these must be configured.

EXAMPLE 3-1: PICKIT 2, LESSON 1: "HELLO WORLD"

```
; PICkit 2 Lesson 1 - "Hello World"
;
#include <p16F887.inc>
       __CONFIG   _CONFIG1, _LVP_OFF & _FCMEN_OFF & _IESO_OFF &
                  _BOR_OFF & _CPD_OFF & _CP_OFF & _MCLRE_OFF &
                  _PWRTE_ON & _WDT_OFF & _INTRC_OSC_NOCLKOUT
      __CONFIG    _CONFIG2, _WRT_OFF & _BOR21V

     org 0
Start:
     BSF     STATUS,RP0  ; select Register Bank 1
     BCF     TRISD,0     ; make IO Pin RD0 an output
     BCF     STATUS,RP0  ; back to Register Bank 0
     BSF     PORTD,0     ; turn on LED RD0 (DS0)
     GOTO     $          ; wait here
     END
```

Now lets look at the program that makes this happen.

;	Starts a comment. Any text on the line following the semicolon is ignored.
#include	Brings in an include file defining all the Special Function Registers available on the PIC16F887. Also, it defines valid memory areas. These definitions match the names used in the device data sheet.
__Config	Defines the Configuration Word. The labels are defined in the p16F887.inc file. The labels may be logically ANDed together to form the word.
Org 0	Tells the assembler where to start generating code. Code may be generated for any area of the part. Mid-range PIC® microcontroller devices start at address '0', also called the Reset vector.
BCF TRISC,0	Tells the processor to clear a bit in a file register. TRISD is the tri-state register for pin 0 of PORTD. A '1' in the register makes the pin an input; a '0' makes it an output. We want to make it an output, so the bit must be cleared.
BSF PORTD,0	Tells the processor to set pin 0 of PORTD. This will force the I/O pin to a high condition turning on the LED.
GOTO $	Tells the processor to go to the current instruction.

For more information, refer to the I/O Ports section of the PIC16F882/883/884/886/887 Data Sheet (DS41291).

3.2.2 Blink (Delay Loop)

The first lesson showed how to turn on a LED, this lesson shows how to make it blink. While this might seem a trivial change from Lesson 1, it gives a context to explore several more instructions.

New Instructions

`CLRF`	Clear file register
`INCF`	Increment file register
`DECF`	Decrement file register
`INCFSZ`	Increment file register, Skip next instruction if zero
`DECFSZ`	Decrement file register, Skip next instruction if zero
`GOTO`	Jump to a new location in the program

EXAMPLE 3-2: PICKIT 2, LESSON 2: BLINK

```
Loop
   BSF   PORTD,0      ;turn on LED D0
   BCF   PORTD,0      ;turn off LED D0
   GOTO  Loop         ;do it again
```

While adding a `BCF` instruction and making it loop will make it blink. It will blink so fast you won't see it, it will only look dim. That loop requires 4 instruction times to execute. The first instruction turns it on. The second one turns it off. The `GOTO` takes two instruction times, which means it will be on for 25% of the time.

As configured, the PIC® microcontroller executes 1 million instructions per second. At this rate, the blinking needs to be slowed down so that the blinking can be seen, which can be done by using a delay loop.

> **Note:** Counting cycles – Relating clock speed to instruction speed. The processor requires 4 clocks to execute an instruction. Since the internal oscillator as used in these lessons runs at 4 MHz, the instruction rate is 1 MHz.

Increment or Decrement a File Register

The INCFSZ and DECFSZ instructions add or subtract one from the contents of the file register and skips the next instruction when the result is zero. One use is in the delay loop as shown in Example 3-3.

CLRF Clears the counter location.

DECFSZ Decrements the location, and if the result is zero, the next instruction is skipped.

EXAMPLE 3-3: DELAY LOOP

```
Short Loop

   CLRF      Delay
Loop
   DECFSZ    Delay,f
   GOTO      Loop

Long Loop

   CLRF      Delay1
   CLRF      Delay2
Loop
   DECFSZ    Delay1,f
   GOTO      Loop
   DECFSZ    Delay2,f
   GOTO      Loop
```

The GOTO Loop (in Example 3-3) backs up and does it again. This loop takes 3 instruction times; one for the decrement and two for the GOTO (see note) and the counter will force it to go around 256 times, which takes it a total of 768 instruction times (768 μs) to execute.

Even that is still too fast for the eye to see. It can be slowed down even more by adding a second loop around this one.

The inner loop still takes 768 μs plus 3 for the outer loop, but now it's executed another 256 times, (768 + 3) * 256 = 197376 μs = 0.197s.

> **Note:** GOTO instructions take two instructions due to the pipelined design of the processor. The processor fetches the next instruction while executing the current instruction. When a program branch occurs, the fetched instruction is not executed.

Open Blink.asm and build the lesson. Next, import the hex file into the PICkit 2 and program the device. Note the LED now flashes at about a 2.5 Hz rate.

3.2.3 Lesson 3: Rotate (Move the LED)

Building on Lessons 1 and 2, which showed how to light up a LED and then make it blink with a delay loop, this lesson adds rotation. It will light up DS8 and then shift it to DS7, then DS6 and on down to DS1, and then back to DS8.

New Instructions

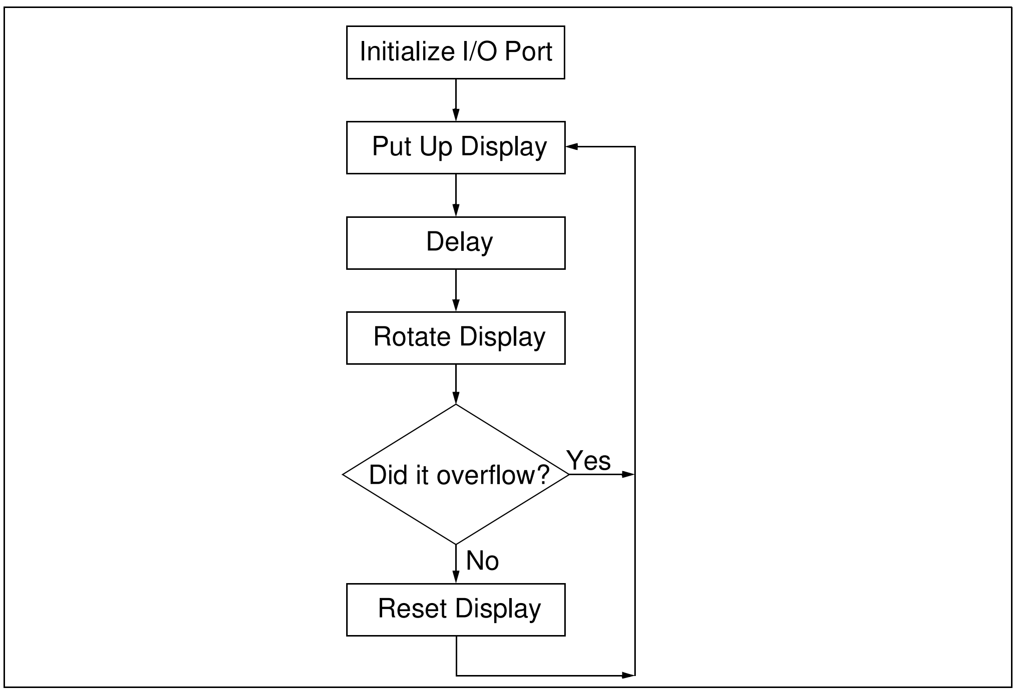
`MOVLW`	Loads WREG with a literal value
`MOVWF`	Moves the contents of WREG to a file register
`MOVF`	Moves the contents of a file register, either to WREG or back into the file register (see note)
`RRF`	Rotate file register right
`RLF`	Rotate file register left

> **Note:** Moving a file register to itself looks like a `NOP` at first. However, it has a useful side effect in that the Z flag is set to reflect the value. In other words, `MOVF fileregister,f` is a convenient way to test whether or not the value is zero without affecting the contents of the WREG.

Rotate Program Flow

- First, initialize the I/O port and the Display,
- Copy the Display variable to the I/O Port, then
- Delay for a little while
- Rotate the display

FIGURE 3-1: ROTATE PROGRAM FLOW

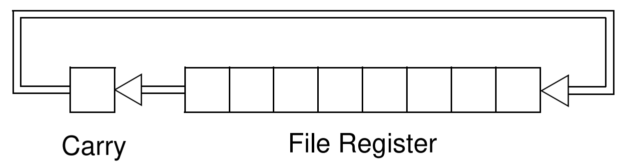


Rotate

The rotate instructions (`RRF` or `RLF`) shift all the bits in the file register right or left by one position, through the Carry bit. The Carry bit is shifted into the byte and receives the bit shifted out of the byte. The Carry bit should be cleared before rotation so unwanted bits are not introduced into the display byte. The Carry bit also indicates when the display byte is empty. When it is, reinsert the '`1`' at bit 3.

PIC microcontrollers have two rotate instructions: Rotate Left (`RLF`) and Rotate Right (`RRF`). These instructions rotate the contents of a file register and Carry bit one place. The Carry bit is found in the STATUS Special Function Register.

FIGURE 3-2: ROTATE LEFT



EXAMPLE 3-4: ROTATE EXAMPLE

```
Start:
  BSF        STATUS,RP0  ; select Register Bank 1
  CLRF       TRISD       ; make IO PortD all output
  BCF        STATUS,RP0  ; back to Register Bank 0
  MOVLW      0x80
  MOVWF      Display
MainLoop:
  MOVF       Display,w   ; Copy the display to the LEDs
  MOVWF      PORTD
OndelayLoop:
  DECFSZ     Delay1,f    ; Delay .197 s
  GOTO       OndelayLoop
  DECFSZ     Delay2,f
  GOTO       OndelayLoop

  BCF        STATUS,C    ; ensure the carry bit is clear
  RRF        Display,f   ; rotate display right
  BTFSC      STATUS,C    ; Did the bit rotate into the carry?
  BSF        Display,7   ; yes, put it into bit 7.
  GOTO       MainLoop
```

3.2.4 Lesson 4: Analog-to-Digital

This lesson shows how to configure the ADC, run a conversion, read the analog voltage controlled by the potentiometer (RP1) on the board, and display the high order 8 bits on the display.

The PIC16F887 has an on-board Analog-to-Digital Converter (ADC) with 10 bits of resolution on any of 14 channels. The converter can be referenced to the device's V_{DD} or an external voltage reference. The 44-pin Demo Board references it to V_{DD} as provided by the PICkit 2 Microcontroller Programmer. The answer from the ADC is represented by a ratio of the voltage to the reference.

$$ADC = V/V_{REF} * 1023$$

Converting the answer from the ADC back to voltage requires solving for V.

$$V = ADC/1023 * V_{REF}$$

Two of the three factors on the right side of the equation are constants and may be calculated in advance. This eliminates the need to actually divide, but still requires fixed or floating point multiply to solve the equation on the fly.

However, sometimes, such as when reading a sensor, calculating the voltage is only the first step. There may be additional math to calculate the meaningful data from the sensor. For example, when reading a thermistor, calculating the voltage is only the first step on the way to getting the temperature.

There are other means to convert ADC values, including a straight table look-up or a piece-wise linear interpolation. Each of these represents different speed/memory trade-offs.

The schematic (**Appendix A. "Hardware Schematics"**) shows the wiper on the potentiometer is connected to pin RA0 on the PIC16F887.

Here's the checklist for this lesson:

- Configure PORTA as an analog input, TRISA<0> = `1`, ANSEL<0> = `1`
- Select justification and V_{REF} source in ADCON1.
- Select clock scaling and channel in ADCON0.

3.2.4.1 ADCON1

The ADCON1 register sets the justification of the 10-bit result in the 16-bit result read through registers ADRESL and ADRESH. Setting the result to Left Justified means the 8 Most Significant bits and read from ADRESH and the 2 Least Significant bits are read from bits 7 and 6 of ADRESL. ADCON1 also sets the voltage reference sources VREF+ and VREF-. VREF- is the voltage at which the result will be zero. VREF+ is the voltage at which the result will be maximum (1023). We select the PIC16F887 VSS and VDD voltages respectively.

REGISTER 3-1: ADCON1: A/D CONTROL REGISTER 1

R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
ADFM	—	VCFG1	VCFG0	—	—	—	—
bit 7							bit 0

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

bit 7	**ADFM:** A/D Conversion Result Format Selection bit 1 = Right justified 0 = Left justified
bit 6	**Unimplemented:** Read as ‘0’
bit 5	**VCFG1:** Voltage Reference bit 1 = VREF- pin 0 = Vss
bit 4	**VCFG0:** Voltage Reference bit 1 = VREF+ pin 0 = VDD
bit 3-0	**Unimplemented:** Read as ‘0’

3.2.4.2 ADCON0

ADCON0 controls the ADC operation. Bit 0 turns on the ADC module and bit 1 starts a conversion. Bits <7:6> select the ratio between the processor clock and conversion speed and bits <5:2> select which channel the ADC will operate on. The ratio between the processor clock and conversion speed is important because the ADC needs at least 1.6 µs per bit. Accuracy degrades if the clock speed is too high. As the processor clock speed increases, an increasingly large divider is necessary to keep the conversion bit speed above 1.6 µs. Four MHz gives the fastest conversion rate above the minimum at 8:1 ratio. This results in a conversion speed of 2 µs per bit. Refer to the “TAD vs. Device Operating Frequencies” Table in the Analog-to-Digital section of the PIC16F882/883/884/886/887 Data Sheet (DS41291) for recommended configurations.

For purposes of this lesson, the ADC must be turned on and pointed to channel AN0 on pin RA0.

The ADC needs about 5 µs, after changing channels, to allow the ADC sampling capacitor to settle. Finally, we can start the conversion by setting the GO bit in ADCON0. The bit also serves as the $\overline{\text{DONE}}$ flag. That is, the ADC will clear the same bit when the conversion is complete. The answer is then available in ADRESH:ADRESL. This lesson takes the high order 8 bits of the result and copies them to the display LEDs attached to PORTD.

See the Analog-to-Digital section in the PIC16F882/883/884/886/887 Data Sheet (DS41291) for more details on the ADC module.

REGISTER 3-2: ADCON0: A/D CONTROL REGISTER 0

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	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

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

bit 7-6 **ADCS<1:0>:** A/D Conversion Clock Select bits
00 = FOSC/2
01 = FOSC/8
10 = FOSC/32
11 = FRC (clock derived from a dedicated internal oscillator = 500 kHz max)

bit 5-2 **CHS<3:0>: Analog Channel Select bits**
0000 = AN0
0001 = AN1
0010 = AN2
0011 = AN3
0100 = AN4
0101 = AN5
0110 = AN6
0111 = AN7
1000 = AN8
1001 = AN9
1010 = AN10
1011 = AN11
1100 = AN12
1101 = AN13
1110 = CVREF
1111 = Fixed Ref (0.6 volt fixed reference)

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
1 = A/D conversion cycle in progress. Setting this bit starts an A/D conversion cycle. This bit is automatically cleared by hardware when the A/D conversion has completed.
0 = A/D conversion completed/not in progress

bit 0 **ADON:** ADC Enable bit
1 = ADC is enabled
0 = ADC is disabled and consumes no operating current