



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





Low Pin Count 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.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, PowerSmart, rfPIC, and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


AmpLab, FilterLab, Migratable Memory, MXDEV, MXLAB, PICMASTER, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Linear Active Thermistor, MPASM, MPLIB, MPLINK, MPSIM, PICKit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, rLAB, rfPICDEM, Select Mode, Smart Serial, SmartTel, Total Endurance and WiperLock are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

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

© 2005, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

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

Microchip received ISO/TS-16949:2002 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona and Mountain View, California in October 2003. The Company's quality system processes and procedures are for its PICmicro® 8-bit MCUs, 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.

Table of Contents

Preface 1

Chapter 1. Low Pin Count (LPC) Demo Board Overview

 1.1 Introduction 7

 1.2 Highlights 7

 1.3 Devices Supported by the LPC Demo Board 7

 1.4 LPC Demo Board Overview 8

 1.5 Running the PICkit™ 2 Flash Starter Kit Default Demonstration 8

Chapter 2. Mid-Range PICmicro® Architectural Overview

 2.1 Introduction 9

 2.2 Memory Organization 10

 2.3 Instruction formats 10

 2.3.1 Assembler Basics 11

Chapter 3. LPC Demo Board Lessons

 3.1 Introduction 13

 3.2 LPC Demo Board lessons 13

 3.2.1 Lesson 1: Hello World (Light a LED) 14

 3.2.2 Lesson 2: Delay Loop (Blink a LED) 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 25

 3.2.8 Lesson 8: Function Calls 26

 3.2.9 Lesson 9: Timer0 27

 3.2.10 Lesson 10: Interrupts 29

 3.2.11 Lesson 11: Indirect Data Addressing 31

 3.2.12 Lesson 12: Look-up Table (ROM Array) 33

Appendix A. Hardware Schematics

 A.1 Introduction 37

Worldwide Sales and Service 38

Low Pin Count Demo Board User's Guide

NOTES:

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 “DSXXXXA”, where “XXXX” 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 Low Pin Count (LPC) Demo Board. Items discussed in this chapter include:

- About this Guide
- Warranty Registration
- Recommended Reading
- Troubleshooting
- The Microchip Web Site
- Development Systems Customer Notification Service
- Customer Support

DOCUMENT LAYOUT

This document describes how to use the Low Pin Count Demo Board User's Guide as a development tool to emulate and debug firmware on a target board. The manual layout is as follows:

- **Chapter 1. “Low Pin Count (LPC) Demo Board Overview”** – An overview of Microchip's Low Pin Count Demo Board.
- **Chapter 2. “Mid-Range PICmicro® Architectural Overview”** – An overview of the Mid-range PICmicro® Architecture.
- **Chapter 3. “LPC Demo Board Lessons”** – Contains a variety of lessons that demonstrate how to utilize and experiment with the Low Pin Count Demo Board.

Low Pin Count Demo Board User's Guide

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

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

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 Low Pin Count (LPC) Demo Board. Other useful documents are listed below. The following Microchip documents are available and recommended as supplemental reference resources.

Readme for Low Pin Count (LPC) Demo Board

For the latest information on using the Low Pin Count (LPC) Demo Board, read the "Readme for Low Pin Count Demo Board.txt" file (an ASCII text file) in the PICkit 2 installation directory. The Readme file contains update information and known issues that may not be included in this user's guide.

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.

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

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

PIC16F685/687/689/690 Data Sheet (DS41262)

Consult this document for information regarding the PIC16F685/687/689/690 20-pin Flash based, 8-bit CMOS Microcontroller device specifications.

MPLAB® IDE, Simulator, Editor User's Guide (DS51025)

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

Low Pin Count Demo Board 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® 1 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
- Development Systems Information Line

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>

In addition, there is a Development Systems Information Line which lists the latest versions of Microchip's development systems software products. This line also provides information on how customers can receive currently available upgrade kits.

The Development Systems Information Line numbers are:

1-800-755-2345 – United States and most of Canada

1-480-792-7302 – Other International Locations

DOCUMENT REVISION HISTORY

Revision A (May 2005)

- Initial Release of this Document.

Low Pin Count Demo Board User's Guide

NOTES:



LOW PIN COUNT DEMO BOARD USER'S GUIDE

Chapter 1. Low Pin Count (LPC) Demo Board Overview

1.1 INTRODUCTION

This chapter introduces the Low Pin Count (LPC) Demo Board and describes the LPC Demo Board features.

1.2 HIGHLIGHTS

This chapter discusses:

- Devices supported by the LPC Demo Board
- The LPC Demo Board Overview
- Running the PICkit™ 2 Starter Kit Default Demonstration

1.3 DEVICES SUPPORTED BY THE LPC DEMO BOARD

For a list of supported devices, see the LPC Demo Board README file on the PICkit™ 2 Starter Kit CD-ROM.

8-pin DIP Flash Devices:

PIC12F508	PIC12F629	PIC12F635
PIC12F509	PIC12F675	PIC12F683
PIC12F510		

14-pin DIP Flash Devices:

PIC16F505	PIC16F630	PIC16F684
PIC16F506	PIC16F676	PIC16F688

20-pin DIP Flash Devices:

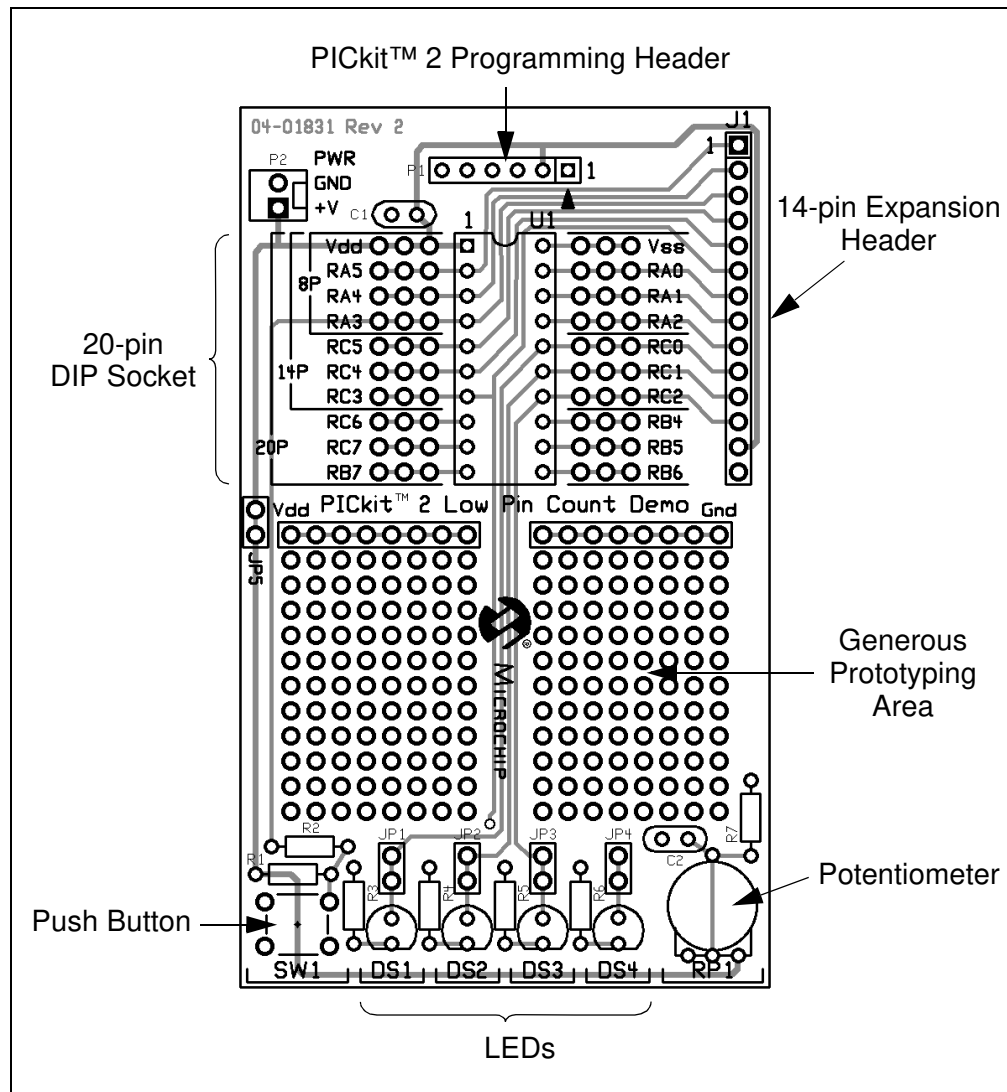
PIC16F685	PIC16F689	PIC16F785
PIC16F687	PIC16F690	

Low Pin Count Demo Board User's Guide

1.4 LPC DEMO BOARD OVERVIEW

The Low Pin Count Demo Board Works with the PICKit™ 2 Microcontroller Programmer to help the user get up to speed quickly using PICmicro® microcontrollers. This user's guide is written in the form of Lessons intended for a person with some exposure to assembly language but has never used a PICmicro® microcontroller. The LPC Demo Board overview is shown in Figure 1-1.

FIGURE 1-1: LPC DEMO BOARD



1.5 RUNNING THE PICKit™ 2 STARTER KIT DEFAULT DEMONSTRATION

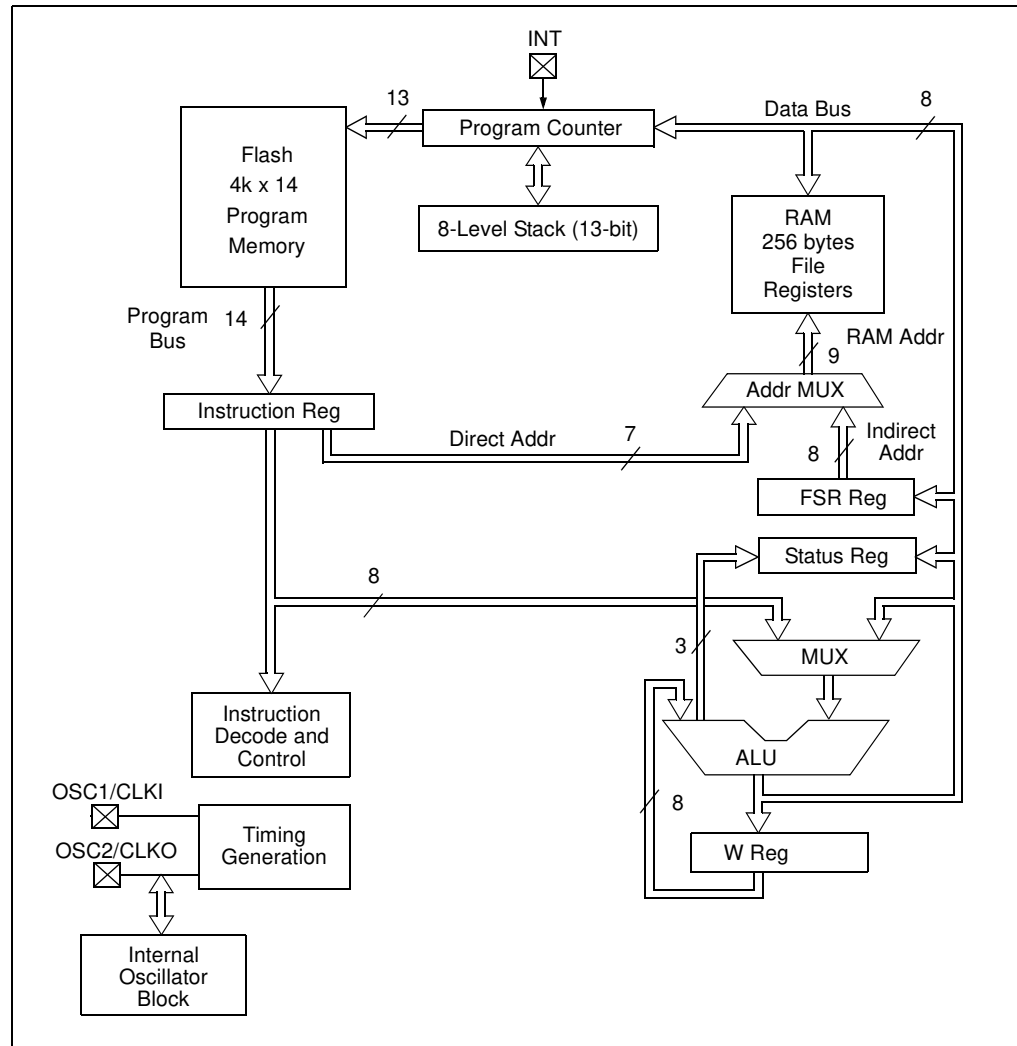
The Low Pin Count Demo Board comes preprogrammed with a demonstration program. To use this program, connect the PICKit™ 2 Starter Kit to the PC's USB port using the USB cable. Start the PICKit™ 2 Microcontroller Programmer application and check the target power box. The demo program will blink the four red lights in succession. Press the Push Button Switch, labeled SW1, and the sequence of the lights will reverse. Rotate the potentiometer, labeled RP1, and the light sequence will blink at a different rate. This demo program is developed through the first 7 lessons in this guide.

Chapter 2. Mid-Range PICmicro® Architectural Overview

2.1 INTRODUCTION

This chapter describes the Mid-range PICmicro® Architectural Overview for the LPC Demo Board.

FIGURE 2-1: SIMPLIFIED MID-RANGE PICmicro® BLOCK DIAGRAM



Low Pin Count Demo Board User's Guide

2.2 MEMORY ORGANIZATION

PICmicro[®] 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 "pages". Two extra bits RP0 and RP1 (in the Status register) allow accessing multiple pages. These two bits effectively become the top two bits of the file register address. The additional pages may or may not be implemented, depending on the device.

Mid-range devices reserve the first 32 addresses of each page for **Special Function Registers** (SFRs). SFRs are how the program interacts with the peripherals. The controls and data registers are memory mapped into the SFR space. Addresses above 0x20 to the end of each page 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) 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 PIC16F685/687/689/690 Data Sheet's (DS41262) 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 PICmicro[®] 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 7-bit data address, a destination bit, and 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 W or back in the original file register. For example:

```
ADDWF data, f
```

adds the contents of Wreg and 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, 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 Wreg.

Mid-Range PICmicro[®] Architectural Overview

2.3.1 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 where to start generating code. Normally we start coding at 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).

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
```

Unless there is a reason to want a name to a specific location, 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.

Low Pin Count Demo Board User's Guide

NOTES:

Chapter 3. LPC Demo Board Lessons

3.1 INTRODUCTION

The following lessons cover basic LPC 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 are installed in `C:\Microchip\PICKit 2 Lessons\`. They may also be found on the PICKit™ 2 CD-ROM under directory `\PICKit 2 Lessons\`.

3.2 LPC DEMO BOARD LESSONS

- Lesson 1: Hello World (Light a LED)
- Lesson 2: Delay Loop (Blink a LED)
- 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)

Low Pin Count Demo Board User's Guide

3.2.1 Lesson 1: Hello World (Light a LED)

The first lesson shows how to turn on a LED. This is the PICmicro[®] 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 RC0 through RC3. When one of these I/O pins drive 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 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 <p16F690.inc>
__config (_INTRC_OSC_NOCLKOUT & _WDT_OFF & _PWRTE_OFF &
_MCLRE_OFF & _CP_OFF & _BOD_OFF & _IESO_OFF & _FCMEN_OFF)
org 0
Start
BSF STATUS,RP0 ;select Register Page 1
BCF TRISC,0 ;make I/O Pin C0 an output
BCF STATUS,RP0 ;back to Register Page 0
BSF PORTC,0 ;turn on LED C0
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 PIC16F690. 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 p16F690.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 PICmicro [®] 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. TRISC is the Tri-state register for pin 0 of PORTC. 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 PORTC,0	Tells the processor to set pin 0 of PORTC. 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 PIC16F685/687/689/690 Data Sheet (DS41262).

3.2.2 Lesson 2: Delay Loop (Blink a LED)

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, the reasons will soon become apparent.

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  PORTC,0      ;turn on LED C0
  BCF  PORTC,0      ;turn off LED C0
  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 PICmicro 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.

Low Pin Count Demo Board User's Guide

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.

<code>CLRF</code>	Clears the counter location.
<code>DECFSZ</code>	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 μ s) 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 μ s plus 3 for the outer loop, but now it's executed another 256 times, $771 * 256 = 197376 \mu$ s = 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 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 DS4 and then shift it to DS3, then DS2, then DS1 and back to DS4.

New Instructions

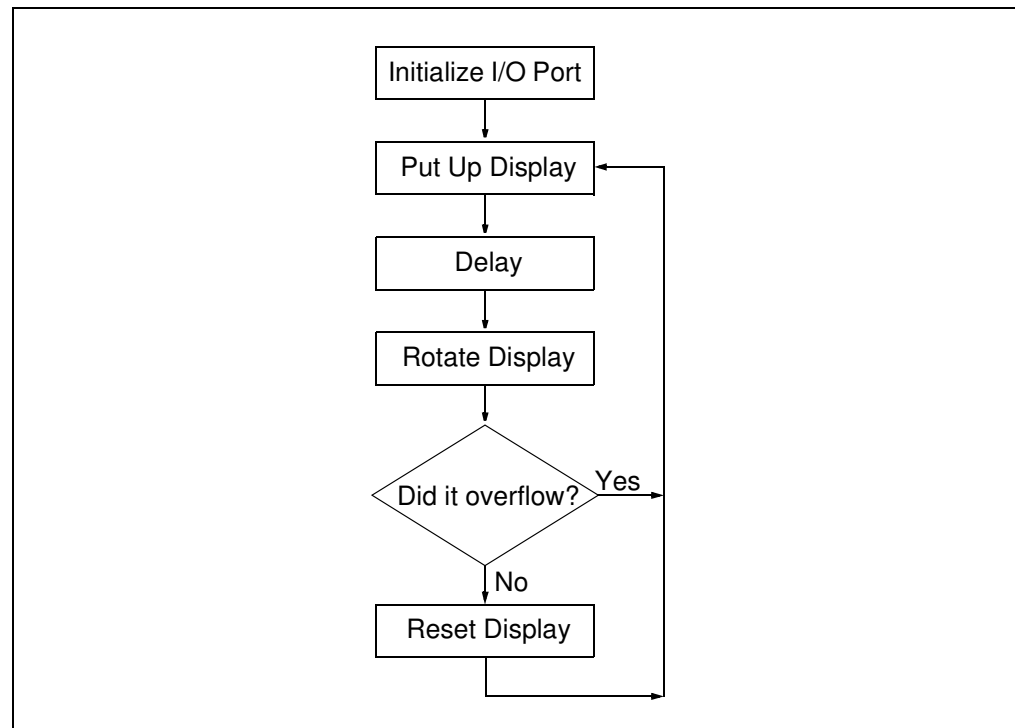
MOVLW	Loads Wreg with a literal value
MOVWF	Moves the contents of Wreg to a file register
MOVF	Moves the contents of a file register, either to Wreg or back into the file register (see note)
RRF	Rotate file register right
RLF	Rotate file register left

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



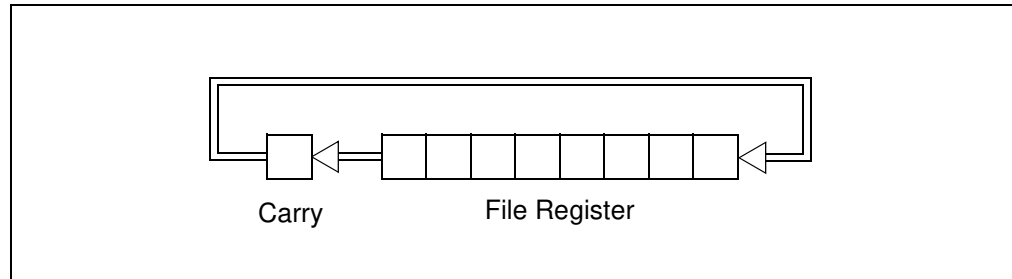
Low Pin Count Demo Board User's Guide

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.

PICmicro MCUs 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.

FIGURE 3-2: ROTATE LEFT



EXAMPLE 3-4: ROTATE EXAMPLE

```
Start
  BSF    STATUS,RP0    ;select Register Page 1
  CLRF   TRISC         ;make I/O PORTC all output
  BCF    STATUS,RP0    ;back to Register Page 0
  MOVLW  0x08
  MOVWF  Display

MainLoop
  MOVF   Display,w     ;Copy the display to the LEDs
  MOVWF  PORTC

OndelayLoop                                ;Delay .197S
  DECFSZ Delay1,f
  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,3     ;yes, put it into bit 3.
  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 4 bits on the display.

The PIC16F690 has an on board Analog-to-Digital Converter (ADC) with 10 bits of resolution on any of 11 channels. The converter can be referenced to the device's VDD or an external voltage reference. The LPC Demo Board references it to VDD as provided by the USB cable. The answer from the ADC is represented by a ratio of the voltage to the reference.

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

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

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

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 PIC16F690.

Here's the checklist for this lesson:

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

Low Pin Count Demo Board User's Guide

3.2.4.1 ADCON1

ADCON1 selects the ratio between processor clock speed and conversion speed. This is important because the ADC needs at least 1.6 μ s 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 speed. Four MHz is fastest at 8:1 ratio with a conversion speed of 2 μ s per bit. Refer to the "TAD vs. Device Operating Frequencies" Table in the Analog-to-Digital Section of the PIC16F685/687/689/690 Data Sheet (DS41262) for recommended configurations.

REGISTER 3-1: ADCON1 – A/D CONTROL REGISTER 1 (ADDRESS: 9Fh)

U-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	ADCS2	ADCS1	ADCS0	—	—	—	—
bit 7							bit 0

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

bit 6-4 **ADCS<2:0>:** A/D Conversion Clock Select bits

- 000 = Fosc/2
- 001 = Fosc/8
- 010 = Fosc/32
- x11 = FRC (clock derived from a dedicated internal oscillator = 500 kHz max)
- 100 = Fosc/4
- 101 = Fosc/16
- 110 = Fosc/64

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

3.2.4.2 ADCON0

ADCON0 controls the ADC operation. Bit 0 turns on the ADC module. Bit 1 starts a conversion and bits <5:2> selects which channel the ADC will operate. VCFG bit <6> selects the ADC reference, which may be either VDD or a separate reference voltage on VREF. ADFM bit <7> selects whether the 10 bits are right or left justified in the 16 bits.

For purposes of this lesson, the ADC must be turned on and pointed to RA0. Choose the internal voltage reference and 8TOSC conversion clock.

The ADC needs about 5 μ s, 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{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 4 bits of the result and copies them to the display LEDs attached to PORTC.

See the Analog-to-Digital section in the PIC16F685/687/689/690 Data Sheet (DS41262) for more details on the ADC module.

REGISTER 3-2: ADCON0 – A/D CONTROL REGISTER (ADDRESS: 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	VCFG	CHS3	CHS2	CHS1	CHS0	GO/ \overline{DONE}	ADON
						bit 0	

bit 7 **ADFM:** A/D Result Formed Select bit
 1 = Right justified
 0 = Left justified

bit 6 **VCFG:** Voltage Reference bit
 1 = VREF pin
 0 = VDD

bit 5-2 **CHS<3:0>:** Analog Channel Select bits
 0000 = Channel 00 (AN0)
 0001 = Channel 01 (AN1)
 0010 = Channel 02 (AN2)
 0011 = Channel 03 (AN3)
 0100 = Channel 04 (AN4)
 0101 = Channel 05 (AN5)
 0110 = Channel 06 (AN6)
 0111 = Channel 07 (AN7)
 1000 = Channel 08 (AN8)
 1001 = Channel 09 (AN9)
 1010 = Channel 10 (AN10)
 1011 = Channel 11 (AN11)
 1100 = CVREF
 1101 = VP6
 1110 = Reserved. Do not use.
 1111 = Reserved. Do not use.

bit 1 **GO/ \overline{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:** A/D Enable bit
 1 = A/D converter module is enabled
 0 = A/D converter 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	'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown