dsPICDEM™ 28-PIN STARTER 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. 11/12/04

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.

dsPICDEM™ 28-PIN STARTER DEMO BOARD USER'S GUIDE

Table of Contents

dsPICDEM™ 28-PIN STARTER DEMO BOARD USER'S GUIDE

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 dsPICDEM 28-Pin Starter 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 LAYOUT

This document describes how to use dsPICDEM 28-Pin Starter Demo Board as a development tool to emulate and debug firmware on a target board. The manual layout is as follows:

- **Chapter 1. Introduction** – This chapter introduces the dsPICDEM 28-Pin Starter Demo Board and provides a brief description of the hardware.
- **Chapter 2. Tutorial** – This chapter details the step-by-step process for getting the dsPICDEM 28-Pin Starter Demo Board up and running with the MPLAB® In-Circuit Debugger 2 (MPLAB ICD 2).
- **Chapter 3. Demonstration Program Description** – This chapter describes the operational functionality of the sample code that is preprogrammed into the dsPIC30F device.
- **Chapter 3. Demonstration Program Description** – TThis chapter describes the hardware on the dsPICDEM 28-Pin Starter Demo Board.

- **Appendix A. Drawings and Schematics** – This appendix provides dsPICDEM 28-Pin Starter Demo Board hardware layout and schematic diagrams.
- **Appendix B. Demonstration Source Code** – This appendix contains a source code listing for the demonstration program included with the dsPICDEM 28-Pin Starter Demo Board.

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	*MPLAB® IDE User's Guide*
	Emphasized text	...is the *only* 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	*File>Save*
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
'b*nnnn*	A binary number where *n* is a digit	'b00100, 'b10
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`
Italic Courier	A variable argument	*`file`*`.o`, where *`file`* can be any valid filename
`0x`*`nnnn`*	A hexadecimal number where *`n`* is a hexadecimal digit	`0xFFFF`, `0x007A`
Square brackets [ ]	Optional arguments	`mcc18 [options]` *`file`* `[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 dsPICDEM 28-Pin Starter Demo Board. Other useful documents are listed below. The following Microchip documents are available and recommended as supplemental reference resources.

dsPIC30F Family Reference Manual (DS70046)

Consult this document for detailed information on dsPIC30F device operation. This reference manual explains the operation of the dsPIC30F MCU family architecture and peripheral modules but does not cover the specifics of each device. Refer to the appropriate device data sheet for device-specific information.

dsPIC30F2010 Data Sheet (DS70118)

Consult this document for detailed information on the dsPIC30F2010 device. Reference information found in this data sheet includes:

- Device memory map
- Device pinout and packaging details
- Device electrical specifications
- List of peripherals included on the device

dsPIC30F Programmer's Reference Manual (DS70030)

This manual is a software developer's reference for the dsPIC30F 16-bit MCU family of devices. It describes the instruction set in detail and also provides general information to assist in developing software for the dsPIC30F MCU family.

dsPIC30F Family Overview (DS70043)

This document provides an overview of the functionality of the dsPIC® product family. It helps determine how the dsPIC30F 16-bit Digital Signal Controller Family fits a specific product application. This document is a supplement to the *dsPIC30F Family Reference Manual.*

MPLAB ASM30, MPLAB LINK30 and Utilities User's Guide (DS51317)

This document details Microchip Technology's language tools for dsPIC devices based on GNU technology. The language tools discussed are:

- MPLAB ASM30 Assembler
- MPLAB LINK30 Linker
- MPLAB LIB30 Archiver/Librarian
- Other Utilities

MPLAB C30 C Compiler User's Guide and Libraries (DS51284)

This document details the use of Microchip's MPLAB C30 C Compiler for dsPIC devices to develop an application. MPLAB C30 is a GNU-based language tool, based on source code from the Free Software Foundation (FSF). For more information about the FSF, see www.fsf.org.

Other GNU language tools available from Microchip are:

- MPLAB ASM30 Assembler
- MPLAB LINK30 Linker
- MPLAB LIB30 Librarian/Archiver

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

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

To obtain any of these documents, visit the Microchip web site at www.microchip.com.

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 (FAQ), 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® 1development 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

NOTES:

Chapter 1. Introduction

1.1 INTRODUCTION

The dsPICDEM 28-Pin Starter Demo Board serves as a development kit and evaluation tool for the dsPIC30F High Performance Digital Signal Controller family. Topics covered in this chapter include:

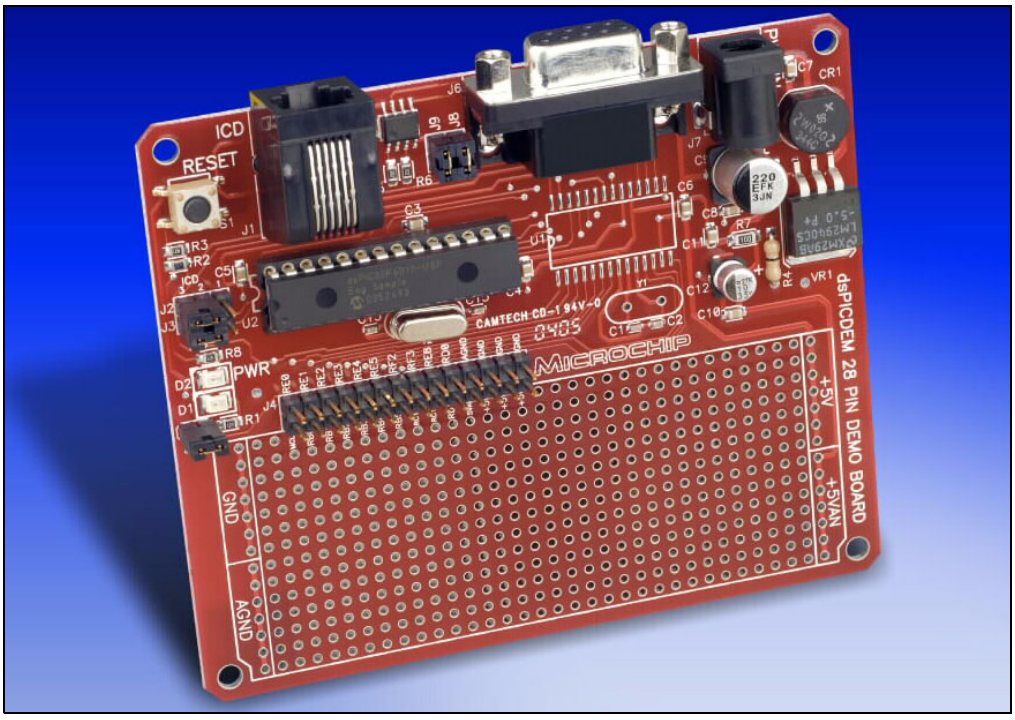
- Development Kit Contents
- dsPICDEM 28-Pin Starter Demo Board Functionality and Features
- dsPIC30F2010 Demonstration Program

1.2 DEVELOPMENT KIT CONTENTS

The following items comprise the dsPICDEM 28-Pin Starter Demo Board Development Kit:

- The dsPICDEM 28-Pin Starter Demo Board printed circuit board (see Figure 1-1).
- A preprogrammed dsPIC30F2010 device
- The dsPICDEM 28-Pin Starter Demo Board CD-ROM containing this manual, dsPIC30F documentation and demonstration program code

FIGURE 1-1: dsPICDEM 28-PIN STARTER DEMO BOARD



For information on the components used on the dsPICDEM 28-Pin Starter Demo Board see **Chapter 4. "dsPICDEM™ Development Hardware"**.

1.3 dsPICDEM 28-PIN STARTER DEMO BOARD FUNCTIONALITY AND FEATURES

The dsPICDEM 28-Pin Starter Demo Board is a simple tool that allows you to begin development with dsPIC30F devices. It provides these capabilities:

Development Board Power

- On-board +5V regulator for V_{DD} and AV_{DD} with direct input from 9V AC or DC wall adapter
- 9V DC power source input jack for development board
- Power-on indicator LED

MPLAB ICD 2 Connections

- MPLAB ICD 2 programming connector
- Pad locations for 28-pin SOIC or SDIP package

Serial Communication Channel

- Single RS-232 communication channel

Device Clocking

- 7.37 MHz crystal for dsPIC device
- Two crystal locations on board support either SDIP or SOIC package

Miscellaneous

- Reset push button for resetting the dsPIC device
- LED connected to pin RD0 for status indicator
- All device I/O pins are brought out to a header for test point and prototyping access
- Prototype area for user hardware

1.4 dsPIC30F2010 DEMONSTRATION PROGRAM

The dsPICDEM 28-Pin Starter Demo Board is supplied with a pre-loaded program that demonstrates some of the CPU and peripheral functions of the dsPIC30F devices:

- Demonstrates interrupt handling by using Timer1 to schedule the I/O pin activity.
- Demonstrates RS-232 functionality by using the UART peripheral to echo characters sent from a PC terminal program at 2400 baud.

Refer to **Chapter 3. "Demonstration Program Description"** for information on the demonstration program code.

Chapter 2. Tutorial

2.1 INTRODUCTION

This chapter is a self-paced tutorial to get you started using the dsPICDEM 28-Pin Starter Demo Board. Topics covered in this chapter include:

- Tutorial Overview
- Equipment Needed
- Creating the Project
- Building the Code
- Programming the Chip
- Debugging the Code
- Summary

2.2 TUTORIAL OVERVIEW

The tutorial program is located on the CD-ROM provided with the development kit, in the `2010_demo.s` file. The tutorial program is written in assembly code. This program echoes any characters that are sent to the UART from the RS-232 interface. In addition, the program pulses all remaining I/O lines in succession. Each I/O pin is pulsed high for approximately 50 milliseconds at 1 second intervals. Timer1 is used to schedule the I/O pin pulse events.

The source file is used with a linker script file (`p30f2010.gld`) and an include file (`p30f2010.inc`) to form a complete project. This simple project uses a single source code file; however, more complex projects might use multiple assembler and compiler source files as well as library files and precompiled object files.

There are four steps to this tutorial:

1. Creating a project in MPLAB IDE.
2. Assembling and linking the code.
3. Programming the chip with the MPLAB ICD 2.
4. Debugging the code with the MPLAB ICD 2.

2.3 EQUIPMENT NEEDED

To complete this tutorial, you will need the following items:

1. dsPICDEM 28-Pin Starter Demo Board
2. 9V, 500 mA Plug-in Power Supply with barrel style plug
3. MPLAB ICD 2 In-Circuit Debugger/Programmer
4. 9-pin, straight-through RS-232 cable
5. PC running Microsoft Windows® with MPLAB IDE 6.41 or later

2.4 CREATING THE PROJECT

The first step is to create a project and a workspace in MPLAB IDE. Typically, there is one project in one workspace.

Note: These instructions presume the use of MPLAB 6.41 or newer. Older versions of MPLAB IDE do not support the dsPIC30F2010 device.

A project contains the files needed to build an application (source code, linker script files, etc.) along with their associations to various build tools and build options.

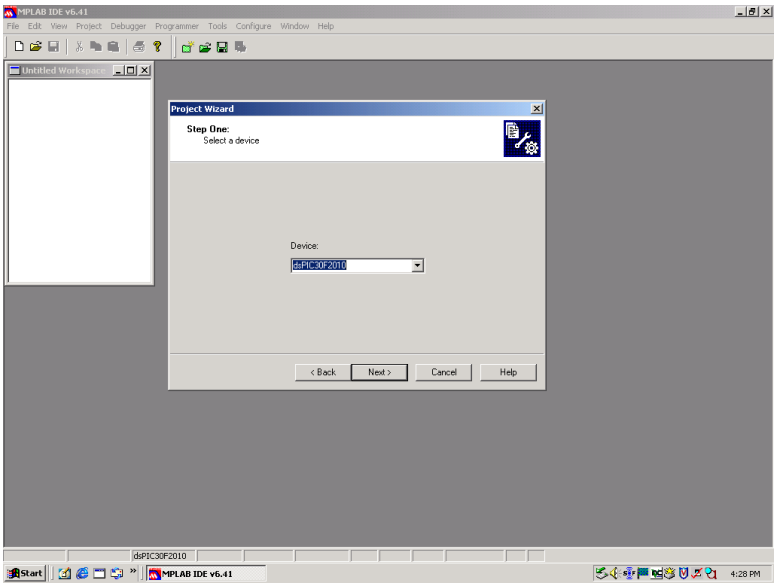
A workspace contains one or more projects and information on the selected device, debug tool and/or programmer, open windows and their location and other IDE configuration settings.

MPLAB IDE contains a Project Wizard to help create new projects. Before starting, create a folder for the project files for this tutorial (`C:\Tutorial` is assumed in the instructions that follow). From the Example Code directory on the dsPICDEM 28-Pin Starter Demo Board Kit CD, copy the `2010_demo.s` file into the `C:\Tutorial` folder.

2.4.1 Select a Device

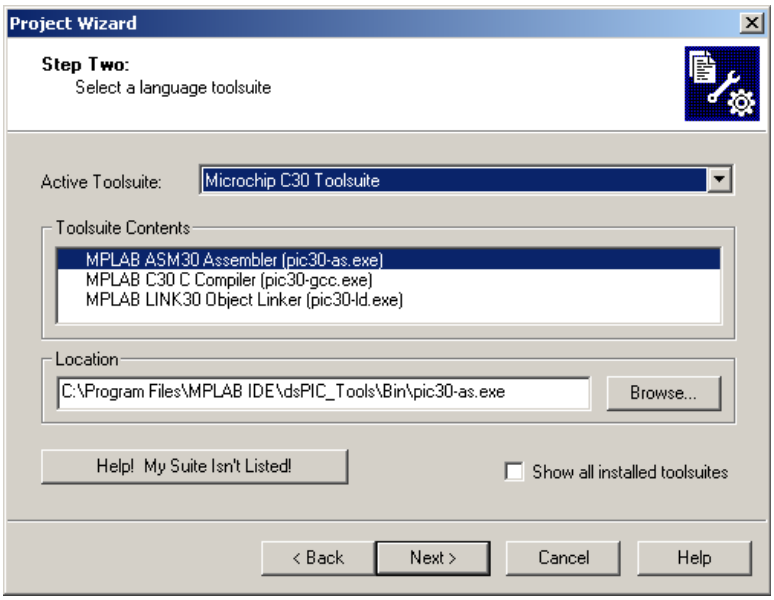
1. Start MPLAB IDE.
2. Close any workspace that might be open (*File>Close Workspace*).
3. From the *Project* menu, select *Project Wizard*.
4. From the Welcome screen, click **Next>** to display the Project Wizard Step One dialog (see Figure 2-1).

FIGURE 2-1: PROJECT WIZARD, STEP 1, SELECT A DEVICE



5. From the **Device:** pull-down list, select `dsPIC30F2010` and click **Next >**. The Project Wizard Step Two dialog displays (see Figure 2-2).

FIGURE 2-2: PROJECT WIZARD, STEP 2, SELECT LANGUAGE TOOLSUITE



2.4.2 Select Language Toolsuite

1. From the **Active Toolsuite:** pull-down menu, select **Microchip C30 Toolsuite**. This toolsuite includes the assembler and linker that will be used (the C Compiler is not used).
2. In the **Toolsuite Contents** block, select **MPLAB ASM 30 Assembler (pic30-as.exe)**
3. In the **Location** block, click **Browse...** and navigate to:

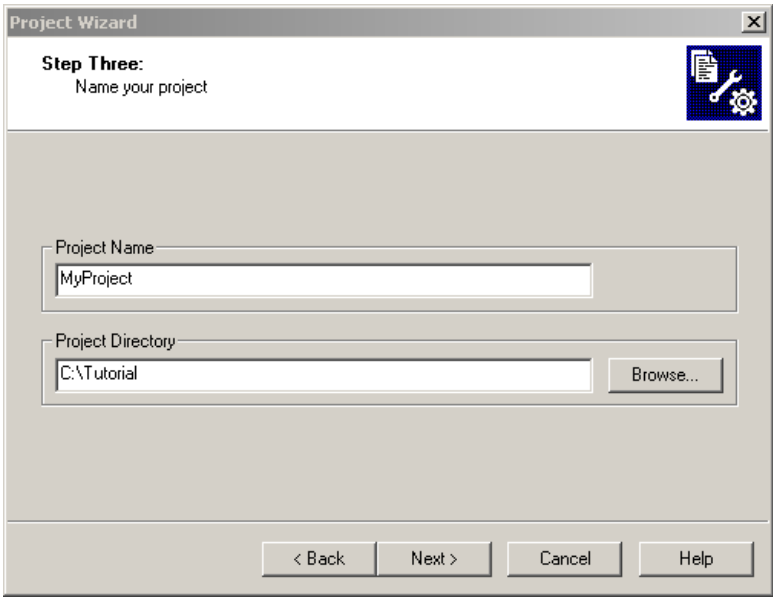
   `C:\Program Files\MPLAB IDE\dsPIC_Tools\Bin\pic30-as.exe`
4. With **MPLAB LINK 30 Object Linker (pic30-ld.exe)** selected in **Toolsuite Contents**, click **Browse...** and navigate to:

   `C:\Program Files\MPLAB IDE\dsPIC_Tools\Bin\pic30-ld.exe`

Note: If you have the MPLAB C30 Toolsuite installed, browse to the `C:\pic30-tools\` directory to set tool locations instead of `C:\Program Files\MPLAB IDE\dsPIC_tools\`.

5. Click **Next >** to continue. The Project Wizard Step Three dialog displays (see Figure 2-3).

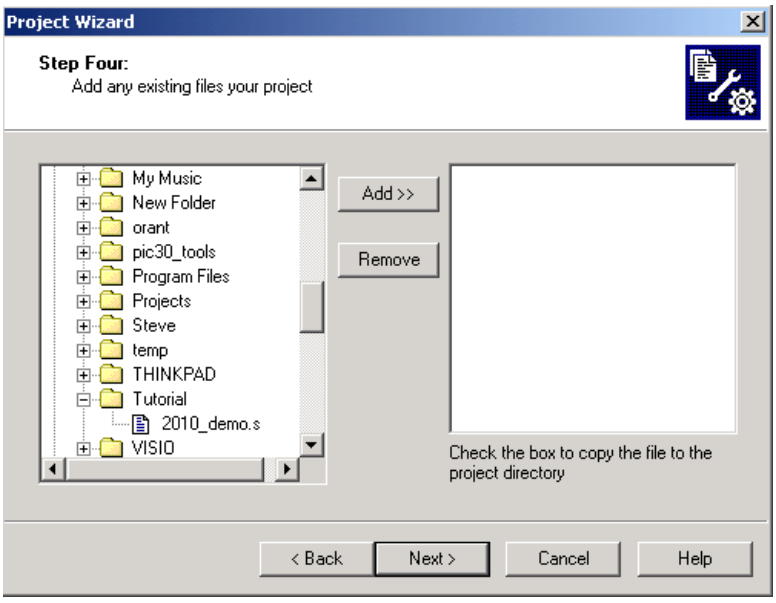
FIGURE 2-3: PROJECT WIZARD, STEP 3, NAME YOUR PROJECT



2.4.3 Name Your Project

1. In the **Project Name** text box, type **MyProject.**
2. Click **Browse...** and navigate to `C:\Tutorial\` to place your project in the Tutorial folder.
3. Click **Next >** to continue. The Project Wizard Step Four dialog displays (see Figure 2-4).

FIGURE 2-4: PROJECT WIZARD, STEP 4, ADD FILES TO PROJECT

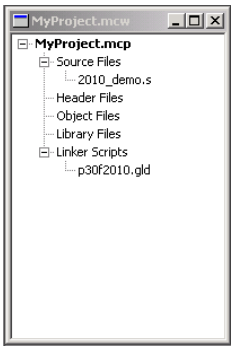


2.4.4 Add Files to Project

1. Locate the `C:\Tutorial` folder and select the `2010_demo.s` file.
2. Click **Add>>** to include the file in the project.
3. Expand the `C:\Program Files\MPLAB IDE\dsPIC_Tools\support\gld` folder and select the `p30f2010.gld` file.
4. Click **Add>>** to include this file in the project. There should now be two files in the project.
5. Click **Next>** to continue.
6. When the summary screen displays, click **Finish.**

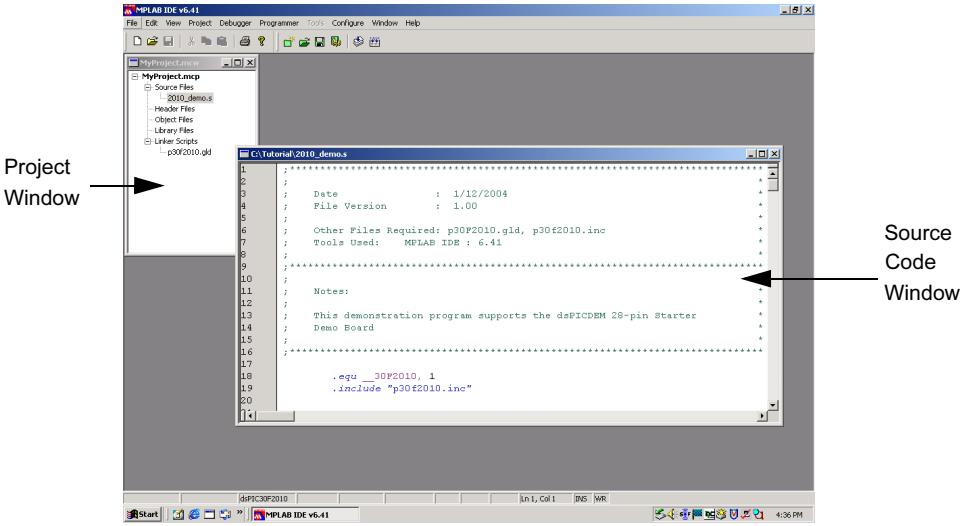
After the project wizard completes, the MPLAB IDE project window shows the `2010_demo.s` file in the **Source Files** folder and the `p30f2010.gld` file in the **Linker Scripts** folder (see Figure 2-5).

FIGURE 2-5: MPLAB IDE PROJECT WINDOW



A project and workspace has now been created in MPLAB IDE. `MyProject.mcw` is the workspace file and `MyProject.mcp` is the project file. Double-click the `2010_demo.s` file in the project window to open the file. MPLAB IDE should now look similar to Figure 2-6.

FIGURE 2-6: MPLAB IDE WORKSPACE WINDOWS



2.5 BUILDING THE CODE

In this project, building the code consists of assembling the `2010_demo.s` file to create an object file, `2010_demo.o` and then linking the object file to create the `MyProject.hex` and `MyProject.cof` output files. The `.hex` file contains the data necessary to program the device and the `.cof` file contains additional information that lets you debug at the source code level. Note that these files have the same name as the project name that you selected earlier.

Before building, there are settings required to tell MPLAB IDE where to find the include files and to reserve space for the extra debug code when the MPLAB ICD 2 is used.

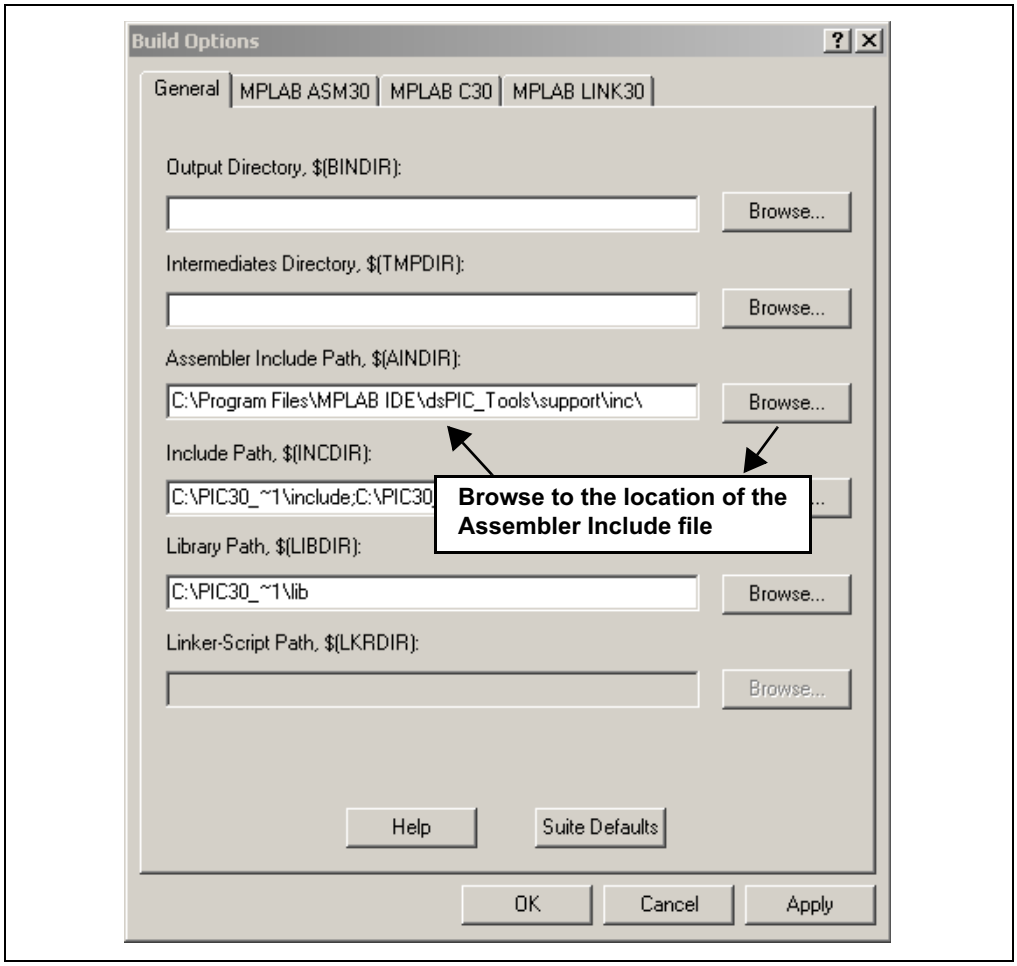
The following line is near the top of the `2010_demo.s` file:

```
.include "p30f2010.inc"
```

This line causes a standard include file to be used. Microchip provides these files with all the Special Function Register (SFR) labels already defined for convenience.

To build the code, select *Build Options>Project* from the *Project* menu. The Build Options dialog displays (see Figure 2-7).

FIGURE 2-7: BUILD OPTIONS



2.5.1 Identify Assembler Include Path

1. Select the General tab.
2. At the **Assembler Include Path, $(AINDIR):** box, click **Browse...** and navigate to:

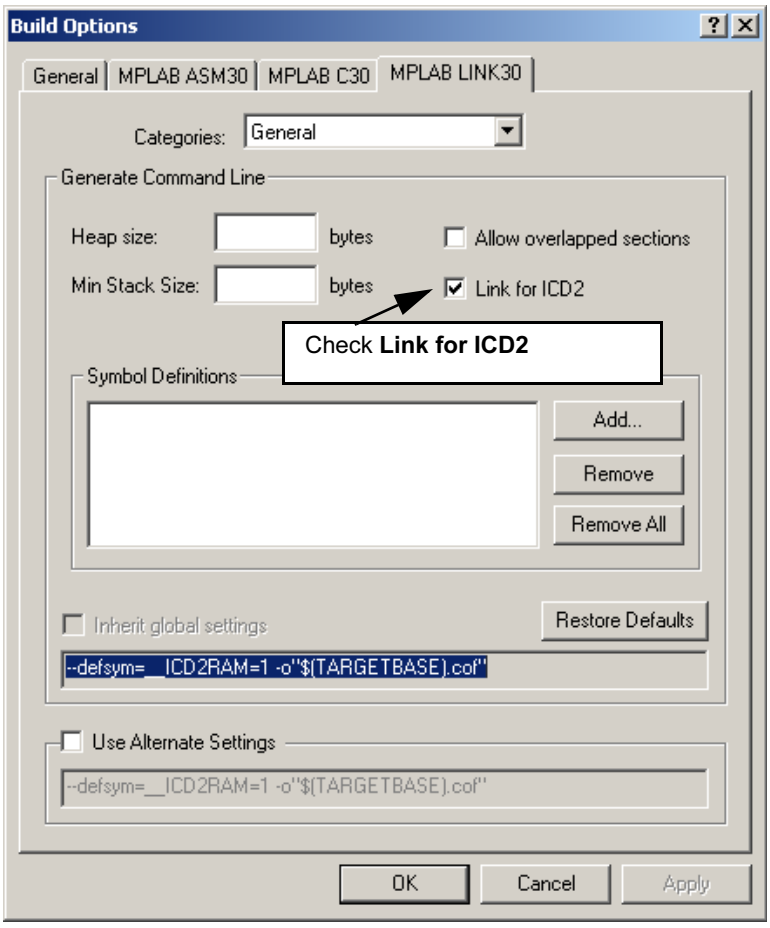
   `C:\Program Files\MPLAB IDE\dsPIC_Tools\support\inc`

   This path tells MPLAB IDE where to find the include files.

2.5.2 Link for MPLAB ICD 2

1. Select the MPLAB LINK30 tab to view the linker settings (see Figure 2-8).
2. Check **Link for ICD 2**.
3. Click **OK**. The text box closes while the linker reserves space for the debug code used by the MPLAB ICD 2.
4. Click **OK** again to save these changes. The project is now ready to build

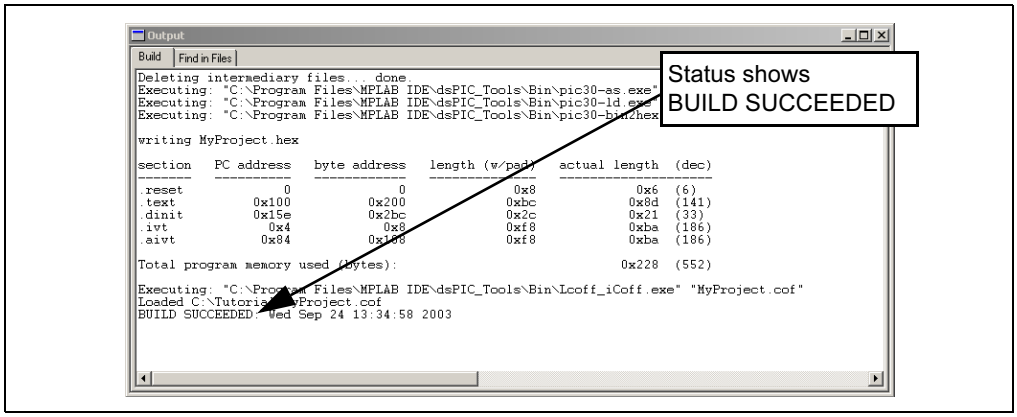
FIGURE 2-8: MPLAB LINK30 BUILD OPTIONS



2.5.3 Build the Project

1. Select the *Make>Project* menu to display the Build Output window display (Figure 2-9).
2. Observe the progress of the build.
3. When the BUILD SUCCEEDED message displays, you are ready to program the device.

FIGURE 2-9: EXAMPLE BUILD OUTPUT



2.6 PROGRAMMING THE CHIP

The MPLAB ICD 2 In-Circuit Debugger can be used to program and debug the dsPIC30F2010 device in circuit on the dsPICDEM 28-Pin Starter Demo Board.

Note:	Before proceeding, make sure that the USB driver for the MPLAB ICD 2 has been installed on the PC (see the *MPLAB ICD 2 User's Guide*, DS51331) for more details regarding the installation of the MPLAB ICD 2.

Use the following procedures to program the dsPIC30F2010 device.

2.6.1 Set Up The Device Configuration

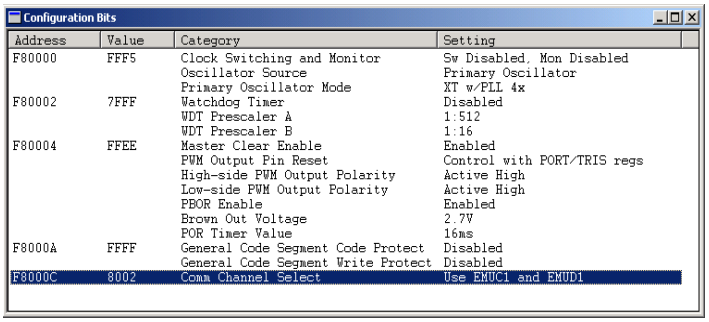
Use the *Configure>Configuration Bits* menu to display the configuration settings. The configuration bits window is shown in Figure 2-10.

The device configuration bits determine global device operating parameters, such as clock source, brown out threshold voltage and so forth. For this code example, the configuration bit settings have been specified by using special directive statements in the source code file. If these directives were not used, you would need to specify the settings manually in the configuration bits window. For this project, the Primary Oscillator has been selected as the oscillator source and XT w/ 4X PLL has been selected as the Primary Oscillator Mode. A 7.37-MHz crystal has been installed on the board. With the 4X PLL enabled, the device instruction clock frequency (FCY) is 7.37 MHz.

FIGURE 2-10: CONFIGURATION SETTINGS

Category settings can be specified in the source code or set manually on this screen.

Set Comm Channel Select to 'Use EMUC1 and EMUD1'



2.6.2 Select the MPLAB ICD 2 Communication Pins

All dsPIC30F devices use a pair of I/O pins (PGC/EMUC and PGD/EMUD) for initially loading your application program into the device and for communicating with the MPLAB ICD 2 In-Circuit Debugger. Typically, these pins can be used by your application program for other functions after your program is loaded into the device. However, these application functions would be not be available while you were connected to the MPLAB ICD 2 for debugging.

To circumvent this issue, most dsPIC30F devices use one or more sets of alternate pins for MPLAB ICD 2 communication. These alternate pins are identified as EMUC*x* and EMUD*x*, where *x* designates the number of the pin pair. By selecting an alternate set of pins for the MPLAB ICD 2, you can safely use the original I/O pins for your application.

The dsPIC30F2010 device has four pairs of MPLAB ICD 2 communication pins. The dsPICDEM 28-pin Starter Demo Board directly supports two selections for MPLAB ICD 2 communication via jumpers J2 and J3. For this demo application, alternate pins EMUC1 and EMUD1 are used for debugging because the EMUC and EMUD pins are multiplexed with the U1RX and U1TX pins, which are used for RS-232 communication. In practice, you may need to select other pairs of debugging pins for your application to avoid I/O pin conflicts.

Note: See Section 4.2 and Section 4.3 for more information about MPLAB ICD 2 communication pins. See the *MPLAB ICD 2 In-Circuit Debugger User's Guide* (DS51331) for more information about MPLAB ICD 2 support.

To select the MPLAB ICD 2 communication pins:

1. On the Configuration Bits screen, go to the **Comm Channel Select** category.
2. In the **Setting** column, set this parameter to **Use EMUC1 and EMUD1**.

2.6.3 Connect the MPLAB ICD 2 In-Circuit Debugger

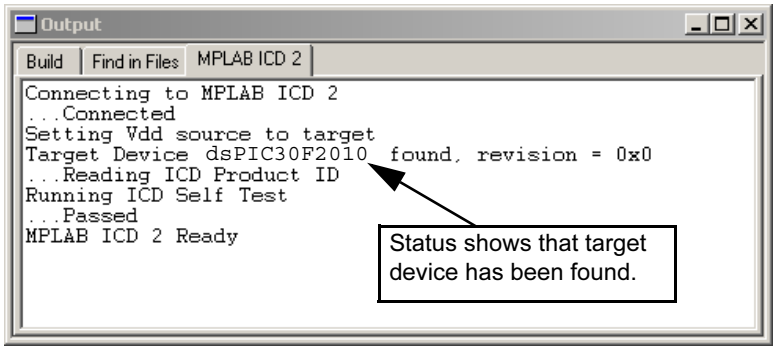
1. Apply power to the board.
2. Connect the MPLAB ICD 2 to the PC with the USB cable.
3. Connect the MPLAB ICD 2 to J1 on the dsPICDEM 28-Pin Starter Demo Board with the short RJ-11 (telephone) cable.
4. On the PCB, make sure that jumpers J2 and J3 are installed across pins 2 and 3 (the left two jumper pins). This configuration connects the MPLAB ICD 2 to the PGC and PGD programming pins.

2.6.4 Enable MPLAB ICD 2 Connection

1. From the *Debugger* menu, click *Select Tool>MPLAB ICD 2* to designate the MPLAB ICD 2 as the debug tool in MPLAB IDE.
2. From the *Debugger* menu, select *Connect* to connect the debugger to the device. The MPLAB IDE should report that it found the dsPIC2010 device, as shown in Figure 2-11.

Note: MPLAB IDE may need to download new firmware if this is the first time the MPLAB ICD 2 is being used with a dsPIC30F device. Allow it to do so. If any errors are shown, double-click the error message to get more information.

FIGURE 2-11: ENABLING MPLAB ICD 2

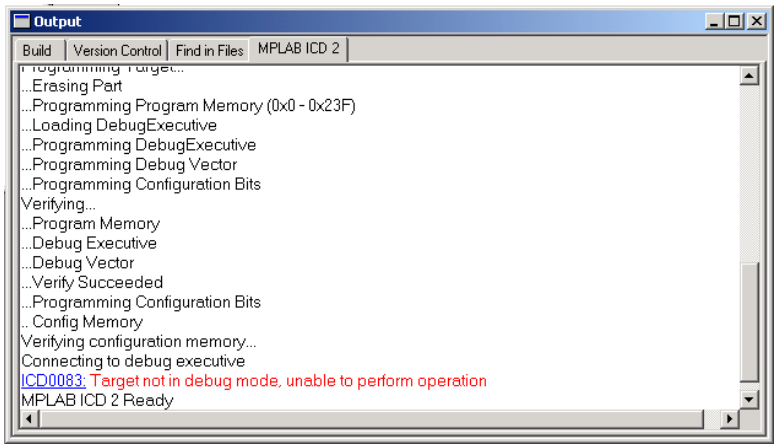


2.6.5 Program the dsPIC30F2010 Device

1. From the *Debugger* menu, select *Program* to program the part. The output window (Figure 2-12) displays the program steps as they occur.
2. When the programming stops, you will get a "Target not in debug mode" message, which is normal. This message is produced because the device has been programmed to use the alternate EMUC1 and EMUD1 communication pins. Move jumpers J2 and J3 to the right hand position, shorting pins 1 and 2 of the 3-way headers.

Note: If you are using the alternate MPLAB ICD 2 communication pins for debugging, you will need to restore jumpers J2 and J3 to the left hand position each time the device needs to be re-programmed (see Section 2.6.2).

FIGURE 2-12: PROGRAMMING THE dsPIC DEVICE



3. Use the *Debugger>Run* menu option to run the code. LED1 should start blinking at a 1 second rate.
4. Start the Windows HyperTerminal program and set up a connection to an available COM port for 2400 baud, no parity, 8 data bits and 1 stop bit. Connect a serial cable between the DB-9 connector on the PCB and the PC. When you type characters on the PC keyboard, they should be echoed on the HyperTerminal display when the demo program is running. A message will appear on the HyperTerminal display each time the application is reset and run again.

2.7 DEBUGGING THE CODE

The MPLAB ICD 2 In-Circuit Debugger can be used to run, halt and step the code. A breakpoint can be set to halt the program once the code has executed the instruction at the breakpoint. The contents of the RAM and registers can be viewed whenever the processor has been halted.

The MPLAB ICD 2 In-Circuit Debugger uses the following function keys to access the main debugging functions:

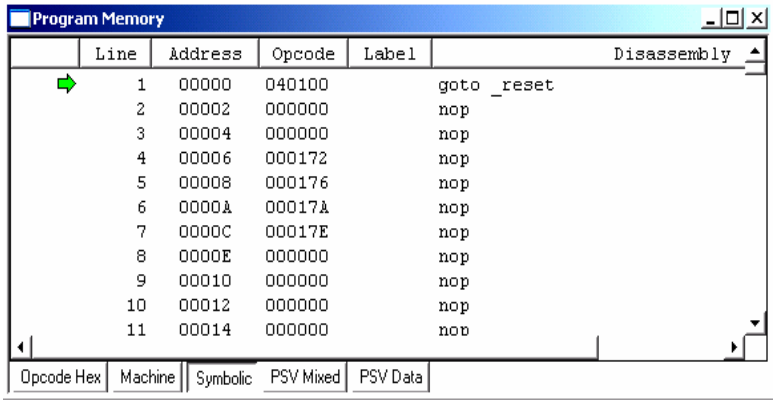
<F5> Halt
<F6> Reset
<F7> Single Step
<F9> Run

In addition, there are more functions available by right clicking on a line of source code. The most important of these are "Set Breakpoint" and "Run to Cursor".

2.7.1 Display the Code

1. From the *View* menu, select *Program Memory*.
2. On the Program Memory window, select the **Symbolic** tab, as shown in Figure 2-13.

FIGURE 2-13: PROGRAM MEMORY WINDOW



3. Press <F5> to halt the processor and press <F6> to reset. The program memory now shows a green arrow pointing to the line of code at address 00000, the reset location.

   The instruction at this location is `goto _reset`. This code is added by the linker to make the program branch to the start of the code in the `2010_demo.s` file. The code uses the `_reset` label at the start of the executable code and declares the label as global to have visibility outside the source file (see Example 2-1).

EXAMPLE 2-1: CODE START-UP

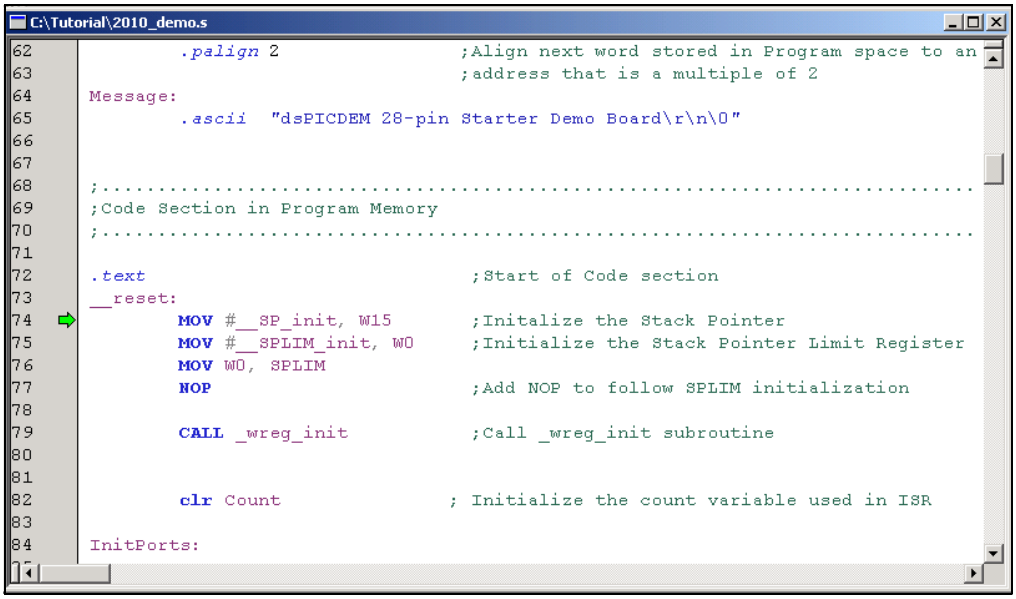
```
.global __reset
..text
    __reset:      mov    #__SP_init, W15 Pointer
                  mov    #__SPLIM_init, W0
                  mov    W0, SPLIM
```

The linker also provides values for the `__SP_init` and `__SPLIM_init` constants to initialize the stack pointer (w15), since the linker determines what RAM is available for the stack.

2.7.2 Step the Program

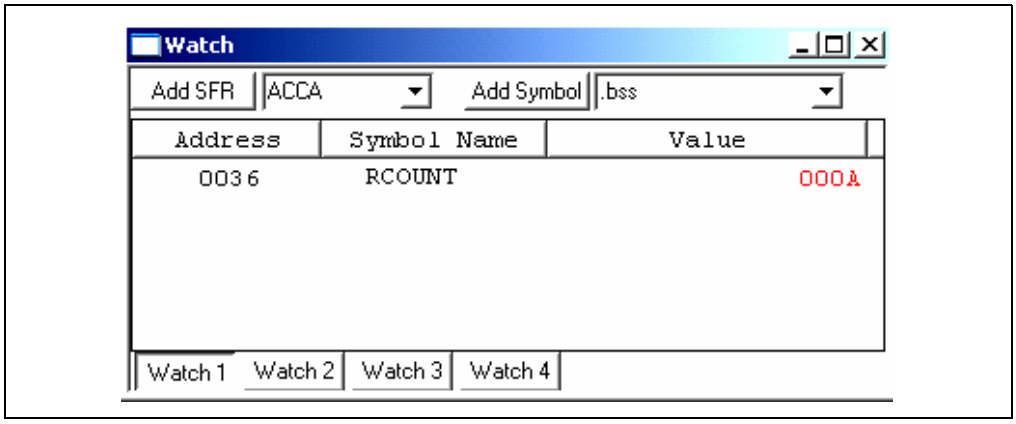
1. Press <F7> to single step the code. The green arrow moves to the code at `__reset` in the `2010_demo.s` source code, as shown in Figure 2-14.

FIGURE 2-14: SOURCE CODE WINDOW



2. Right click the line of code `mov W0,W14` and choose *Run to Cursor*. The green arrow moves to `repeat #12` because it has executed the prior lines of code up to and including `mov W0,W14`.
3. From the *View* menu, select *Watch* to open a Watch Window.
4. From the **Add SFR** pull-down list, display **RCOUNT**.
5. Click **Add SFR** to add the RCOUNT register to the Watch window.
6. Press <F7> a few times and watch the RCOUNT value decrement (see Figure 2-15. RCOUNT is the repeat loop counter and decrements to zero as the instruction in a repeat loop is executed several times.

FIGURE 2-15: WATCH WINDOW DISPLAY



2.7.3 Set Breakpoint

1. To set a breakpoint, right-click a code line and select *Set Breakpoint* from the pop-up menu.

Note: An alternate method is to simply double-click the line. This feature may need to be enabled using the *Edit>Properties* menu.

As an example, find the following line of code and set a breakpoint on this line:

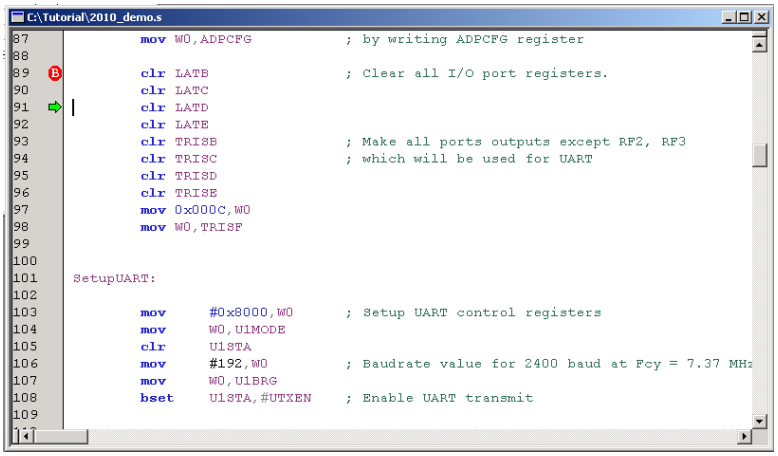
```
clr    LATB            ; Clear all I/O port registers.
```

A red stop sign should appear in the gutter (grey bar on the left) of the source code window.

2. Press <F6> to reset the device, then <F9> to run the code. The program halts on the second instruction following the breakpoint as shown in Figure 2-16. When ICD 2 is halted, the next instruction in the pipeline is executed. The arrow points to the instruction that will execute when the program resumes.

Note: The instruction on which the code halts could be elsewhere in the code if the breakpoint is set on a branch or call instruction. Refer to Section 12 titled "Important Notes" in the Readme file `MPLAB ICD 2.txt` located in the `C:\MPLAB IDE\README`S directory for additional operational information on the MPLAB ICD 2.

FIGURE 2-16: SETTING BREAKPOINT



2.8 PROGRAMMING THE DEVICE FOR STANDALONE OPERATION

The previous example showed you the basics of code debugging using the MPLAB ICD 2. When you have fully debugged your application, you will want to run the code without using the MPLAB ICD 2. In this case, the MPLAB ICD 2 is enabled as a device programmer instead of a debugger. The following steps describe how to accomplish this.

1. Ensure that J2 and J3 are both installed across the left two jumper pins (pins 2 and 3 shorted). This connects the MPLAB ICD 2 to the PGC and PGD programming pins.
2. Starting with the project you have created in this Tutorial, select MPLAB ICD 2 as the device programmer in the *Programmer* menu. Select the *Select Programmer>MPLAB ICD 2* option. If you were previously using the MPLAB ICD 2 as a debugger tool, you will get a warning message indicating that the tool cannot be enabled a as a programmer and a debugger at the same time. Click 'OK' on the warning message to continue.
3. From the *Program* menu, select *Program* to program the part. The output window will look similar to Figure 2-12, except that the debugging features of the device will not be enabled.
4. Remove the MPLAB ICD 2 programming cable connected to J1. When the cable is unplugged, the device will begin to run the application.

2.9 SUMMARY

This tutorial demonstrates the main features of the MPLAB IDE and MPLAB ICD 2 as they are used with the dsPICDEM 28-Pin Starter Demo Board. Upon completing this tutorial, you should be able to:

- Create a project using the Project Wizard.
- Assemble and link the code and set the configuration bits.
- Set up MPLAB IDE to use the MPLAB ICD 2 In-Circuit Debugger.
- Program the chip with the MPLAB ICD 2.
- View the code execution in program memory and source code.
- View registers in a Watch Window.
- Set a breakpoint and make the code halt at a chosen location.
- Use the function keys to Reset, Run, Halt and Single-Step the code.
- Program the device for debugger mode or standalone operation.

NOTES:

Chapter 3. Demonstration Program Description

3.1 INTRODUCTION

This chapter provides an overview of the dsPICDEM 28-Pin Starter Demo Board demonstration program. Detailed information on the dsPICDEM 28-Pin Starter Demo Board hardware is presented in **Chapter 4. "dsPICDEM™ Development Hardware"** and **Appendix A. "Drawings and Schematics"**.

3.2 HIGHLIGHTS

Items presented in this chapter include:

- Demonstration Program Summary
- Demonstration Program Description

3.3 DEMONSTRATION PROGRAM SUMMARY

The dsPICDEM 28-Pin Starter Demo Board is shipped with a simple example application programmed into the dsPIC30F2010 device. This program demonstrates the use of key functionality.

3.4 DEMONSTRATION PROGRAM DESCRIPTION

When power is applied to the dsPICDEM 28-Pin Starter Demo Board the dsPIC device begins executing the demonstration program. The program demonstrates the following functions:

- Configuration bits setup
- Global symbol declaration
- Declaration of variables and constant data
- I/O and peripheral initialization
- Interrupt function declaration and processing
- UART communication

3.4.1 Configuration Bits Setup

The demo program takes advantage of the `config` directive to define the configuration bit values. The `p30f2010.inc` file has defined values for different configuration register settings.

Figure 3-1 shows this portion of the code.

FIGURE 3-1: CONFIGURATION BITS CODE

```
;.....................................................................
;Configuration bits:
;.....................................................................

    config __FOSC, CSW_FSCM_OFF & XT_PLL4     ;Turn off clock switching and
                                              ;fail-safe clock monitoring and
                                              ;use a crystal oscillator with
                                              ;the 4x PLL

    config __FWDT, WDT_OFF                    ;Turn off Watchdog Timer

    config __FBORPOR, PBOR_ON & BORV_27 & PWRT_16 & MCLR_EN
                                              ;Set Brown-out Reset voltage
                                              ;and set Power-up Timer to
                                              ;16msecs

    config __FGS, CODE_PROT_OFF               ;Set Code Protection Off for the
                                              ;General Segment
```

3.4.2 Global Symbol Declaration

The demo program declares `__reset` and `__T1Interrupt` as global symbols. These symbols have already been declared in the `p30f2010.gld` device linker script file. The `.global` declarations in the source code provides linker script visibility to these two symbols.

The `__reset` symbol is defined in the device linker script as the beginning of the executable code area (PC address 0x100). When this symbol is used in the source code, it defines the beginning of executable code.

The name of each interrupt vector has been defined in the device linker script. For a function to be used as an interrupt service routine (ISR), the name of the ISR declared in the device linker script must be used as the ISR function name in the source code file. In addition, the ISR function name must be declared as a global symbol, which causes the linker to place the address of the interrupt function in the interrupt vector table.

3.4.3 Variables and Constant Data

The demo program defines one 16-bit variable in data memory, `Count`. The `.space` directive is used to tell the linker to reserve 2 bytes of memory.

A string of constant character data is declared in program memory. The `.ascii` directive is used to declare an ASCII string.

3.4.4 I/O and Peripheral Setup

The demo program writes to the ADPCFG (A/D Port Configuration) register, which designates whether input pins associated with the A/D converter are for digital or analog signals. The A/D pins are set to analog mode by default. For this application, the A/D pins are used for digital signals. Consequently, information must be written to ADPCFG register designate the pins as digital.

The software configures all device I/O pins as outputs, with the exception of RF2 and RF3. These pins are used for the UART1 peripheral. When UART1 is initialized and enabled, it overrides the I/O functionality of pins RF2 and RF3. UART1 is enabled for transmit and receive. The baud rate register is set to provide 2400 baud communication at an internal instruction execution frequency of 7.37 MHz.

3.4.5 Interrupt Processing

To illustrate interrupt processing, the demonstration program uses Timer1 to generate periodic interrupts. The clock prescaler and period register for Timer1 are configured to produce an interrupt every 1/19th of a second. All I/O pins except for RF2 and RF3 (UART pins) are pulsed high for one Timer1 count period. Each I/O pin is pulsed in succession. There are 19 pins total in the I/O pin sequence, so the pulse period for each pin is 1 second.

The demo software declares one 16-bit uninitialized variable in RAM, `Count`, which is incremented by the `IO_Pin_Scan` function each time an interrupt occurs. The `IO_Pin_Scan` function demonstrates the use of relative branch instructions. The `Count` variable is used to calculate a jump address that writes the correct I/O pin. There are three instructions at each jump location in the `IO_Pin_Scan` function, so `Count` is multiplied by 3 to form the jump value.

The LED connected to pin RD0 flashes at 1 second intervals when the demonstration program is run. You can use an oscilloscope to observe the pulse created on the other I/O pins.

3.4.6 UART Communication

Basic UART functionality and the usage of constant data stored in program memory are demonstrated by the UART communication code.

The Program Space Visibility (PSV) feature of the dsPIC30F device family is used to access the character string stored in program memory. PSV allows a portion of constant user data to be located in program space, conserving available RAM. PSV maps a selected range of program memory into the upper half of the 64 Kbyte data space memory map. The data can be accessed as if it were located in data space. Any data memory reads from address 0x8000 or higher come from program space when PSV is enabled. The lower 15 bits of the program memory address are provided from the W register that holds the read pointer. The upper 8 bits of the program memory address are provided from the PSVPAG register.

The PSV access code used in the demonstration program is shown in Figure 3-2. Two assembler macros, `psvpage()`, and `psvoffset()` are used to derive the address values for the PSVPAG register and the W register used for the read address. W1 holds the lower 15 bits of the read address. Since character information is to be read from memory, the `mov.b` (byte mode move) instruction is used to read the data. The use of the byte mode instruction also results in a byte pointer adjustment of W1.

After a character has been read from program memory, it is checked for the null character (0). If it is not 0, the character is written to the UART and the loop continues.

After the message in program space has been sent to the UART, the main program loop polls the URXDA (UART receive data available) status bit to see if any received data is waiting in the buffer. If so, this data is read and transmitted back to the sender.

FIGURE 3-2: PSV ACCESS EXAMPLE

```
SetupUART:

    mov     #0x8000,W0      ; Setup UART control registers
    mov     W0,U1MODE
    clr     U1STA
    mov     #192,W0         ; Baudrate value for 2400 baud at Fcy = 7.37 MHz
    mov     W0,U1BRG
    bset    U1STA,#UTXEN    ; Enable UART transmit


    bset    CORCON,#PSV     ; Enable program space visibility

    ; Load PSV page register and setup W1 as a pointer to the
    ; ASCII text string stored in program memory

    mov     #psvpage(Message),W0
    mov     W0,PSVPAG
    mov     #psvoffset(Message),W1

    ; The next block of code writes the ascii string stored in
    ; program memory to the UART.  The loop repeats until a
    ; null character is found.
MsgLoop:
    mov.b   [W1++],W0       ; Read the data from P.M. using PSV
    cp0.b   W0              ; Is the character 0?
    bra     Z,MsgDone       ; If the character is 0, we're done.
    call    WriteChar       ; If not, write the character to UART
    bra     MsgLoop         ; Go back to the top of the loop.
```

To use the UART for communication, connect the J6 connector on the dsPICDEM 28-Pin Starter Demo Board to the RS-232 serial port on the PC with a DB9 cable. Using the HyperTerminal program available as a Microsoft Windows communications accessory, configure the serial port to 2400 baud, 8 bits with 1 stop bit, no parity and no flow control. When the device is reset, the message stored in program memory will appear on the HyperTerminal screen. After that, any characters that are typed on the PC keyboard should be echoed back to the HyperTerminal screen by the demonstration program.

Chapter 4. dsPICDEM™ Development Hardware

4.1 dsPICDEM™ 28-PIN STARTER DEMO BOARD HARDWARE OVERVIEW

This chapter describes the dsPICDEM 28-Pin Starter Demo Board hardware.

FIGURE 4-1: dsPICDEM 28-PIN STARTER DEMO BOARD

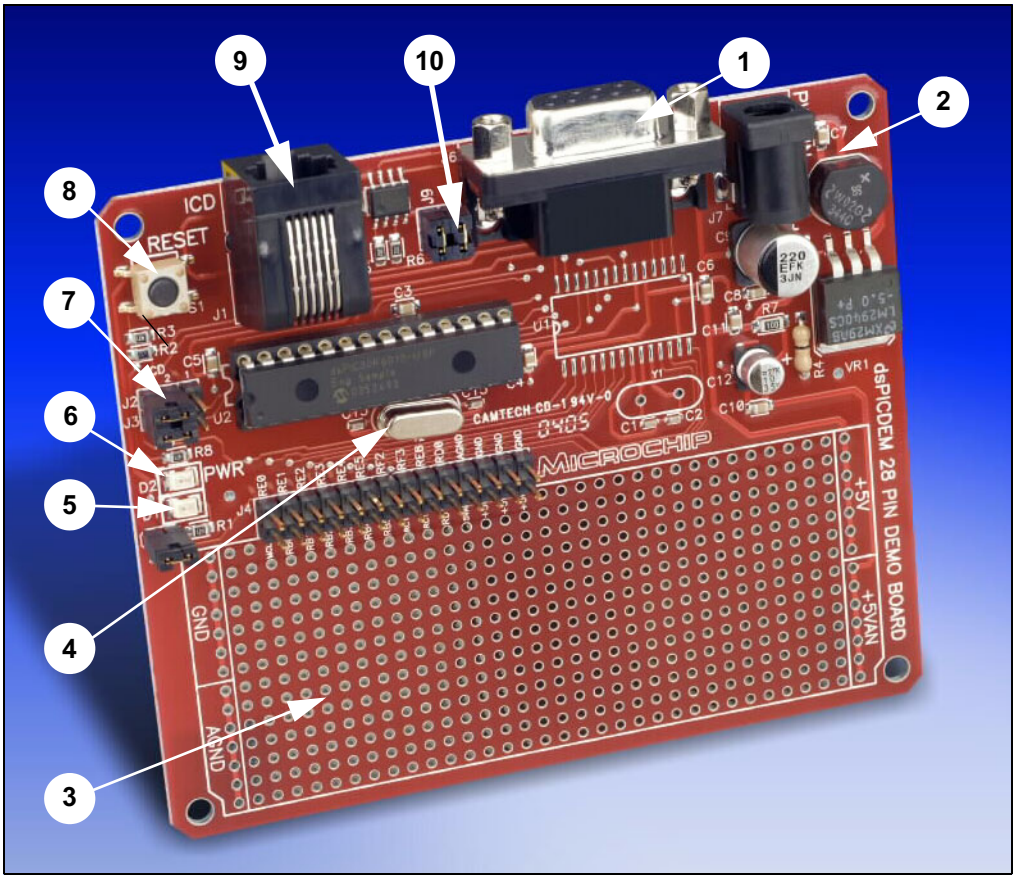


TABLE 4-1: dsPICDEM™ 28-PIN STARTER DEMO BOARD HARDWARE ELEMENTS

No.	Hardware Element
1	RS-232 Serial Port (Section 4.1.1)
2	Power Supply (Section 4.1.3)
3	Prototyping Area (Section 4.1.8)
4	Oscillator (Section 4.1.6)
5	LED Indicator (Section 4.1.5)
6	Power On Indicator (Section 4.1.4)
7	MPLAB ICD 2 debug pin-selection jumpers (Section 4.1.9)
8	Reset Button (Section 4.1.7)
9	MPLAB ICD 2 Connector (Section 4.1.2)
10	RS-232 Transceiver connection jumpers (Section 4.1.10)

4.1.1 RS-232 Serial Port

The dsPICDEM Starter Demo Board provides one RS-232 serial communication channel. The serial communication channel, labeled J6, is configured as an RS-232 communication channel. The dsPIC UART channel 1 U1RX and U1TX pins are connected to an RS-232 level-shifting IC (U3). The serial port is configured as Data Communication Equipment (DCE), and can be connected to a PC using a straight-through cable.

4.1.2 MPLAB ICD 2 Connector

By way of the modular connector J1, the MPLAB ICD 2 can be connected for low cost programming and debugging of the dsPIC device.

4.1.3 Power Supply

The dsPICDEM™ Starter Demo Board is powered by a 9V AC/DC wall adapter with a standard 2.1mm barrel plug. If a 9V adapter is not available, any voltage between 8V and 15V may be used. Separate +5V DC regulators (V_{DD} and AV_{DD}) provide power to their respective processor pins and prototyping area. A ground trace connects all V_{SS} points.

4.1.4 Power On Indicator

A red LED is connected to the input of the regulators to indicate the presence of power.

4.1.5 LED Indicator

An LED is connected to the RD0 pin by way of JP1 to indicate the status of pin RD0 when the JP1 jumper pin is in place.

4.1.6 Oscillator

A crystal oscillator (7.37 MHz) is supplied. The crystal oscillator can be used with the on-chip PLL circuit to provide internal instruction execution frequencies (F_{CY}) up to 29.5 MHz.

4.1.7 Reset Button

The MCLR Reset button (S1) connected to the processor $\overline{MCLR}$ pin provides a hard reset to the dsPIC device.

4.1.8 Prototyping Area

A prototyping area and associated header, J4, is provided which enables additional ICs and attachment boards to be added.

4.1.9 Debug Pin-Selection Jumpers

J2 and J3 determine which pair of communication pins is selected for use during MPLAB ICD 2 debugging. The jumpers can select either the primary pins (EMUC/EMUD) or one pair of secondary pins (EMUC2/EMUD1). J2 and J3 must always select the primary position to program the device (pins 2 and 3 shorted).

The dsPIC30F2010 device supports four different pairs of debug communication pins. These could be connected by installing jumper wires from pin 2 of J2 and J3 to the appropriate pins on connector J4.

For more information about MPLAB ICD 2 Debugger communication see **2.6.2 "Select the MPLAB ICD 2 Communication Pins"**

4.1.10 RS-232 Transceiver Jumpers

Jumpers J8 and J9 allow the RS-232 transceiver to be disconnected from the UART1 pins on the PIC18FXXXX device. These jumpers provide two functions. First, the transceiver can be removed from the circuit to avoid conflicts with the primary ICD communication pins. Second, the transceiver could be connected to alternate UART pins available on the PIC18FXXXX device. The alternate pins are available on connector J4.

4.2 DEBUGGING TIPS FOR SMALL PIN-COUNT DEVICES

When you use the Microchip MPLAB ICD 2 to debug application code, you will need to use two I/O pins on the device to provide the debugger communication signal connections. On a 28-pin device like the PIC18FXXXX, there is a good possibility that the pins used for MPLAB ICD 2 communication may have other important peripheral functions multiplexed onto the same pin. The multiplexing can present a challenge to the user while debugging the application. To avoid potential debugger pin conflicts, the dsPIC family of devices has four pairs of MPLAB ICD 2 communication pins.

Device programming is always performed on a dedicated pair of pins. After the device is programmed with the application code, debugger operations can take place using the same pair of programming pins or one of three other pairs of alternate MPLAB ICD 2 communication pins. The pair of pins used during MPLAB ICD 2 debugging is selected by device configuration bits.

For more information, see the appropriate data sheet for the device that you are using. This document will provide the locations of the debugger communication pins.

4.3 HARDWARE JUMPERS THAT CONTROL DEBUGGER OPERATION

The dsPICDEM 28-Pin Starter Demo Board has four jumpers that simplify debugger operation. Jumpers J2 and J3 select whether the EMUC/EMUD communication pins or the EMUC1/EMUD1 communication pins are used for debugging. When the device is programmed, J2 and J3 must be installed on the left two pins to connect the EMUC and EMUD pins. After programming, the jumpers can optionally be moved to connect EMUC1 and EMUD1. You could also connect the center pin of J2 and J3 to the appropriate pins on connector J4. J2 and J3 only allow connection to two available pairs of debugger pins, but the other two pairs are available on J4.

Jumpers J8 and J9 are used to optionally disconnect the UART communication pins from the RS-232 transceiver on the PCB. The UART1 receive and transmit functions are multiplexed with the EMUC and EMUD pins on the dsPIC30F2010 device. If you need to use the primary debugger communication pins EMUC and EMUD for debugging, you should disconnect the RS-232 transceiver using J8 and J9.

If your application requires a UART, there are two possible solutions for simultaneous use of the debugger and UART. First, you could select the alternate EMUC1 and EMUD1 pins for debugging. This would free the EMUC and EMUD pins for the UART function. Secondly, the UART on the dsPIC30F2010 device can support an alternate pair of communication pins by setting a bit in the control registers. The transceiver could be connected to the alternate set of UART pins by connecting wires between J8/J9 and J4. This configuration would allow the EMUC/EMUD pins to be used for both debugging and programming. J2 and J3 would not have to be changed between programming and debugging operations.

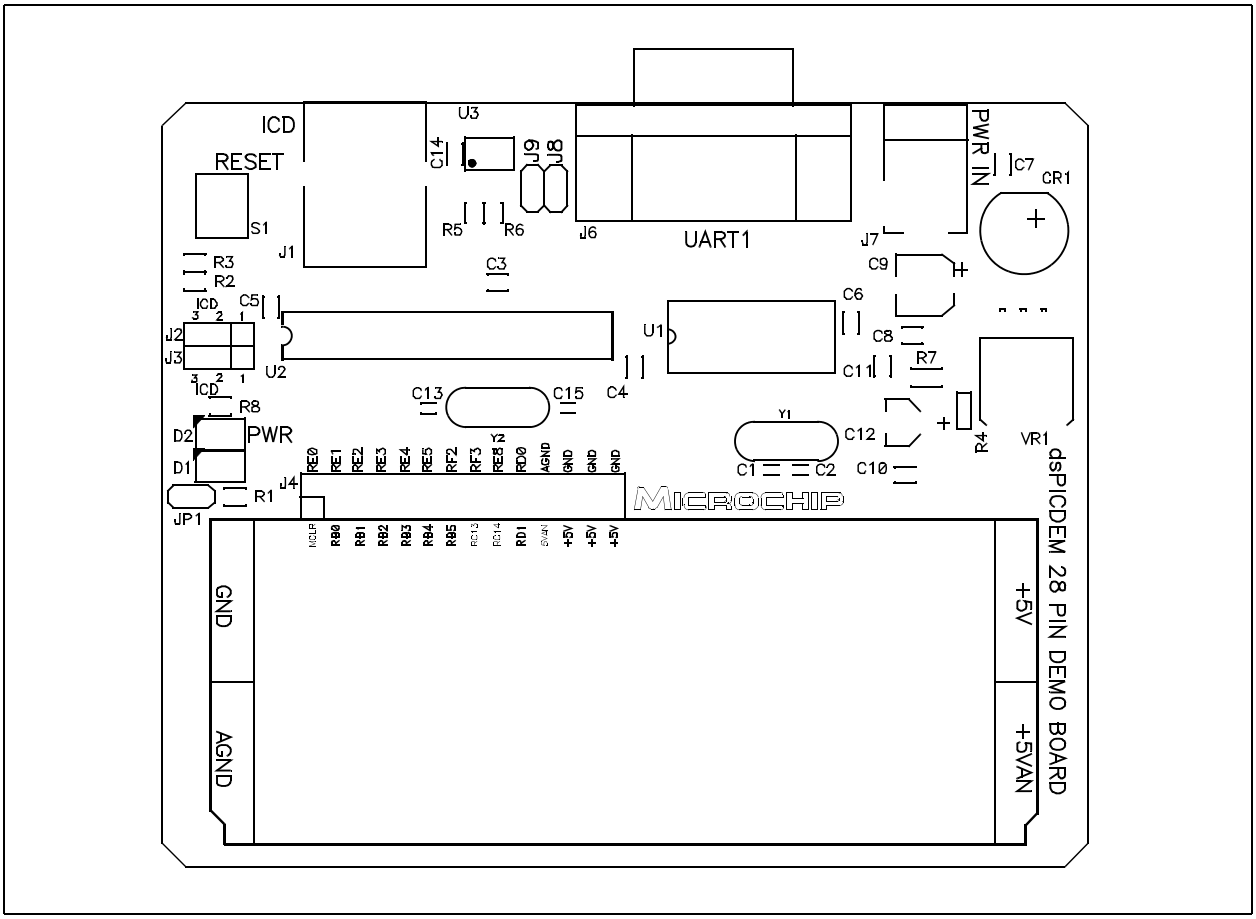
NOTES:

Appendix A. Drawings and Schematics

A.1 dsPICDEM™ 28-PIN STARTER DEMO BOARD LAYOUT

Figure A-1 shows the parts layout for the dsPICDEM 28-Pin Starter Demo Board.

FIGURE A-1: dsPICDEM™ 28-PIN STARTER DEMO BOARD LAYOUT



A.2 dsPICDEM™ 28-PIN STARTER DEMO BOARD SCHEMATICS

FIGURE A-2: dsPICDEM 28-PIN STARTER DEMO BOARD SCHEMATIC (PAGE 1 OF 3)

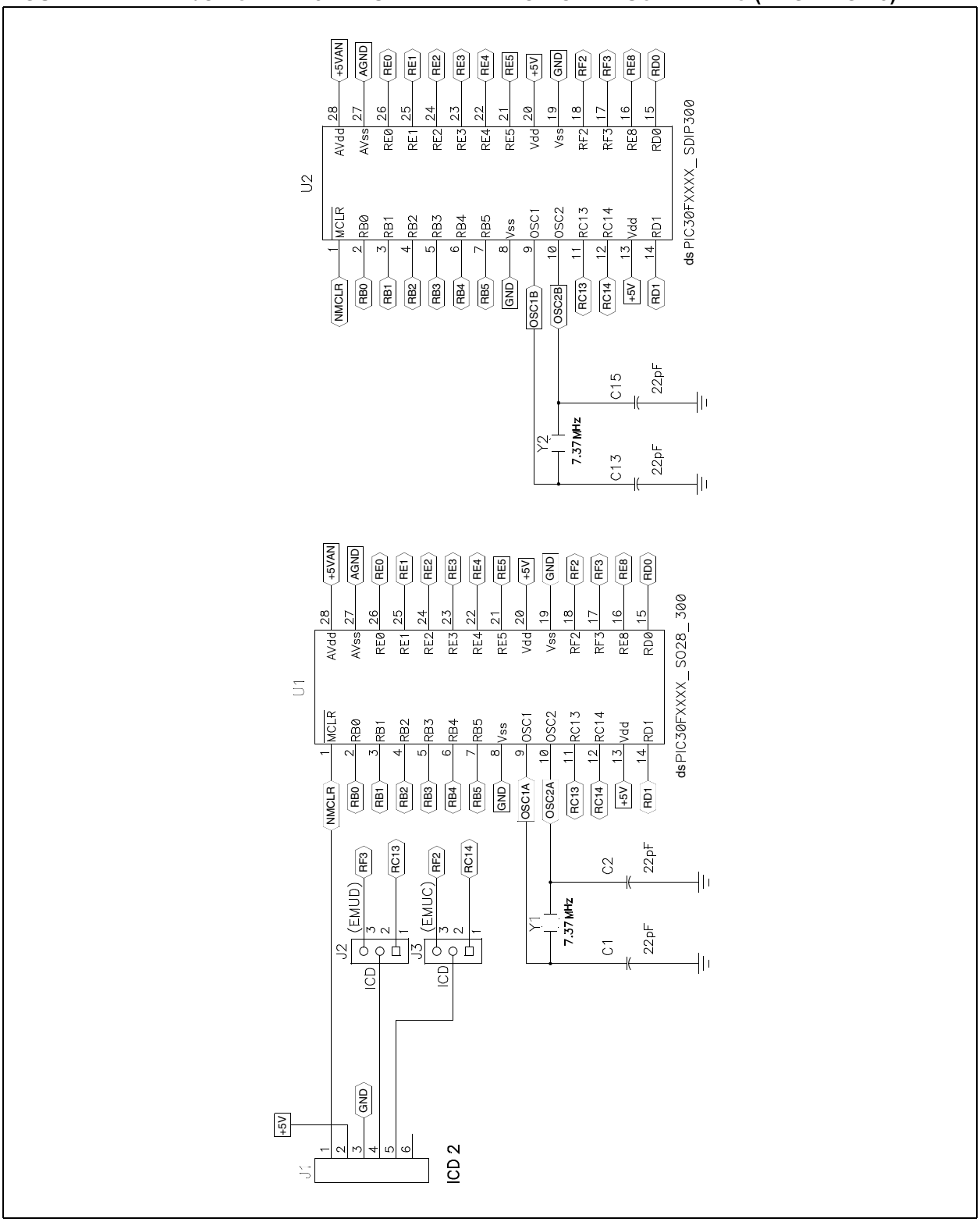


FIGURE A-3: dsPICDEM 28-PIN STARTER DEMO BOARD SCHEMATIC (PAGE 2 OF 3)

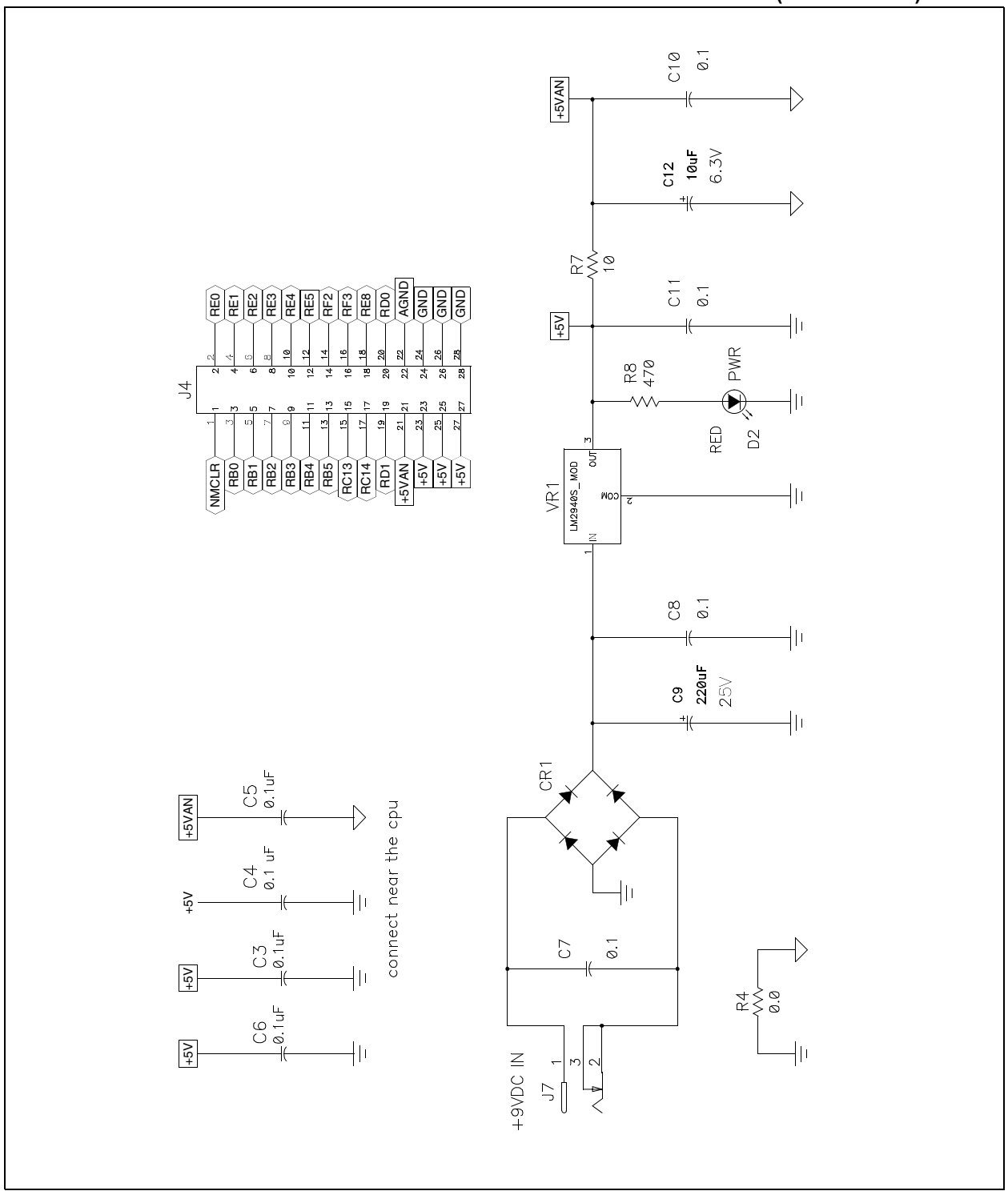
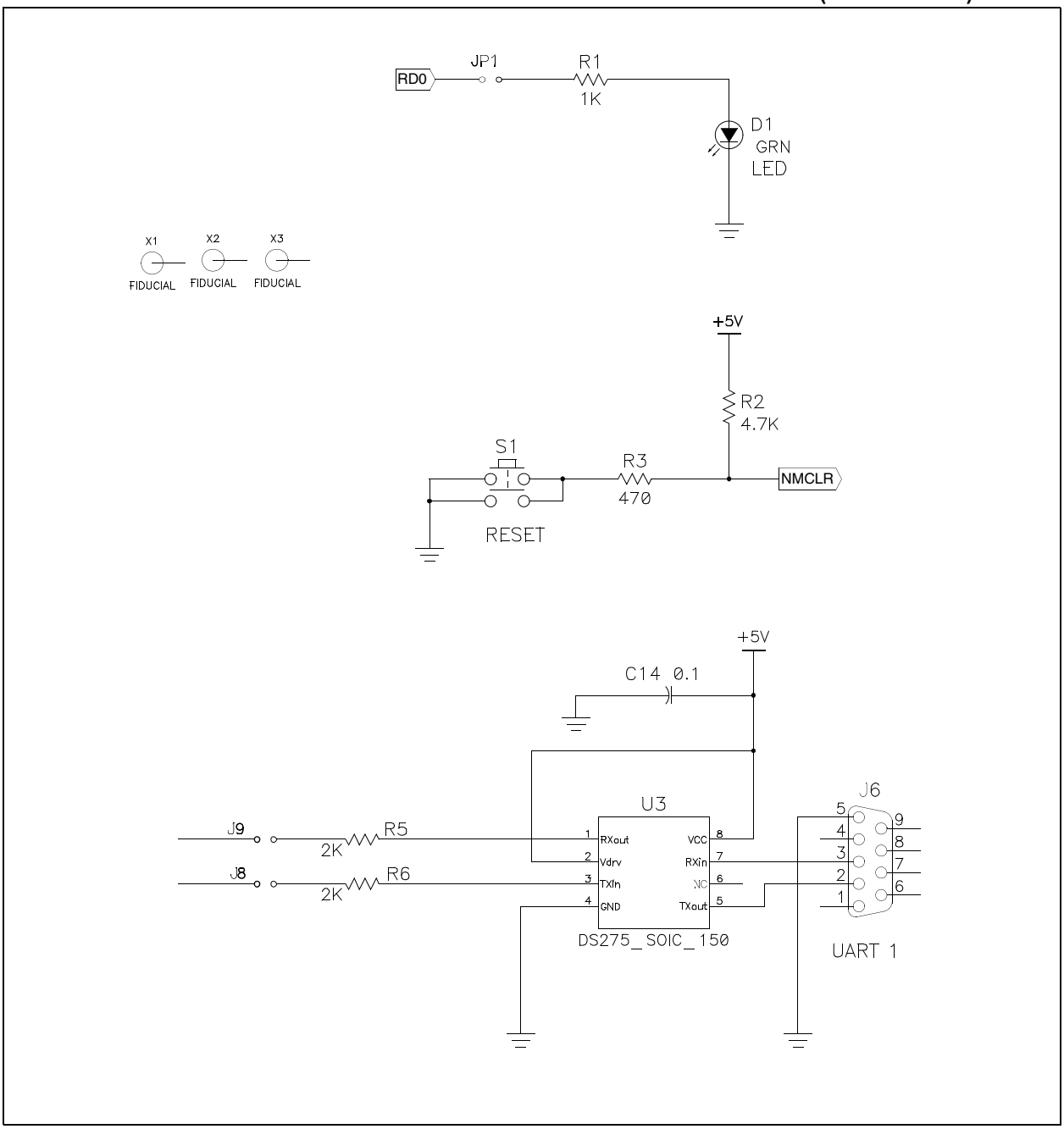


FIGURE A-4: dsPICDEM 28-PIN STARTER DEMO BOARD SCHEMATIC (PAGE 3 OF 3)



Appendix B. Demonstration Source Code

B.1 DEMONSTRATION SOURCE CODE

```
;*************************************************************************
;
;   Date                    :   1/12/2004
;   File Version            :   1.00
;
;   Other Files Required    :   p30F2010.gld, p30f2010.inc
;   Tools Used              :   MPLAB IDE : 6.41
;
;*************************************************************************
;
;   Notes:
;
;   This demonstration program supports the
;   dsPICDEM 28-pin Starter Demo Board
;
;*************************************************************************

        .equ __30F2010, 1
        .include "p30f2010.inc"

;.........................................................................
;   Configuration bits:
;.........................................................................

        config __FOSC, CSW_FSCM_OFF & XT_PLL4 ;Turn off clock switching and
                                              ;fail-safe clock monitoring and
                                              ;use a crystal oscillator with
                                              ;the 4x PLL

        config __FWDT, WDT_OFF                ;Turn off Watchdog Timer

        config __FBORPOR, PBOR_ON & BORV_27 & PWRT_16 & MCLR_EN
                                              ;Set Brown-out Reset voltage and
                                              ;set Power-up Timer to 16 msecs

        config __FGS, CODE_PROT_OFF           ;Set Code Protection Off for the
                                              ;General Segment
```

```
;.....................................................................
;   Global Declarations:
;.....................................................................

        .global __reset                     ;Label for first line of code.

        .global __T1Interrupt               ;Declare Timer 1 ISR name global

;.....................................................................
;   Unintialized storage in data space
;.....................................................................

        .section .bss

Count:.space 2

;.....................................................................
;   Constants stored in Program space
;.....................................................................

        .section .asciitext, "x"
        .palign 2                                   ;Align next word stored in
                                                    ;Program space to an address
                                                    ;that is a multiple of 2
Message:
        .ascii"dsPICDEM 28-pin Starter Demo Board\r\n\0"

;.....................................................................
;   Code Section in Program Memory
;.....................................................................

.text                           ;Start of Code section
__reset:
        MOV     #__SP_init, W15         ;Initalize the Stack Pointer
        MOV     #__SPLIM_init, W0       ;Initialize the Stack Pointer Limit
                                        ;Register
        MOV     W0, SPLIM
        NOP                             ;Add NOP to follow SPLIM initialization

        CALL    _wreg_init              ;Call _wreg_init subroutine

        clr     Count                   ;Initialize the count variable used in
                                        ;ISR

InitPorts:

        mov     #0x003F,W0              ; Setup all analog pins for digital mode
        mov     W0,ADPCFG               ; by writing ADPCFG register

        clr     LATB                    ; Clear all I/O port registers.
        clr     LATC
        clr     LATD
        clr     LATE
        clr     TRISB                   ; Make all ports outputs except RF2, RF3
        clr     TRISC                   ; which will be used for UART
        clr     TRISD
        clr     TRISE
        mov     0x000C,W0
        mov     W0,TRISF
```



```
SetupUART:

        mov     #0x8000,W0              ; Setup UART control registers
        mov     W0,U1MODE
        clr     U1STA
        mov     #192,W0                 ; Baud rate value for 2400 baud
                                        ; at Fcy = 7.37 MHz
        mov     W0,U1BRG
        bset    U1STA,#UTXEN            ; Enable UART transmit


        bset CORCON,#PSV            ; Enable program space visibility

    ; Load PSV page register and setup W1 as a pointer to the
    ; ASCII text string stored in program memory

        mov     #psvpage(Message),W0
        mov     W0,PSVPAG
        mov     #psvoffset(Message),W1

    ; The next block of code writes the ascii string stored in
    ; program memory to to the UART.  The loop repeats until a
    ; null character is found.

MsgLoop:
        mov.b   [W1++],W0           ; Read the data from P.M. using PSV
        cp0.b   W0                  ; Is the character 0?
        bra     Z,MsgDone           ; If the character is 0, we're done.
        call    WriteChar           ; If not, write the character to UART
        bra     MsgLoop             ; Go back to the top of the loop.

MsgDone:

        mov     #54084,W0           ; Gives 1/19th second period with 7.38 MHz
        mov     W0,PR1              ; clock and 1:8 prescaler
        bset    T1CON,#TCKPS0       ; 1:8 timer prescaler
        bset    T1CON,#TON          ; Turn on the timer
        bclr    IFS0,#T1IF
        bset    IEC0,#T1IE

;.....................................................................
;   Software main loop
;.....................................................................

; The main loop looks for incoming characters on the UART and echoes
; them back to the sender.

Loop:   clrwdt                      ; Clear the watchdog timer.

        btss    U1STA,#URXDA        ; Is receive data available?
        bra     Loop                ; If not, go back to top of loop.

        mov     U1RXREG,W0          ; If so, get the data from the RX buffer
        call    WriteChar           ; Transmit the character back.

        bra     Loop                ; Go back to top of loop.
```

```
;.....................................................................
;   Subroutine: Initialization of W registers to 0x0000
;.....................................................................

_wreg_init:
        CLR     W0
        MOV     W0, W14
        REPEAT  #12
        MOV     W0, [++W14]
        CLR     W14
        RETURN

;.....................................................................
;   Subroutine: Writes character to UART
;.....................................................................

WriteChar:
        btss    U1STA,#TRMT         ; Wait for present transmission to complete
        bra     WriteChar
        mov     W0,U1TXREG          ; Write the character to TX buffer
        return

;.....................................................................
;Timer 1 Interrupt Service Routine
;.....................................................................

; The Timer1 interrupt service routine toggles all of the dsPIC30F2010 I/O
; lines in succession.  Each time the interrupt occurs, the last I/O line
; is turned off and the next one is turned on.  A count value is maintained
; and a computed jump table is used to determine which I/O pin to activate.

__T1Interrupt:
        BCLR    IFS0, #T1IF          ; Clear the timer interrupt flag Status
        push.d  W0                  ; Save W0 and W1 on stack

        inc     Count               ; Increment the count variable
        mov     #19,W0
        cp      Count               ; Is the count value 19?
        btsc    SR,#Z
        clr     Count               ; If so, reset count to 0.

        call    IO_Pin_Scan         ; Call the routine that writes the I/O pins.

        pop.d   W0                  ; Restore W0 and W1
        retfie                      ; Return from Interrupt Service routine

;.....................................................................
;Subroutine for toggling I/O pins
;.....................................................................

; This routine is called after every Timer1 interrupt.  The count value
; adjusted in the Timer1 ISR is used to form a computed jump address. The jump
; address determines which I/O pin to turn on.

IO_Pin_Scan:

        mov     Count,W0
        mov     #3,W1

        mul.uu  W0,W1,W0

        bra     W0
```



```
; Count = 0
bclr    LATE,#0
bset    LATB,#0
return

; Count = 1
bclr    LATB,#0
bset    LATB,#1
return

; Count = 2
bclr    LATB,#1
bset    LATB,#2
return

; Count = 3
bclr    LATB,#2
bset    LATB,#3
return

; Count = 4
bclr    LATB,#3
bset    LATB,#4
return

; Count = 5
bclr    LATB,#4
bset    LATB,#5
return

; Count = 6
bclr    LATB,#5
bset    LATC,#13
return

; Count = 7
bclr    LATC,#13
bset    LATC,#14
return

; Count = 8
bclr    LATC,#14
bset    LATD,#1
return

; Count = 9
bclr    LATD,#1
bset    LATD,#0
return

; Count = 10
bclr    LATD,#0
bset    LATE,#8
return

; Count = 11
bclr    LATE,#8
bset    LATF,#3
return

; Count = 12
bclr    LATF,#3
bset    LATF,#2
return
```

```
        ; Count = 13
        bclr    LATF,#2
        bset    LATE,#5
        return

        ; Count = 14
        bclr    LATE,#5
        bset    LATE,#4
        return

        ; Count = 15
        bclr    LATE,#4
        bset    LATE,#3
        return

        ; Count = 16
        bclr    LATE,#3
        bset    LATE,#2
        return

        ; Count = 17
        bclr    LATE,#2
        bset    LATE,#1
        return

        ; Count = 18
        bclr    LATE,#1
        bset    LATE,#0
        return

;--------End of All Code Sections---------------------------------------------

        .end                            ;End of program code in this file
```


Index

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http:\\support.microchip.com
Web Address:
www.microchip.com

Atlanta
Alpharetta, GA
Tel: 770-640-0034
Fax: 770-640-0307

Boston
Westford, MA
Tel: 978-692-3848
Fax: 978-692-3821

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

San Jose
Mountain View, CA
Tel: 650-215-1444
Fax: 650-961-0286

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8676-6200
Fax: 86-28-8676-6599

China - Fuzhou
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Shunde
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

China - Qingdao
Tel: 86-532-502-7355
Fax: 86-532-502-7205

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-2229-0061
Fax: 91-80-2229-0062

India - New Delhi
Tel: 91-11-5160-8632
Fax: 91-11-5160-8632

Japan - Kanagawa
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Taiwan - Hsinchu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

EUROPE

Austria - Weis
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark - Ballerup
Tel: 45-4420-9895
Fax: 45-4420-9910

France - Massy
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Ismaning
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

England - Berkshire
Tel: 44-118-921-5869
Fax: 44-118-921-5820

10/20/04