Getting Started with PIC32 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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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 devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. 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, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, rfPIC and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, Linear Active Thermistor, MXDEV, MXLAB, 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, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, PICkit, PICDEM, PICDEM.net, PICtail, PIC^{32} logo, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, Select Mode, Total Endurance, UNI/O, WiperLock and ZENA 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.

♻ Printed on recycled paper.

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

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

Table of Contents

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 “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 online help. Select the Help menu, and then Topics to open a list of available online help files.

INTRODUCTION

This document is intended to enable new PIC32 users to get a basic hardware and software understanding of the PIC32 architecture and quickly get started with software development for PIC32 products.

This document is primarily written for anyone wanting to get overview knowledge of PIC32 microcontrollers and associated technical collateral provided by Microchip and its third party tools partners. A basic understanding of embedded systems development and 'C' programming knowledge is required, if you are planning to follow step-by-step instructions provided later in this document.

This chapter contains general information that will be useful to know before using the PIC32. Items discussed in this chapter include:

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

DOCUMENT LAYOUT

This document is divided into 6 sections. Sections 1 through 4 provide an overview of PIC32 products. Section 5 provides an overview of PIC32 development tools. Section 6 describes the procedure to setup a development environment and debug a sample application. The manual layout is as follows:

- **Chapter 1. PIC32 Features**
- **Chapter 2. PIC32 Product Family**
- **Chapter 3. PIC32 Architecture**
- **Chapter 4. PIC32 Tools**
- **Chapter 5. Step-by-Step Procedures to Setup, Build, and Run a Demo Project**
- **Chapter 6. Technical Support Resources**

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
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 New font:		
Plain Courier New	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 New	A variable argument	*`file`*`.o`, where *`file`* can be any valid filename
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)`
		`{ ...`
		`}`

RECOMMENDED READING

This user's guide describes how to use PIC32. Other useful documents are listed below. The following Microchip documents are available and recommended as supplemental reference resources.

Readme for the PIC32MX

For the latest information on using PIC32 microcontrollers, read the file (an ASCII text file) at the root level of the CD included in the PIC32 Starter Kit. The file generally contains the most current update information, as well as any issues that may not have been available when this document was published.

Readme Files

For the latest information on using other tools, read the tool-specific readme files in the `Readmes` subdirectory of the PIC32MX Starter Kit installation directory. The files contain update information, as well as any issues that may not have been available when this document was published.

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 C32 C compilers; 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, PICkit® 1 and PICkit® 2 development programmers.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

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

DOCUMENT REVISION HISTORY

Revision A (October 2007)

- Initial Release of this Document.

Revision B (March 2008)

- Updates

NOTES:

Chapter 1. PIC32 Features

1.1 INTRODUCTION

The PIC32 is a 32-bit family of general purpose microcontrollers from Microchip Technology. It offers 80+ DMIPS performance with a wide variety of on-chip peripherals. It employs industry leading M4K MIPS32 core from MIPS Technologies, Inc. All members in the PIC32 family use programming interface similar to other Microchip PIC® microcontrollers. In addition, PIC32 microcontrollers are pin-to-pin compatible with the PIC24FJ128GA family of 16-bit microcontrollers.

1.2 HIGHLIGHTS

The PIC32 family offers a number of features to enable a wide variety of applications. The following subsections list all the key features grouped in major categories.

Performance:

- Up to 80 MHz, MIPS M4K 32-bit core with 5 stage pipeline
- High-performance hardware multiply/divide unit – 1 multiply per clock
- Programmable user and kernel memory partition for enhanced application stability
- Multiple register sets for reduced interrupt latency
- Hardware assisted single-cycle register bits manipulations
- 128-bit wide Flash memory to shorten individual instruction fetch time
- 256 bytes of high-speed cache memory with instruction and ROM data prefetch buffer
- Available DMA controller with integrated CRC calculation and pattern-based transfer termination
- Includes USB On-The-Go controller for USB device, host, or dual-role applications
- USB controller has own dedicated DMA interface

Power Management:

- 2.3 to 3.6V operation
- Full-speed operation over entire voltage range
- Various low-power modes including RUN, IDLE and SLEEP
- I/O transfers via DMA in IDLE mode
- Programmable peripheral clock
- Individual peripheral ON/OFF control and operation during IDLE mode
- Multiple clock sources

Scalability:

- Industry known MIPS32-compatible M4K CPU core with 5 stage pipeline
- Large family of devices with Flash memory options from 32 KB to 512 KB
- Pin compatible with 64/100 pin PIC24FJXXXGA family of 16-bit microcontrollers

Ease of Use:

- PIC® microcontroller “look and feel” peripherals
- Standard MPLAB® tool-suite – MPLAB IDE, MPLAB C32 C Compiler, MPLAB REAL ICE™, and MPLAB ICD 2.
- Software Peripheral Libraries compatible with those for Microchip 16-bit microcontrollers
- Microchip developed middleware modules such as TCP/IP and 16-bit file system

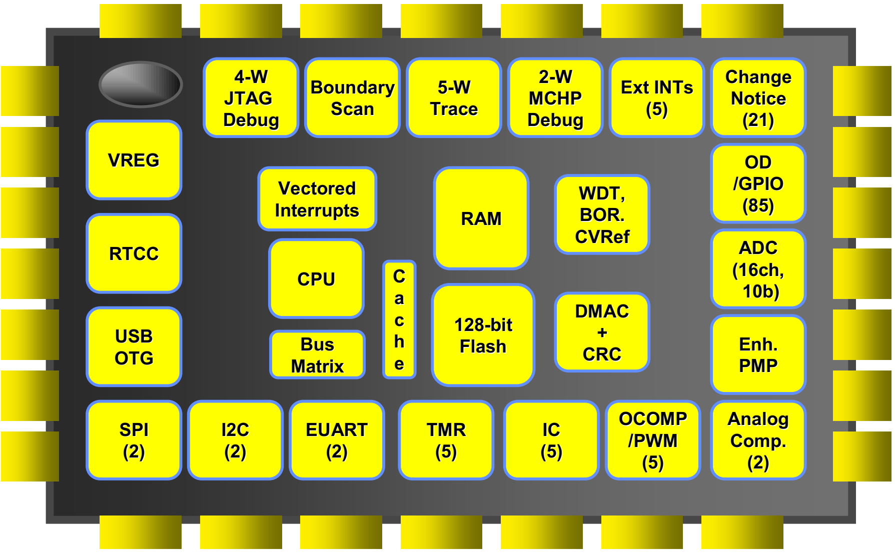
Chapter 2. PIC32 Product Family

2.1 INTRODUCTION

The PIC32 family includes scalable devices ranging from 32KB to 512KB of Flash memory. Also a rich set of peripherals – Five timers, 16 channels of 10-bit A/D Converters and communication interfaces: SPI, I^{2}C™ and UART.

Please consult the "*PIC32MX Family Data Sheet*" (DS61143) for a complete list of family variants, core and peripheral characteristics.

FIGURE 2-1: PIC32 MCU MODULES



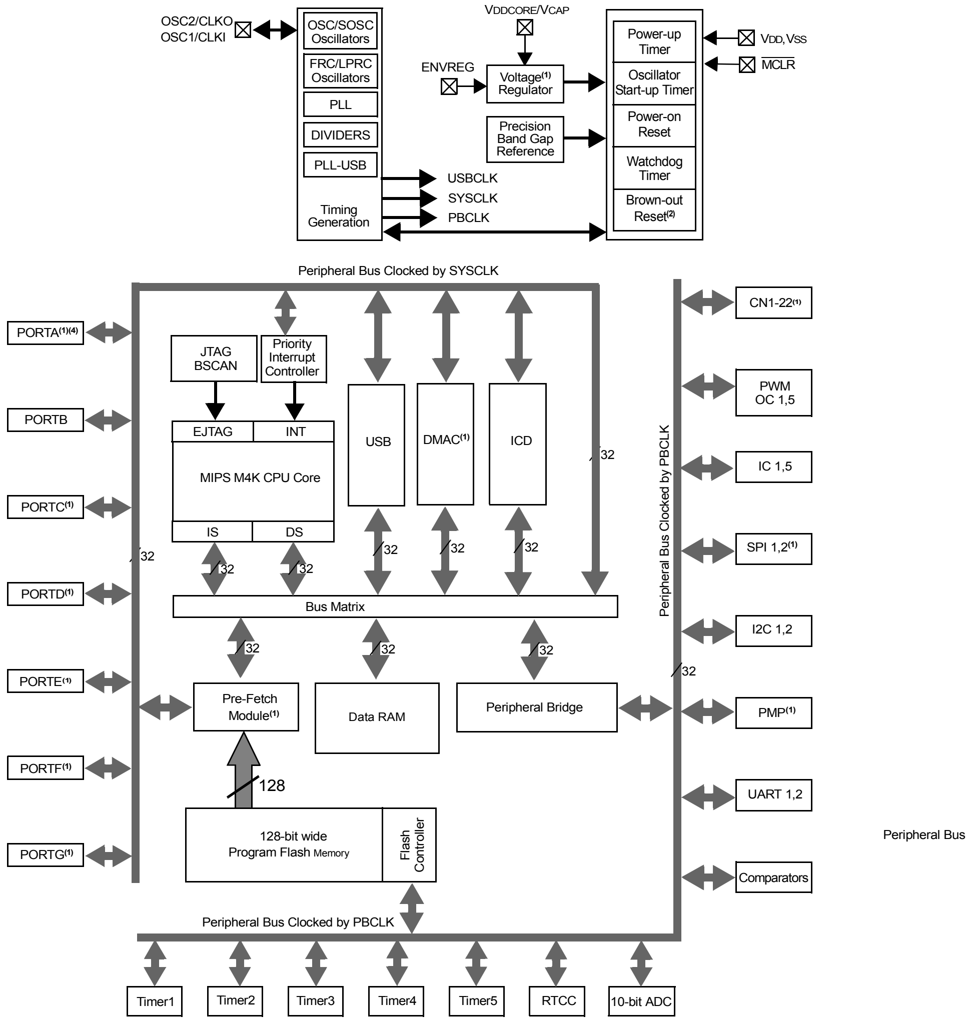
NOTES:

Chapter 3. PIC32 Architecture

3.1 INTRODUCTION

The PIC32 family of MCUs combines the MIPS M4K core together with powerful peripherals and embedded Flash and RAM memory to address a wide range of applications.

FIGURE 3-1: PIC32 BLOCK DIAGRAM



Note 1: Not all pins or features are implemented on all device pinout configurations. See Table 1-4 for I/O port pin descriptions.

2: Some features are not available on certain device variants.

3: BOR functionality is provided when the on-board voltage regulator is enabled.

4: PORTA is not present on 64 pin devices

Processor core:

- MIPS M4K with 5-stage pipeline
- MIPS32-compatible Release 2 Instruction Set
- MIPS16e™ Code Compression to improve code density by up to 40%
- GPR shadow registers to minimize latency for interrupt handlers
- Bit field manipulation instructions
- High-performance Multiply/Divide Unit:
  - Maximum issue rate of one 32x16 multiply per clock
  - Maximum issue rate of one 32x32 multiply every other clock
- Static implementation: minimum operating frequency 0 MHz
- 2.3 to 3.6V operation with full speed over entire range
- Low-power modes including RUN, IDLE, and SLEEP

Memory:

- Unified 4GB virtual memory space
- Fixed Memory Mapping Translation (FMT) mechanism
- Flexible partitioning into kernel and user accessible memory segments for increased application stability

Pre Fetch Cache:

- 16 lines, each 128-bit wide, instruction Prefetch buffer
- Ability to load and lock lines – useful to create SW breakpoints in Flash and minimize interrupt latency

Interrupt Controller:

- Fully programmable interrupt controller with Single or Multi vector mode, supporting up to 95 IRQs.
- Multiple priorities and subpriorities for each vector
- Highest priority interrupt has dedicated register set for reduced interrupt latency

DMA Controller:

- Up to 4 independent channels
- Memory-to-Memory, Memory-to-Peripheral, and Peripheral-to-Memory transfers
- Programmable trigger from any IRQ
- Chainable channels, stop on match detection, Auto-Enable mode
- Data transfers can occur while the core is in IDLE mode
- Integrated programmable CRC engine: calculates on the fly while the data is transferred.

Enhanced Parallel Master Port:

- 8- and 16-bit data interface
- Up to 16-bit address lines, expandable using GPIO lines
- 2 Chip Select lines

Communication channels:

- USB 2.0 compliant (FS. 12 Mbps), OTG, Host and Device-only capable
- 2 enhanced UART channels with hardware IrDA®
- 2 Master/Slave/Frame mode SPI channels
- 2 Master/Slave I^{2}C channels, 10/7 bits mode addressing, broadcast capable

Analog-to-Digital Converters:

- Up to 16 Channels, each 10-bit resolution ADCs
- Up to 500+ kilo-samples per second (ksps) conversion speed
- Software selectable Internal or External voltage reference
- Automatic Channel Scan mode
- Selectable conversion trigger source
- 16 word conversion result buffer
- Selectable Buffer Fill modes
- Eight result alignment options
- Operation during CPU Sleep mode

Timers:

- 5 16-bit timer/counter with the ability to form up to (2) 32-bit timer/counters
- Software-selectable internal or external clock source
- Asynchronous timer/counter with built-in oscillator
- Programmable interrupt generation and priority
- Gated external pulse counter
- Software-selectable prescalers.
- Operational during CPU Sleep mode

Core Timer:

- 32 bit timer in CPU for implementing a timer interrupt function.

RTCC (Real-Time Clock and Calendar):

- Time with hours, minutes and seconds
- Calendar with weekday, date, month and year
- Leap year detection
- Highly configurable alarm
- Calibration of up to 260 ppm of crystal error

Debug and Programming:

- 6 instructions and 2 data breakpoints
- 2 complex breakpoint logic blocks with qualified/primed breakpoint triggers, Pass counters, and stopwatch timers.
- 4-wire EJTAG and 2-wire Microchip interface
- 2-wire Microchip Interface:
  - 6 real-time read/write capture logic blocks
  - Read/write access to all data RAM and SFRs without stopping CPU
- Instruction Trace Port:
  - 5-Wire, nonintrusive trace port
  - Triggered by complex breakpoint logic block

GPIO:

- 5V tolerant inputs
- Individual output pin open-drain enable/disable
- Individual input pin weak pull-up enable/disable
- Monitor selective inputs and generate interrupt on mismatch condition

Note:	Input pin weak pull-up and interrupt on mismatch features are available only on select I/O pins.

Chapter 4. PIC32 Tools

4.1 INTRODUCTION

The PIC32 microcontrollers are supported by the MPLAB Integrated Development Environment and its full range of hardware and software tools.

4.2 HIGHLIGHTS

Items discussed in this chapter are:

- MPLAB IDE
- MPLAB C32 C Compiler
- Peripheral Libraries
- Software Solutions
- Demonstration, Development and Evaluation Boards
- Technical Documentation

4.3 MPLAB IDE

The Microchip MPLAB IDE is a FREE development toolsuite for Microsoft Windows® that contains:

- A single graphical interface to all debugging tools:
  - Simulator
  - Programmers
  - Emulators
  - In-Circuit Debuggers
- A full-featured editor with color-coded context
- A project manager
- Customizable data windows with in-place editable contents
- High-level source code debugging
- Extensive on-line help

The MPLAB IDE provides integrated debugging and programming facilities using any of the following probes connected to the target processor/Development Board:

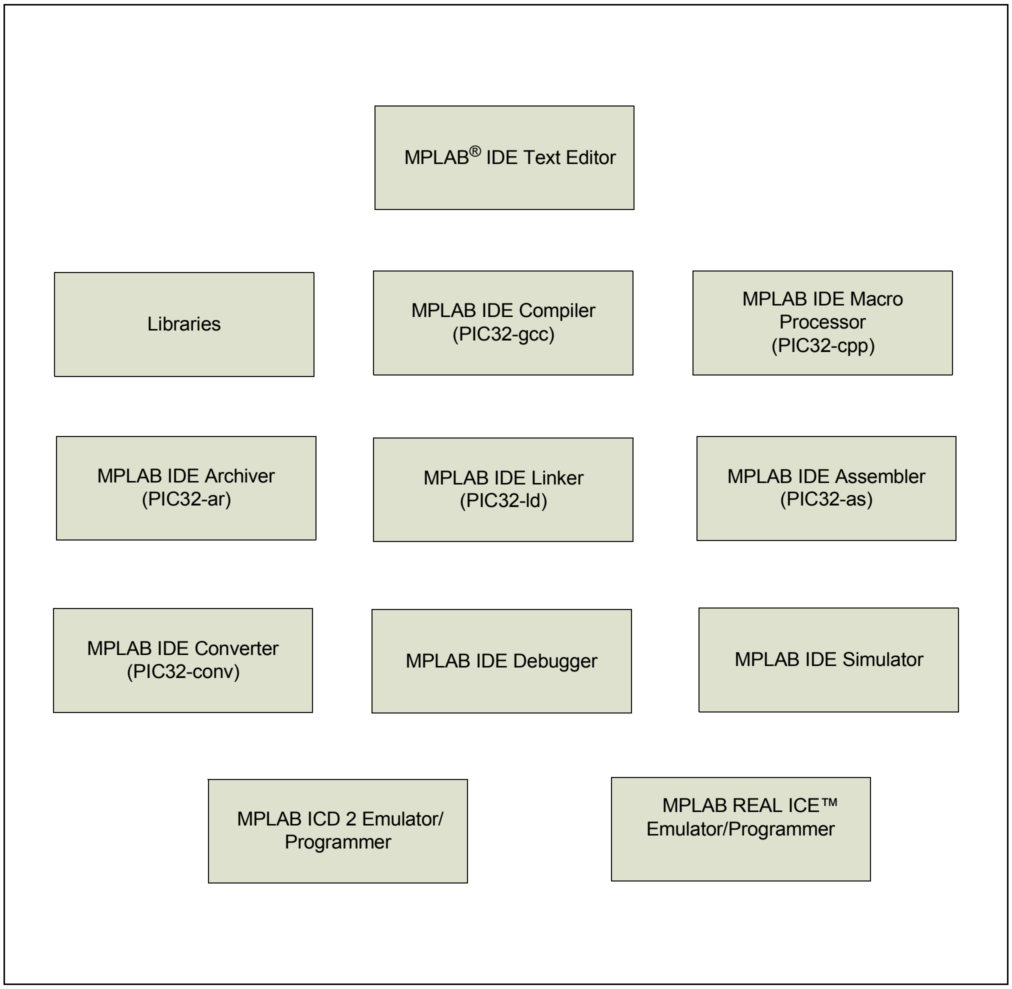
- MPLAB ICD 2 In-Circuit Debugger: a low-cost, run-time development tool.
- MPLAB REAL ICE In-Circuit Emulator: a high speed in-circuit emulator with hardware and software trace capabilities.
- FS2 JTAG probe: Developed by First Silicon Solutions, this probe uses 4-wire EJTAG interface to debug and program the PIC32 microcontrollers.
- MPLAB PM3 Device Programmer: a Microchip universal device programmer suitable for development and manufacturing purposes.
- MPLAB SIM32 Device Simulator: Accurately simulates many PIC32 peripherals and the CPU in selectable cycle and clock-accurate modes

4.4 MPLAB C32 C COMPILER

The MPLAB C32 C Compiler package allows you to build your applications and contains the following key software components (see Figure 4-1).

- **PIC32-gcc** Compiler: complete ANSI C compiler with powerful integration capabilities and efficient code optimization. The compiler provides symbol information that is used by the MPLAB IDE debugger.
- **PIC32-gpp** Macro Processor: used automatically by the compiler to transform the program before compilation.
- **PIC32-as** Assembler: full-featured macro assembler.
- **PIC32-ld** Object Linker: links relocatable objects created by the assembler/compiler with those from precompiled libraries.
- **PIC32-ar** Archiver and Librarian: manages the creation and modification of library files of precompiled code.
- **PIC32-conv**: converts ELF executable files into ASCII or binary format, suitable for downloading onto a PROM programmer or evaluation board.

FIGURE 4-1: MPLAB® IDE DEVELOPMENT TOOLS COMPONENTS


MPLAB® IDE Text Editor
Libraries
MPLAB IDE Compiler (PIC32-gcc)
MPLAB IDE Macro Processor (PIC32-cpp)
MPLAB IDE Archiver (PIC32-ar)
MPLAB IDE Linker (PIC32-ld)
MPLAB IDE Assembler (PIC32-as)
MPLAB IDE Converter (PIC32-conv)
MPLAB IDE Debugger
MPLAB IDE Simulator
MPLAB ICD 2 Emulator/ Programmer
MPLAB REAL ICE™ Emulator/Programmer

4.5 PERIPHERAL LIBRARIES

PIC32 MCUs integrate a large number of on-chip high-performance peripherals.

To accelerate the usage of these peripherals, the MPLAB C32 compiler for PIC32 includes software peripheral libraries compatible with the 16-bit Microchip MCUs. The peripheral libraries are distributed in source and object format along with a detailed API description document. Software applications using MPLAB C32 compiler may call peripheral library functions by simply including the appropriate header file in their source files – the MPLAB C32 compiler has built-in knowledge of library header and archive files.

The peripheral include files are located in `C:\Program Files\Microchip\MPLAB C32\pic32mx\include\peripheral` and the full source code is located in `C:\Program Files\Microchip\MPLAB C32\pic32-libs\peripheral` location in corresponding peripheral sub-directory.

The Peripheral Library contains following include files:

TABLE 4-1: PERIPHERAL LIBRARY

Peripheral	Header File	Description
ADC	`adc10.h`	Library for the Analog-to-Digital converters support
Memory Org	`bmx.h`	Library for configuring the Bus Matrix internal buses, priorities and memory layout
Analog Comparators	`cmp.h`	Library for the Comparator modules support
Comparator Voltage Reference	`cvref.h`	Library for the Comparators Voltage reference support
DMA Controller	`dma.h`	Library for the DMA Controller and CRC module support
Nonvolatile (Flash) Memory	`nvm.h`	Library providing access to Flash erase/program functions
I^{2}C™	`i2c.h`	Library for I^{2}C™ module support
Input Capture	`incap.h`	Library for the Input Capture module suport
Interrupts	`int.h`	Library for the vectored Interrupt Controller support
Lock	`lock.h`	Library for the system lock and unlock support
Output Compare	`outcompare.h`	Library for Output Compare module support
Oscillator	`osc.h`	Library for Oscillators support
Prefetch Cache	`pcache.h`	Library for prefetch-cache module support
Power Modes	`power.h`	Library for Sleep and Idle operationsl power modes support
Parallel Master Port	`pmp.h`	Library for Parallel Master Port support
GPIO	`ports.h`	Library for configuring the I/O ports, reading or writing I/O data
Reset	`reset.h`	Library for Reset control and status
RTCC	`rtcc.h`	Library for Real-Time Clock and Calendar support
SYSTEM	`system.h`	Library for system level operations
SPI	`spi.h`	Library for Serial Peripheral Interface support
Timer	`timer.h`	Library to support 16 and 32-bit timers
UART	`uart.h`	Library supporting the enhanced UART module
Watchdog	`wdt.h`	Library to support the Watchdog Timer module

4.6 SOFTWARE SOLUTIONS

At the time of writing this document, Microchip had not released any drivers. Microchip does intend to develop drivers for key communication peripherals.

4.6.1 Middleware

The following list provides the Microchip middleware components available at the time of writing this document:

- 16-bit file system on SD Memory to support Microsoft MS-DOS file system.
- Two types of TCP/IP stack – 1) Microchip legacy TCP/IP Stack – fully compatible with Microchip 8- and 16-bit Stack, 2) Microchip BSD TCP/IP Stack – A Berkeley socket API stack with many advanced features.
- TCP/IP middleware:
  - FTP server to allow the application running on the target PIC32 machine to be a File Transfer Protocol server.
  - SNMP Agent to monitor PIC32 products using SNMP protocol.
  - Web server to accept HTTP requests from Web browsers clients.
- USB middleware:
  - USB Embedded Host Stack (HID and Mass Storage)
  - USB Device Stack (HID, Mass Storage and CDC class)

4.6.2 3rd Party Tools

Check the PIC32 home page (www.microchip.com/PIC32) to learn about the list of companies with support for PIC32.

4.7 DEMONSTRATION, DEVELOPMENT AND EVALUATION BOARDS

A wide variety of demonstration, development and evaluation boards for PIC32 MCUs allow quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification. These boards support a variety of features, including LEDs, switches, RS-232 interfaces, LCD displays, etc. The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

Currently, the following boards are available:

1. PIC32MX Starter kit (DM320001).
2. Explorer 16 board (DM240001) with PIC32 Plug-in Modules.

4.8 TECHNICAL DOCUMENTATION

The current set of documents that are available for PIC32 MCUs:

- Application Notes:
  - AN833, *“Microchip TCP/IP Stack”* (DS00833)
  - AN1107, *“HTTP Server for the Microchip BSD TCP/IP Stack”* (DS01107)
  - AN1108, *“Microchip TCP/IP Stack with BSD Socket API”* (DS01108)
  - AN1109, *“An SNMP Agent for the Microchip TCP/IP Stack”* (DS01109)
  - AN1111, *“The Microchip FTP Server Using BSD Socket API”* (DS01111)
  - AN1140, “*USB Embedded Host Stack*” (DS01140)
  - AN1141, “*USB Embedded Host Stack Programmer’s Guide*” (DS01141)
  - AN1142, “*USB Mass Storage Class on an Embedded Host*” (DS01142)
  - AN1143, “*USB Generic Client on an Embedded Host*” (DS01143)
  - AN1144, “*USB HID Class on an Embedded Host*” (DS01144)
  - AN1145, “*Using a USB Flash Drive on an Embedded Host*” (DS01145)
  - AN1176, “*USB Devcie Stack for PIC32 Programmer’s Guide*” (DS01176)
  - AN1166, “*USB Generic Function on an Embedded Device*” (DS01166)
  - AN1163, “*USB HID Class on an Embedded Device*” (DS01163)
  - AN1169, “*USB Mass Storage Class on an Embedded Device*” (DS01169)
  - AN1164, “*USB CDC Class on an Embedded Device*” (DS01164)
- Data Sheets:
  - DS61143 – *PIC32MX Family Data Sheet*
- Family Reference Manuals:
  - DS61132 – *PIC32MX Family Reference Manual*
- Code Examples:
  - PIC32 examples available in `C:\ProgramFiles\Microchip\MPLAB C32\examples` and on www.microchip.com/pic32.
- Errata (DS80350, DS80367)
- Migration Documents – available in future
- Design Notes, Tips and Tricks – available in future
- Development Tool Ordering Guide – available in future

Chapter 5. Step-by-Step Procedures to Setup, Build, and Run a Demo Project

5.1 INTRODUCTION

In this step-by-step procedure, the basic concepts of the MPLAB Project Manager, Editor and Debugger will be presented. You will create a simple project and understand the debug capabilities of MPLAB IDE.

No previous MPLAB IDE knowledge is assumed. For complete features set and comprehensive technical details of MPLAB IDE and its components, please visit our web site (www.microchip.com/ide).

5.2 HIGHLIGHTS

Items discussed in this chapter are:

- MPLAB IDE Setup
- Step-by-Step Guide Overview
- Selecting the Device
- Creating the Project
- Setting Up Language Tools
- Naming the Project
- Adding Files to the Project
- Attaching the Debugger
- Building the Project
- Testing the Code

5.3 MPLAB IDE SETUP

5.3.1 Install MPLAB IDE

To install the MPLAB on your system, you could either use the supplied installation CD or download the latest MPLAB IDE from the Microchip web site.

- To install from a CD-ROM, just place the disk into a CD drive and follow the on-screen prompts (you could use Windows Explorer to find and execute the CD-ROM menu, `menu.exe`).
- If the MPLAB IDE was downloaded from the Microchip web site, unzip the file and execute the resulting file to install.

Note: Administrative access will be required to install the MPLAB on a PC.

5.3.2 To uninstall MPLAB IDE

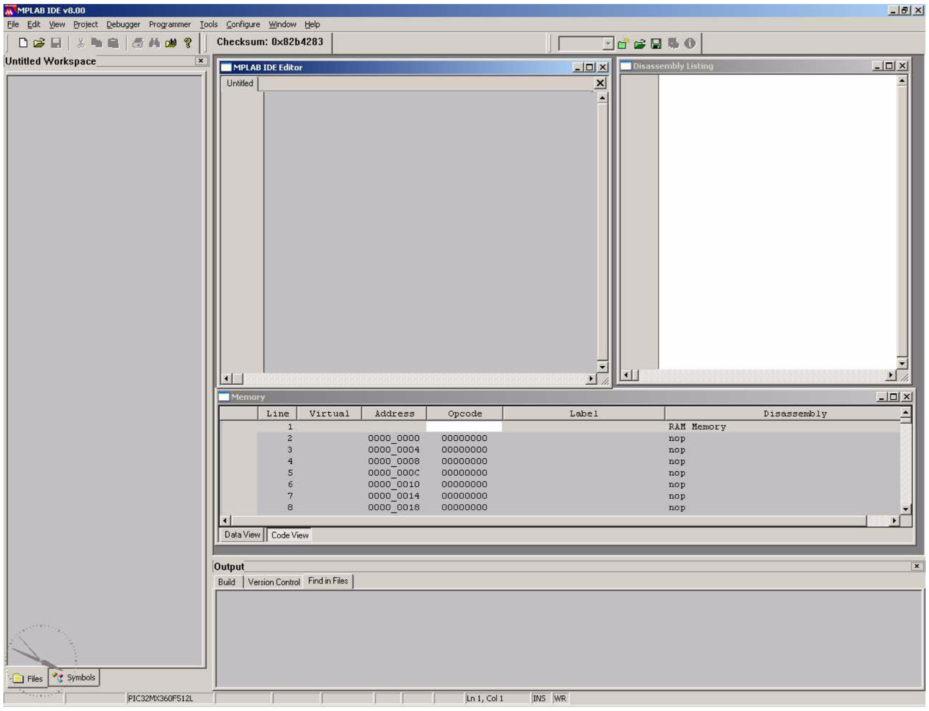
- Select *Start>Settings>Control Panel* to open the control panel.
- Double click on Add/Remove Programs. Find MPLAB IDE and select it.
- Click Change/Remove to remove the program from your system.

Note: Administrative access might be required in order to uninstall the MPLAB.

5.3.3 Running MPLAB IDE

To start MPLAB IDE, double click on the icon installed on the desktop after installation or select *Start>Programs>Microchip>MPLAB IDE vx.xx>MPLAB IDE*. A screen will display the MPLAB IDE logo followed by the MPLAB IDE desktop.

FIGURE 5-1: MPLAB IDE START SCREEN



Line	Virtual	Address	Opcode	Label	Disassembly
1					RAM Memory
2		0000_0000	00000000		nop
3		0000_0004	00000000		nop
4		0000_0008	00000000		nop
5		0000_000C	00000000		nop
6		0000_0010	00000000		nop
7		0000_0014	00000000		nop
8		0000_0018	00000000		nop

5.4 STEP-BY-STEP GUIDE OVERVIEW

To create code that is executable by the PIC32 MCU, source files need to be part of a project. The code can then be built into executable code using selected language tools (assemblers, compilers, linkers, etc.). In MPLAB IDE, the project manager controls this process and will guide us through most of these steps.

All projects will have these basic steps:

- Select Device

The capabilities of MPLAB IDE vary according to which device is selected. Device selection should be completed before starting a project.

- Create Project

MPLAB IDE Project Wizard will be used to create a project.

- Select Language Tools

In the Project Wizard the language tools will be selected. For this tutorial, the PIC32 tools will be used. For other projects, either other Microchip or third party tools might be selected.

- Add Files in Project

We'll add a template file and a linker script to the project.

- Create Code

Some very simple code will be added to the template file to print a “Hello World…” string to a serial console connected to the evaluation board. We will use the UART Peripheral Library provided by Microchip.

- Build Project

The project will be built – causing the source files to be compiled and linked into machine code that can run on the selected PIC32 MCU.

- Test Code

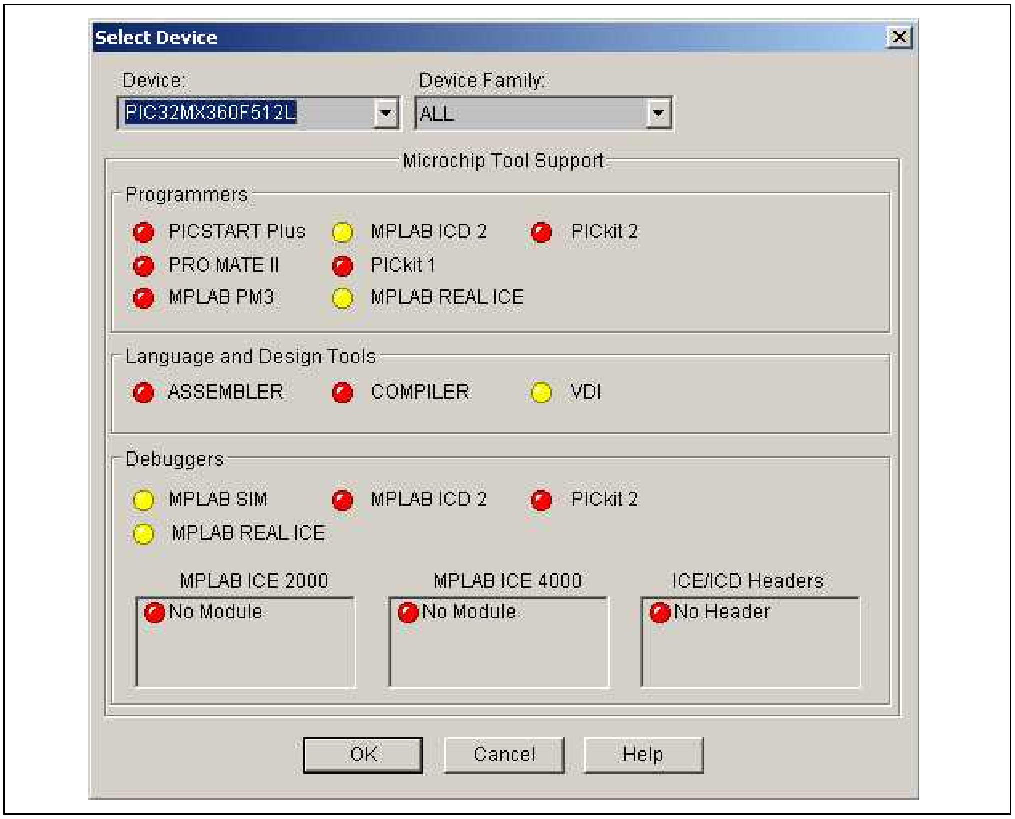
Finally, the code will be tested by running it on the evaluation board.

> **Note:** Some aspects of the user interface will change in future product releases and the screen shots in this guide may not exactly match the appearance of the MPLAB IDE desktop in later releases.

5.5 SELECTING THE DEVICE

Choose *Configure>Select Device* from the top IDE menu. In the Device dialog, select the PIC32 variant from the drop-down list.

FIGURE 5-2: MPLAB IDE SELECT DEVICE



Select Device

Device: PIC32MX360F512L

Device Family: ALL

Microchip Tool Support

Programmers

- 🔴 PICSTART Plus
- 🟡 MPLAB ICD 2
- 🔴 PICkit 2
- 🔴 PRO MATE II
- 🔴 PICkit 1
- 🔴 MPLAB PM3
- 🟡 MPLAB REAL ICE

Language and Design Tools

- 🔴 ASSEMBLER
- 🔴 COMPILER
- 🟡 VDI

Debuggers

- 🟡 MPLAB SIM
- 🔴 MPLAB ICD 2
- 🔴 PICkit 2
- 🟡 MPLAB REAL ICE

MPLAB ICE 2000: 🔴 No Module

MPLAB ICE 4000: 🔴 No Module

ICE/ICD Headers: 🔴 No Header

The “lights” indicate which MPLAB IDE components support this device.

- A green light indicates full support.
- A yellow light indicates preliminary support for an upcoming part by the particular MPLAB IDE tool component. Components with a yellow light instead of a green light are often intended for early adopters of new parts who need quick support and understand that some operations or functions may not be available.
- A red light indicates no support for this device. Support may be forthcoming or inappropriate for the tool.

5.6 CREATING THE PROJECT

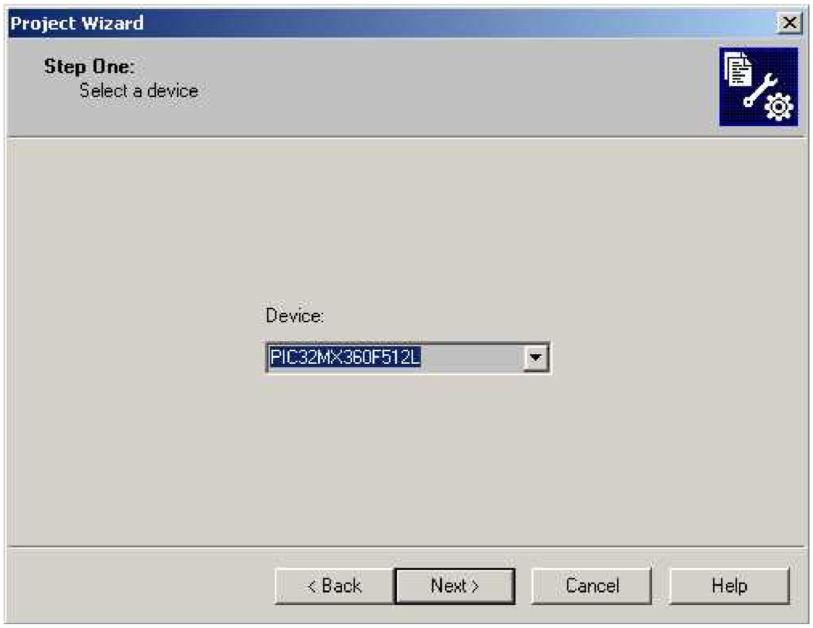
The next step is to create a project using the Project Wizard. A project is the way the files are organized to be compiled, assembled and linked. We will use a single “C” file for this project and a linker script.

Choose *Project>Project Wizard*.

From the Welcome dialog, click on **Next>** to advance.

The Step One dialog allows you to select the device, which we’ve already done. Make sure that it displays the proper PIC32 variant. If not, select the required PIC32 variant from the drop down menu. Click **Next>**.

FIGURE 5-3: MPLAB IDE WIZARD SELECT DEVICE



Step One:
Select a device

Device: PIC32MX360F512L

5.7 SETTING UP LANGUAGE TOOLS

Step Two of the Project Wizard sets up the language tools that are used with this project. Make sure the “Show all installed toolsuites” checkbox is checked. Select Microchip PIC32 C Compiler Toolsuite in the Active Toolsuite list box.

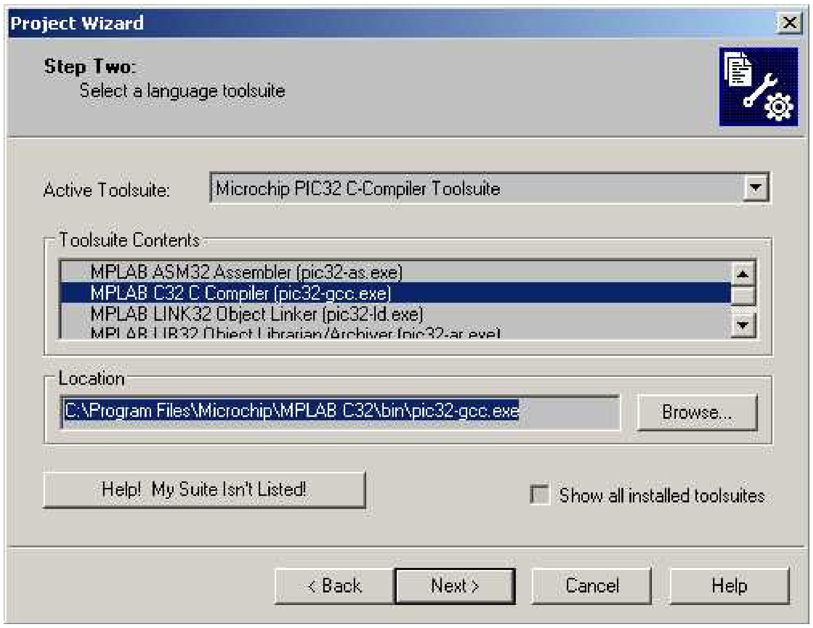
Then “MPLAB PIC32 Assembler (PIC32-as.exe)”, “MPLAB PIC32 C Compiler (PIC32-gcc.exe)”, “MPLAB PIC32 Object Linker (PIC32-ld.exe)”, and “MPLAB PIC32 Archiver (PIC32-ar.exe)”, should be visible in the Toolsuite Contents box. Click on each one to see its location. If MPLAB IDE was installed into the default directory, the paths for these files will be:

- for the MPLAB PIC32 assembler:
  - `C:\Program Files\Microchip\MPLAB IDE\ MPLAB C32\bin\PIC32-as.exe`
- for the MPLAB PIC32 compiler:
  - `C:\Program Files\Microchip\MPLAB IDE\ MPLAB C32\bin\PIC32-gcc.exe`
- for the MPLAB PIC32 Object Linker:
  - `C:\Program Files\Microchip\MPLAB IDE\ MPLAB C32\bin\PIC32-ld.exe`
- for the MPLAB PIC32 Archiver:
  - `C:\Program Files\Microchip\MPLAB IDE\ MPLAB C32\bin\PIC32-ar.exe`

If these paths do not show up correctly, use the **Browse** button to set them to the proper files in the MPLAB IDE subfolders.

When you are finished, click **Next>**.

FIGURE 5-4: MPLAB IDE SELECT LANGUAGE TOOLSUITE



Project Wizard

Step Two:
Select a language toolsuite

Active Toolsuite: Microchip PIC32 C-Compiler Toolsuite

Toolsuite Contents

MPLAB ASM32 Assembler (pic32-as.exe)
MPLAB C32 C Compiler (pic32-gcc.exe)
MPLAB LINK32 Object Linker (pic32-ld.exe)
MPLAB LIB32 Object Librarian/Archiver (pic32-ar.exe)

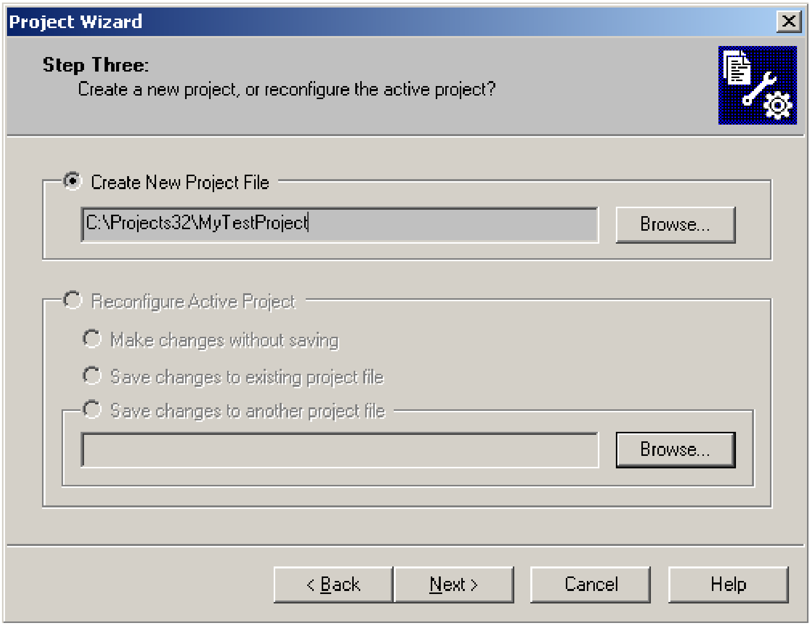
Location: C:\Program Files\Microchip\MPLAB C32\bin\pic32-gcc.exe

□ Show all installed toolsuites

5.8 NAMING THE PROJECT

Step Three of the wizard allows you to name the project and put it into a folder. This sample project will be called MyTestProject. Using the **Browse** button, place the project in a folder named Projects32. Click **Next>**.

FIGURE 5-5: MPLAB IDE NAMING THE PROJECT



Project Wizard

Step Three:
Create a new project, or reconfigure the active project?

● Create New Project File

C:\Projects32\MyTestProject

○ Reconfigure Active Project

○ Make changes without saving

○ Save changes to existing project file

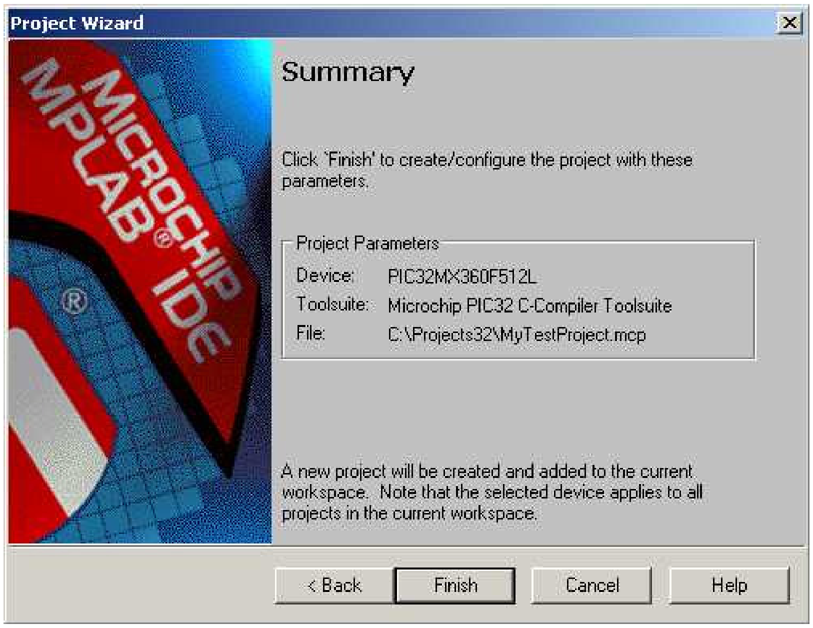
○ Save changes to another project file

5.9 ADDING FILES TO THE PROJECT

Step Four of the Project Wizard allows file selection for the project. This is where we can add existing files to our project. In the current example we don't have a previously created file but we'll create one once we're done with the new project setting.

Just press **Next>** and the following Summary Screen will show up:

FIGURE 5-6: MPLAB IDE SUMMARY SCREEN



Summary

Click `Finish' to create/configure the project with these parameters.

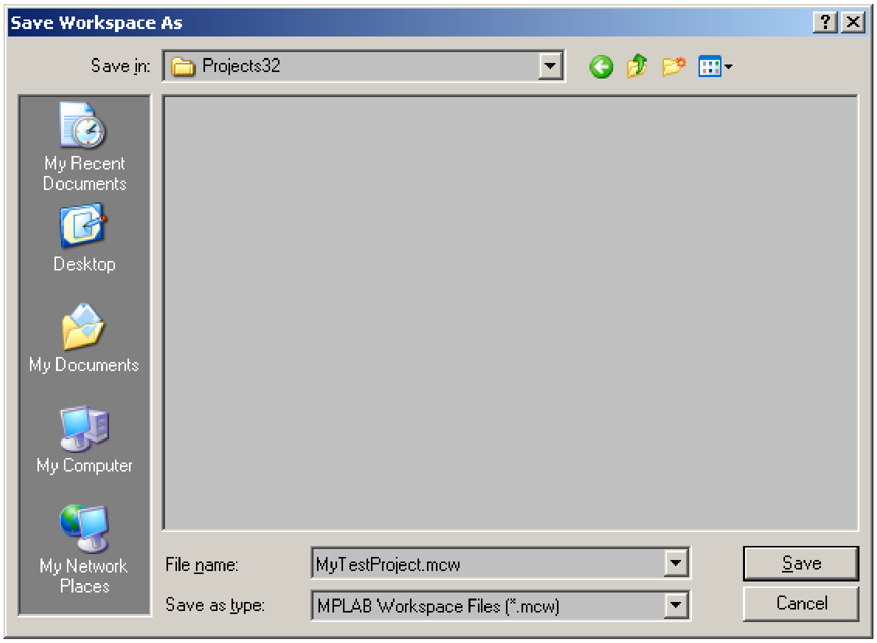
Project Parameters

Device: PIC32MX360F512L
Toolsuite: Microchip PIC32 C-Compiler Toolsuite
File: C:\Projects32\MyTestProject.mcp

A new project will be created and added to the current workspace. Note that the selected device applies to all projects in the current workspace.

Press **Finish** and the workspace dialog shows up:

FIGURE 5-7: MPLAB IDE SAVE WORKSPACE



Save in: Projects32

File name: MyTestProject.mcw

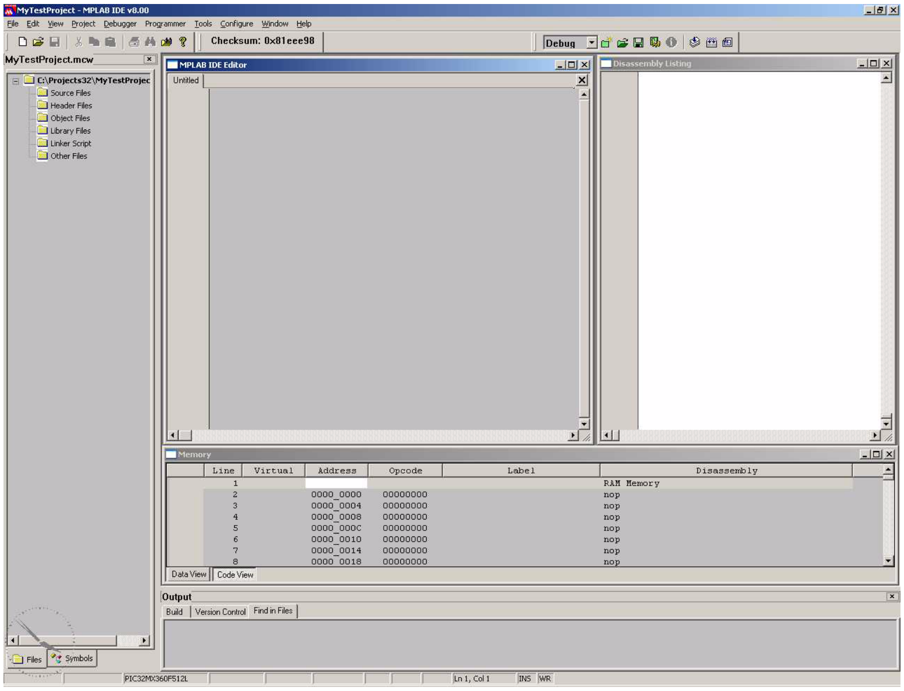
Save as type: MPLAB Workspace Files (*.mcw)

Be sure to save the workspace in the Projects32 directory and name this workspace using the same name as for the project: MyTestProject.

Click **Save**.

The project space is now completely created and it should look like that:

FIGURE 5-8: MPLAB IDE PROJECT SAVE

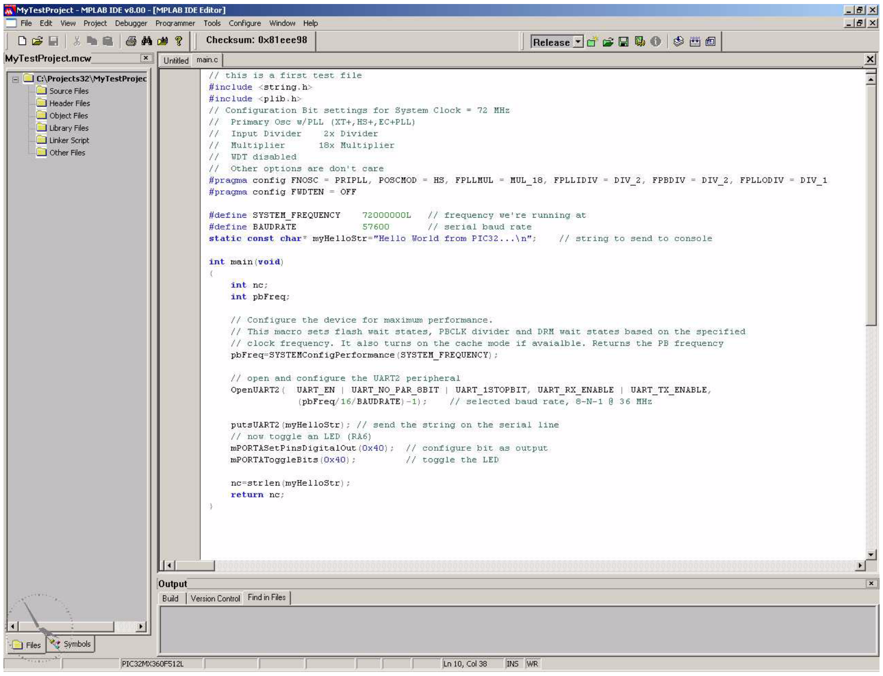


Line	Virtual	Address	Opcode	Label	Disassembly
1					RAM Memory
2		0000_0000	00000000		nop
3		0000_0004	00000000		nop
4		0000_0008	00000000		nop
5		0000_000C	00000000		nop
6		0000_0010	00000000		nop
7		0000_0014	00000000		nop
8		0000_0018	00000000		nop

Now we'll create our C source file. If the MPLAB IDE Editor window is not open, click *File->New* from the top menu or the New File menu shortcut on the standard toolbar. The Editor window will show up.

Let's type in a very simple Hello World program in the Editor window and save it in Projects32 directory as `main.c`. We'll use the UART Peripheral Library provided by Microchip:

FIGURE 5-9: MPLAB IDE MAIN.C FILE



```
// this is a first test file
#include <string.h>
#include <plib.h>
// Configuration Bit settings for System Clock = 72 MHz
//  Primary Osc w/PLL (XT+,HS+,EC+PLL)
//  Input Divider    2x Divider
//  Multiplier      18x Multiplier
//  WDT disabled
//  Other options are don't care
#pragma config FNOSC = PRIPLL, POSCMOD = HS, FPLLMUL = MUL_18, FPLLIDIV = DIV_2, FPBDIV = DIV_2, FPLLODIV = DIV_1
#pragma config FWDTEN = OFF

#define SYSTEM_FREQUENCY    72000000L   // frequency we're running at
#define BAUDRATE            57600       // serial baud rate
static const char* myHelloStr="Hello World from PIC32...\n";    // string to send to console

int main(void)
{
    int nc;
    int pbFreq;

    // Configure the device for maximum performance.
    // This macro sets flash wait states, PBCLK divider and DRM wait states based on the specified
    // clock frequency. It also turns on the cache mode if avaialble. Returns the PB frequency
    pbFreq=SYSTEMConfigPerformance(SYSTEM_FREQUENCY);

    // open and configure the UART2 peripheral
    OpenUART2(  UART_EN | UART_NO_PAR_8BIT | UART_1STOPBIT, UART_RX_ENABLE | UART_TX_ENABLE,
                (pbFreq/16/BAUDRATE)-1);    // selected baud rate, 8-N-1 @ 36 MHz

    putsUART2(myHelloStr); // send the string on the serial line
    // now toggle an LED (RA6)
    mPORTASetPinsDigitalOut(0x40);  // configure bit as output
    mPORTAToggleBits(0x40);         // toggle the LED

    nc=strlen(myHelloStr);
    return nc;
}
```

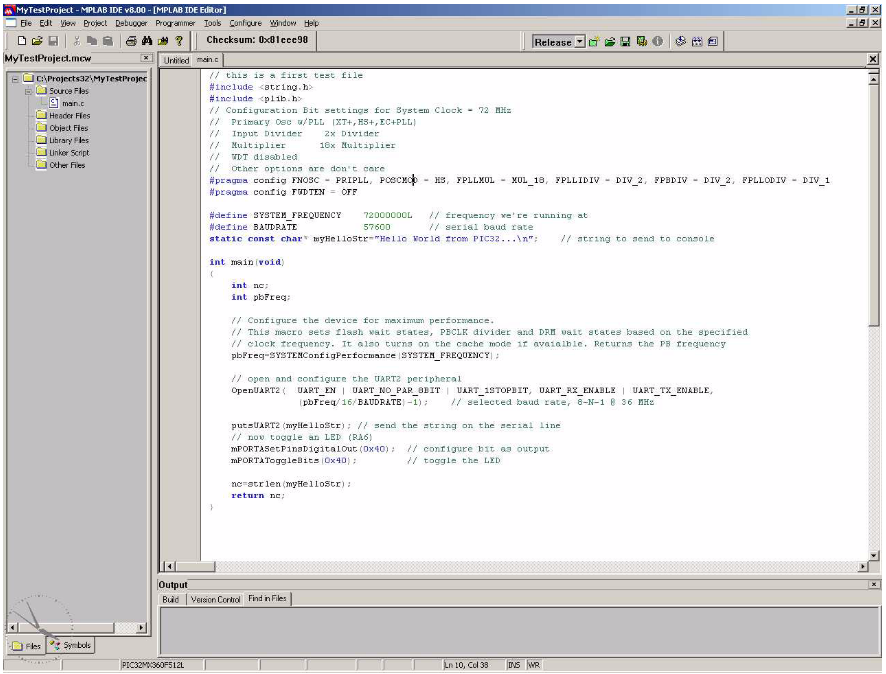
To add the newly created `main.c` file to our project just right click on the Source Files folder in the MPLAB IDE MyTestProject.mcp window and select Add File. Browse and select `main.c` file from the Projects32 directory.

Note: If the Project window is not open, select View->Project from the top IDE menu.

The linker script that's needed for this project is selected automatically by the MPLAB IDE. We don't need to add any other file to our project.

Right click on MyTestProject.mcp window and select **Save**. The test project should be saved. The MyTestProject.mcp should look like:

FIGURE 5-10: MPLAB IDE PROJECT SAVED



```
// this is a first test file
#include <string.h>
#include <plib.h>
// Configuration Bit settings for System Clock = 72 MHz
//  Primary Osc w/PLL (XT+,HS+,EC+PLL)
//  Input Divider    2x Divider
//  Multiplier      18x Multiplier
//  WDT disabled
//  Other options are don't care
#pragma config FNOSC = PRIPLL, POSCMOD = HS, FPLLMUL = MUL_18, FPLLIDIV = DIV_2, FPBDIV = DIV_2, FPLLODIV = DIV_1
#pragma config FWDTEN = OFF

#define SYSTEM_FREQUENCY    72000000L   // frequency we're running at
#define BAUDRATE            57600       // serial baud rate
static const char* myHelloStr="Hello World from PIC32...\n";    // string to send to console

int main(void)
{
    int nc;
    int pbFreq;

    // Configure the device for maximum performance.
    // This macro sets flash wait states, PBCLK divider and DRM wait states based on the specified
    // clock frequency. It also turns on the cache mode if avaialble. Returns the PB frequency
    pbFreq=SYSTEMConfigPerformance(SYSTEM_FREQUENCY);

    // open and configure the UART2 peripheral
    OpenUART2(  UART_EN | UART_NO_PAR_8BIT | UART_1STOPBIT, UART_RX_ENABLE | UART_TX_ENABLE,
                (pbFreq/16/BAUDRATE)-1);    // selected baud rate, 8-N-1 @ 36 MHz

    putsUART2(myHelloStr); // send the string on the serial line
    // now toggle an LED (RA6)
    mPORTASetPinsDigitalOut(0x40);  // configure bit as output
    mPORTAToggleBits(0x40);         // toggle the LED

    nc=strlen(myHelloStr);
    return nc;
}
```

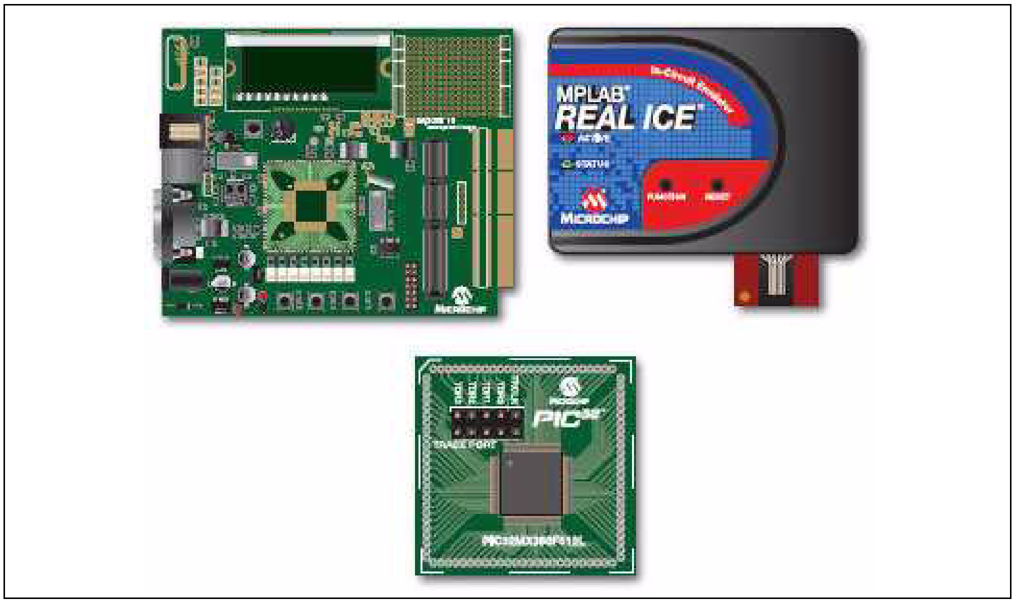
TIP: Files can be added and projects saved by using the right mouse button in the project window. In case of error, files can be manually deleted by selecting them and using the right mouse click menu.

5.10 ATTACHING THE DEBUGGER

In order to test the code using the PIC32 Starter Kit, please refer to the PIC32MX Starter Kit User's Guide (DS61144) for a sample project and a step-by-step getting started info.

For the purpose of testing our code in this document we will use an Explorer 16 Development board (DM240001), a MPLAB REAL ICE In-Circuit Debugger (DV244005) and a PIC32MX360F512L PIM (MA320001) together with a 9V universal power supply, a serial cable and a USB cable for connecting the REAL ICE to the development board.

FIGURE 5-11: EXPLORER 16, MPLAB® REAL ICE™ AND PIC32MX360F512L PIM



Take the following steps to ensure proper connection of the REAL ICE to the Explorer 16 development board:

1. Connect the MPLAB REAL ICE module to the PC with the USB cable.
2. Connect the MPLAB REAL ICE to the Explorer 16 Development Board with the short RJ-11 cable.
3. Apply power to the Explorer 16 board.
4. From the Debugger menu, click *Select Tool > MPLAB REAL ICE* to set the MPLAB REAL ICE as the debug tool in MPLAB IDE.
5. From the Debugger menu, select Connect to connect the debugger to the device. MPLAB IDE should report in the Output window that it found the PIC32MX360F512L device.

> **Note:** MPLAB IDE may need to download new firmware if this is the first time the MPLAB REAL ICE is being used with a PIC32 device. Allow it to do so.

Once you have performed these steps, go to the MPLAB IDE window and from the *Debugger->Select Tool* menu select the debugging tool you're using to connect to the board (i.e., MPLAB REAL ICE should be within the available choices).

Once you have selected the tool, the "Debug Toolbar" should be present just below the main menu bar, together with other toolbars that may be selected.

> **Note:** Be sure to select Debug from the MPLAB IDE Build Configuration drop down list.

5.11 BUILDING THE PROJECT

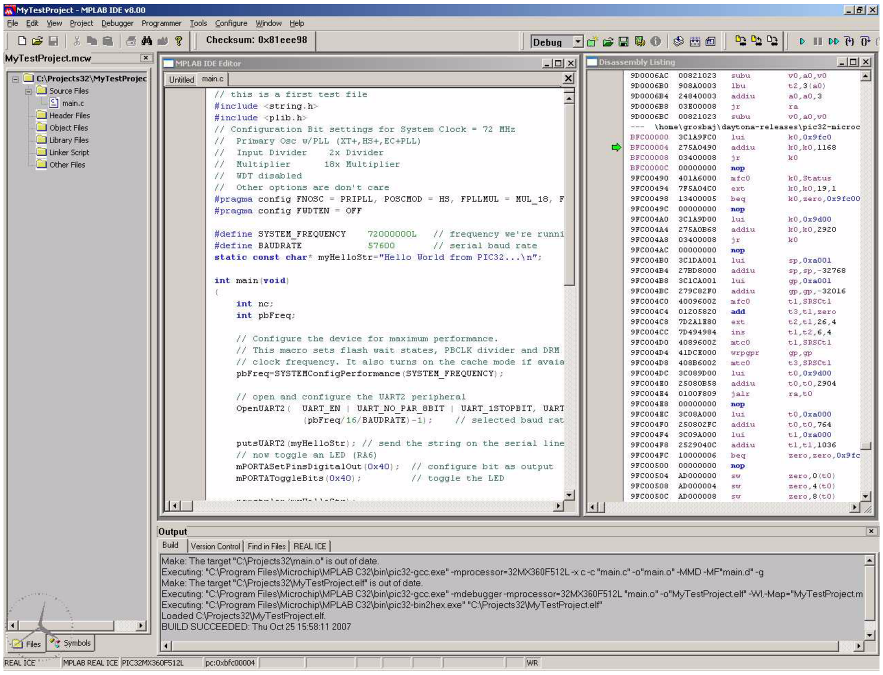
From the Project menu, we can compile and link the current files.

To build the project, select either:

- *Project>Build All*
- Right click on the project name in the project window and select Build All
- Click the Build All icon on the Project toolbar. Hover the mouse over icons to see pop-up text of what they represent.

The Output window shows the result of the build process. There should be no errors on any step.

FIGURE 5-12: MPLAB IDE PROJECT BUILD



```
// this is a first test file
#include <string.h>
#include <plib.h>
// Configuration Bit settings for System Clock = 72 MHz
//  Primary Osc w/PLL (XT+,HS+,EC+PLL)
//  Input Divider    2x Divider
//  Multiplier      18x Multiplier
//  WDT disabled
//  Other options are don't care
#pragma config FNOSC = PRIPLL, POSCMOD = HS, FPLLMUL = MUL_18, F
#pragma config FWDTEN = OFF

#define SYSTEM_FREQUENCY    72000000L   // frequency we're runni
#define BAUDRATE            57600       // serial baud rate
static const char* myHelloStr="Hello World from PIC32...\n";

int main(void)
{
    int nc;
    int pbFreq;

    // Configure the device for maximum performance.
    // This macro sets flash wait states, PBCLK divider and DRM
    // clock frequency. It also turns on the cache mode if avaia
    pbFreq=SYSTEMConfigPerformance(SYSTEM_FREQUENCY);

    // open and configure the UART2 peripheral
    OpenUART2(  UART_EN | UART_NO_PAR_8BIT | UART_1STOPBIT, UART
                (pbFreq/16/BAUDRATE)-1);    // selected baud rat

    putsUART2(myHelloStr); // send the string on the serial line
    // now toggle an LED (RA6)
    mPORTASetPinsDigitalOut(0x40);  // configure bit as output
    mPORTAToggleBits(0x40);         // toggle the LED
```

```
9D0006AC  00821023   subu      v0,a0,v0
9D0006B0  908A0003   lbu       t2,3(a0)
9D0006B4  24840003   addiu     a0,a0,3
9D0006B8  03E00008   jr        ra
9D0006BC  00821023   subu      v0,a0,v0
---  \home\grosbaj\daytona-releases\pic32-microc
BFC00000  3C1A9FC0   lui       k0,0x9fc0
BFC00004  275A0490   addiu     k0,k0,1168
BFC00008  03400008   jr        k0
BFC0000C  00000000   nop
9FC00490  401A6000   mfc0      k0,Status
9FC00494  7F5A04C0   ext       k0,k0,19,1
9FC00498  13400005   beq       k0,zero,0x9fc00
9FC0049C  00000000   nop
9FC004A0  3C1A9D00   lui       k0,0x9d00
9FC004A4  275A0B68   addiu     k0,k0,2920
9FC004A8  03400008   jr        k0
9FC004AC  00000000   nop
9FC004B0  3C1DA001   lui       sp,0xa001
9FC004B4  27BD8000   addiu     sp,sp,-32768
9FC004B8  3C1CA001   lui       gp,0xa001
9FC004BC  279C82F0   addiu     gp,gp,-32016
9FC004C0  40096002   mfc0      t1,SRSCtl
9FC004C4  01205820   add       t3,t1,zero
9FC004C8  7D2A1E80   ext       t2,t1,26,4
9FC004CC  7D494984   ins       t1,t2,6,4
9FC004D0  40896002   mtc0      t1,SRSCtl
9FC004D4  41DCE000   wrpgpr    gp,gp
9FC004D8  408B6002   mtc0      t3,SRSCtl
9FC004DC  3C089D00   lui       t0,0x9d00
9FC004E0  25080B58   addiu     t0,t0,2904
9FC004E4  0100F809   jalr      ra,t0
9FC004E8  00000000   nop
9FC004EC  3C08A000   lui       t0,0xa000
9FC004F0  250802FC   addiu     t0,t0,764
9FC004F4  3C09A000   lui       t1,0xa000
9FC004F8  2529040C   addiu     t1,t1,1036
9FC004FC  10000006   beq       zero,zero,0x9fc
9FC00500  00000000   nop
9FC00504  AD000000   sw        zero,0(t0)
9FC00508  AD000004   sw        zero,4(t0)
9FC0050C  AD000008   sw        zero,8(t0)
```

```
Make: The target "C:\Projects32\main.o" is out of date.
Executing: "C:\Program Files\Microchip\MPLAB C32\bin\pic32-gcc.exe" -mprocessor=32MX360F512L -x c -c "main.c" -o"main.o" -MMD -MF"main.d" -g
Make: The target "C:\Projects32\MyTestProject.elf" is out of date.
Executing: "C:\Program Files\Microchip\MPLAB C32\bin\pic32-gcc.exe" -mdebugger -mprocessor=32MX360F512L "main.o" -o"MyTestProject.elf" -Wl,-Map="MyTestProject.m
Executing: "C:\Program Files\Microchip\MPLAB C32\bin\pic32-bin2hex.exe" "C:\Projects32\MyTestProject.elf"
Loaded C:\Projects32\MyTestProject.elf.
BUILD SUCCEEDED: Thu Oct 25 15:58:11 2007
```

5.12 TESTING THE CODE

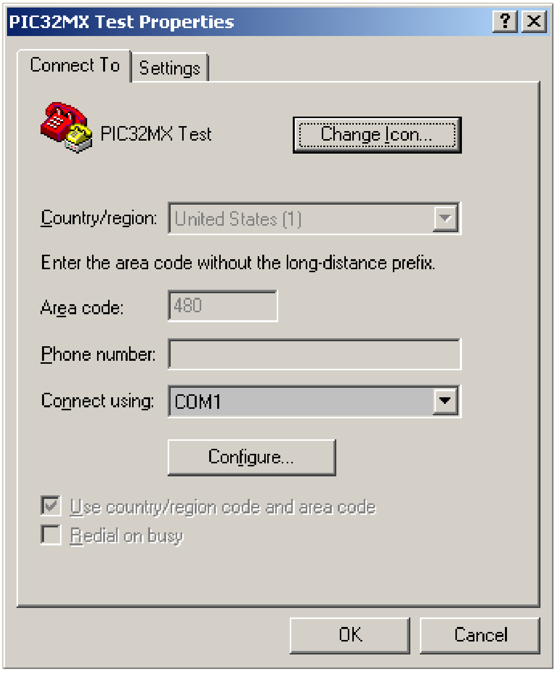
First download the hex image of the program we just built by selecting Debugger->Program from the Debugger menu or directly program from the “Debug Toolbar”. The programming operation should occur with no errors.

In order to test our example application, we need to set-up the PC part too.

Connect a serial cable between the board RS-232 connector and one of the PC’s COM ports and open a Hyper Terminal application on the PC (Programs->Accessories->Communications->Hyper Terminal).

Select a proper name for this connection, click **OK** and then, in the “Connect To” screen select the COM port that’s connected to the development board.

FIGURE 5-13: HYPER TERMINAL CONNECTION SCREEN



PIC32MX Test Properties

PIC32MX Test

Country/region: United States (1)

Enter the area code without the long-distance prefix.

Area code: 480

Phone number:

Connect using: COM1

☑ Use country/region code and area code

□ Redial on busy

Click **OK** and, on the next screen, select the following communication settings:

- Bits Per Second: 57600
- Data Bits: 8
- Parity: None
- Stop Bits: 1
- Flow Control: None

Click **OK** and the Hyper Terminal session should be connected to the serial port on the development board.

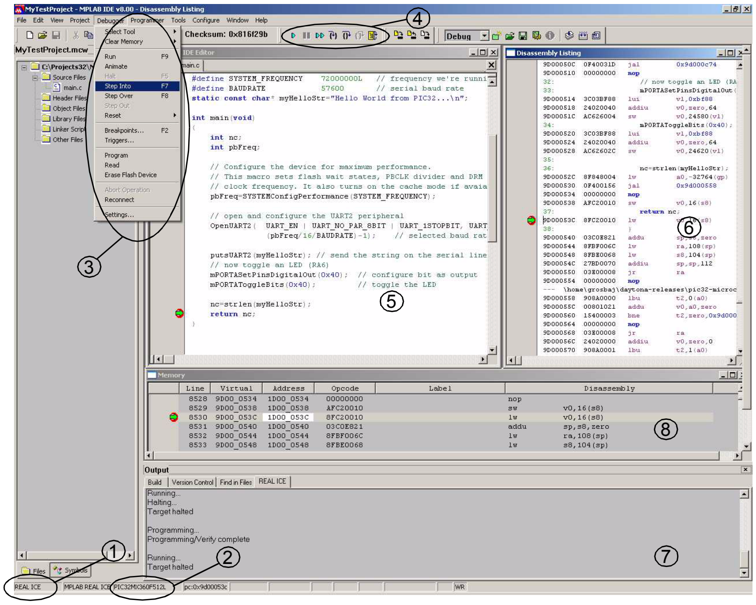
Now return to the MPLAB IDE and double click on the line:

return nc;

in the MPLAB IDE Editor window, the `main.c` program, to set a breakpoint on that line, just after the call to `strlen()`.

Click Debugger->Run from the main menu, or Run from the Debug toolbar. The program should start running and it will reach the set breakpoint. The MPLAB IDE window will look very similar to this one:

FIGURE 5-14: MPLAB IDE BREAKPOINT REACHED



```
#define SYSTEM_FREQUENCY    72000000L   // frequency we're runni
#define BAUDRATE            57600       // serial baud rate
static const char* myHelloStr="Hello World from PIC32...\n";

int main(void)
{
    int nc;
    int pbFreq;

    // Configure the device for maximum performance.
    // This macro sets flash wait states, PBCLK divider and DRM
    // clock frequency. It also turns on the cache mode if avaia
    pbFreq=SYSTEMConfigPerformance(SYSTEM_FREQUENCY);

    // open and configure the UART2 peripheral
    OpenUART2(  UART_EN | UART_NO_PAR_8BIT | UART_1STOPBIT, UART
                (pbFreq/16/BAUDRATE)-1);    // selected baud rat

    putsUART2(myHelloStr); // send the string on the serial line
    // now toggle an LED (RA6)
    mPORTASetPinsDigitalOut(0x40);  // configure bit as output
    mPORTAToggleBits(0x40);         // toggle the LED

    nc=strlen(myHelloStr);
    return nc;
}
```

```
9D00050C  0F40031D   jal        0x9d000c74
9D000510  00000000   nop
32:                    // now toggle an LED (RA
33:                    mPORTASetPinsDigitalOut(
9D000514  3C03BF88   lui        v1,0xbf88
9D000518  24020040   addiu      v0,zero,64
9D00051C  AC626004   sw         v0,24580(v1)
34:                    mPORTAToggleBits(0x40);
9D000520  3C03BF88   lui        v1,0xbf88
9D000524  24020040   addiu      v0,zero,64
9D000528  AC62602C   sw         v0,24620(v1)
35:
36:                    nc=strlen(myHelloStr);
9D00052C  8F848004   lw         a0,-32764(gp)
9D000530  0F400156   jal        0x9d000558
9D000534  00000000   nop
9D000538  AFC20010   sw         v0,16(s8)
37:                    return nc;
9D00053C  8FC20010   lw         v0,16(s8)
38:                    }
9D000540  03C0E821   addu       sp,s8,zero
9D000544  8FBF006C   lw         ra,108(sp)
9D000548  8FBE0068   lw         s8,104(sp)
9D00054C  27BD0070   addiu      sp,sp,112
9D000550  03E00008   jr         ra
9D000554  00000000   nop
---  \home\grosbaj\daytona-releases\pic32-microc
9D000558  908A0000   lbu        t2,0(a0)
9D00055C  00801021   addu       v0,a0,zero
9D000560  15400003   bne        t2,zero,0x9d000
9D000564  00000000   nop
9D000568  03E00008   jr         ra
9D00056C  24020000   addiu      v0,zero,0
9D000570  908A0001   lbu        t2,1(a0)
```

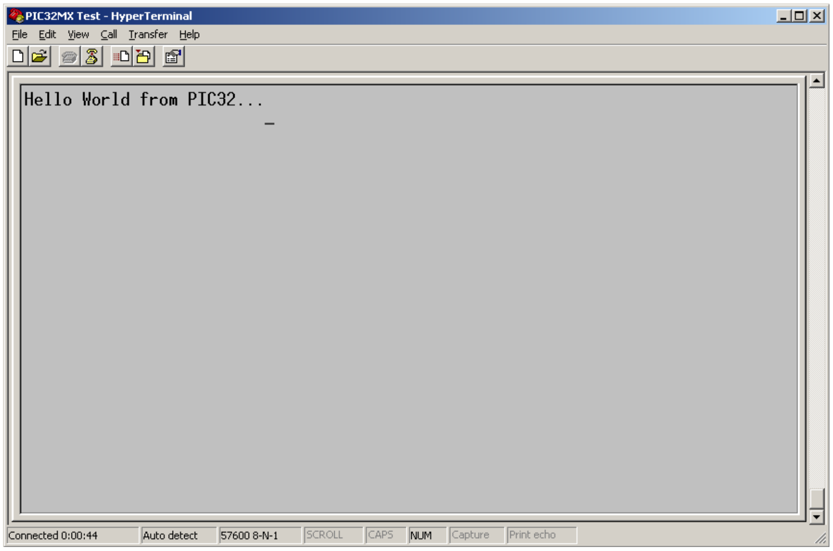
Line	Virtual	Address	Opcode	Label	Disassembly
8528	9D00_0534	1D00_0534	00000000		nop
8529	9D00_0538	1D00_0538	AFC20010		sw v0,16(s8)
8530	9D00_053C	1D00_053C	8FC20010		lw v0,16(s8)
8531	9D00_0540	1D00_0540	03C0E821		addu sp,s8,zero
8532	9D00_0544	1D00_0544	8FBF006C		lw ra,108(sp)
8533	9D00_0548	1D00_0548	8FBE0068		lw s8,104(sp)

```
Running...
Halting...
Target halted

Programming...
Programming/Verify complete

Running...
Target halted
```

1. The debugging tool selected to connect to the demo board
2. The device selected on the target board
3. The menus available under the Debugger top menu
4. The Debug toolbar
5. The Source window, with the breakpoint reached
6. The Disassembly window
7. The Output window
8. The Program Memory window

Now, if you take a look to the PC Hyper Terminal window, you should see the output sent by our simple test program to the serial line:

FIGURE 5-15: HYPER TERMINAL RECEIVED CHARACTERS


PIC32MX Test - HyperTerminal
File Edit View Call Transfer Help

```
Hello World from PIC32...
```

```
Connected 0:00:44   Auto detect   57600 8-N-1   SCROLL   CAPS   NUM   Capture   Print echo
```

Also, if you look at the Explorer 16 board, you should see the LED D9 lit on.

This shows that the PIC32 Explorer16 demo board and the MPLAB IDE operate correctly.

We just made our first “Hello World” program for PIC32.

There are many other things that you can do in the MPLAB IDE to debug your program. For example, you can notice in that by hovering the mouse over the variable “nc” we can see its actual value (which should be the number of characters sent to the serial port by the `putsUART2()` function).

Other MPLAB IDE debug features:

- Watch window: you can add variables of peripheral SFRs to the watch window and monitor their values
- Complex Breakpoint manipulation
- All the useful debug commands: step into, step over, etc, using both the C Source window and the disassembly listing window.
- Profile your code execution and calculate delay times between different points in your program.

We presented here the essential steps for getting started with PIC32 using the MPLAB IDE. You are now ready to continue exploring the capabilities of PIC32 and MPLAB IDE. For further information about PIC32 please see the documentation list provided in this document. For more information about the MPLAB IDE please see the “*MPLAB IDE Quick Start Guide*” (DS51281) available on our site at www.microchip.com following the Design link.

NOTES:

Chapter 6. Technical Support Resources

6.1 INTRODUCTION

For information about the Technical Support provided please visit Support.microchip.com.

NOTES:

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
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

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

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

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

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

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: 86-28-8665-5511
Fax: 86-28-8665-7889

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

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

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 - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

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

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

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

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

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

Taiwan - Hsin Chu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

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

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

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

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

Germany - Munich
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

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/02/08