



**MPLAB<sup>®</sup> C18  
C COMPILER  
GETTING STARTED**

---

---

**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contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

#### **Trademarks**

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


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

Analog-for-the-Digital Age, Application Maestro, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, PICKit, PICDEM, PICDEM.net, PICLAB, PICTail, PowerCal, PowerInfo, PowerMate, PowerTool, rLAB, rPICDEM, Select Mode, Smart Serial, SmartTel and Total Endurance 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.

© 2004, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

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

*Microchip received ISO/TS-16949:2002 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona and Mountain View, California in October 2003. The Company's quality system processes and procedures are for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

---

---

**Table of Contents**

---

---

<b>Preface .....</b>	<b>1</b>
<b>Chapter 1. Overview</b>	
1.1 Introduction .....	7
1.2 Highlights .....	7
1.3 System Requirements .....	7
1.4 Quick Directory Tour .....	8
1.5 About the Language Tools .....	9
<b>Chapter 2. Installation</b>	
2.1 Introduction .....	11
2.2 Highlights .....	11
2.3 Installing MPLAB C18 .....	11
2.4 Uninstalling MPLAB C18 .....	17
<b>Chapter 3. Examples of Use</b>	
3.1 Introduction .....	19
3.2 Highlights .....	19
3.3 Example 1 .....	20
3.4 Example 2 .....	39
3.5 Example 3 .....	43
3.6 Example 4 .....	47
3.7 Example 5 .....	51
<b>Glossary .....</b>	<b>53</b>
<b>Index .....</b>	<b>59</b>
<b>Worldwide Sales and Service .....</b>	<b>60</b>

# MPLAB® C18 Compiler Getting Started

---

NOTES:

---

## Preface

---

### INTRODUCTION

The purpose of this document is to help users get up and running with Microchip's MPLAB C18 C compiler.

### HIGHLIGHTS

Items discussed in this chapter are:

- About This Guide
- Recommended Reading
- The Microchip Web Site
- Development Systems Customer Notification Service
- Customer Support

### ABOUT THIS GUIDE

#### Document Layout

This document describes how to install/uninstall MPLAB C18 and provides several examples of writing C code for PICmicro® microcontroller applications. For a detailed discussion about basic MPLAB IDE v6.xx functions, refer to the MPLAB IDE on-line help file.

This document includes:

- **Chapter 1: Overview** – Defines system requirements and provides a brief description of the installed programs and directories created by the installation process.
- **Chapter 2: Installation** – Provides instructions on how to install the compiler onto your system. Also provides uninstall instructions.
- **Chapter 3: Examples of Use** – uses a tutorial style to illustrate effective use of the MPLAB C18 C compiler. The examples use MPLAB IDE v6.xx with PIC18F452 or PIC18F4620 as the selected device and MPLAB SIM simulator as a debug tool. Some examples use the additional tools, MPLAB ICD 2 in-circuit debugger and PICDEM™ 2 Plus demo board.
  - **Example 1** demonstrates how to set up and build a project; run, step and set breakpoints in the example code; and debug the code.
  - **Example 2** demonstrates the use of the MPLAB C18 peripheral libraries and the C standard library, as well as the allocation of variables into program memory.
  - **Example 3** demonstrates some of the differences between Extended and Non-extended modes.
  - **Example 4** demonstrates the allocation of variables in access RAM.
  - **Example 5** demonstrates the use of interrupt service routines with MPLAB C18 and provides an example of the use of the MPLAB C18 peripheral libraries.

# MPLAB® C18 C Compiler Getting Started

---

- **Glossary** – A glossary of terms used in this guide.
- **Index** – Cross-reference listing of terms, features and sections of this document.

## Conventions Used in this Guide

This guide uses the following documentation conventions:

### DOCUMENTATION CONVENTION

Description	Represents	Examples
<b>Code (Courier font):</b>		
Courier font	Sample source code	<code>distance -= time * speed;</code>
	Filenames and paths	<code>c:\mcc18\h</code>
	Keywords	<code>_asm, _endasm, static</code>
	Command-line options	<code>-Opa+, -Opa-</code>
Italic Courier font	Variable name argument	<code>file.o</code> , where <i>file</i> can be any valid file name
Square brackets [ ]	Optional arguments	<code>mcc18 [options] file [options]</code>
Ellipses...	Replaces repeated instances of text	<code>var_name [, var_name...]</code>
	Represents code supplied by user.	<pre>void main (void) {     ... }</pre>
<code>0xnnnn</code>	A hexadecimal number where <i>n</i> is a hexadecimal digit	<code>0xFFFF, 0x007A</code>
<b>Documents (Arial font):</b>		
Italic characters	Referenced books	<i>MPLAB User's Guide</i>

## Documentation Updates

All documentation becomes dated, and this guide is no exception. Since MPLAB IDE, MPLAB C18 and other Microchip tools are constantly evolving to meet customer needs, some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site to obtain the latest documentation available.

## Documentation Numbering Conventions

Documents are numbered with a “DS” number. The 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 = The document number.  
A = The revision level of the document.

## RECOMMENDED READING

For more information on included libraries and precompiled object files for the compilers, the operation of MPLAB IDE and the use of other tools, the following are recommended reading.

### **readme.c18**

For the latest information on using MPLAB C18 C Compiler, read the readme.c18 file (ASCII text) included with the software. This readme file contains update information that may not be included in this document.

### **readme.xxx**

For the latest information on other Microchip tools (MPLAB IDE, MPLINK™ linker, etc.), read the associated readme files (ASCII text file) included with the software.

### **MPLAB® C18 C Compiler User's Guide (DS51288)**

Comprehensive guide that describes the operation and features of Microchip's MPLAB C18 C compiler for PIC18 devices.

### **MPLAB® IDE V6.XX Quick Start Guide (DS51281)**

Describes how to set up the MPLAB IDE software and use it to create projects and program devices.

### **MPASM™ User's Guide with MPLINK™ Linker and MPLIB™ Librarian (DS33014)**

Describes how to use the Microchip PICmicro MCU assembler (MPASM), linker (MPLINK) and librarian (MPLIB).

### **PICmicro® 18C MCU Family Reference Manual (DS39500)**

Focuses on the Enhanced MCU family of devices. The operation of the Enhanced MCU family architecture and peripheral modules is explained but does not cover the specifics of each device.

### **PIC18 Device Data Sheets and Application Notes**

Data sheets describe the operation and electrical specifications of PIC18 devices. Application notes describe how to use PIC18 devices.

To obtain any of the above listed documents, visit the Microchip web site ([www.microchip.com](http://www.microchip.com)) to retrieve these documents in Adobe Acrobat (.pdf) format.

## THE MICROCHIP WEB SITE

Microchip provides on-line support on the Microchip World Wide Web (WWW) site. The web site is used by Microchip as a means to make files and information easily available to customers. To view the site, you must have access to the Internet and a web browser, such as, Netscape Navigator® or Microsoft® Internet Explorer.

The Microchip web site is available by using your favorite Internet browser to reach:

<http://www.microchip.com>

The web site provides a variety of services. Users may download files for the latest development tools, data sheets, application notes, user's guides, articles and sample programs. A variety of information specific to the business of Microchip is also available, including listings of Microchip sales offices, distributors and factory representatives.

### Technical Support

- Frequently Asked Questions (FAQ)
- On-line Discussion Groups – conferences for products, development systems, technical information and more
- Microchip Consultant Program Member Listing
- Links to other useful web sites related to Microchip products

### Engineer's Toolbox

- Design Tips
- Device Errata

### Other Available Information

- Latest Microchip Press Releases
- Listing of seminars and events
- Job Postings



## DEVELOPMENT SYSTEMS CUSTOMER NOTIFICATION SERVICE

Microchip started the customer notification service to help our customers keep current on Microchip products with the least amount of effort. Once you subscribe, you will receive e-mail notification whenever we change, update, revise or have errata related to your specified product family or development tool of interest.

Go to the Microchip web site at (<http://www.microchip.com>) and click on Customer Change Notification. Follow the instructions to register.

The Development Systems product group categories are:

- Compilers
- Emulators
- In-Circuit Debuggers
- MPLAB IDE
- Programmers

Here is a description of these categories:

**Compilers** – The latest information on Microchip C compilers and other language tools. These include the MPLAB<sup>®</sup> C17, MPLAB C18 and MPLAB C30 C compilers; MPASM<sup>™</sup> and MPLAB ASM30 assemblers; MPLINK<sup>™</sup> and MPLAB LINK30 object linkers; MPLIB<sup>™</sup> 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<sup>®</sup> IDE, the Windows<sup>®</sup> Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE and MPLAB SIM simulators, MPLAB IDE Project Manager and general editing and debugging features.

**Programmers** – The latest information on Microchip device programmers. These include the MPLAB PM3 and PRO MATE<sup>®</sup> II device programmers and PICSTART<sup>®</sup> Plus development programmer.

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Corporate Applications Engineer (CAE)
- Hotline

Customers should call their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. See the last page for a list of sales offices and locations.

Corporate Applications Engineers (CAEs) may be contacted at (480) 792-7627.

In addition, there is a Systems Information and Upgrade Line. This line provides system users a list of the latest versions of all of Microchip's development systems software products. Plus, this line provides information on how customers can receive any currently available upgrade kits.

The Hotline Numbers are:

1-800-755-2345 for U.S. and most of Canada.

1-480-792-7302 for the rest of the world.

---

## **Chapter 1. Overview**

---

### **1.1 INTRODUCTION**

This document is designed to get users started quickly using Microchip's C compiler – MPLAB C18. PICmicro microcontroller applications can be easily developed using MPLAB C18 with PIC18 PICmicro MCUs, MPLINK linker and MPLAB IDE. Please refer to the *MPLAB® C18 C Compiler User's Guide* (DS51288) for more details on the features mentioned in this document.

### **1.2 HIGHLIGHTS**

Information in this chapter includes:

- System Requirements
- Quick Directory Tour
- About the Language Tools

### **1.3 SYSTEM REQUIREMENTS**

The minimum system requirements for using MPLAB C18 and the MPLINK linker are:

- 25 MB hard disk space (50 MB recommended)
- Microsoft® Windows® operating system (95 or later)

# MPLAB® C18 C Compiler Getting Started

---

## 1.4 QUICK DIRECTORY TOUR

The MPLAB C18 installation directory contains the readme file for the compiler (`readme.c18`) and the readme file for the linker (`readme.lkr`). In addition, a number of subdirectories are also present. A detailed description of the subdirectories is shown in Table 1-1.

**TABLE 1-1: MPLAB C18 SUBDIRECTORY DESCRIPTIONS**

Directory	Description
bin	Contains the executables for the compiler and linker. These are described in more detail in the following section.
cpp	Contains the source code for the MPLAB C18 C preprocessor. This source code is provided for general interest.
doc	Contains the MPLAB C18 electronic documentation. Refer to these documents for questions regarding MPLAB C18.
example	Contains sample applications to help users get started using MPLAB C18, including the examples contained in this document.
h	Contains the header files for the standard C library and the processor-specific libraries for the supported PICmicro MCUs.
lib	Contains the standard C library ( <code>clib.lib</code> or <code>clib_e.lib</code> ), the processor-specific libraries ( <code>p18xxxx.lib</code> or <code>p18xxxx_e.lib</code> , where <code>xxxx</code> is the specific device number) and the startup modules ( <code>c018.o</code> , <code>c018_e.o</code> , <code>c018i.o</code> , <code>c018i_e.o</code> , <code>c018iz.o</code> , <code>c018iz_e.o</code> ).
lkr	Contains the linker script files.
mpasm	Contains the command-line version of the MPASM assembler, the assembly header files for the devices supported by MPLAB C18 ( <code>p18xxxx.inc</code> ) and the assembly header files used by the libraries.
src	Contains the source code, in the form of C and assembly files, for the standard C library, the processor-specific libraries and the startup modules.

## 1.5 ABOUT THE LANGUAGE TOOLS

The `bin` and `mpasm` subdirectories of the MPLAB C18 compiler installation directory contain the executables which comprise the MPLAB C18, MPASM assembler and the MPLINK linker. A brief description of these programs is shown in Table 1-2.

**TABLE 1-2: MPLAB C18, MPASM ASSEMBLER AND MPLINK LINKER EXECUTABLES**

Executable	Description
<code>mcc18.exe</code>	This is the compiler shell. It takes as input a C file (i.e., <code>file.c</code> ) and invokes the Extended or Non-extended mode compiler executable.
<code>mcc18-extended.exe</code>	This is the Extended mode compiler executable. It is invoked by the compiler shell when compiling for Extended mode. It invokes the preprocessor <code>cpp18.exe</code> to preprocess the C file and then compiles the preprocessed output and generates a COFF file (e.g., <code>file.o</code> ) to be passed to the linker.
<code>mcc18-traditional.exe</code>	This is the Non-extended mode compiler executable. It is invoked by the compiler shell when compiling for the Non-extended mode. It invokes the preprocessor <code>cpp18.exe</code> to preprocess the C file and then compiles the preprocessed output and generates a COFF file (e.g., <code>file.o</code> ) to be passed to the linker.
<code>cpp18.exe</code>	This is the C preprocessor.
<code>mplink.exe</code>	This is the driver program for the linker. It takes as input a linker script, object files and library files and passes these to <code>_mplink.exe</code> . It then takes the output COFF file from <code>_mplink.exe</code> and passes it to <code>mp2cod.exe</code> and <code>mp2hex.exe</code> .
<code>_mplink.exe</code>	This is the linker. It takes as input a linker script (e.g., <code>p18f452.lkr</code> ), object files and library files and outputs a COFF executable (e.g., <code>file.out</code> or <code>file.cof</code> ). This COFF file is the result of resolving unassigned addresses of data and code of the input object files and referenced object files from the libraries. <code>_mplink.exe</code> also optionally produces a map file (e.g., <code>file.map</code> ) that contains detailed information on the allocation of data and code.
<code>mp2cod.exe</code>	This is the COFF to COD file converter. The COD file is a symbolic debugging file format which is used by the MPLAB IDE v5.xx. <code>mp2cod.exe</code> takes as input the COFF file produced by <code>_mplink.exe</code> and outputs a COD file (e.g., <code>file.cod</code> ). It also creates a listing file (e.g., <code>file.lst</code> ) that displays the correspondence between the original source code and machine code.
<code>mp2hex.exe</code>	This is the COFF to HEX file converter. The HEX file is a file format readable by a PICmicro programmer, such as the PICSTART Plus or the PRO MATE II. <code>mp2hex.exe</code> takes as input the COFF file produced by <code>_mplink.exe</code> and outputs a HEX file (e.g., <code>file.hex</code> ).
<code>mplib.exe</code>	This is the librarian. It allows for the creation and management of a library file (e.g., <code>file.lib</code> ) that acts as an archive for the object files. Library files are useful for organizing object files into reusable code repositories.

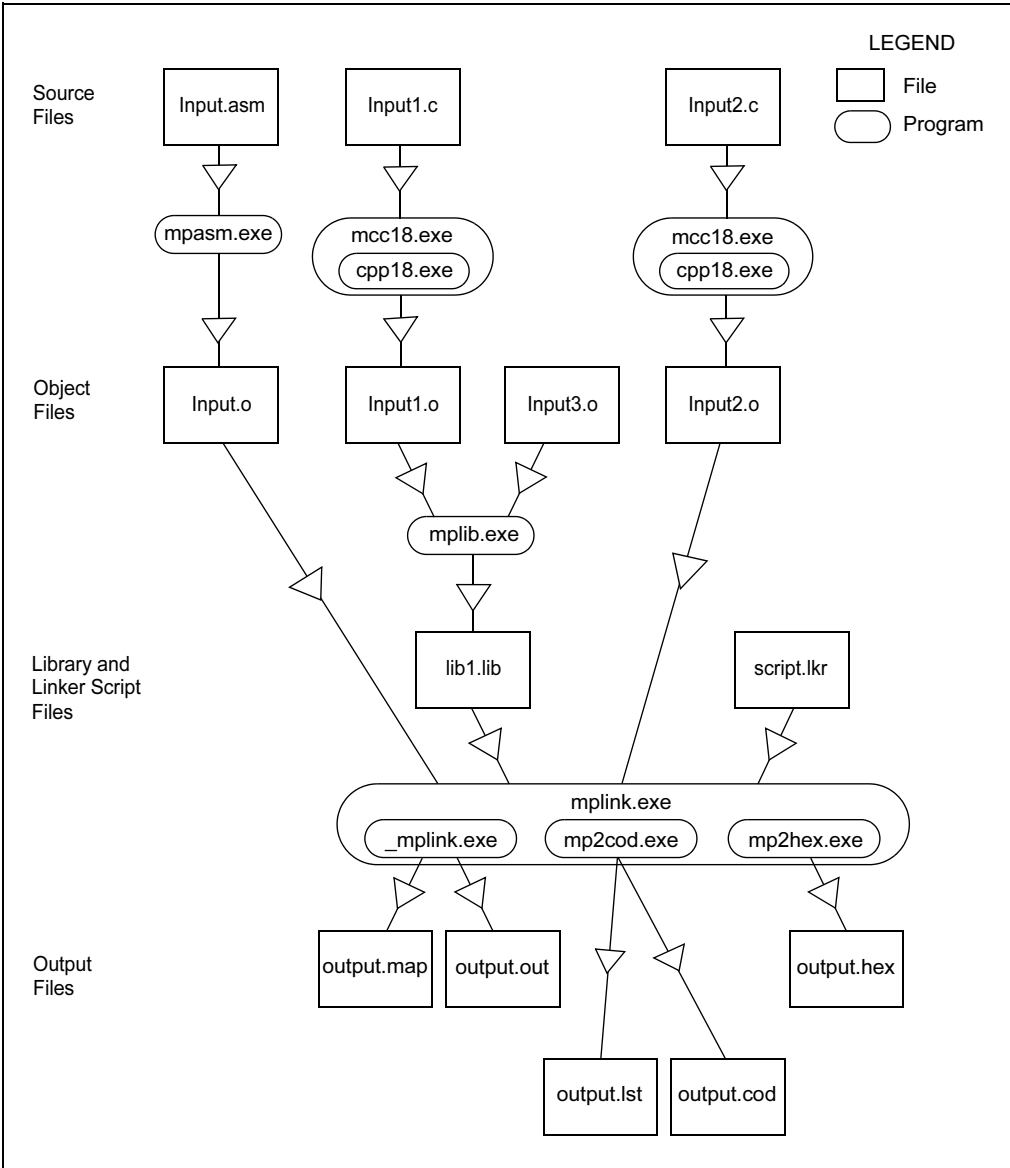
TABLE 1-2: MPLAB C18, MPASM ASSEMBLER AND MPLINK LINKER EXECUTABLES (CONTINUED)

mpasm.exe	This is the command-line assembler. It takes as input an assembly source file (e.g., file.asm) and outputs either a COFF file (e.g., file.o) or a HEX file and COD file (e.g., file.hex and file.cod). It also creates a listing file (e.g., file.lst) and an error file (e.g., file.err), which contains any errors or warnings emitted during the assembly process. Assembly source files may include assembly header files (e.g., p18f452.inc), which also contain assembly source code.
-----------	---

More detailed information on the language tools, including their command-line usage, can be found in the *MPLAB® C18 C Compiler User's Guide* (DS51288) and the *MPASM™ User's Guide with MPLINK™ and MPLIB™* (DS33014).

An example of the flow of execution of the language tools is illustrated in Figure 1-1.

FIGURE 1-1: LANGUAGE TOOLS EXECUTION FLOW



---

## Chapter 2. Installation

---

### 2.1 INTRODUCTION

This chapter will discuss in detail how to install MPLAB C18. Should it become necessary to remove the software, uninstall directions are provided as well.

### 2.2 HIGHLIGHTS

Information discussed in this chapter includes:

- Installing MPLAB C18
- Uninstalling MPLAB C18

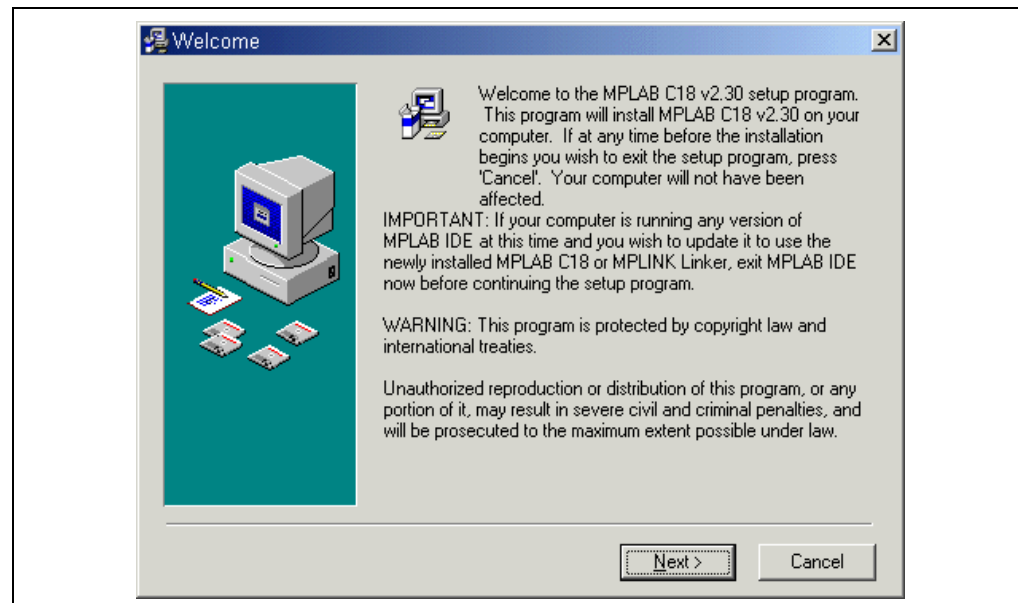
### 2.3 INSTALLING MPLAB C18

To install MPLAB C18, run the setup program from the CD-ROM. If installing an MPLAB C18 upgrade, run the upgrade setup program downloaded from the Microchip web site. A series of dialogs will step through the setup process.

#### 2.3.1 Welcome

A welcome screen displays the version number of MPLAB C18 that the setup program will install, Figure 2-1.

**FIGURE 2-1: MPLAB C18 WELCOME SCREEN**

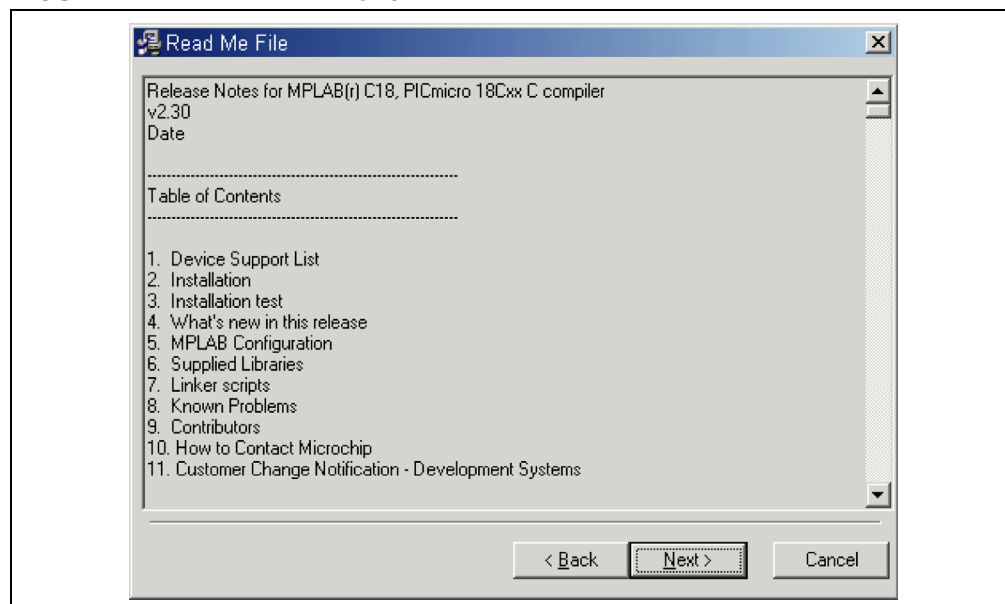


Click **Next** to continue.

## 2.3.2 Readme File

The MPLAB C18 readme file is displayed. This file contains important information about this release of MPLAB C18, such as known bugs, Figure 2-2.

**FIGURE 2-2: MPLAB C18 README FILE**



After reviewing, click **Next** to continue.

## 2.3.3 Select Installation Directory

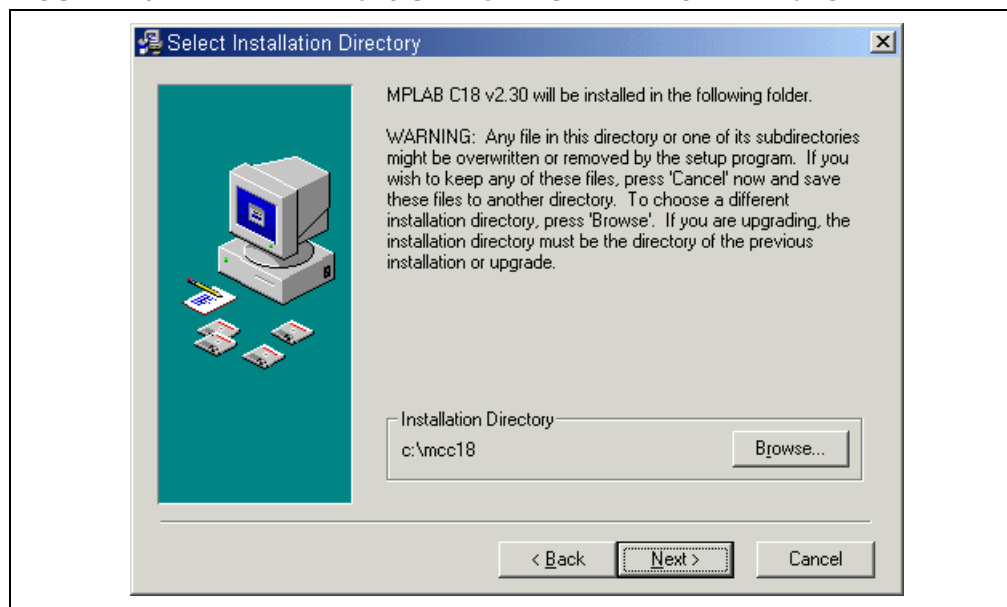
This step allows users to choose the directory where MPLAB C18 will be installed. When installing MPLAB C18 for the first time, the default installation directory is C:\mcc18, as shown in Figure 2-3.

If an upgrade is being installed, the setup program attempts to set the default installation directory to the directory of the previous installation. The installation directory for an upgrade must be the same directory of the previous installation or upgrade.

**Note:** Files in the installation directory and its subdirectories may be overwritten or removed during the installation process. To save any files, such as modified linker scripts or library source code from a previous installation, copy those files to a directory outside the installation directory before continuing.



**FIGURE 2-3: MPLAB C18 SELECT INSTALLATION DIRECTORY**

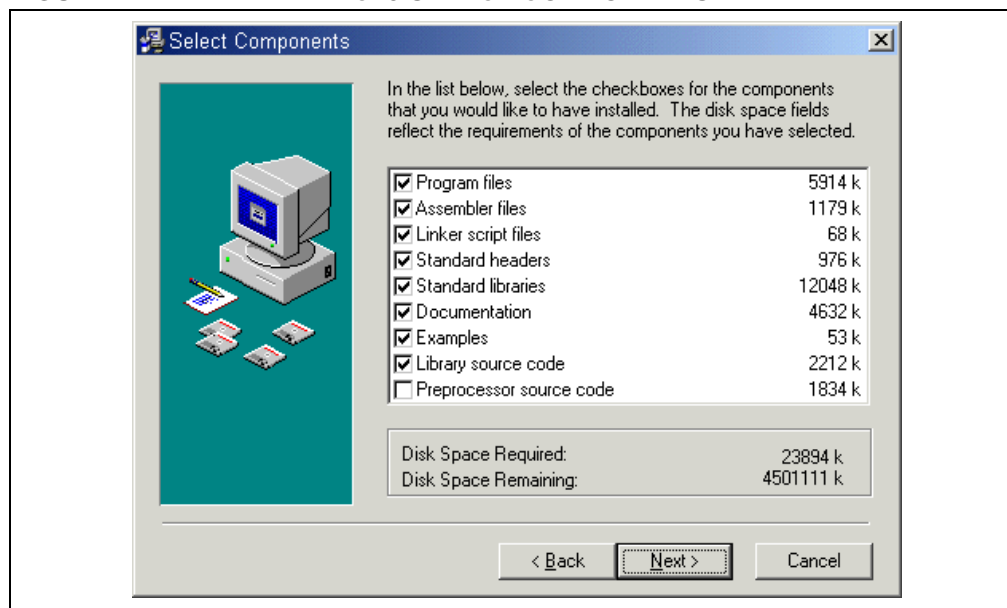


After specifying the directory, click **Next**.

## 2.3.4 Select Components

Choose the components to be installed by checking the appropriate box, Figure 2-4.

**FIGURE 2-4: MPLAB C18 SELECT COMPONENTS**



A detailed description of the available components is shown in Table 2-1.

# MPLAB® C18 C Compiler Getting Started

---

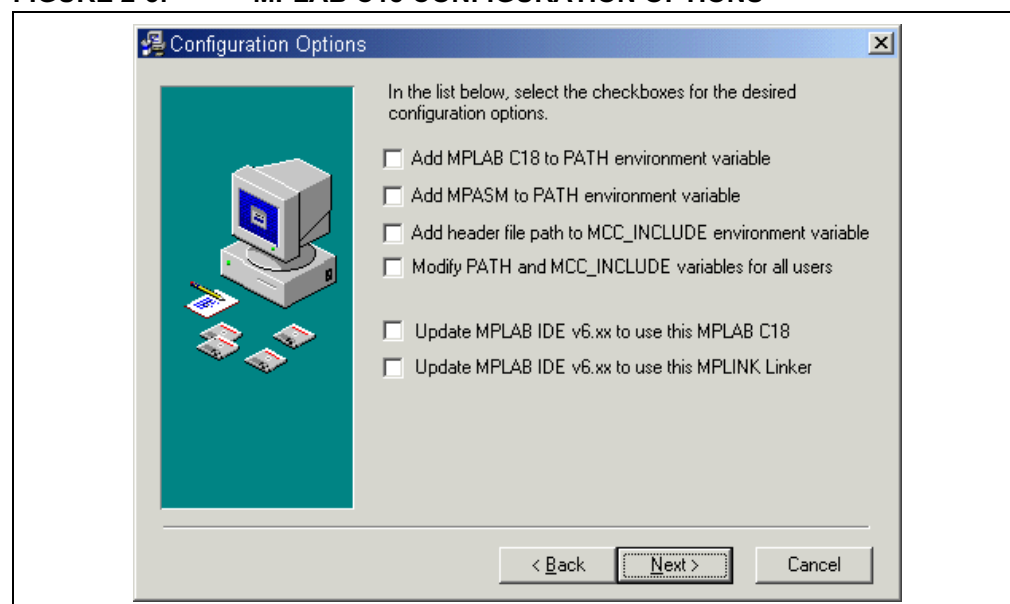
**TABLE 2-1: MPLAB C18 SOFTWARE COMPONENTS**

Component	Description
Program files	These are the executables for the compiler and linker. Users should install this component unless they are upgrading and wish to use the executables from the previously installed version.
Assembler files	These include the command-line version of the MPASM assembler ( <code>mpasm.exe</code> ), the assembly header files for the devices supported by MPLAB C18 ( <code>p18xxxx.inc</code> ) and the assembly header files used by the libraries.
Linker script files	These files are used by the MPLINK linker. There is one file for each supported PICmicro microcontroller. Each file provides a default memory configuration for the processor and directs the linker in the allocation of code and data in the processor's memory. These linker scripts differ from the linker scripts provided with the MPLAB IDE in that these are specifically designed for use with MPLAB C18. Since the MPLINK linker requires a linker script, users should install this component unless they plan on creating their own linker scripts.
Standard headers	These are the header files for the standard C library and the processor-specific libraries. If users choose to install the standard libraries, these will also be installed.
Standard libraries	This component contains the standard C library, the processor-specific libraries, and the startup modules. See the <i>MPLAB® C18 C Compiler Libraries</i> (DS51297) and the <i>MPLAB® C18 C Compiler User's Guide</i> (DS51288) for more information on the libraries and startup modules. Since most typical programs use the libraries and a startup module, it is recommended that users install this component.
Documentation	This is the electronic documentation for MPLAB C18.
Examples	These are sample applications to assist users in getting started with MPLAB C18, including the examples described in this document.
Library source code	This is the source code for the standard C library and the processor-specific libraries. Users should install this component if they plan on rebuilding the libraries.
Preprocessor source code	This is the source code for the preprocessor. It is provided for general interest.

## 2.3.5 Configuration Options

The next dialog screen allows users to select a particular set of MPLAB C18 configuration options for their system, Figure 2-5:

**FIGURE 2-5: MPLAB C18 CONFIGURATION OPTIONS**



A detailed description of the available configuration options is shown in Table 2-2. Select the components to be installed, then click **Next**.

# MPLAB® C18 C Compiler Getting Started

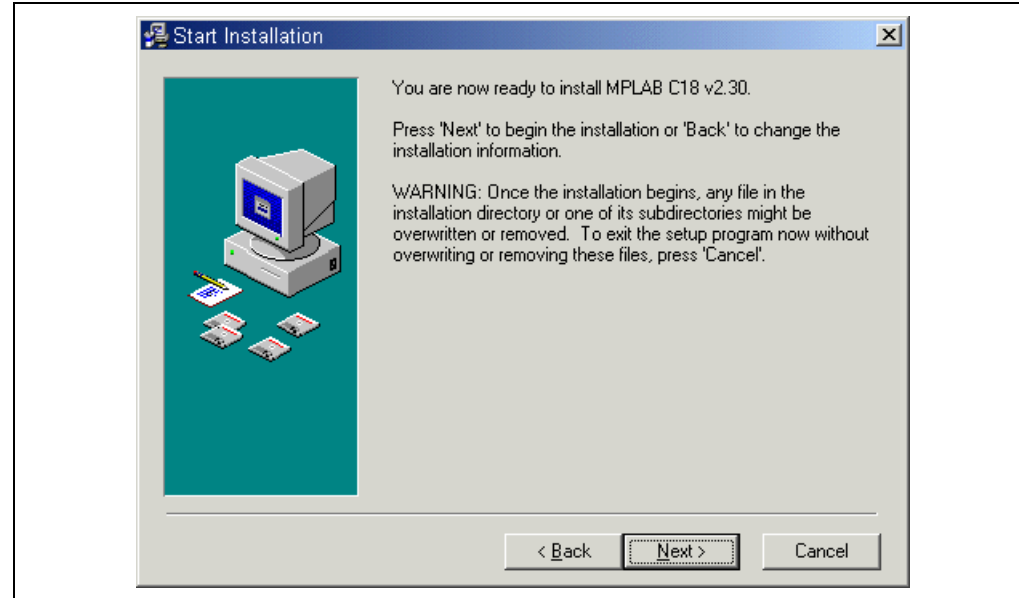
**TABLE 2-2: MPLAB C18 CONFIGURATION OPTIONS**

Configuration	Description
Add MPLAB C18 to PATH environment variable	This adds the path of the MPLAB C18 executable ( <code>mcc18.exe</code> ) and the MPLINK linker executable ( <code>mplink.exe</code> ) to the front of the PATH environment variable. Doing this allows users to launch the newly installed version of MPLAB C18 and the MPLINK linker at the command shell prompt from any directory.
Add MPASM to PATH environment variable	This adds the path of the MPASM executable ( <code>mpasm.exe</code> ) to the front of the PATH environment variable. Doing this allows users to launch the newly installed version of the MPASM assembler at the command shell prompt from any directory.
Add header file path to MCC_INCLUDE environment variable	This adds the path of the MPLAB C18 header file directory to the front of the MCC_INCLUDE environment variable. If this variable does not exist, it is created. MCC_INCLUDE is a list of semicolon-delimited directories that MPLAB C18 will search for in a header file if it cannot find the file in the directory list specified with the <code>-I</code> command-line option. Selecting this configuration option means users will not have to use the <code>-I</code> command-line option when including a standard header file.
Modify PATH and MCC_INCLUDE variables for all users	This option appears only if users are logged into a Windows NT® or Windows 2000 computer as an administrator. Selecting this configuration will perform the modifications to these variables as specified in the three previous options for all users. Otherwise, only the current user's variables will be affected.
Update MPLAB IDE v6.xx to use this MPLAB C18	This option appears only if the MPLAB IDE v6.xx is installed on your system. Selecting this option configures the MPLAB IDE v6.xx to use the newly installed MPLAB C18. This includes using the MPLAB C18 library directory as the default library path for MPLAB C18 projects in the MPLAB IDE v6.xx.
Update MPLAB IDE v6.xx to use this MPLINK linker	This option appears only if the MPLAB IDE v6.xx is installed on your system. Selecting this option configures the MPLAB IDE v6.xx to use the newly installed MPLINK linker.

## 2.3.6 Start Installation

The next dialog screen launches the installation, Figure 2-6. Once the **Next** button is pressed, all files in the installation directory and its subdirectories will be overwritten or removed.

**FIGURE 2-6: MPLAB C18 START INSTALLATION**



## 2.3.7 Complete Installation

MPLAB C18 has now been successfully installed. In the "Installation Complete" dialog, click **Finish**.

For MPLAB C18 to operate properly, it may be necessary to restart the computer. If the "Restart Computer" dialog appears, select **Yes** to restart immediately, or **No** to restart the computer at a later time.

## 2.4 UNINSTALLING MPLAB C18

To uninstall MPLAB C18, open the Windows control panel and launch "Add/Remove Programs". Select the MPLAB C18 installation in the list of programs and follow the directions to remove the program. This will remove the MPLAB C18 directory and its contents from the computer.

**Note:** If uninstalling an upgraded version of MPLAB C18, the entire installation will be removed; MPLAB C18 cannot be "downgraded".

# MPLAB® C18 C Compiler Getting Started

---

NOTES:

---

## Chapter 3. Examples of Use

---

### 3.1 INTRODUCTION

The following examples are intended to illustrate the effective use of MPLAB C18, including how to create and build projects and how to step through programs.

These examples assume that MPLAB C18 and MPLAB IDE v6.xx are installed. Some examples assume MPLAB ICD 2 is installed and connected to a PICDEM™ 2 Plus demo board with a PIC18F452 device. Please refer to the *PIC18FXX2 Data Sheet* (DS39564) for information regarding processor-specific items such as the special function registers, instruction set and interrupt logic.

### 3.2 HIGHLIGHTS

Examples presented in this chapter for using MPLAB C18 include:

- **Example 1** demonstrates how to set up and build a project; run, step and set breakpoints in the example code; and debug the code.
- **Example 2** demonstrates the use of the MPLAB C18 peripheral libraries and the C standard library, as well as the allocation of variables into program memory.
- **Example 3** demonstrates some of the differences between Extended and Non-extended modes.
- **Example 4** demonstrates the allocation of variables in access RAM.
- **Example 5** demonstrates the use of interrupt service routines with MPLAB C18 and provides an example of the use of the MPLAB C18 peripheral libraries.

## 3.3 EXAMPLE 1

This example is designed for use with the MPLAB IDE v6.xx, the MPLAB SIM simulator and the PIC18F452 device. It shows how to set up an MPLAB C18 project in the MPLAB IDE, build the project and step through the source code using the MPLAB SIM simulator. Additionally, running the program using the MPLAB ICD 2 with the PICDEM 2 Plus demo board is demonstrated. The example assumes that the directory `c:\mcc18` is the MPLAB C18 installation directory.

Here is the source code for the example:

```
#include <p18cxxx.h>    /* for TRISB and PORTB
                        declarations */
/* Set configuration bits for use with ICD2 / PICDEM2 PLUS Demo Board:
 * - set HS oscillator
 * - disable watchdog timer
 * - disable low voltage programming
 */

#pragma romdata CONFIG
_CONFIG_DECL(_CONFIG1H_DEFAULT & _OSC_HS_1H,
             _CONFIG2L_DEFAULT,
             _CONFIG2H_DEFAULT & _WDT_OFF_2H,
             _CONFIG3H_DEFAULT,
             _CONFIG4L_DEFAULT & _LVP_OFF_4L,
             _CONFIG5L_DEFAULT,
             _CONFIG5H_DEFAULT,
             _CONFIG6L_DEFAULT,
             _CONFIG6H_DEFAULT,
             _CONFIG7L_DEFAULT,
             _CONFIG7H_DEFAULT);
#pragma romdata

int counter;
void main (void)
{
    counter = 1;
    TRISB = 0;          /* configure PORTB for output */
    while (counter <= 15)
    {
        PORTB = counter; /* display value of 'counter'
                           on the LEDs */
        counter++;
    }
}
```

- TRISB and PORTB are special function registers on the PIC18F452 device. The PORTB pins are connected to the LEDs on the PICDEM 2 demo board; the TRISB pins configure the PORTB pins for input (1) or output (0).
- The configuration bits need to be set appropriately; this is done by invoking the macro `_CONFIG_DECL` with settings for each configuration byte. This includes specifying the oscillator used on the PICDEM 2 Plus demo board; in the example, the crystal (`_OSC_HS_1H`) is used. Additionally, MPLAB ICD 2 requires that the Watchdog Timer and low-voltage programming be disabled (`_WDT_OFF_2H` and `_LVP_OFF_4L`, respectively). The macros to set the configuration bits are defined in the processor-specific header file; see the *MPLAB® C18 C Compiler User's Guide* (DS51288) for details.

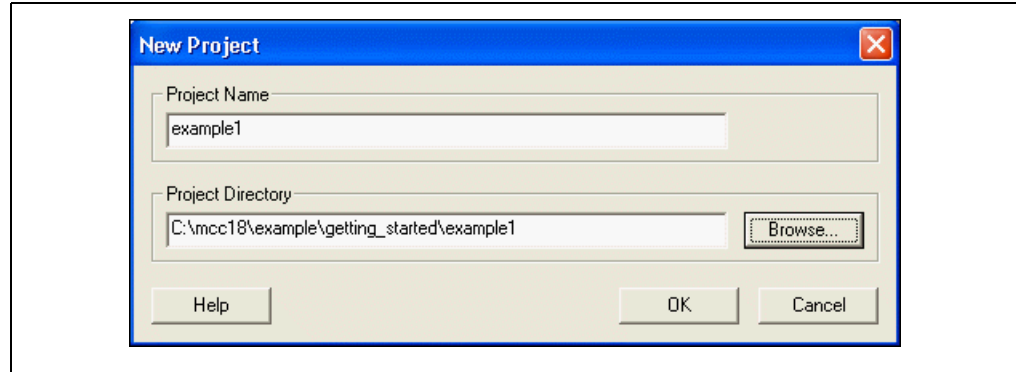


## 3.3.1 Setting Up the Project

Select *Project>New* to create a new project. Then enter the name and directory of the project in the dialog that displays and click **OK**, Figure 3-1.

If the examples with MPLAB C18 were installed, then the `example\getting_started\example1` subdirectory of the MPLAB C18 installation will already contain the source file for this example.

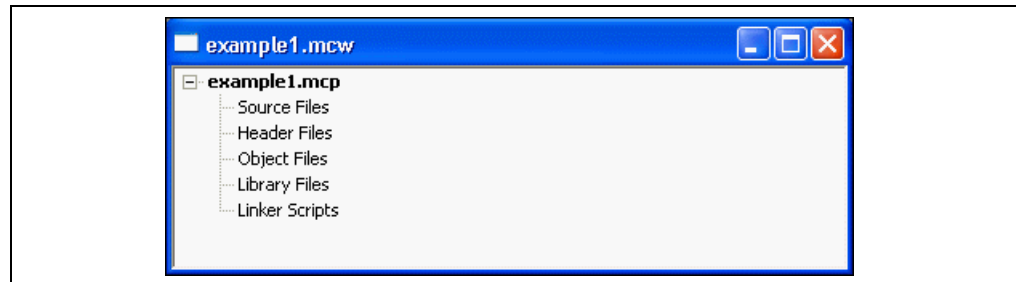
**FIGURE 3-1: NEW PROJECT DIALOG**



**Note:** The project name does not have to be the same as the directory name.

The project tree will now be visible with a branch for each type of project file.

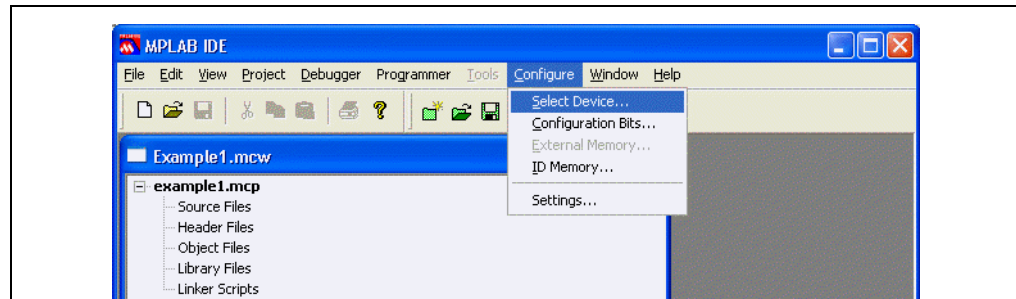
**FIGURE 3-2: PROJECT TREE**



## 3.3.2 Select Target Processor

The target processor must be selected before anything else is done with the project. This is accomplished by choosing *Configure>Select Device*.

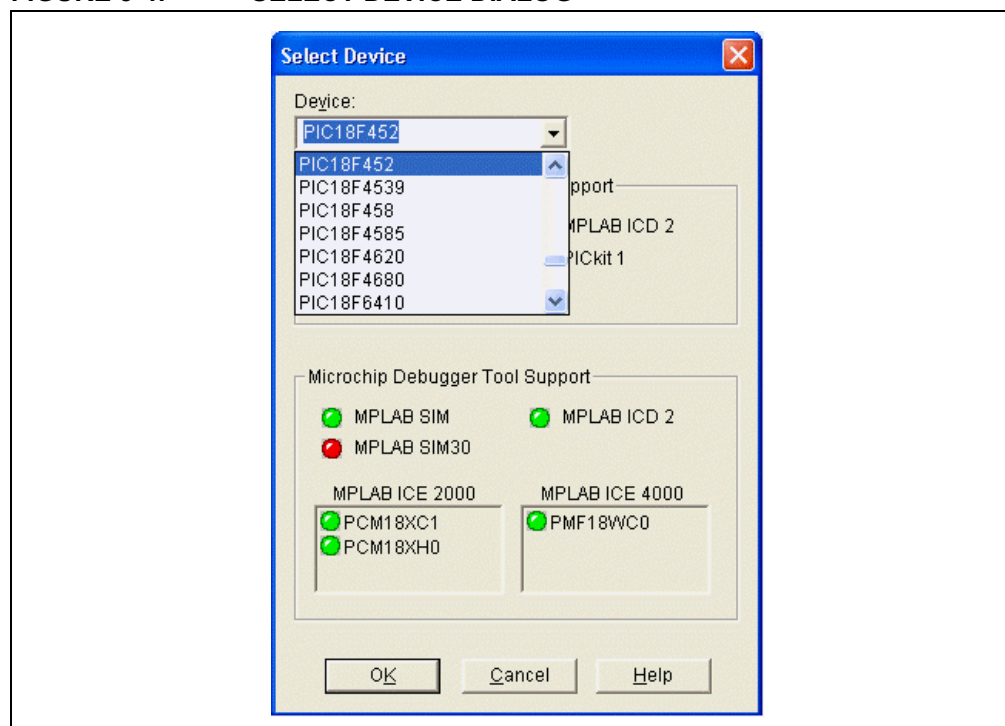
**FIGURE 3-3: SELECT DEVICE OPTION**



# MPLAB® C18 C Compiler Getting Started

For this example, the PIC18F452 device will be used. Select the device and click **OK**.

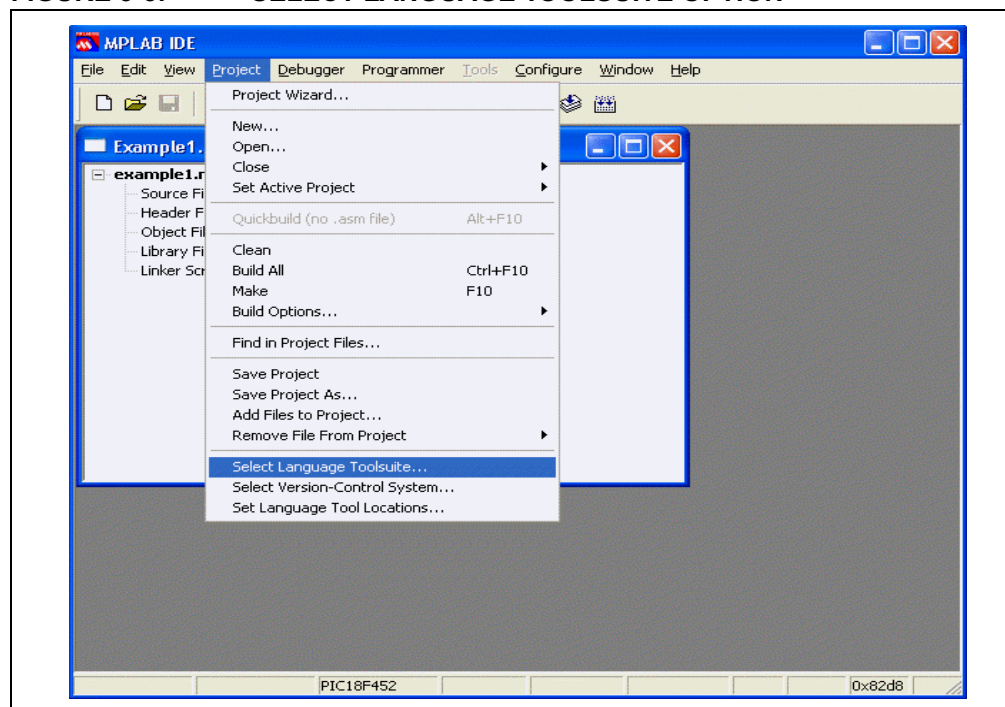
**FIGURE 3-4: SELECT DEVICE DIALOG**



## 3.3.3 Select Project Settings

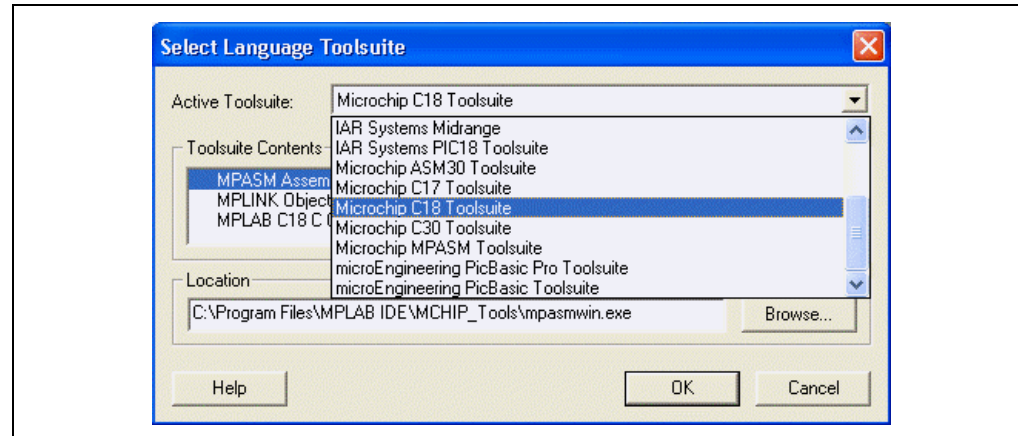
The MPLAB IDE needs to know which compiler and linker to use. To select MPLAB C18 and the MPLINK linker, first choose *Project>Select Language Toolsuite*.

**FIGURE 3-5: SELECT LANGUAGE TOOLSUITE OPTION**



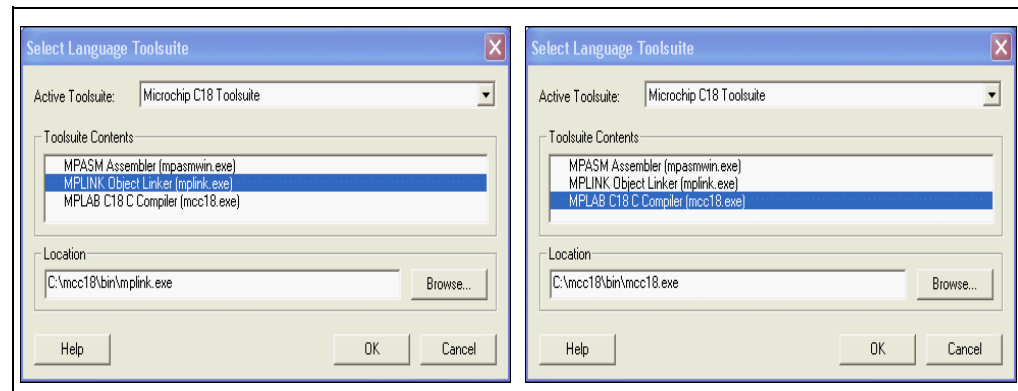
A dialog appears to select the language toolsuite. To use the language tools that include MPLAB C18 and the MPLINK linker, select “Microchip C18 Toolsuite” as the active toolsuite.

**FIGURE 3-6: SELECT LANGUAGE TOOLSUITE DIALOG**



Then, select “MPLINK Object Linker” and “MPLAB C18 C Compiler” and make sure that the paths are to the newly installed versions of the executables, `mplink.exe` and `mcc18.exe`, respectively. Click **OK**.

**FIGURE 3-7: TOOLSUITE CONTENTS**

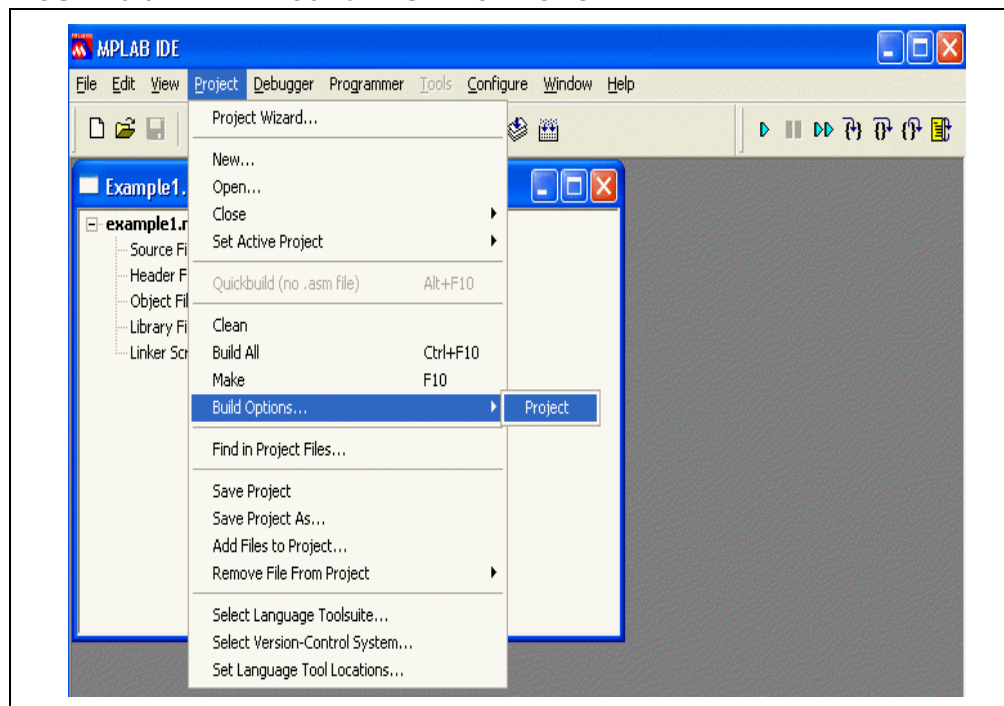


**Note:** If the user chose to update the MPLAB IDE 6.xx to use the newly installed compiler and linker in the MPLAB C18 setup program, these paths should already be set up correctly.

# MPLAB® C18 C Compiler Getting Started

The next step is to set the command-line options for the compiler and linker. Choose *Project>Build Options>Project*.

**FIGURE 3-8: PROJECT BUILD OPTIONS**



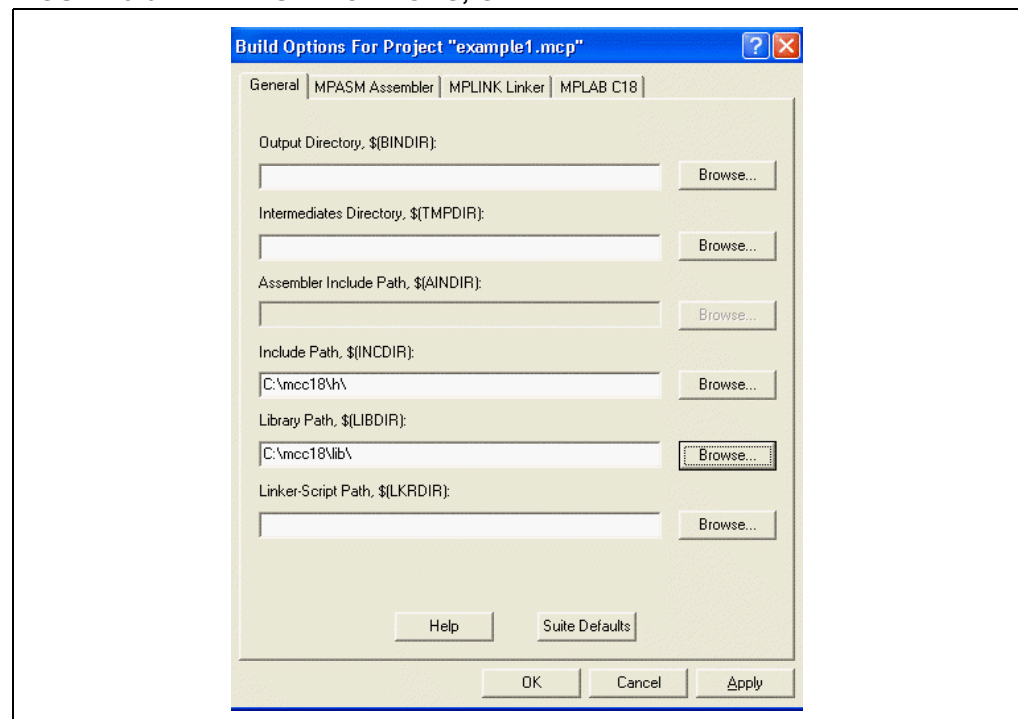
Enter the paths of the header file and library subdirectories of the MPLAB C18 installation directory on the “General” tab as shown in Figure 3-9. You can type the paths or click on **Browse** to designate the path. MPLAB C18 will search for included .h files in the specified header file directory. The MPLINK linker will search for object and library files, including those specified in the linker script, in the library directory.

“Output Directory” is the final destination for files that result only from a complete build of the project — the .cod, .cof and .hex files. Leave “Output Directory” blank; as a result, the output file (example1.cof) will be placed in the project directory.

“Intermediates Directory” is where the object files produced by the compiler will be placed. Leave this entry blank as well; as a result, the object file (example1.o) will be placed in the same directory as the source file.

The MPLINK linker will search the directory specified in “Linker-Script Path” for linker scripts. Since the location of the linker script will be specified when it is added to the project tree, this entry can also be left blank for this example.

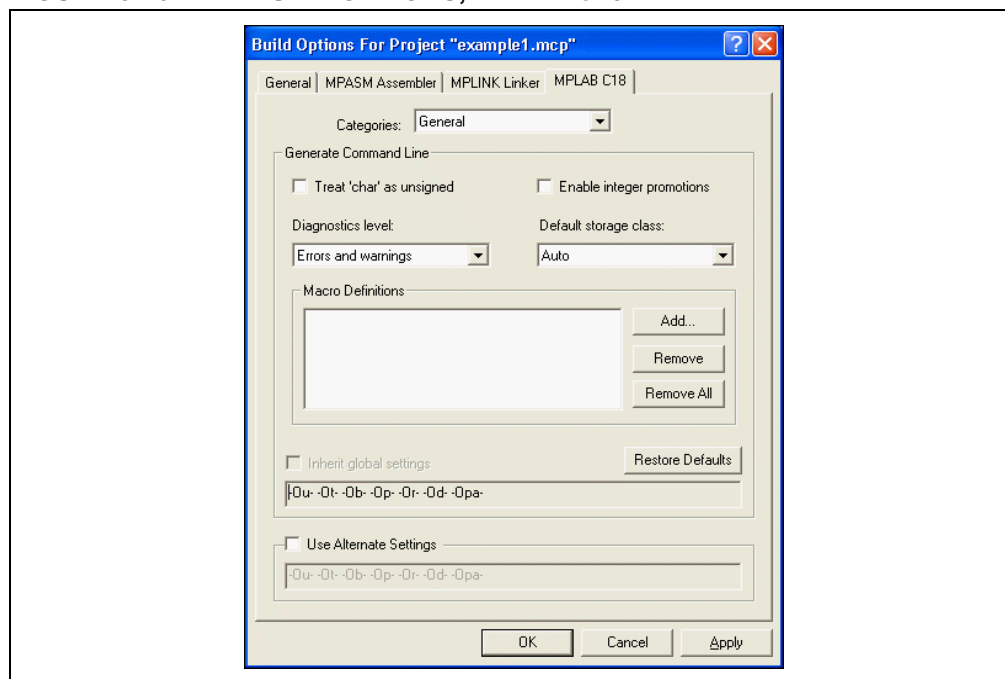
**FIGURE 3-9: BUILD OPTIONS, GENERAL TAB**



## 3.3.4 Select Compiler and Linker Settings

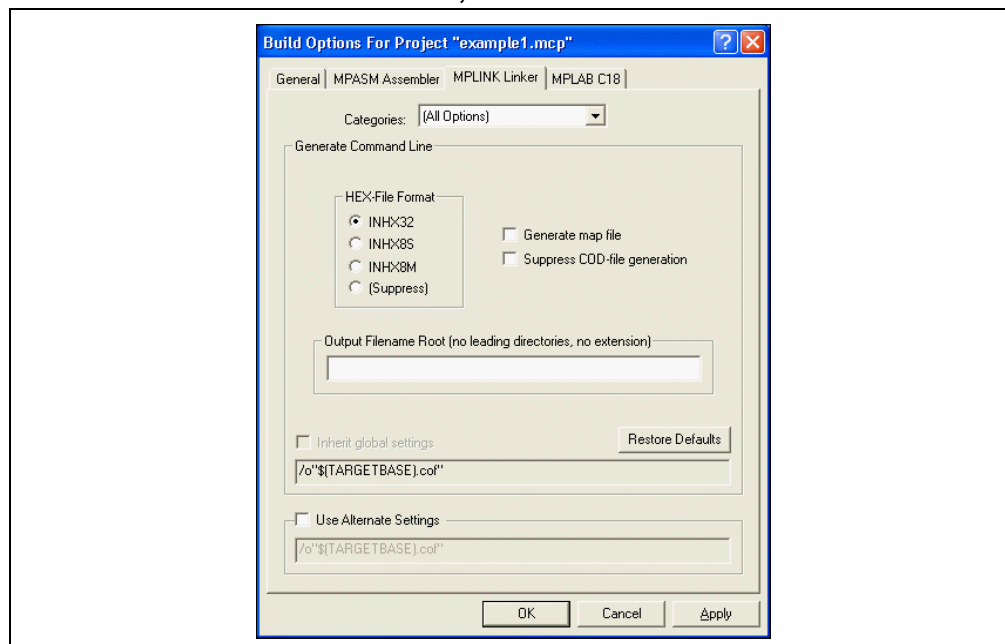
The various command-line options which are passed to the compiler and linker can be set on the “MPLAB C18” and “MPLINK Linker” tabs, respectively, in the Build Options window. For this example, the default command-line options for MPLAB C18 will be accepted.

**FIGURE 3-10: BUILD OPTIONS, MPLAB C18 TAB**



By default, when the MPLINK linker is run from the MPLAB IDE, it will not generate a map (example1.map) file. To change this, select “Generate map file” on the “MPLINK Linker” tab. Click **OK**. The default settings will be used for the remainder of the command-line options.

**FIGURE 3-11: BUILD OPTIONS, MPLINK LINKER TAB**

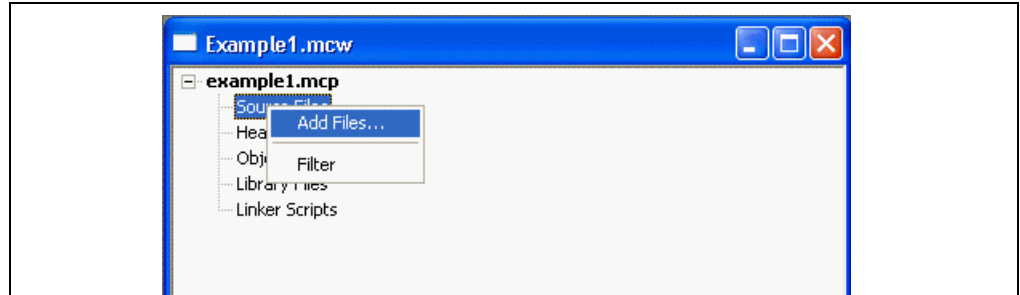




## 3.3.5 Add Files to Project

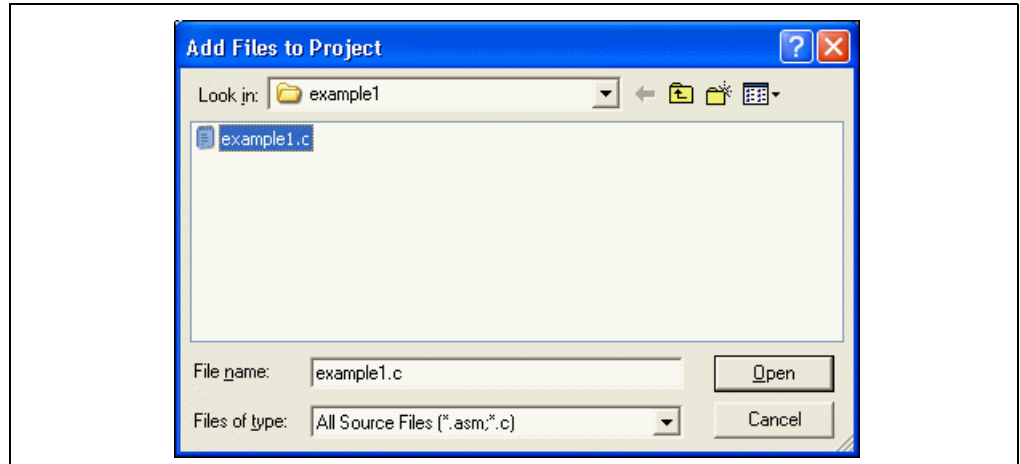
The C source file must be added to the project. Click the right mouse button on “Source Files” in the project window. Select “Add Files”.

**FIGURE 3-12: SOURCE FILES, ADD FILES OPTION**



If `c:\mcc18\example\getting_started\example1` was chosen as the project directory, the source file `example1.c` already exists there. Browse to this directory and select the file `example1.c`. Click **Open** to add the file to the project.

**FIGURE 3-13: ADD FILES DIALOG**



The source file should appear in the project tree.

**FIGURE 3-14: SOURCE FILE IN PROJECT TREE**

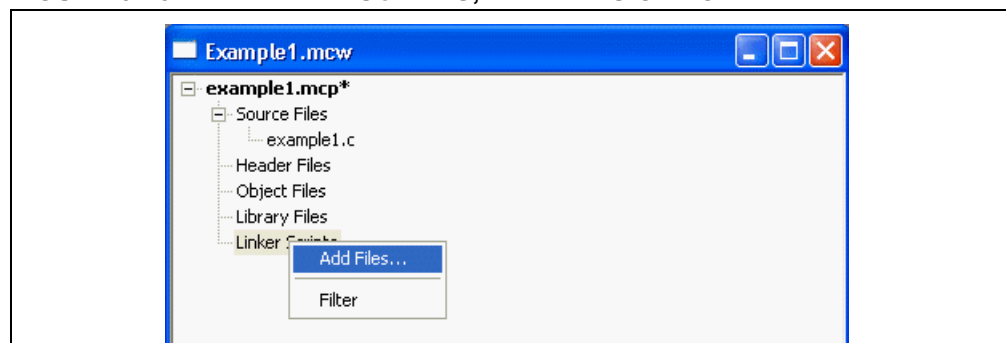


# MPLAB® C18 C Compiler Getting Started

The header file is specified in the C source file; therefore no file needs to be added to “Header Files” in the project tree. A header file may be added to “Header Files” for convenient viewing of the file, but it is only required that the header file be included in the C source code to build the project. The required startup module, standard library and processor library are specified in the linker script, and so no file needs to be added to “Object Files” or “Library Files” in the project tree. If there were other object files or library files to link in the project, they would be added under these branches.

The MPLINK linker requires a linker script to be specified. Click the right mouse button on “Linker Scripts” in the project window, and select **Add Files**.

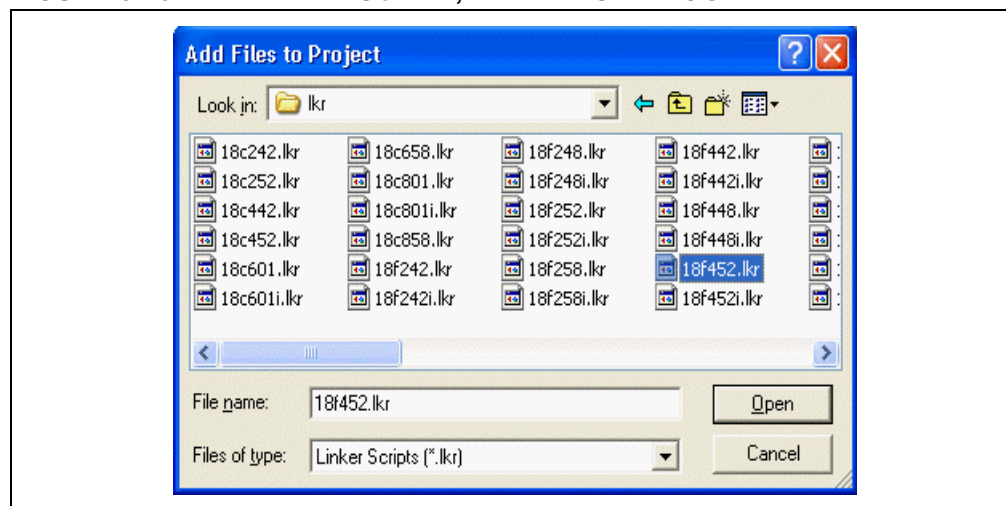
**FIGURE 3-15: LINKER SCRIPTS, ADD FILES OPTION**



Use the linker script 18f452.lkr in the lkr subdirectory of the MPLAB C18 installation directory. This script is for the PIC18F452 device.

Click **Open** to add the file to the project tree.

**FIGURE 3-16: LINKER SCRIPT, ADD FILES DIALOG**

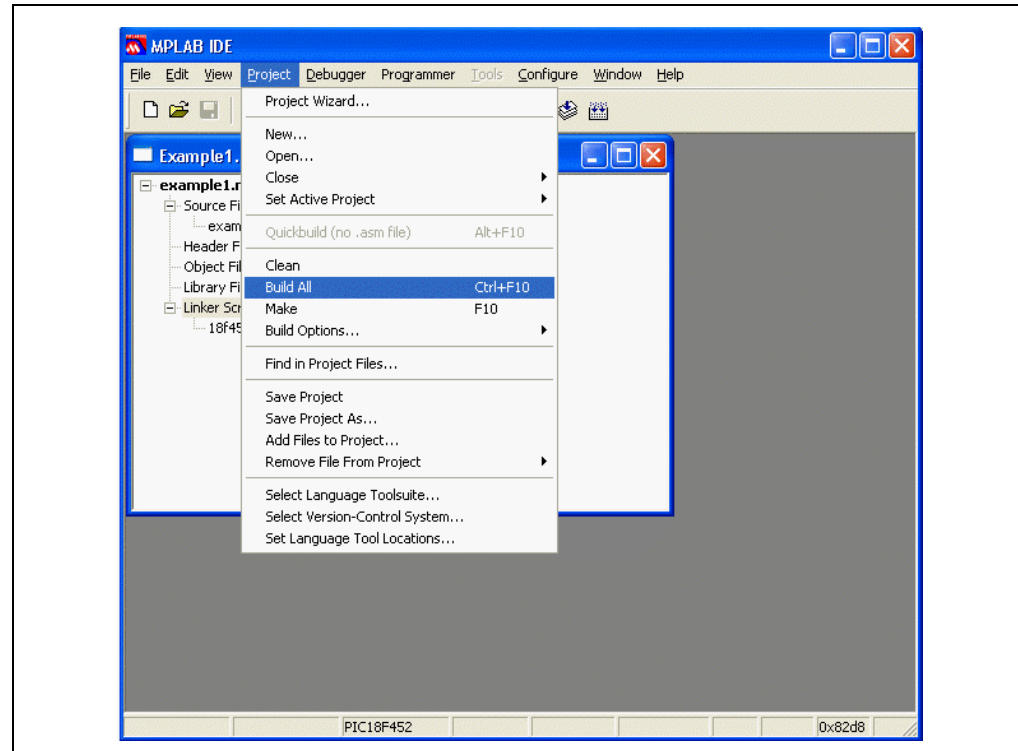




## 3.3.6 Build the Project

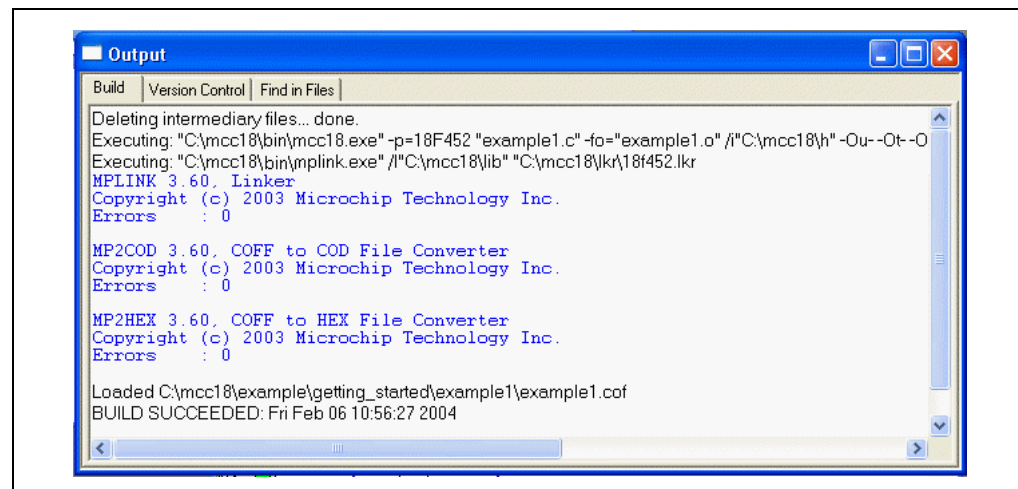
Select **Project>Build All** to compile and link the project. If there are any error or warning messages, they will appear in the output window.

**FIGURE 3-17: BUILD ALL OPTION**



For this example, the output window should display no errors and a message stating the output file was successfully built should be visible. If there were any errors, check to see that the content of the source file matches the program text displayed at the beginning of 3.3 “Example 1”.

**FIGURE 3-18: OUTPUT FOR EXAMPLE 1**

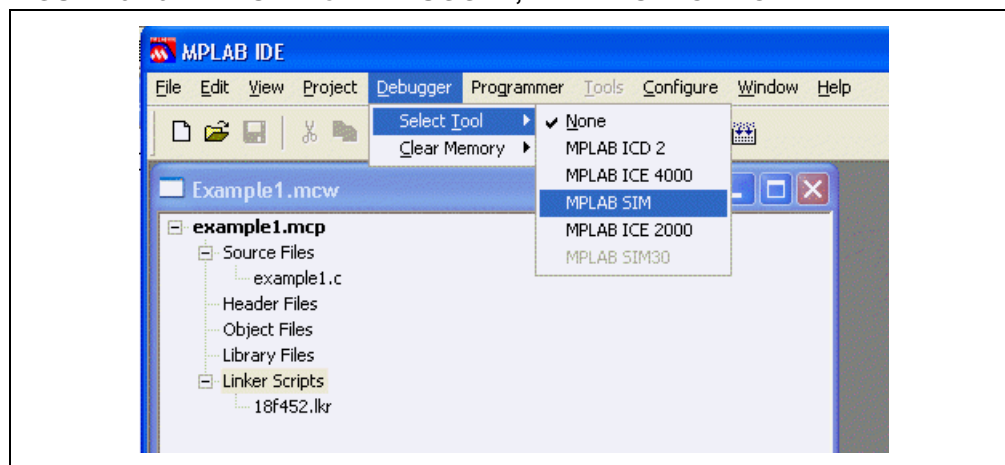


# MPLAB® C18 C Compiler Getting Started

## 3.3.7 Debugging with the MPLAB SIM Simulator

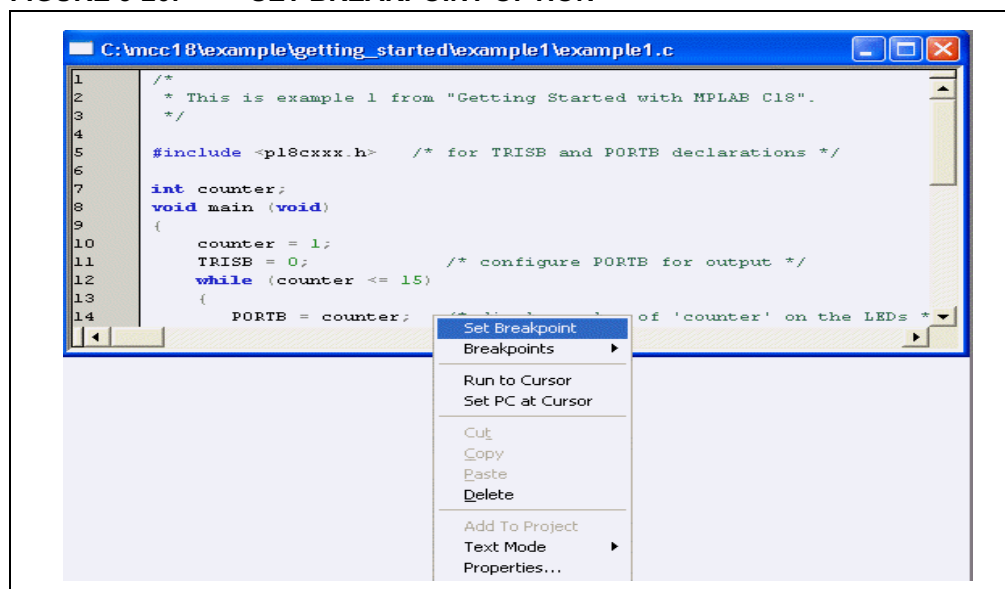
With the MPLAB SIM Simulator, breakpoints can be set in the source code to observe the value of variables with a watch window. First, make sure that the MPLAB SIM Simulator is selected as the debugging tool by selecting *Debugger>Select Tool>MPLAB SIM*.

FIGURE 3-19: SELECT DEBUGGER, MPLAB SIM OPTION



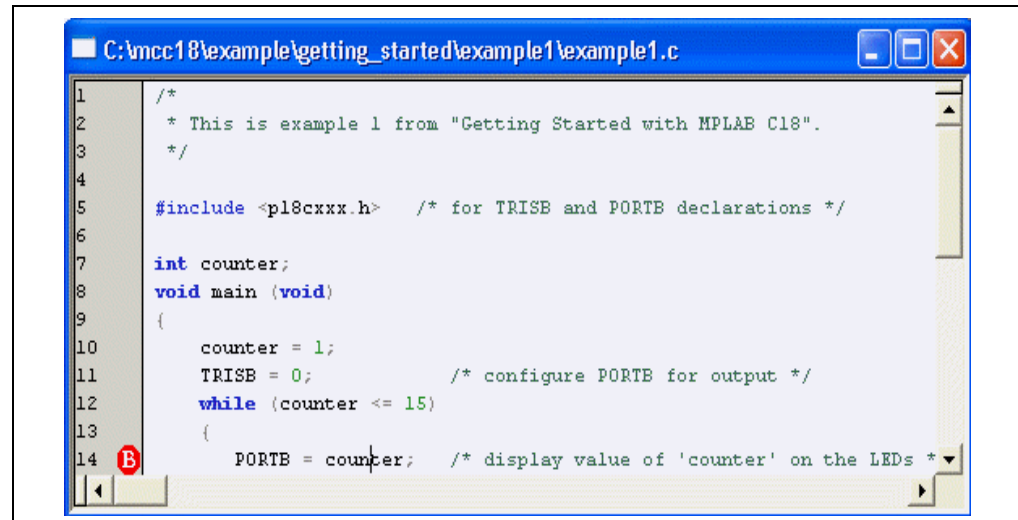
Open the source file by double clicking on it in the project tree. In the source file, place the cursor over the line where the breakpoint is desired to be set, and click the right mouse button. Select "Set Breakpoint".

FIGURE 3-20: SET BREAKPOINT OPTION



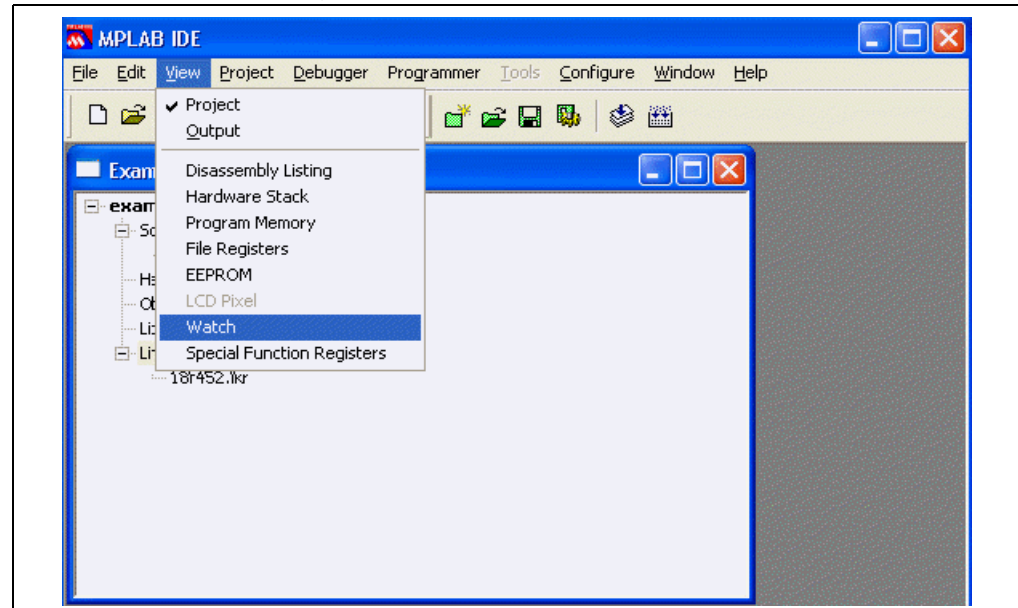
The red dot in the gutter along the side of the source window indicates that the breakpoint has been set and is enabled.

**FIGURE 3-21: BREAKPOINT ENABLED**



To open a watch window on the variable counter, select View>Watch.

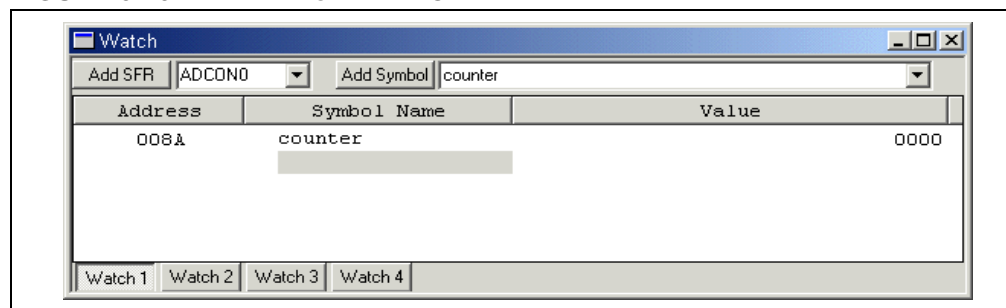
**FIGURE 3-22: VIEW, WATCH WINDOW OPTION**




# MPLAB® C18 C Compiler Getting Started

Select `counter` from the menu next to **Add Symbol**, and click **Add Symbol**.

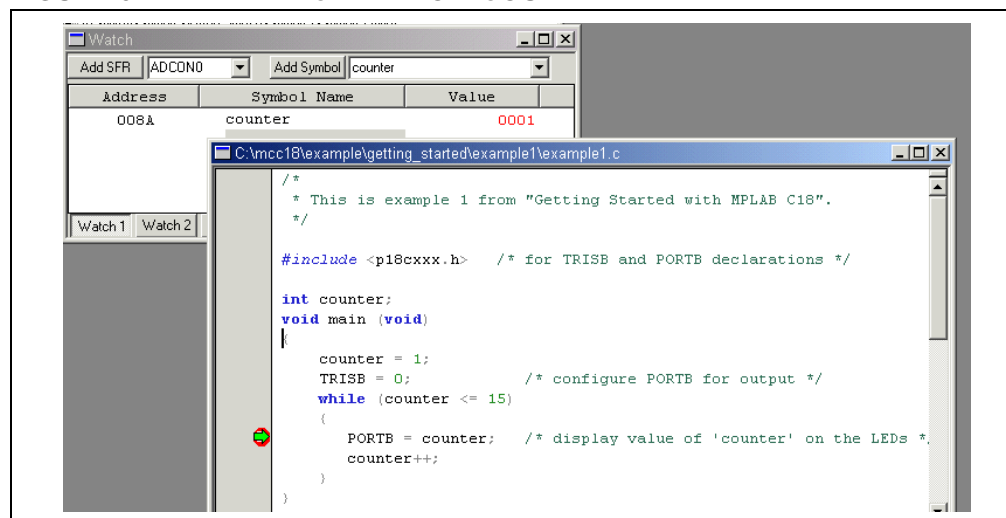
**FIGURE 3-23: WATCH WINDOW**



Click **Run** on the toolbar  to run the program.

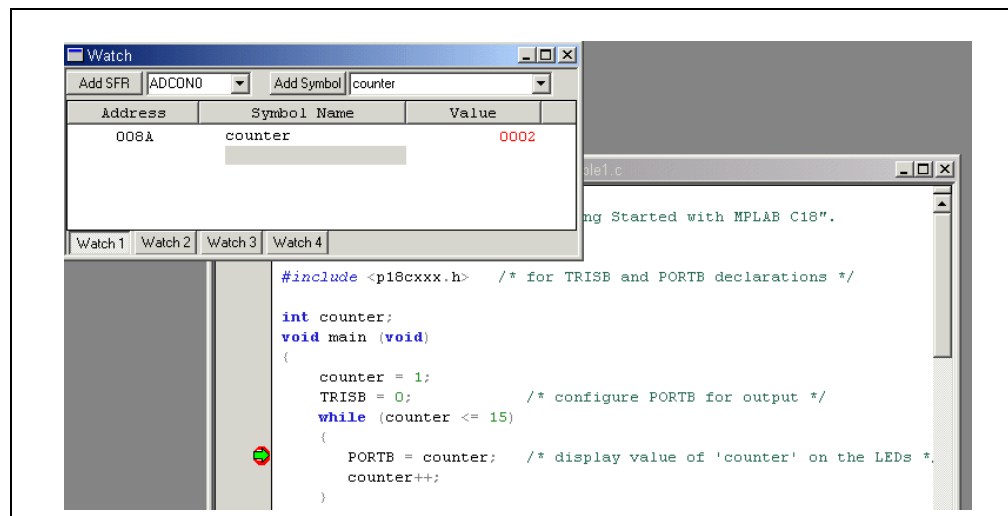
The program should halt just before the statement at the breakpoint is executed. The green arrow in the gutter of the source window points to the next statement to be executed. The watch window should show `counter` with a value of 1.

**FIGURE 3-24: WATCH WINDOW COUNTER AT 1**



Click **Run** again to continue the program. Execution should halt again at the breakpoint. The watch window should show `counter` with a value of 2.

**FIGURE 3-25: WATCH WINDOW COUNTER AT 2**



To step through the source code one statement at a time, use **Step Into** on the toolbar. As each statement executes, the green arrow in the gutter of the source window moves to the next statement to be executed.

If the program is running, it can be halted by clicking **Halt** on the toolbar .

## 3.3.8 Map and Listing Files

The map file (example1.map) and listing file (example1.lst) are present in the project directory and may be opened by selecting *File>Open*, and then browsing to the project directory. These files provide additional information which may be useful in debugging, such as details of allocation of variables and the correspondence between machine code and source code. For example, the map file shows that the variable counter has been allocated to address 0x80 in data memory, and it was defined in example1.c as a non-static global variable, thus giving it external linkage (visibility to other modules).

### EXAMPLE 3-1: MAP FILE

Symbols - Sorted by Address			
Name	Address	Location	Storage File
-----	-----	-----	-----
counter	0x00008a	data	extern c:\mcc18\getting_started\example1\example1.c

The listing file shows the machine code generated for each statement the main function. For each instruction, its address, raw value and disassembly is displayed.

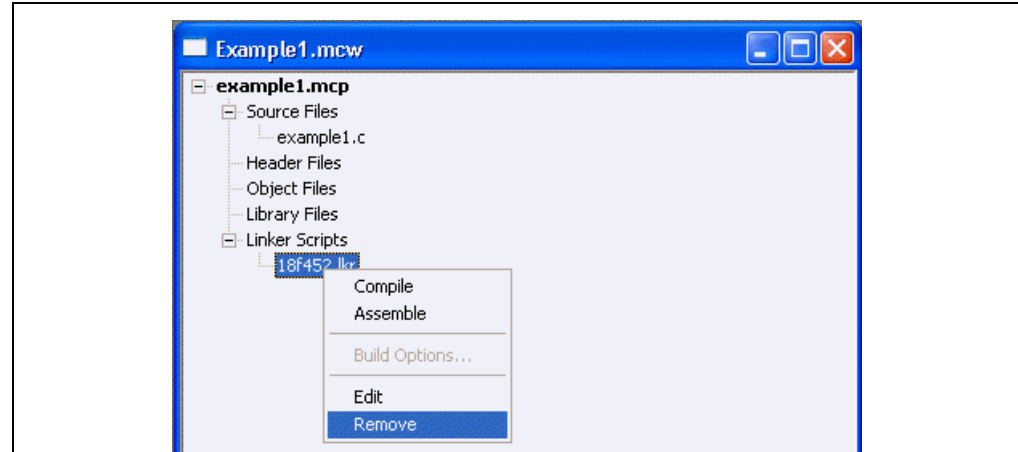
### EXAMPLE 3-2: LISTING FILE

Address	Value	Disassembly	Source
-----	-----	-----	-----
			#include <p18cxxx.h> /* for TRISB and PORTB declarations */
			int counter;
			void main (void)
			{
			counter = 1;
0000e2	0e01	MOVLW 0x1	
0000e4	0100	MOVLB 0x0	
0000e6	6f8a	MOVWF 0x8a,0x1	
0000e8	6b8b	CLRF 0x8b,0x1	
0000ea	6a93	CLRF 0x93,0x0	TRISB = 0; /* configure PORTB for output */
0000ec	518b	MOVF 0x8b,0x0,0x1	while (counter <= 15)
0000ee	0a00	XORLW 0x0	
0000f0	ae08	BTFSS 0xe8,0x7,0x0	
0000f2	d002	BRA 0xf8	
0000f4	358b	RLCF 0x8b,0x0,0x1	
0000f6	d005	BRA 0x102	
0000f8	0e0f	MOVLW 0xf	
0000fa	80d8	BSF 0xd8,0x0,0x0	
0000fc	558a	SUBFWB 0x8a,0x0,0x1	
0000fe	0e00	MOVLW 0x0	
000100	558b	SUBFWB 0x8b,0x0,0x1	
000102	e306	BNC 0x110	
00010e	d7ee	BRA 0xec	
			{
000104	c08a	MOVFF 0x8a,0xf81	PORTB = counter; /* display 'counter' on the LEDs */
000106	ff81		
000108	2b8a	INCF 0x8a,0x1,0x1	counter++;
00010a	0e00	MOVLW 0x0	
00010c	238b	ADDWFC 0x8b,0x1,0x1	
			}
000110	0012	RETURN 0x0	}

## 3.3.9 Debugging with the MPLAB ICD 2

The MPLAB ICD 2 can be used to actually program the device and step through the application. To do this, the project must be rebuilt with a linker script designed for use with the MPLAB ICD 2. In the project window, click the right mouse button on the file `18f452.lkr` under “Linker Scripts”, and click **Remove**.

**FIGURE 3-26: LINKER SCRIPTS, REMOVE OPTION**

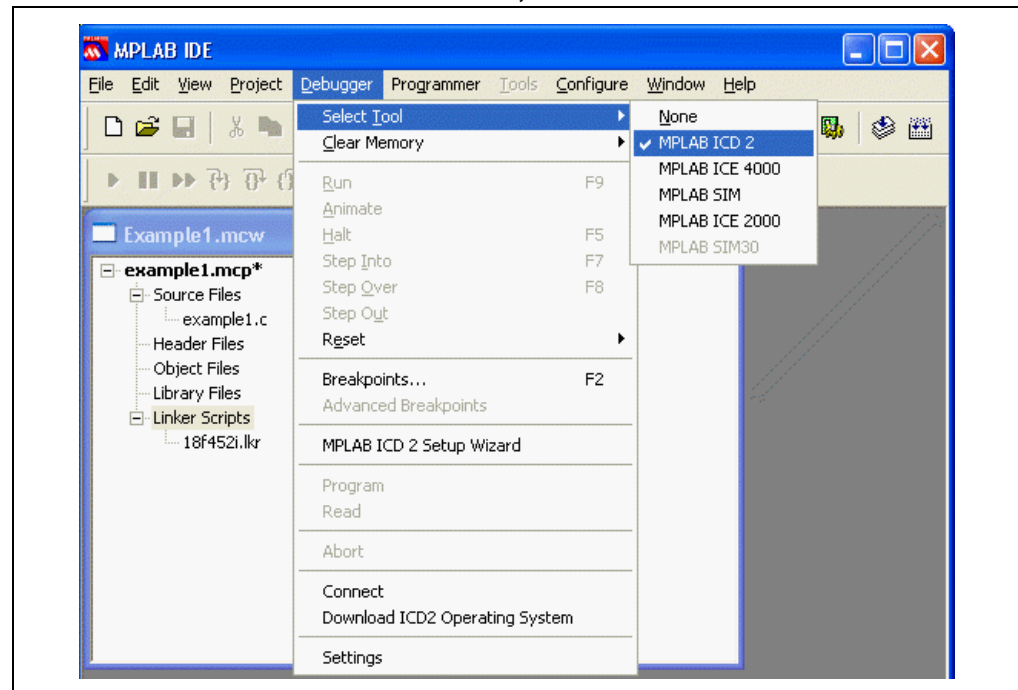


Add the linker script file `18f452i.lkr` from the `lkr` subdirectory of the MPLAB C18 installation directory under “Linker Scripts” in the project tree. This linker script allocates memory for resources used by the MPLAB ICD 2. The ‘i’ in the file’s name indicates this linker script is for use with the MPLAB ICD 2.

Rebuild the project by selecting *Project>Build All*.

To use the MPLAB ICD 2, select *Debugger>Select Tool>MPLAB ICD 2*.

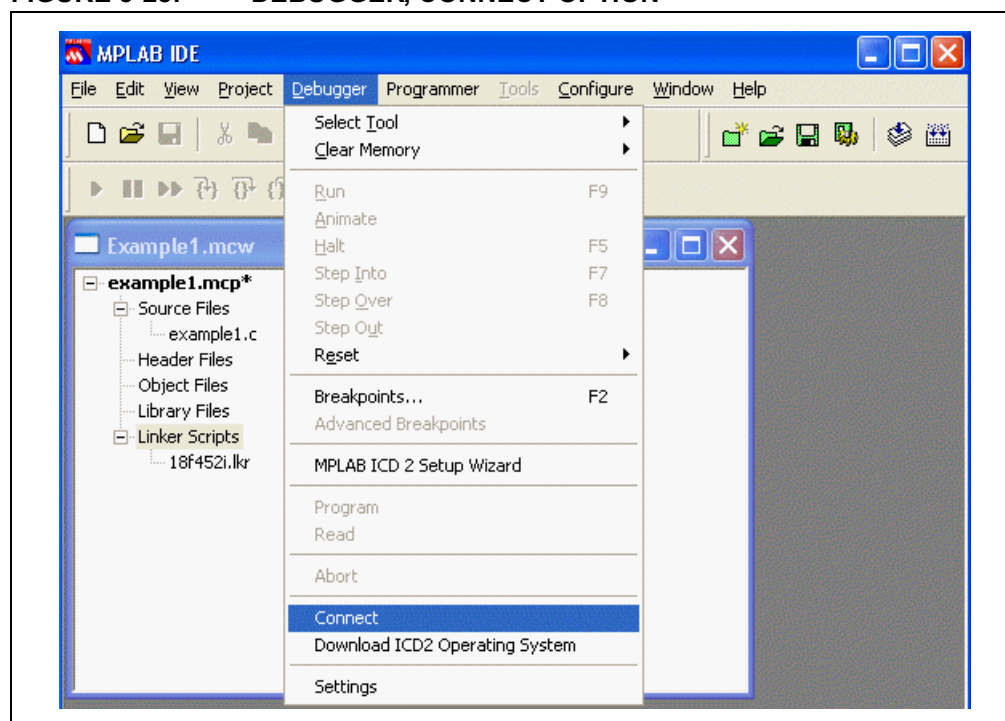
**FIGURE 3-27: SELECT DEBUGGER, MPLAB ICD 2 OPTION**



# MPLAB® C18 C Compiler Getting Started

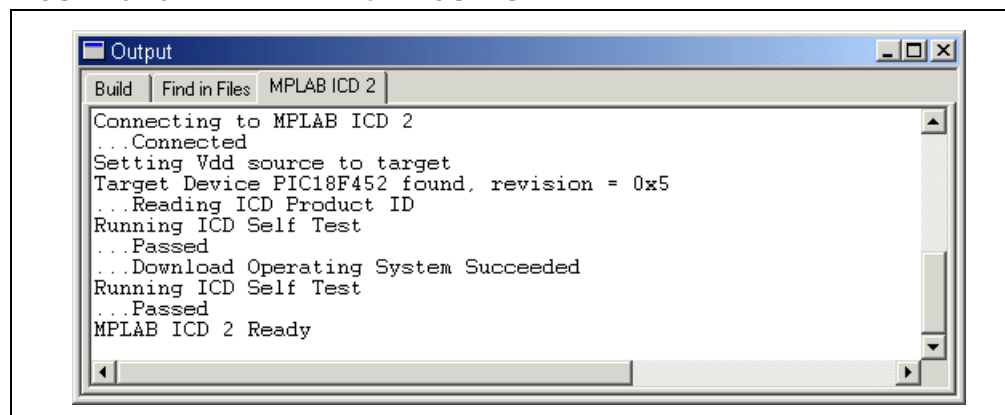
To connect to the MPLAB ICD 2, choose *Debugger>Connect*. See the MPLAB ICD 2 documentation for information on how to configure the MPLAB ICD 2 connection settings.

**FIGURE 3-28: DEBUGGER, CONNECT OPTION**



The output window should show that the MPLAB ICD 2 passed its self-test and is ready to be programmed. If any errors occur, refer to documentation for the MPLAB ICD 2.

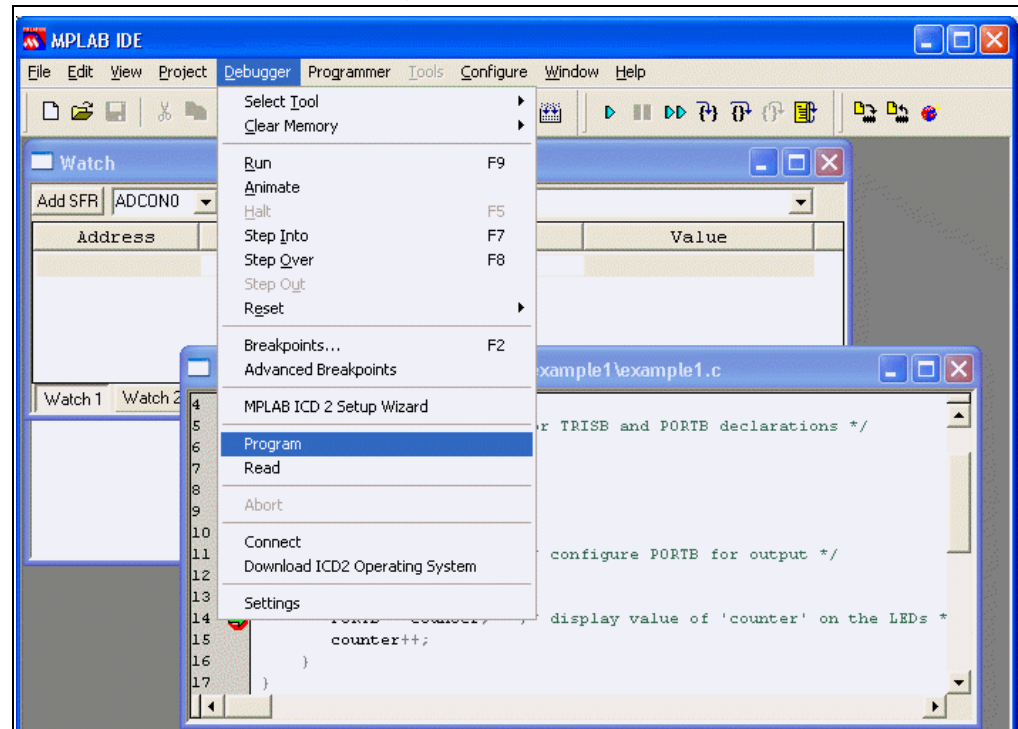
**FIGURE 3-29: MPLAB ICD 2 OUTPUT**





To program the device, select *Debugger>Program*.

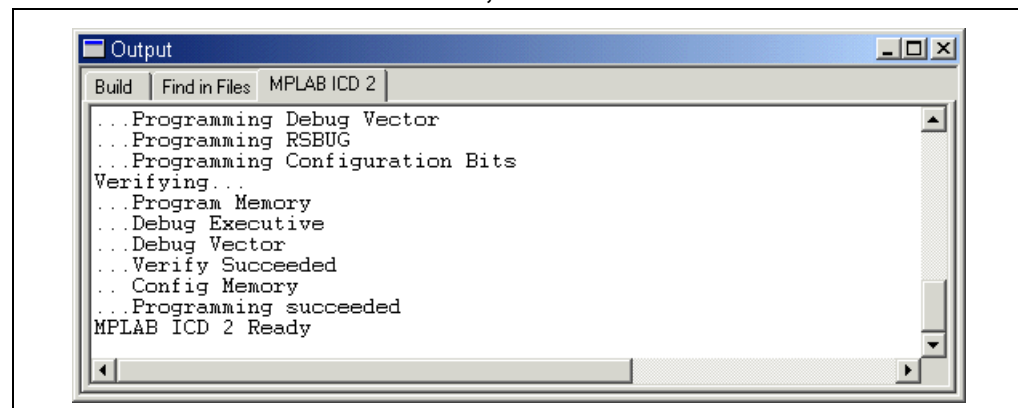
**FIGURE 3-30: DEBUGGER, PROGRAM OPTION**



**Note:** Programming the device requires reducing the program memory space available to allow for MPLAB ICD 2 resources, disabling low voltage programming and disabling the Watchdog Timer. If an MPLAB ICD 2 Warning dialog is received concerning any of these issues, simply click **OK** to proceed.

The output window should show that the programming operation succeeded.

**FIGURE 3-31: OUTPUT WINDOW, PROGRAMMING SUCCEEDED**

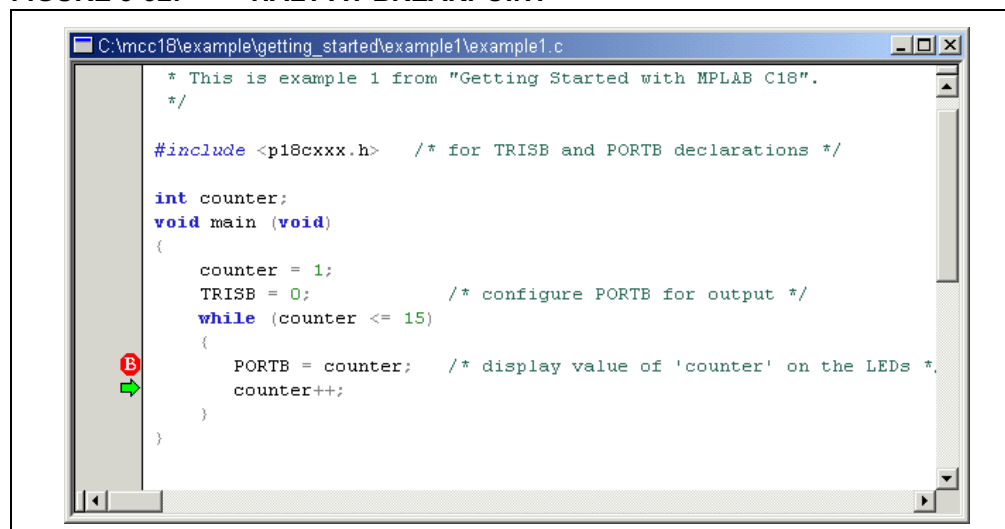


# MPLAB® C18 C Compiler Getting Started

A breakpoint may be set in the source file as demonstrated for the MPLAB SIM Simulator. When **Run** on the toolbar is clicked, the program halts immediately after the statement where the breakpoint has been executed.

**Note:** This is different than the MPLAB SIM Simulator, which halts before the statement where the breakpoint is executed. The green arrow points to the next statement to be executed.

**FIGURE 3-32: HALT AT BREAKPOINT**



The PORTB register has been assigned the value of 1. The LEDs on the PICDEM 2 Plus demo board, which are multiplexed with the PORTB pins, should display the binary representation of 1.

**Note:** The J6 connection on the demo board must be jumpered in order to connect the PORTB pins with the LEDs.

Each time **Run** on the toolbar is clicked, execution halts after the assignment to PORTB and the value on the LEDs should reflect the incremented value of the counter.

## 3.4 EXAMPLE 2

This example is designed for use with the MPLAB IDE v6.xx, the MPLAB ICD 2, the PICDEM 2 Plus demo board and the PIC18F452 device. It demonstrates the use of the MPLAB C18 peripheral libraries and the C standard library. It also demonstrates the allocation of variables into program memory. The program cycles through a list of strings, each representing a number from 0 to 15. Each string is converted into its integer representation for display on the LEDs. The program pauses after displaying each number to give the user an opportunity to observe the LEDs. For this program, the J6 connection on the PICDEM 2 Plus demo board must be jumpered.

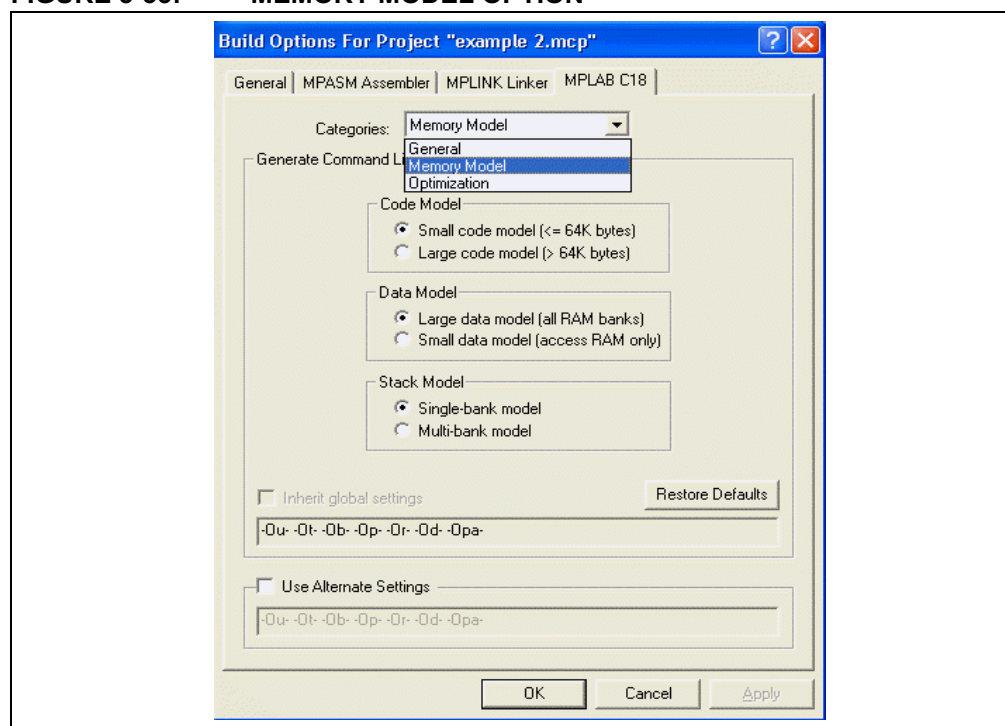
- MPLAB C18 places string literals in program memory; therefore, the `rom` keyword is required in the declaration of the array `string_table`. The `const` keyword alone will not place the data in program memory; the `rom` keyword is required. Since program memory in general cannot be modified without additional specialized code, the `const` keyword is appropriate.
- The configuration bits need to be set appropriately; this is done by invoking the macro `_CONFIG_DECL` with settings for each configuration byte. This includes specifying the oscillator used on the PICDEM 2 Plus demo board; in the example, the crystal (`_OSC_HS_1H`) is used. Additionally, MPLAB ICD 2 requires that the Watchdog Timer and low-voltage programming be disabled (`_WDT_OFF_2H` and `_LVP_OFF_4L`, respectively). Finally, since this example exclusively uses the MPLAB ICD 2, background debugging should always be enabled (`_DEBUG_ON_4L`). The macros to set the configuration bits are defined in the processor-specific header file; see the *MPLAB® C18 C Compiler User's Guide* (DS51288) for details.
- The standard C library function `atoi`, which converts the string to an integer representation, expects a character pointer located in data memory. However, as the string literals are in program memory, they must be copied to data memory first. The function `strcpypgm2ram`, which is an MPLAB C18 variant of `strcpy`, does just that.
- The `PORTB` register, which is connected to the LEDs on the PICDEM 2 demo board, and the `TRISB` register, which configures the `PORTB` pins for input or output, are declared in the processor-specific header file `p18f452.h`.
- MPLAB C18 provides several functions which provide delays of various lengths, such as `Delay10RTCYx` used below. See the header file `delays.h` for more details.

# MPLAB® C18 C Compiler Getting Started

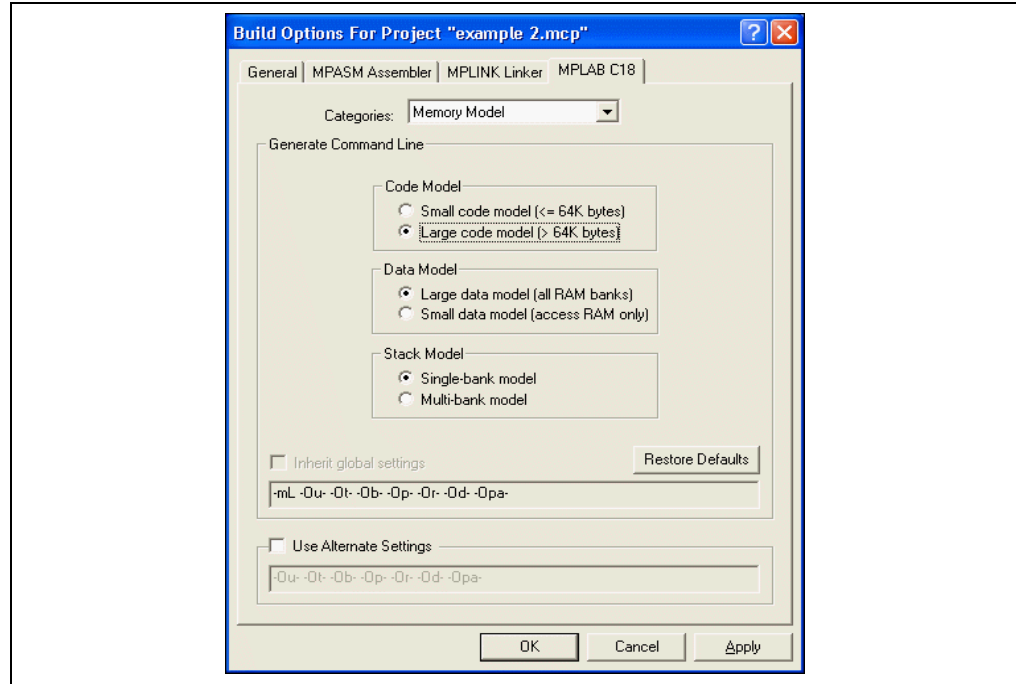
This example can be built and linked in the MPLAB IDE v6.xx and used with the MPLAB ICD 2 by following the steps in Example 1.

**Note:** When compiling with the small code model, MPLAB C18 may emit a warning about a type qualifier mismatch in an assignment with respect to the call of `strcpypgm2ram`. This happens because the second argument passed to the function is a pointer to near program memory, while the parameter's type in the function prototype is a far program memory pointer. Since the conversion from a near pointer to a far pointer is always safe, this warning can be ignored. Alternatively, the warning may be eliminated by compiling with the large code model (at the expense of possibly a larger code image). To do this, choose "Build Options" and then "Project" from the "Project" menu. Select "MPLAB C18" and choose "Memory Model" from the drop-down menu. Finally, select the large code model.

FIGURE 3-33: MEMORY MODEL OPTION



**FIGURE 3-34: LARGE CODE MODEL OPTION**



```
#include <string.h>    /* for 'strcpypgm2ram'    */
#include <stdlib.h>    /* for 'atoi'          */
#include <delays.h>    /* for 'Delay10KTCYx'  */
#include <p18cxxx.h>   /* for 'PORTB' and 'TRISB' */

/* MPLAB C18 places string literals in program memory */
#define STRING_TABLE_SIZE 16
const rom char *string_table[STRING_TABLE_SIZE] =
{
    "0", "1", "2", "3", "4", "5", "6", "7",
    "8", "9", "10", "11", "12", "13", "14", "15"
};

/* Set configuration bits for use with ICD2 / PICDEM2 PLUS Demo Board:
 * - set HS oscillator
 * - disable watchdog timer
 * - disable low voltage programming
 * - enable background debugging
 */
#pragma romdata CONFIG
_CONFIG_DECL(_CONFIG1H_DEFAULT & _OSC_HS_1H,
             _CONFIG2L_DEFAULT,
             _CONFIG2H_DEFAULT & _WDT_OFF_2H,
             _CONFIG3H_DEFAULT,
             _CONFIG4L_DEFAULT & _LVP_OFF_4L & _DEBUG_ON_4L,
             _CONFIG5L_DEFAULT,
             _CONFIG5H_DEFAULT,
             _CONFIG6L_DEFAULT,
             _CONFIG6H_DEFAULT,
             _CONFIG7L_DEFAULT,
             _CONFIG7H_DEFAULT);
#pragma romdata

void main (void)
```

# MPLAB® C18 C Compiler Getting Started

---

```
{
    int index;
    int integer;
    char string[3];

    PORTB = 0;
    TRISB = 0;    /* configure all PORTB pins for output */

    for (index = 0; index < STRING_TABLE_SIZE; index++)
    {
        /* copy the string from program memory to data memory for 'atoi' */
        strcpypgm2ram (string, string_table[index]);
        /* get the number's integer representation from its
           string representation */
        integer = atoi (string);

        PORTB = integer;    /* output the value to the LEDs */
        Delay10KTCYx (255); /* pause for a moment (255 * 10,000 cycles) */
    }
}
```

## 3.5 EXAMPLE 3

Beginning with v2.30, MPLAB C18 supports the PIC18 devices' Extended mode, which is targeted toward smaller code size for reentrant code; however, not all PIC18 devices support the Extended mode. See the device data sheet for more details and other implications.

This example is designed for use with the MPLAB IDE v6.xx, the MPLAB ICD 2, the PICDEM 2 Plus demo board and the PIC18F4620 device. This example builds on Example 2 and demonstrates some of the differences between Extended and Non-extended mode.

### 3.5.1 Selecting the Processor

This example will utilize the PIC18F4620 device as the target processor. Steps to select the target processor can be found in Example 1.

### 3.5.2 Utilizing the Extended Mode

MPLAB C18, by default, operates in Non-extended mode and it generates code that will work on a device operating in Non-extended mode. This default behavior can be changed with the command-line option `--extended`. To make this change, choose **Project>Build Options>Project** and click on the **MPLAB C18** tab. Then, click in the *Alternate Settings* checkbox to enable the text area below the checkbox. Type a space followed by `--extended`. Click **OK**.

FIGURE 3-35: PROJECT BUILD OPTIONS

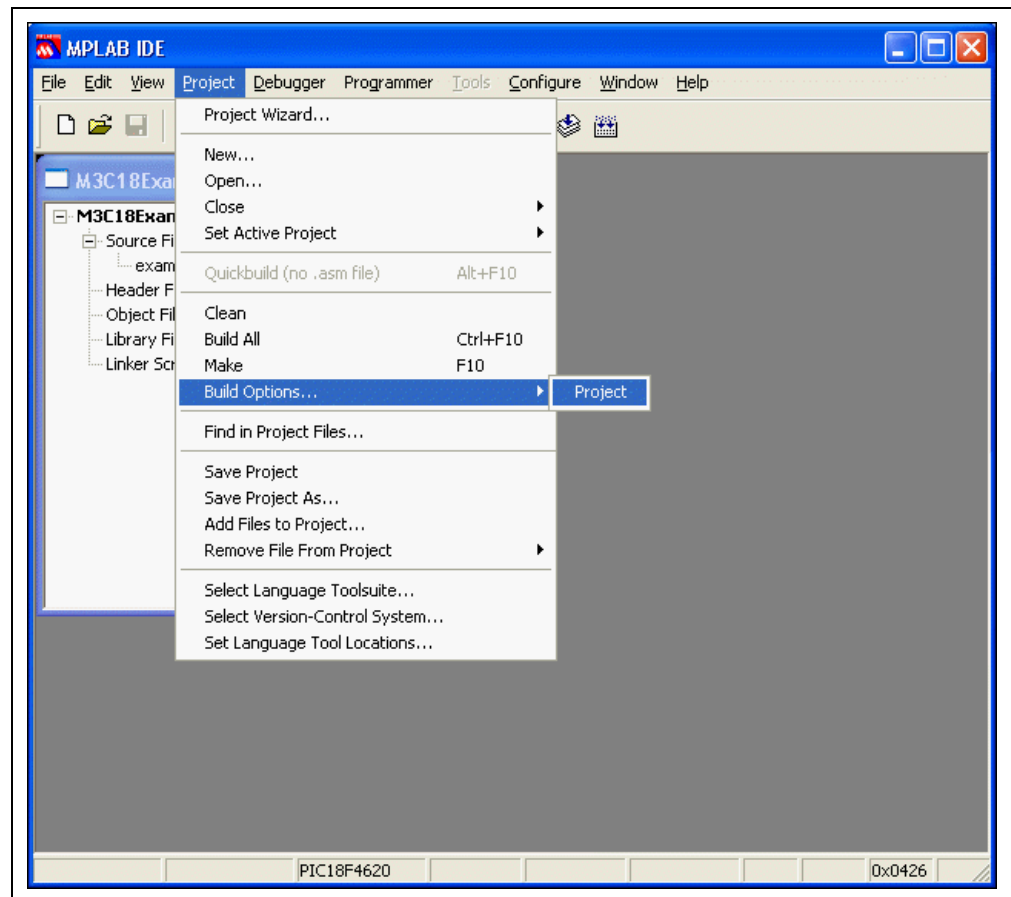
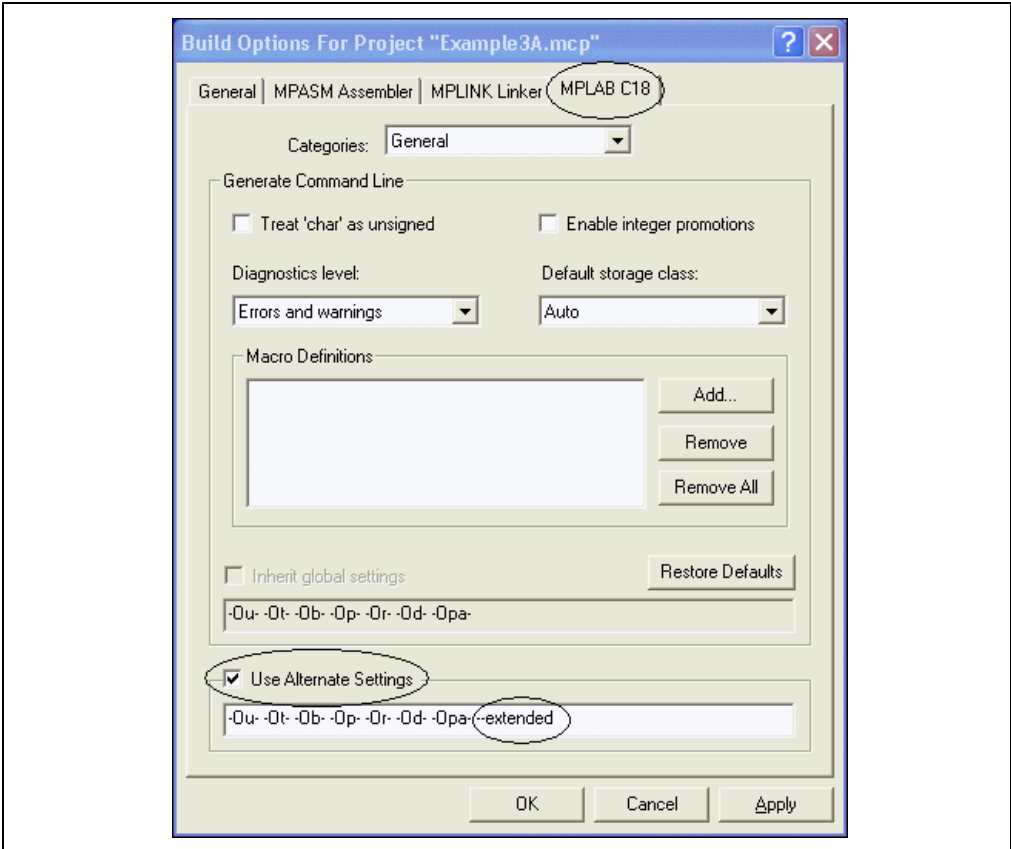


FIGURE 3-36: ALTERNATE SETTINGS FOR EXTENDED MODE



Up to four different types of linker scripts are distributed with the MPLAB C18 C compiler for each processor. These linker scripts are different from those distributed with the MPLAB IDE in that they automatically link in the compiler startup code and libraries, as well as setting aside a stack area. The four linker scripts distributed for the PIC18F4620 processor are:

18f4620.lkr	For use with applications compiled in Non-extended mode.
18f4620i.lkr	For use with applications compiled in Non-extended mode and being debugged with the MPLAB ICD 2. The "i" represents that this linker script allocates memory for resources used by the MPLAB ICD 2.
18f4620_e.lkr	For use with applications compiled in Extended mode.
18f4620i_e.lkr	For use with applications compiled in Extended mode and being debugged with the MPLAB ICD 2. The "i" represents that this linker script allocates memory for resources used by the MPLAB ICD 2.



For this portion of the example, the `18f4620i_e.lkr` must be added to the project files of the MPLAB IDE project. See **Section 3.3 “Example 1”** for information on how to add files to a project.

Next, the `__EXTENDED18__` predefined macro will be utilized to set up the configuration words. The following will replace the CONFIG section of Example 2:

```
/* Configuration bit settings for use with ICD2 / PICDEM2 PLUS Demo Board:
 * - set the HS oscillator (for the crystal)
 * - disable watchdog timer and low-voltage programming (0 for disable)
 * - enable background debugging (0 for enable)
 * - for extended mode compilation, enable extended mode (1 for enable)
 */
#define CONFIG1H_DEFAULT 0x07 /* default value for configuration byte 1H */
#define HS_OSC      0x02      /* setting for HS oscillator */

#define CONFIG2H_DEFAULT 0x1F /* default value for configuration byte 2H */
#define WDT_BIT      0x01      /* enables/disables watchdog timer */

#define CONFIG4L_DEFAULT 0x85 /* default value for configuration byte 4L */
#define DEBUG_BIT 0x80      /* disables/enables background debugging */
#define XINST_BIT 0x40      /* enables/disables extended mode */
#define LVP_BIT 0x04      /* enables/disables low-voltage programming */

#pragma romdata CONFIG1H=0x300001
const rom unsigned char config1H = CONFIG1H_DEFAULT & HS_OSC;
#pragma romdata CONFIG2H=0x300003
const rom unsigned char config2H = CONFIG2H_DEFAULT & ~WDT_BIT;
#pragma romdata CONFIG4L=0x300006
#ifdef __EXTENDED18__
const rom unsigned char config4L =
    (CONFIG4L_DEFAULT | XINST_BIT) & ~LVP_BIT & ~DEBUG_BIT;
#else
const rom unsigned char config4L =
    CONFIG4L_DEFAULT & ~LVP_BIT & ~DEBUG_BIT;
#endif
#pragma romdata
```

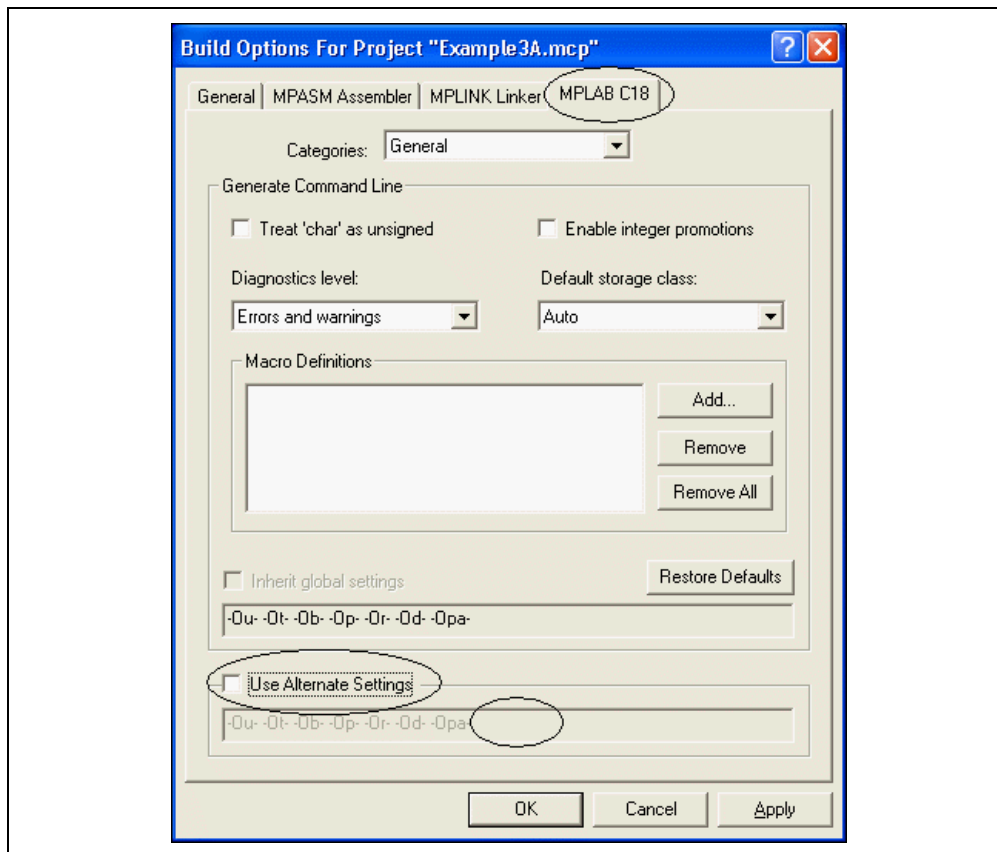
The above code will enable the Extended mode instructions when compiling in the Extended mode, and disable the Extended mode instructions when compiling in Non-extended mode.

## 3.5.3 Utilizing the Non-extended Mode

To change the above example from Extended mode to Non-extended mode, the following steps must occur:

1. Disable Extended mode. Choose *Project>Build Options>Project* and click on the **MPLAB C18** tab. Then, if the *Alternate Settings* checkbox has a check in it, remove the `--extended` from the Alternate Settings text box and clear the checkbox by clicking on it. Click **OK**.

**FIGURE 3-37: ALTERNATE SETTINGS FOR NON-EXTENDED MODE**



2. Remove the Extended mode, MPLAB ICD 2 linker script (18f4620i\_e.lkr) from the MPLAB IDE project.
3. Add the Non-extended mode, MPLAB ICD 2 linker script (18f4620i.lkr) to the MPLAB IDE project.

## 3.6 EXAMPLE 4

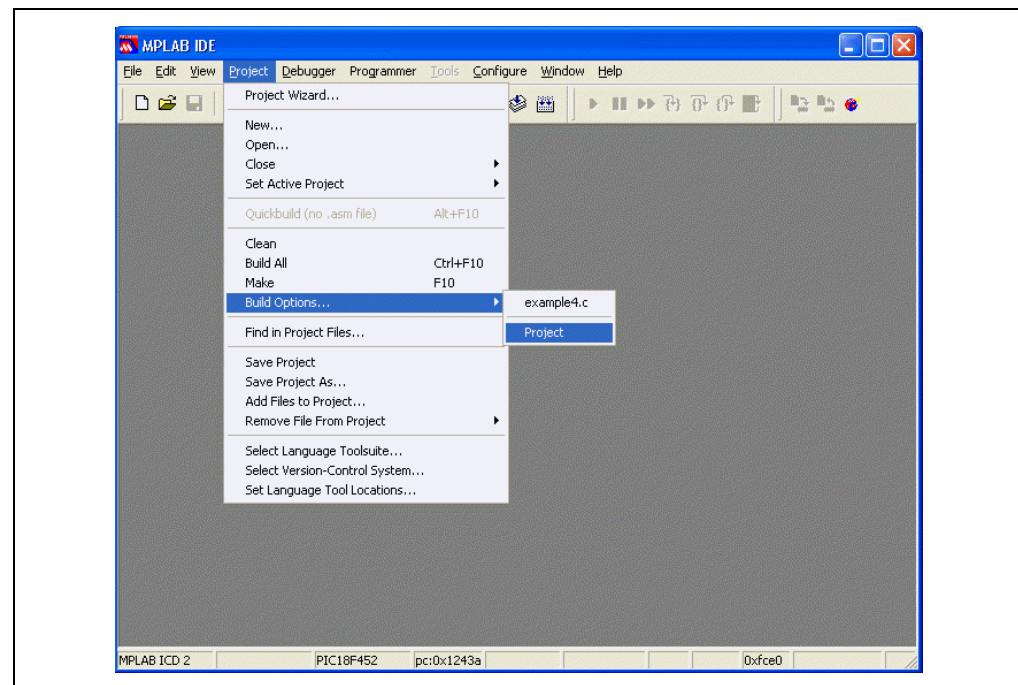
This example is designed for use with the MPLAB IDE v6.xx, the MPLAB SIM simulator and the PIC18F452 device. It demonstrates the allocation of variables in access RAM. For each value from 0 to 99, the program finds the square root of the value with the fractional part truncated. It then squares this root to obtain the greatest perfect square less than or equal to the original value.

- The square root function is implemented as a table in program memory. This has several advantages. If the table were in data memory, it would need to be copied from program memory to data memory at program initialization. Locating the table in program memory also saves data memory space. Finally, the code associated with calculating the square root at runtime may occupy more program memory than a table when the domain of the function is small.
- Data located in access RAM does not require the BSR register to be loaded, thus resulting in fewer instructions. The variables `root` and `square` are located in an access RAM section named `MY_ACS_DATA`. The type qualifier `near` must be used to ensure that MPLAB C18 knows bank selection is not required for these objects.

This example can be built and linked in the MPLAB IDE v6.xx and used with the MPLAB SIM simulator by following the steps in Example 1.

MPLAB C18, by default, assumes all statically allocated data objects, unless explicitly specified with the `near` type qualifier, reside in banked (non-access) RAM. This default behavior can be changed with the command-line option `-Oa+`. To make this change, choose Project>Build Options>Project.

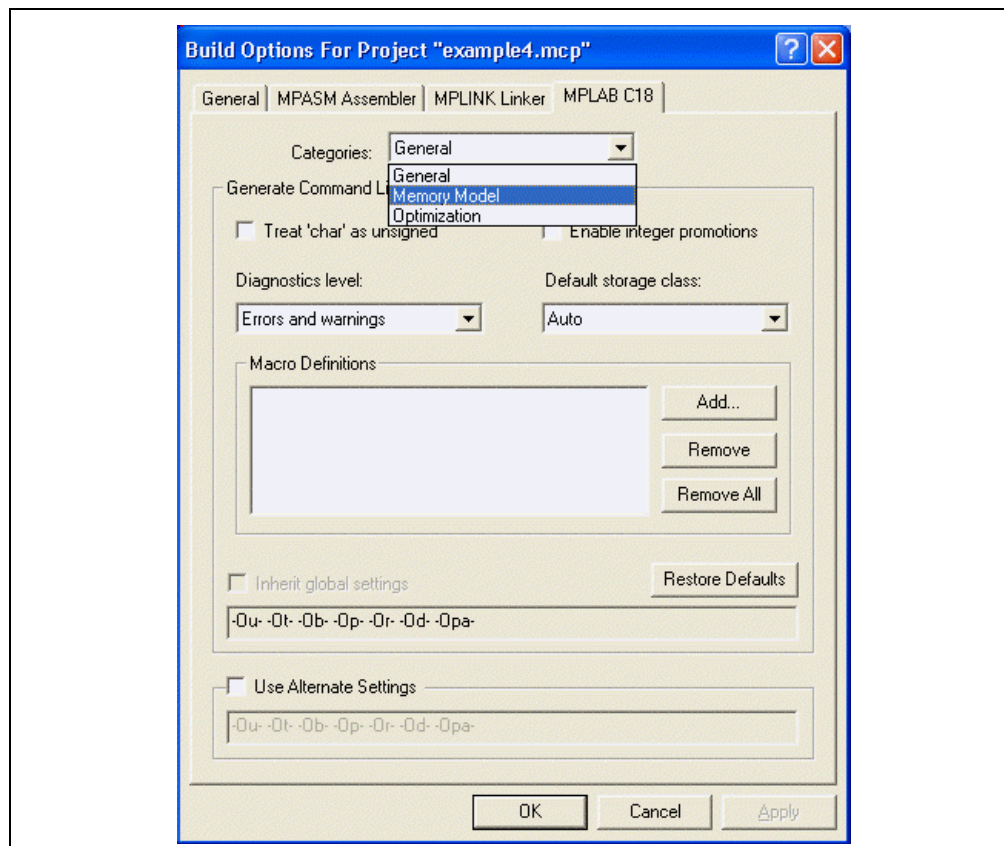
**FIGURE 3-38: PROJECT BUILD OPTIONS**



# MPLAB® C18 C Compiler Getting Started

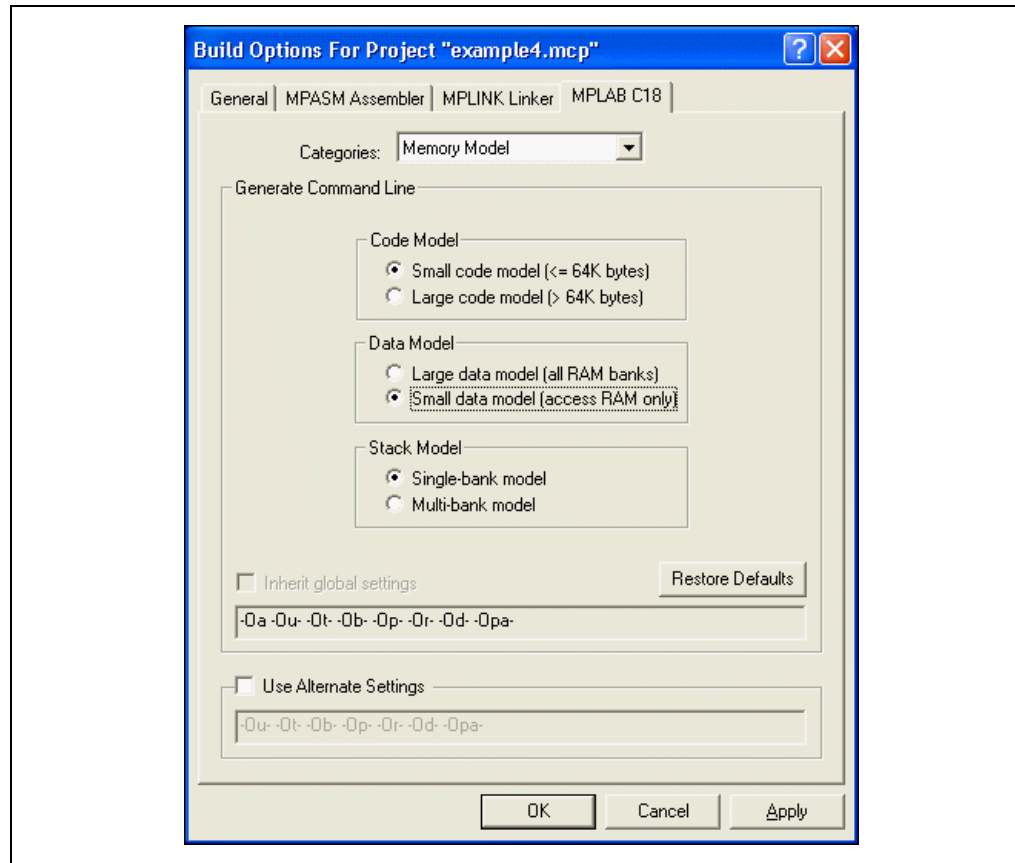
Select the tab labeled “MPLAB C18” and choose “Memory Model” from the drop-down menu.

**FIGURE 3-39: MEMORY MODEL OPTION**



Finally, select the small data model to tell MPLAB C18 that statically allocated data objects without an explicit `near` or `far` qualifier are located in access RAM. See the *MPLAB® C18 C Compiler User's Guide* (DS51288) for details on access and banked (non-access) RAM. Click **OK**.

**FIGURE 3-40: SMALL DATA MODE OPTION**



# MPLAB® C18 C Compiler Getting Started

---

```
#include <p18f452.h> /* for 'PRODL' declaration and 'ACCESS' macro */

/*
 * Locate the read-only table in program memory at address 0x1000.
 * 'romdata' is used for data, and 'code' is used for instructions.
 */
#pragma romdata ROOT_TABLE = 0x1000
#define TABLE_SIZE 100
const rom unsigned char roots[TABLE_SIZE] =
{ 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3,
  3, 3, 3, 3, 3, 3, 4, 4, 4, 4,
  4, 4, 4, 4, 4, 5, 5, 5, 5, 5,
  5, 5, 5, 5, 5, 5, 6, 6, 6, 6,
  6, 6, 6, 6, 6, 6, 6, 6, 6, 7,
  7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
  7, 7, 7, 7, 8, 8, 8, 8, 8, 8,
  8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
  8, 9, 9, 9, 9, 9, 9, 9, 9, 9,
  9, 9, 9, 9, 9, 9, 9, 9, 9, 9 };

/*
 * Data in access ram does not require banking.
 * When compiled with '-Oa', these pragmas and the 'near' qualifier
 * may be removed.
 */
#pragma udata access MY_ACS_DATA
near unsigned char root, square;
#pragma udata /* continue allocating static data in non-access ram */

/*
 * Returns the truncated root of the value.
 */
unsigned char get_root (int val)
{
    return roots[val];
}

void main (void)
{
    static int val;

    for (val = 0; val < TABLE_SIZE; val++)
    {
        /* 'square' holds the greatest perfect square less than or
         equal to 'val' */
        square = get_root (val);
    }
}
```

**Note:** When compiling this example with static data in access RAM by default, the `udata` pragmas surrounding the declarations of `root` and `square` may be removed, as well as the `near` type qualifier.

## 3.7 EXAMPLE 5

This example is designed for use with the MPLAB ICD 2, the PICDEM 2 Plus demo board, the MPLAB IDE v6.xx and the PIC18F452 device. It demonstrates the use of interrupt service routines with MPLAB C18. It also provides an example of the use of the MPLAB C18 peripheral libraries. For this program, the J6 jumper on the demo board must be removed in order to disconnect the PORTB pins from the LEDs.

- This program generates the Piezo buzzer of the PICDEM 2 Plus demo board. The user may disable the buzzer by pressing the S3 button. The buzzer may be reactivated by pressing the button again.
- The S3 button is connected to the INT0 pin, which is associated with the INT0 external interrupt. This interrupt is a high priority interrupt, and so will always trigger a branch to program memory address 0x8. Located at this address is the interrupt service routine (`high_ISR`), which branches to the procedure that turns the buzzer either off or on.
- The configuration bits need to be set appropriately; this is done by invoking the macro `_CONFIG_DECL` with settings for each configuration byte. This includes specifying the oscillator used on the PICDEM 2 Plus demo board; in the example, the crystal (`_OSC_HS_1H`) is used. Additionally, MPLAB ICD 2 requires that the Watchdog Timer and low-voltage programming be disabled (`_WDT_OFF_2H` and `_LVP_OFF_4L`, respectively). Finally, since this example exclusively uses the MPLAB ICD 2, background debugging should always be enabled (`_DEBUG_ON_4L`). The macros to set the configuration bits are defined in the processor-specific header file; see the *MPLAB® C18 C Compiler User's Guide* (DS51288) for details.
- `toggle_buzzer` is declared as a high priority interrupt routine. This means the `WREG`, `BSR` and `STATUS` registers will be saved and restored via their shadow registers without explicit instructions upon interrupt routine entry and exit. Additionally, upon return, the `GIEH` bit in the `INTCON` register will be set, which was cleared when the interrupt was triggered. Refer to the *PIC18FXX2 Data Sheet* (DS39564) for details on interrupt logic.

This example can be built and linked in the MPLAB IDE v6.xx and used with the MPLAB ICD 2 by following the steps in Example 1.

```
#include <p18f452.h> /* for the special function register declarations */
#include <portb.h> /* for the RB0/INT0 interrupt */

/* Set configuration bits for use with ICD2 / PICDEM2 PLUS Demo Board:
 * - set HS oscillator
 * - disable watchdog timer
 * - disable low voltage programming
 * - enable background debugging
 */
#pragma romdata CONFIG
_CONFIG_DECL(_CONFIG1H_DEFAULT & _OSC_HS_1H,
            _CONFIG2L_DEFAULT,
            _CONFIG2H_DEFAULT & _WDT_OFF_2H,
            _CONFIG3H_DEFAULT,
            _CONFIG4L_DEFAULT & _LVP_OFF_4L & _DEBUG_ON_4L,
            _CONFIG5L_DEFAULT,
            _CONFIG5H_DEFAULT,
            _CONFIG6L_DEFAULT,
            _CONFIG6H_DEFAULT,
            _CONFIG7L_DEFAULT,
            _CONFIG7H_DEFAULT);
#pragma romdata
```

# MPLAB® C18 C Compiler Getting Started

---

```
/*
 * For high interrupts, control is transferred to address 0x8.
 */
void toggle_buzzer (void); /* prototype needed for 'goto' below */
#pragma code HIGH_INTERRUPT_VECTOR = 0x8
void high_ISR (void)
{
    _asm
        goto toggle_buzzer
    _endasm
}
#pragma code /* allow the linker to locate the remaining code */

/*
 * If the buzzer is on, turn it off.  If it is off, turn it on.
 */
#pragma interrupt toggle_buzzer
void toggle_buzzer (void)
{
    CCP1CON = ~CCP1CON & 0x0F; /* turn the buzzer off or on */
    INTCONbits.INT0IF = 0; /* clear flag to avoid another interrupt */
}

void EnableHighInterrupts (void)
{
    RCONbits.IPEN = 1; /* enable interrupt priority levels */
    INTCONbits.GIEH = 1; /* enable all high priority interrupts */
}

void InitializeBuzzer (void)
{
    T2CON = 0x05; /* postscale 1:1, Timer2 ON, prescaler 4 */
    TRISCBits.TRISC2 = 0; /* configure the CCP1 module for the buzzer */
    PR2 = 0x80; /* initialize the PWM period */
    CCP1L = 0x80; /* initialize the PWM duty cycle */
}

void SoundBuzzer (void)
{
    CCP1CON = 0x0F; /* turn the buzzer on */
    while (1); /* wait for the S3 button to be pressed */
}

void main (void)
{
    EnableHighInterrupts ( );
    InitializeBuzzer ( );

    OpenRB0INT (PORTB_CHANGE_INT_ON & /* enable the RB0/INT0 interrupt */
               PORTB_PULLUPS_ON & /* configure the RB0 pin for input */
               FALLING_EDGE_INT); /* trigger interrupt upon S3 button
                                   depression */

    SoundBuzzer ( );
}
```



---

## Glossary

---

### A

**Absolute Section**

A section with a fixed address that cannot be changed by the linker.

**Access Memory**

Special general purpose registers on the PIC18 PICmicro microcontrollers that allow access regardless of the setting of the Bank Select Register (BSR).

**Address**

The code that identifies where a piece of information is stored in memory.

**Anonymous Structure**

An unnamed object.

**ANSI**

American National Standards Institute

**Assembler**

A language tool that translates assembly source code into machine code.

**Assembly**

A symbolic language that describes the binary machine code in a readable form.

**Assigned Section**

A section that has been assigned to a target memory block in the linker command file.

**Asynchronously**

Multiple events that do not occur at the same time. This is generally used to refer to interrupts that may occur at any time during processor execution.

### B

**Binary**

The base two numbering system that uses the digits 0-1. The right-most digit counts ones, the next counts multiples of 2, then  $2^2 = 4$ , etc.

### C

**Central Processing Unit**

The part of a device that is responsible for fetching the correct instruction for execution, decoding that instruction and then executing that instruction. When necessary, it works in conjunction with the Arithmetic Logic Unit (ALU) to complete the execution of the instruction. It controls the program memory address bus, the data memory address bus and accesses to the stack.

**Compiler**

A program that translates a source file written in a high-level language into machine code.

## **Conditional Compilation**

The act of compiling a program fragment only if a certain constant expression, specified by a preprocessor directive, is true.

## **CPU**

Central Processing Unit

## **E**

### **endianness**

The ordering of bytes in a multi-byte object.

### **Error File**

A file containing the diagnostics generated by the MPLAB C18.

### **Extended Mode**

In Extended mode, the compiler will utilize the extended instructions (i.e., ADDFSR, ADDULNK, CALLW, MOVSF, MOVSS, PUSH, SUBFSR and SUBULNK) and the indexed with literal offset addressing.

## **F**

### **Fatal Error**

An error that will halt compilation immediately. No further messages will be produced.

### **Frame Pointer**

A pointer that references the location on the stack that separates the stack-based arguments from the stack-based local variables.

### **Free-standing**

An implementation that accepts any strictly conforming program that does not use complex types and in which the use of the features specified in the library clause (ANSI '89 standard clause 7) are confined to the contents of the standard headers `<float.h>`, `<iso646.h>`, `<limits.h>`, `<stdarg.h>`, `<stdbool.h>`, `<stddef.h>` and `<stdint.h>`.

## **H**

### **Hexadecimal**

The base 16 numbering system that uses the digits 0-9 plus the letters A-F (or a-f). The digits A-F represent decimal values of 10 to 15. The right-most digit counts ones, the next counts multiples of 16, then  $16^2 = 256$ , etc.

### **High-level Language**

A language for writing programs that is further removed from the processor than assembly.

## **I**

### **ICD**

In-Circuit Debugger

### **ICE**

In-Circuit Emulator

### **IDE**

Integrated Development Environment

## **IEEE**

Institute of Electrical and Electronics Engineers

## **Interrupt**

A signal to the CPU that suspends the execution of a running application and transfers control to an ISR so that the event may be processed. Upon completion of the ISR, normal execution of the application resumes.

## **Interrupt Service Routine**

A function that handles an interrupt.

## **ISO**

International Organization for Standardization

## **ISR**

Interrupt Service Routine

## **L**

### **Latency**

The time between when an event occurs and the response to it.

### **Librarian**

A program that creates and manipulates libraries.

### **Library**

A collection of relocatable object modules.

### **Linker**

A program that combines object files and libraries to create executable code.

### **Little Endian**

Within a given object, the least significant byte is stored at lower addresses.

## **M**

### **Memory Model**

A description that specifies the size of pointers that point to program memory.

### **Microcontroller**

A highly integrated chip that contains a CPU, RAM, some form of ROM, I/O ports and timers.

### **MPASM Assembler**

Microchip Technology's relocatable macro assembler for PICmicro microcontroller families.

### **MPLIB Object Librarian**

Microchip Technology's librarian for PICmicro microcontroller families.

### **MPLINK Object Linker**

Microchip Technology's linker for PICmicro microcontroller families.

## **N**

### **Non-extended Mode**

In Non-extended mode, the compiler will not utilize the extended instructions nor the indexed with literal offset addressing.

## O

### Object File

A file containing object code. It may be immediately executable or it may require linking with other object code files (e.g., libraries), to produce a complete executable program.

### Object Code

The machine code generated by an assembler or compiler.

### Octal

The base 8 number system that only uses the digits 0-7. The right-most digit counts ones, the next digit counts multiples of 8, then  $8^2 = 64$ , etc.

## P

### Pragma

A directive that has meaning to a specific compiler.

## R

### RAM

Random Access Memory

### Random Access Memory

A memory device in which information can be accessed in any order.

### Read Only Memory

Memory hardware that allows fast access to permanently stored data but prevents addition to or modification of the data.

### ROM

Read Only Memory

### Recursive

Self-referential (e.g., a function that calls itself).

### Reentrant

A function that may have multiple, simultaneously active instances. This may happen due to either direct or indirect recursion or through execution during interrupt processing.

### Relocatable

An object whose address has not been assigned to a fixed memory location.

### Run-time Model

Set of assumptions under which the compiler operates.

## S

### Section

A portion of an application located at a specific address of memory.

### Section Attribute

A characteristic ascribed to a section (e.g., an `access` section).

### Special Function Register

Registers that control I/O processor functions, I/O status, timers or other modes or peripherals.

**Storage Class**

Determines the lifetime of the memory associated with the identified object.

**Storage Qualifier**

Indicates special properties of the objects being declared (e.g., `const`).

**V****Vector**

The memory locations that an application will jump to when either a reset or interrupt occurs.

# MPLAB® C18 C Compiler Getting Started

---

NOTES:

## Index

### Symbols

\_\_EXTENDED18\_\_ ..... 45

### A

Access RAM ..... 1, 19, 47  
Add Files to Project ..... 27

### B

Breakpoint ..... 30, 32, 38  
Build Options ..... 26  
Build Project ..... 29

### C

COD ..... 9  
Customer Notification Service ..... 5  
Customer Support ..... 6

### D

Directory ..... 12  
Directory Contents ..... 8  
Documentation Conventions ..... 2

### E

Electronic Documentation ..... 14  
Examples ..... 8, 14, 20, 39, 43, 47, 51  
Extended Mode ..... 1, 9, 19, 43, 45

### H

HEX ..... 9

### I

Installing MPLAB C18 ..... 11  
Interrupt Service Routine ..... 1, 19, 51  
Interrupt Service Routine ..... 55

### L

Language Tools ..... 9, 22  
Libraries ..... 3, 8, 14  
Linker Scripts ..... 8, 14, 28, 35  
Listing Files ..... 34  
Little Endian ..... 55

### M

Map Files ..... 34  
MCC\_INCLUDE ..... 16  
Memory Model ..... 48  
Microchip Web Site ..... 4  
MPLAB ICD 2 ..... 35  
MPLAB SIM Simulator ..... 30

### N

New Project ..... 21  
Non-extended Mode ..... 1, 9, 19, 44, 46

### P

Paths ..... 25  
Project Paths ..... 25  
Project Settings ..... 22  
Project Tree ..... 21

### R

README File ..... 3  
Readme File ..... 12  
References ..... 3  
Run-time Model ..... 19

### S

Select Device ..... 21  
Select Language Toolsuite ..... 22  
System Requirements ..... 7

### T

Troubleshooting ..... 4

### U

Uninstalling MPLAB C18 ..... 17

### W

Watch Window ..... 31–33



## 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: 480-792-7627  
Web Address: <http://www.microchip.com>

#### Atlanta

3780 Mansell Road, Suite 130  
Alpharetta, GA 30022  
Tel: 770-640-0034  
Fax: 770-640-0307

#### Boston

2 Lan Drive, Suite 120  
Westford, MA 01886  
Tel: 978-692-3848  
Fax: 978-692-3821

#### Chicago

333 Pierce Road, Suite 180  
Itasca, IL 60143  
Tel: 630-285-0071  
Fax: 630-285-0075

#### Dallas

4570 Westgrove Drive, Suite 160  
Addison, TX 75001  
Tel: 972-818-7423  
Fax: 972-818-2924

#### Detroit

Tri-Atria Office Building  
32255 Northwestern Highway, Suite 190  
Farmington Hills, MI 48334  
Tel: 248-538-2250  
Fax: 248-538-2260

#### Kokomo

2767 S. Albright Road  
Kokomo, IN 46902  
Tel: 765-864-8360  
Fax: 765-864-8387

#### Los Angeles

18201 Von Karman, Suite 1090  
Irvine, CA 92612  
Tel: 949-263-1888  
Fax: 949-263-1338

#### San Jose

1300 Terra Bella Avenue  
Mountain View, CA 94043  
Tel: 650-215-1444  
Fax: 650-961-0286

#### Toronto

6285 Northam Drive, Suite 108  
Mississauga, Ontario L4V 1X5, Canada  
Tel: 905-673-0699  
Fax: 905-673-6509

### ASIA/PACIFIC

#### Australia

Suite 22, 41 Rawson Street  
Epping 2121, NSW  
Australia  
Tel: 61-2-9868-6733  
Fax: 61-2-9868-6755

#### China - Beijing

Unit 706B  
Wan Tai Bei Hai Bldg.  
No. 6 Chaoyangmen Bei Str.  
Beijing, 100027, China  
Tel: 86-10-85282100  
Fax: 86-10-85282104

#### China - Chengdu

Rm. 2401-2402, 24th Floor,  
Ming Xing Financial Tower  
No. 88 TIDU Street  
Chengdu 610016, China  
Tel: 86-28-86766200  
Fax: 86-28-86766599

#### China - Fuzhou

Unit 28F, World Trade Plaza  
No. 71 Wusi Road  
Fuzhou 350001, China  
Tel: 86-591-7503506  
Fax: 86-591-7503521

#### China - Hong Kong SAR

Unit 901-6, Tower 2, Metroplaza  
223 Hing Fong Road  
Kwai Fong, N.T., Hong Kong  
Tel: 852-2401-1200  
Fax: 852-2401-3431

#### China - Shanghai

Room 701, Bldg. B  
Far East International Plaza  
No. 317 Xian Xia Road  
Shanghai, 200051  
Tel: 86-21-6275-5700  
Fax: 86-21-6275-5060

#### China - Shenzhen

Rm. 1812, 18/F, Building A, United Plaza  
No. 5022 Binhe Road, Futian District  
Shenzhen 518033, China  
Tel: 86-755-82901380  
Fax: 86-755-8295-1393

#### China - Shunde

Room 401, Hongjian Building, No. 2  
Fengxiangnan Road, Ronggui Town, Shunde  
District, Foshan City, Guangdong 528303, China  
Tel: 86-757-28395507 Fax: 86-757-28395571

#### China - Qingdao

Rm. B505A, Fullhope Plaza,  
No. 12 Hong Kong Central Rd.  
Qingdao 266071, China  
Tel: 86-532-5027355 Fax: 86-532-5027205

#### India

Divyasree Chambers  
1 Floor, Wing A (A3/A4)  
No. 11, O'Shaughnessy Road  
Bangalore, 560 025, India  
Tel: 91-80-22290061 Fax: 91-80-22290062

#### Japan

Benex S-1 6F  
3-18-20, Shinyokohama  
Kohoku-Ku, Yokohama-shi  
Kanagawa, 222-0033, Japan  
Tel: 81-45-471-6166 Fax: 81-45-471-6122

#### Korea

168-1, Youngbo Bldg. 3 Floor  
Samsung-Dong, Kangnam-Ku  
Seoul, Korea 135-882  
Tel: 82-2-554-7200 Fax: 82-2-558-5932 or  
82-2-558-5934

#### Singapore

200 Middle Road  
#07-02 Prime Centre  
Singapore, 188980  
Tel: 65-6334-8870 Fax: 65-6334-8850

#### Taiwan

Kaohsiung Branch  
30F - 1 No. 8  
Min Chuan 2nd Road  
Kaohsiung 806, Taiwan  
Tel: 886-7-536-4818  
Fax: 886-7-536-4803

#### Taiwan

Taiwan Branch  
11F-3, No. 207  
Tung Hua North Road  
Taipei, 105, Taiwan  
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

### EUROPE

#### Austria

Durisolstrasse 2  
A-4600 Wels  
Austria  
Tel: 43-7242-2244-399  
Fax: 43-7242-2244-393

#### Denmark

Regus Business Centre  
Lautrup høj 1-3  
Ballerup DK-2750 Denmark  
Tel: 45-4420-9895 Fax: 45-4420-9910

#### France

Parc d'Activite du Moulin de Massy  
43 Rue du Saule Trapu  
Batiment A - 1er Etage  
91300 Massy, France  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

#### Germany

Steinheilstrasse 10  
D-85737 Ismaning, Germany  
Tel: 49-89-627-144-0  
Fax: 49-89-627-144-44

#### Italy

Via Quasimodo, 12  
20025 Legnano (MI)  
Milan, Italy  
Tel: 39-0331-742611  
Fax: 39-0331-466781

#### Netherlands

P. A. De Biesbosch 14  
NL-5152 SC Drunen, Netherlands  
Tel: 31-416-690399  
Fax: 31-416-690340

#### United Kingdom

505 Eskdale Road  
Winnersh Triangle  
Wokingham  
Berkshire, England RG41 5TU  
Tel: 44-118-921-5869  
Fax: 44-118-921-5820

02/17/04