

**MPLAB[®] C18
C COMPILER
GETTING STARTED**

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products 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 PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be 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 product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

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contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates.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, KEELOQ, MPLAB, PIC, PICmicro, PICSTART and PRO MATE are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


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

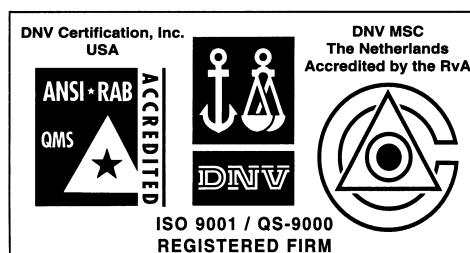
dsPIC, dsPICDEM.net, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, PICC, PICDEM, PICDEM.net, rPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Serialized Quick Turn Programming (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.

© 2003, Microchip Technology Incorporated. Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.



Table of Contents

Preface	1
Chapter 1. Overview	
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
Chapter 2. Installation	
2.1 Introduction	11
2.2 Highlights	11
2.3 Installing MPLAB C18	11
2.4 Uninstalling MPLAB C18	17
Chapter 3. Examples of Use	
3.1 Introduction	19
3.2 Highlights	19
3.3 Example 1	20
3.4 Example 2	37
3.5 Example 3	40
3.6 Example 4	43
Glossary	45
Index	51
Worldwide Sales and Service	52

MPLAB® C18 C Compiler Getting Started

NOTES:



MPLAB® C18 C COMPILER GETTING STARTED

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
- Warranty Registration
- Recommended Reading
- Troubleshooting
- Microchip On-Line Support
- Customer Change 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 Help 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. All examples use MPLAB IDE v6.xx with PIC18F452 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 the allocation of variables in access RAM.
 - **Example 4** 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 manual uses the following documentation conventions:

Description	Represents	Examples
Code (Courier font):		
Plain characters	Sample code Filenames and paths	#define START c:\autoexec.bat
Square brackets: []	Optional arguments	MPASMWIN [main.asm]
Curly brackets and pipe character: { }	Choice of mutually exclusive arguments An OR selection	errorlevel {0 1}
Ellipses...	Used to imply (but not show) additional text	list [<i>list_option...</i> , <i>list_option</i>]
0xnnnn	A hexadecimal number where <i>n</i> is a hexadecimal digit	0xFFFF, 0x007A
Italic characters	A variable argument	char isascii (char <i>ch</i>);
Interface (Arial font):		
Underlined, italic text with “arrow”	A menu selection from the menu bar	<u>File > Save</u>
Bold characters	A window or dialog button to click	OK, Cancel
Characters in angle brackets: < >	A key on the keyboard	<Tab>, <Ctrl-C>
Documents (Arial font):		
Italic characters	Referenced books	<i>MPLAB IDE 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.

WARRANTY REGISTRATION

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

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 MPLAB IDE 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 C18 C Compiler Libraries (DS51297)

Reference guide for MPLAB C18 libraries and precompiled object files. Lists all library functions with a detailed description of their use.

MPLAB IDE User's Guide (DS51025)

Comprehensive guide that describes installation and features of Microchip's MPLAB Integrated Development Environment (IDE), as well as the editor and simulator functions in the MPLAB IDE environment.

MPASM™ User's Guide with MPLINK™ and MPLIB™ (DS33014)

This user's guide describes how to use the Microchip PICmicro MCU MPASM assembler, the MPLINK object linker and the MPLIB object librarian.

PIC18 Device Data Sheets

These data sheets describe the operation and electrical specifications of PIC18 devices and may be found on the Technical CD-ROM or the Microchip web site.

Technical Library CD-ROM (DS00161)

This CD-ROM contains comprehensive application notes, data sheets and technical briefs for all Microchip products. To obtain this CD-ROM, contact the nearest Microchip Sales and Service location (see back page).

Microchip Web Site

The Microchip web site (www.microchip.com) contains a wealth of documentation. Individual data sheets, application notes, tutorials and user's guides are all available for easy download. All documentation is in Adobe® Acrobat® (pdf) format.

Microsoft® Windows® Manuals

This manual assumes that users are familiar with the Microsoft Windows operating system. Many excellent references exist for this software program, and should be consulted for general operation of Windows.

TROUBLESHOOTING

If problems are encountered with any of the procedures or tutorial steps in this document, please see the README files or other recommended reading documents for additional information. If these are not helpful, check out the Technical Support section of our web site or contact customer support.

MPLAB® C18 C Compiler Getting Started

MICROCHIP ON-LINE SUPPORT

Microchip provides on-line support on the Microchip web site at:

www.microchip.com

A file transfer site is also available by using an FTP service connecting to:

<ftp://ftp.microchip.com>

The web site and file transfer site provide 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 Microchip specific business information is also available, including listings of Microchip sales offices and distributors. Other information available on the web site includes:

- Latest Microchip press releases
- Technical support section with FAQs
- Design tips
- Device errata
- Job postings
- Microchip consultant program member listing
- Links to other useful web sites related to Microchip products
- Conferences for products, development systems, technical information and more
- Listing of seminars and events

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip started the customer notification service to help customers stay current on Microchip products with the least amount of effort. Subscribers will receive email notification when we change, update, revise or have errata related to their specified product family or development tool of interest.

Go to the Microchip web site (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 C17, MPLAB C18 and MPLAB C30 C Compilers; MPASM and MPLAB ASM30 assemblers; MPLINK and MPLAB LINK30 linkers; and MPLIB and MPLAB LIB30 librarians.

Emulators - The latest information on Microchip in-circuit emulators. This includes the MPLAB ICE 2000.

In-Circuit Debuggers - The latest information on Microchip in-circuit debuggers. These include the MPLAB ICD and MPLAB ICD 2.

MPLAB - 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 device programmers. These include the PRO MATE II device programmer and PICSTART Plus development programmer.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributors
- Local Sales Office
- Field Application Engineers (FAEs)
- Corporate Applications Engineers (CAEs)
- Systems Information and Upgrade Hot Line

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

Corporate applications engineers (CAEs) may be contacted at (480) 792-7627.

Systems Information and Upgrade Line

The Systems Information and Upgrade Information Line provides system users with a listing of the latest versions of all of Microchip's development systems software products. Plus, this line provides information on how customers can receive the most current upgrade kits. The Information Line Numbers are:

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

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

MPLAB® C18 C Compiler Getting Started

NOTES:



Chapter 1. Overview

1.1 INTRODUCTION

This document is designed to get users started quickly using Microchip's C compiler – MPLAB C18. PICmicro 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>c18.lib</code>), the processor-specific libraries (<code>p18xxxx.lib</code> , where <code>xxxx</code> is the specific device number) and the startup modules (<code>c018.o</code> , <code>c018i.o</code> , <code>c018iz.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>p18xxx.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 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

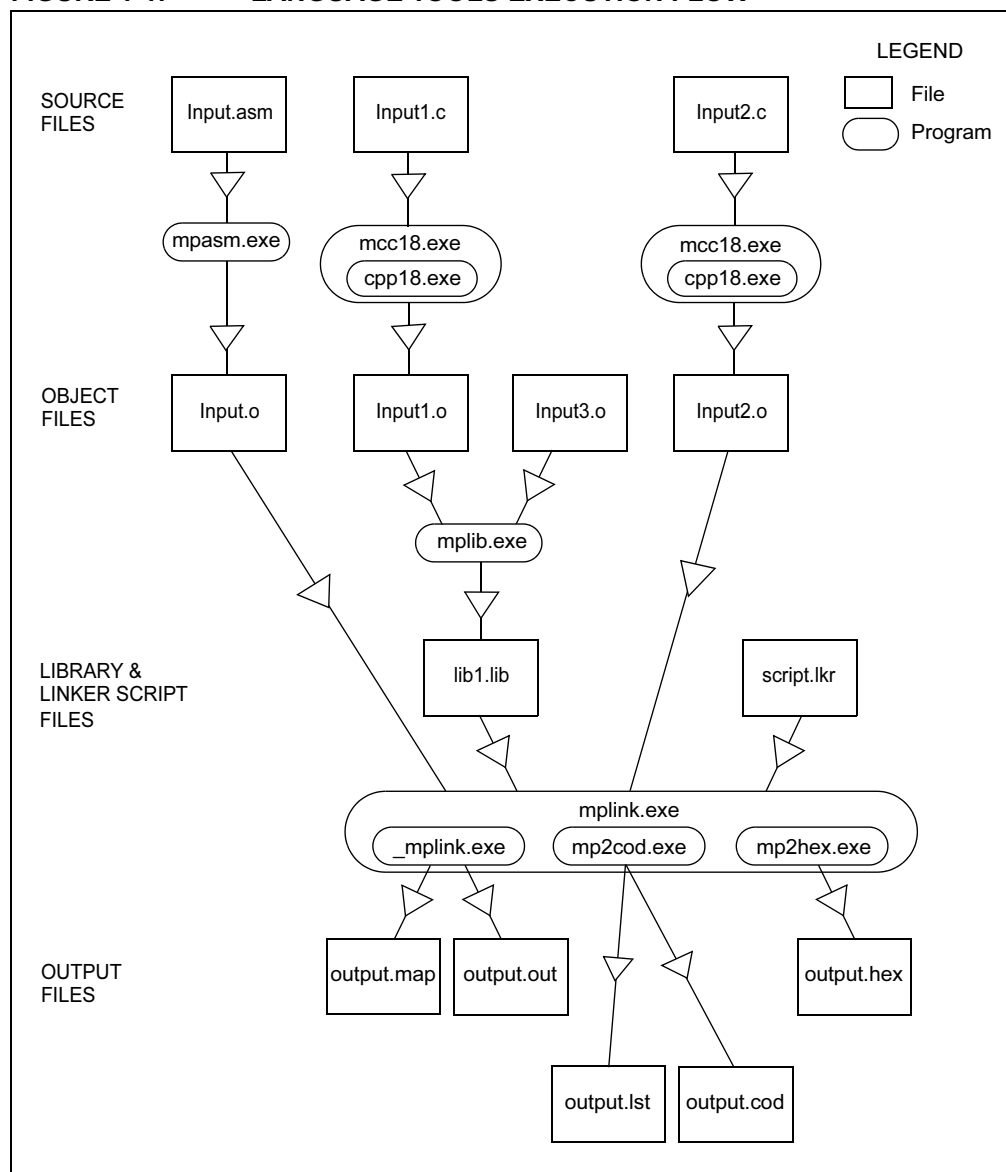
Executable	Description
<code>mcc18.exe</code> <code>c18demo.exe</code>	This is the C compiler. It takes as input a C source file (e.g., <code>file.c</code>), which is passed to <code>cpp18.exe</code> for preprocessing. For demo versions of MPLAB C18, this file is called <code>c18demo.exe</code> . <code>mcc18.exe</code> 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 PROMATE 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.
<code>mpasm.exe</code>	This is the command-line assembler. It takes as input an assembly source file (e.g., <code>file.asm</code>) and outputs either a COFF file (e.g., <code>file.o</code>) or a HEX file and COD file (e.g., <code>file.hex</code> and <code>file.cod</code>). It also creates a listing file (e.g., <code>file.lst</code>) and an error file (e.g., <code>file.err</code>) which contains any errors or warnings emitted during the assembly process. Assembly source files may include assembly header files (e.g., <code>p18f452.inc</code>), 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).

MPLAB® C18 C Compiler Getting Started

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

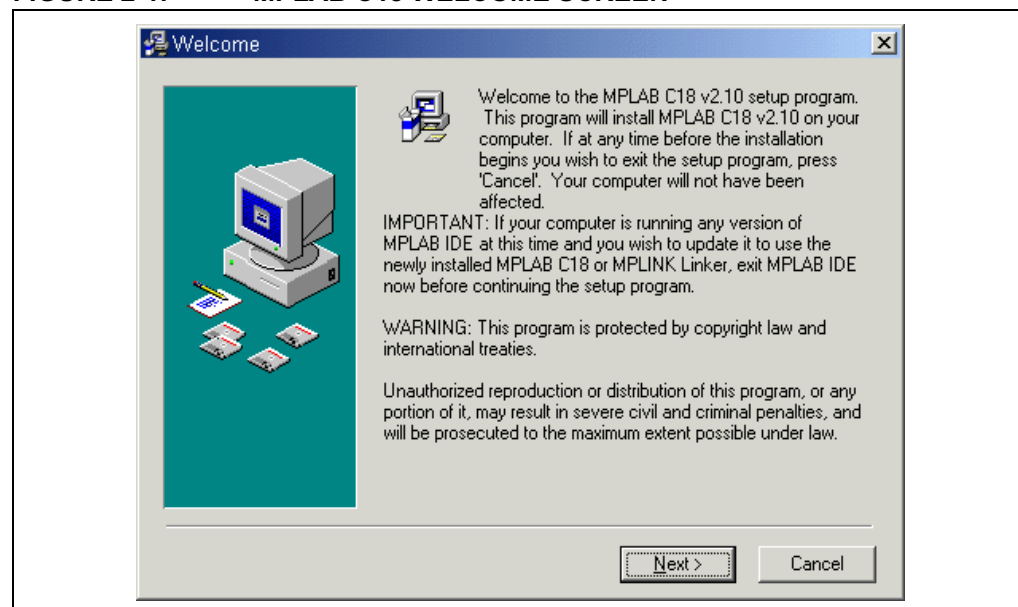
2.3 INSTALLING MPLAB C18

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



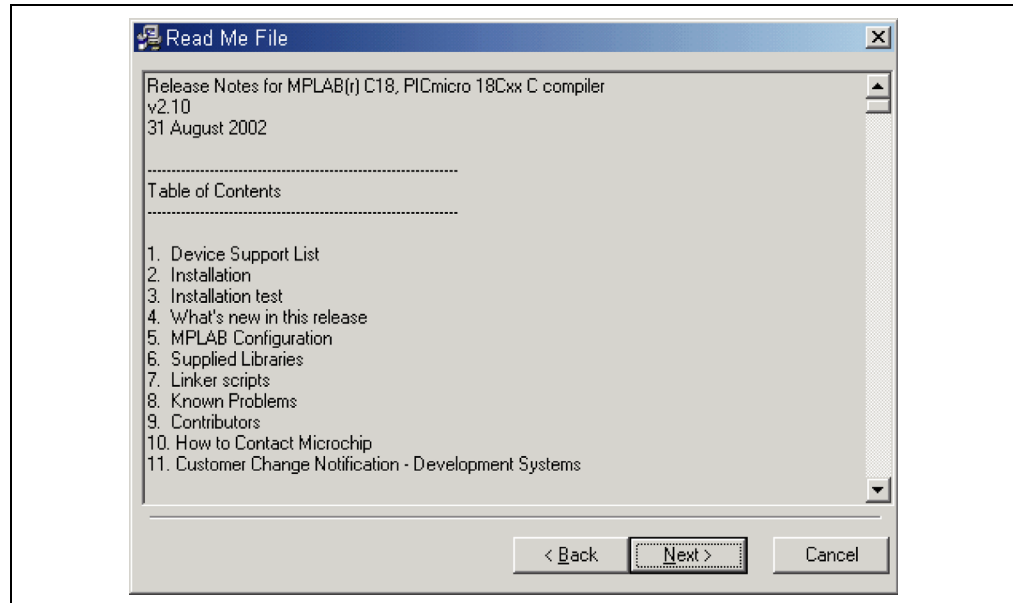
Click **Next** to continue.

MPLAB® C18 C Compiler Getting Started

2.3.2 Read Me 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 READ ME FILE



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

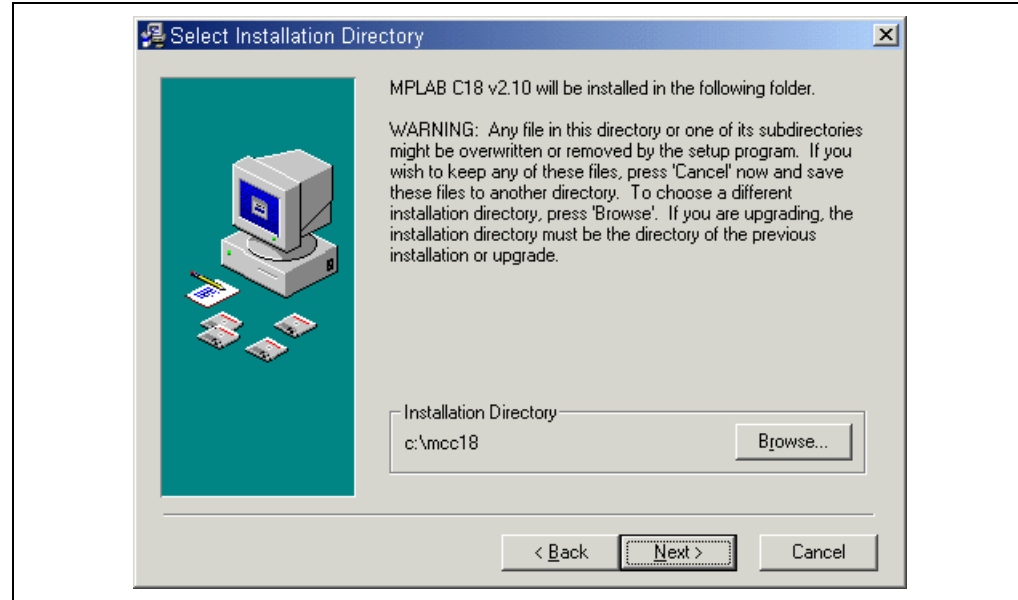
2.3.3 Select Installation Directory

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, 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

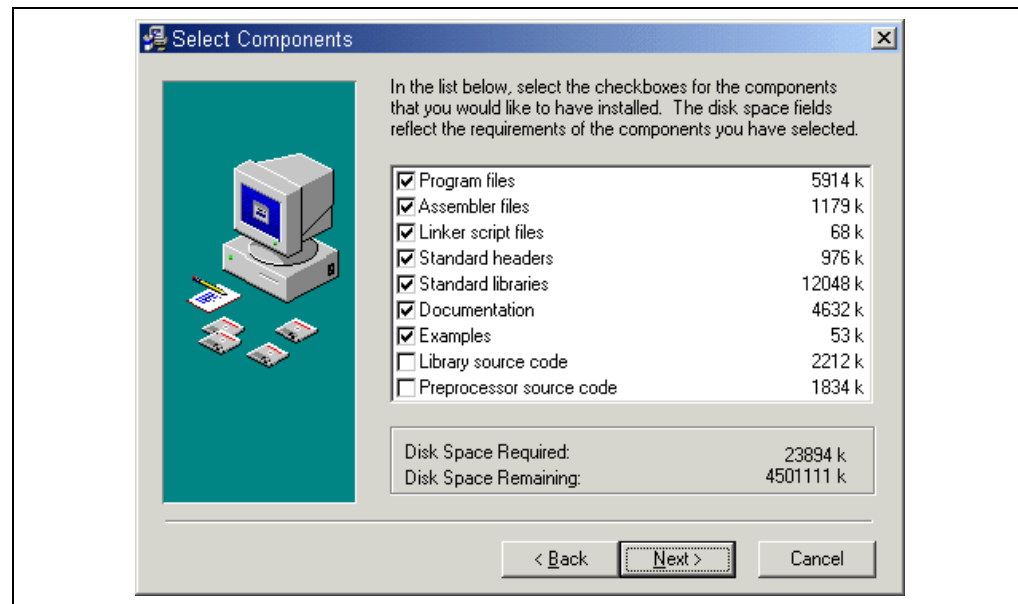


After specifying the directory, click **Next**.

2.3.4 Select Components

Choose the components to be installed by checking the appropriate box.

FIGURE 2-4: 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

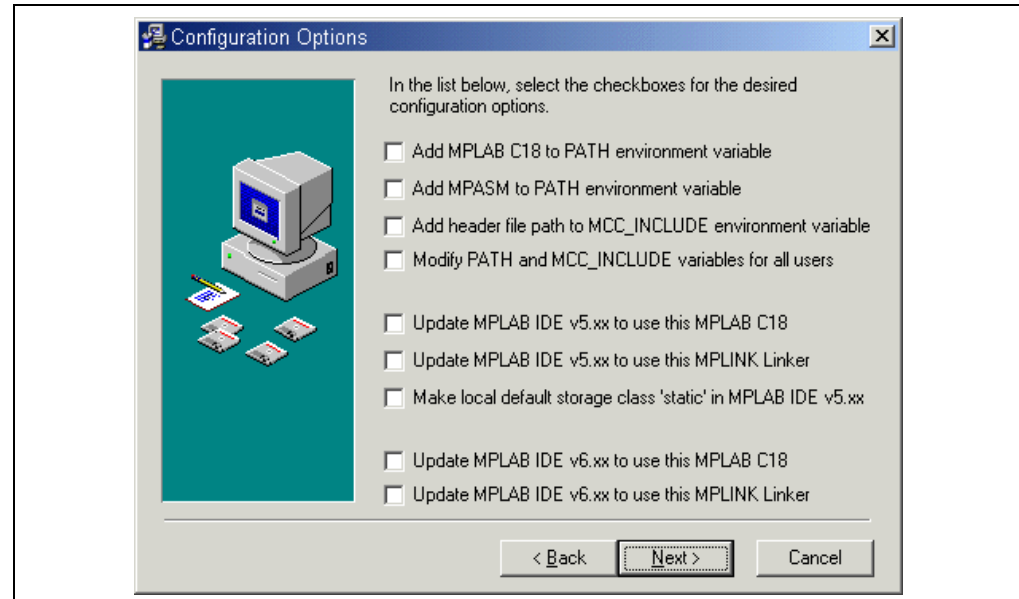
Component	Description
Program files	These are the executables for the compiler and linker. Users should install this component unless they are upgrading and wish to use the executables from the previously installed version.
Assembler files	These include the command-line version of the MPASM assembler (<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.

Select the components to be installed, then click **Next**.

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

MPLAB® C18 C Compiler Getting Started

TABLE 2-2: MPLAB C18 CONFIGURATION OPTIONS

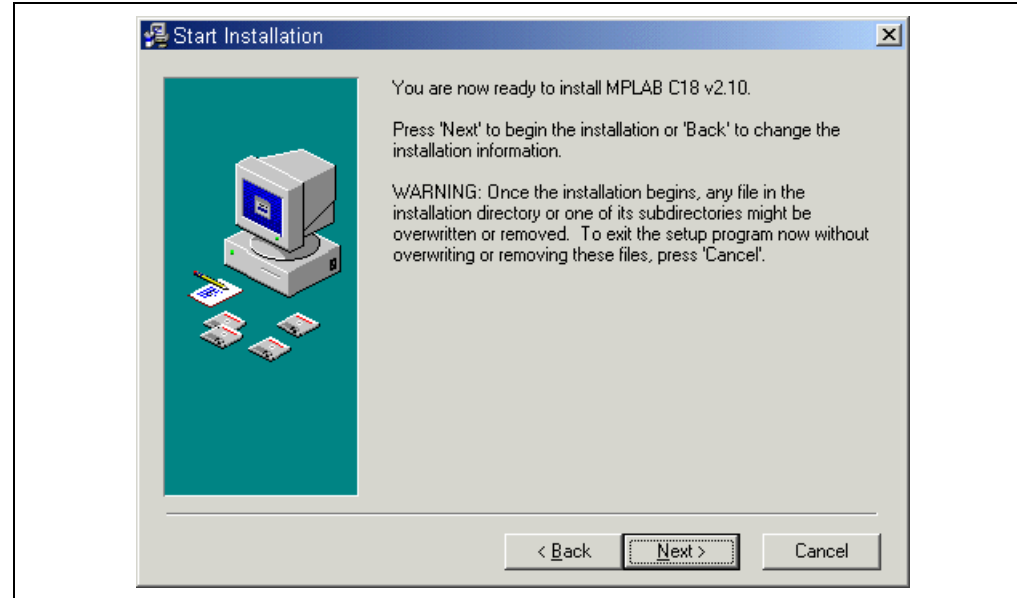
Configuration	Description
Add MPLAB C18 to PATH environment variable	This adds the path of the MPLAB C18 executable (<code>mcc18.exe</code>) and the MPLINK linker executable (<code>mplink.exe</code>) to the front of the <code>PATH</code> environment variable. Doing this allows users to launch the newly 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 <code>PATH</code> 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 <code>MCC_INCLUDE</code> environment variable	This adds the path of the MPLAB C18 header file directory to the front of the <code>MCC_INCLUDE</code> environment variable. If this variable does not exist, it is created. <code>MCC_INCLUDE</code> is a list of semicolon-delimited directories that MPLAB C18 will search for a header file if it cannot find the file in 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 <code>PATH</code> and <code>MCC_INCLUDE</code> 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 v5.xx to use this MPLAB C18	This option appears only if the MPLAB IDE v5.xx is installed on your system. Selecting this option configures the MPLAB IDE v5.xx to use the newly installed MPLAB C18.
Update MPLAB IDE v5.xx to use this MPLINK linker	This option appears only if the MPLAB IDE v5.xx is installed on your system. Selecting this option configures the MPLAB IDE v5.xx to use the newly installed MPLINK linker.
Make local default storage class 'static' in MPLAB IDE v5.xx	This option appears only if the MPLAB IDE v5.xx is installed on your system. When this option is in effect, it is as if the C language storage class specifier <code>static</code> were used with all local and formal parameter variable declarations. This allocates these variables in global memory instead of on the stack. In general, statically allocated variables require fewer instructions to access than stack-allocated variables. However, keep in mind that some functions may behave differently depending on the allocation scheme used.
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.

Select the desired configuration options and click **Next**.

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 the allocation of variables in access RAM.
- **Example 4** 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

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 */

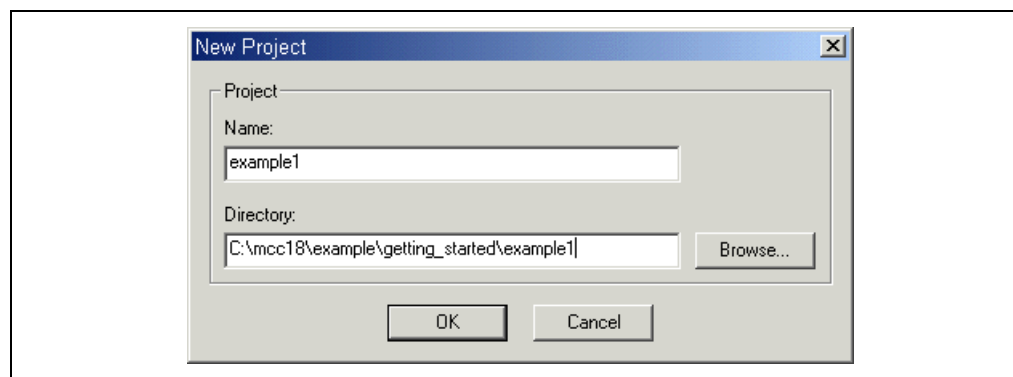
int counter;
void main (void)
{
    counter = 1;
    TRISB = 0;          /* configure PORTB for output */
    while (counter <= 15)
    {
        PORTB = counter; /* display value of 'counter'
                           on the LEDs */
        counter++;
    }
}
```

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).

3.3.1 Setting Up the Project

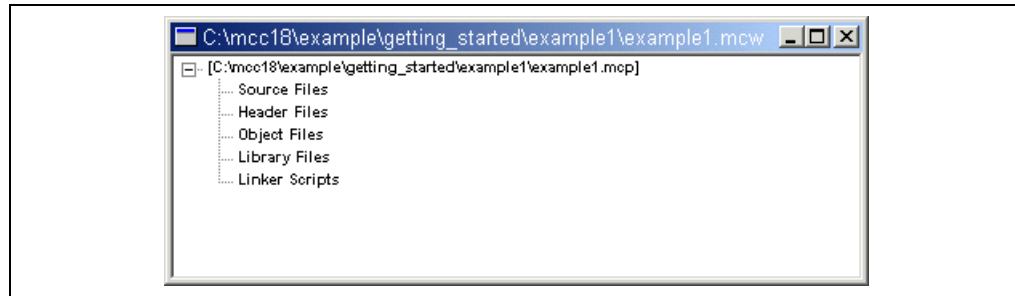
Select *Project>New* to create a new project. Then enter the name and directory of the project in the dialog that displays and click **OK**.

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.



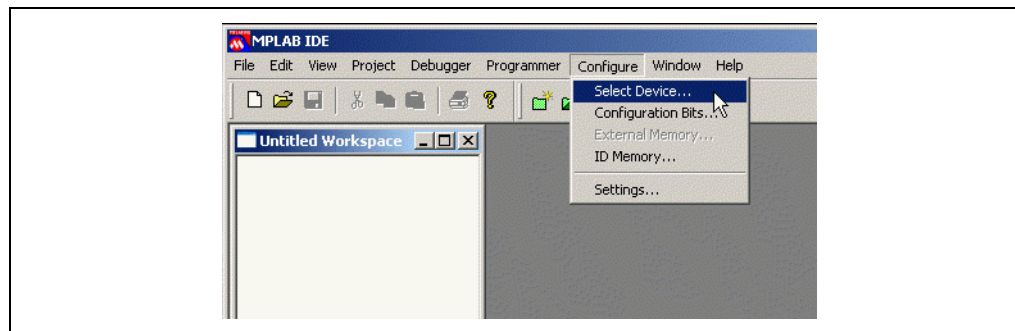
Note: The project name does not have to be the same as the directory name.

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

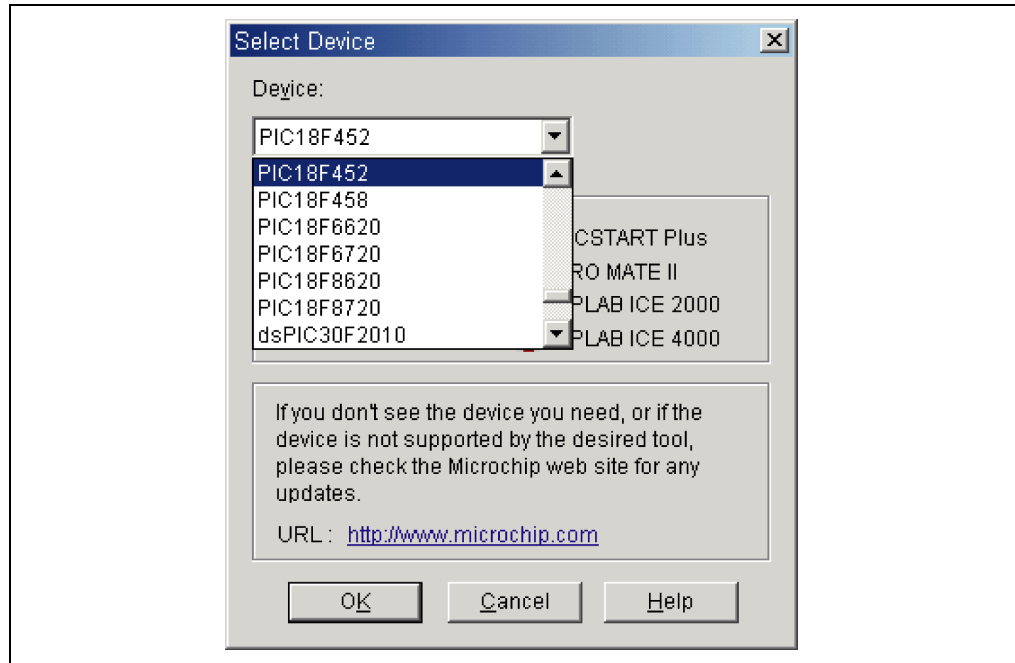


3.3.2 Select Target Processor

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



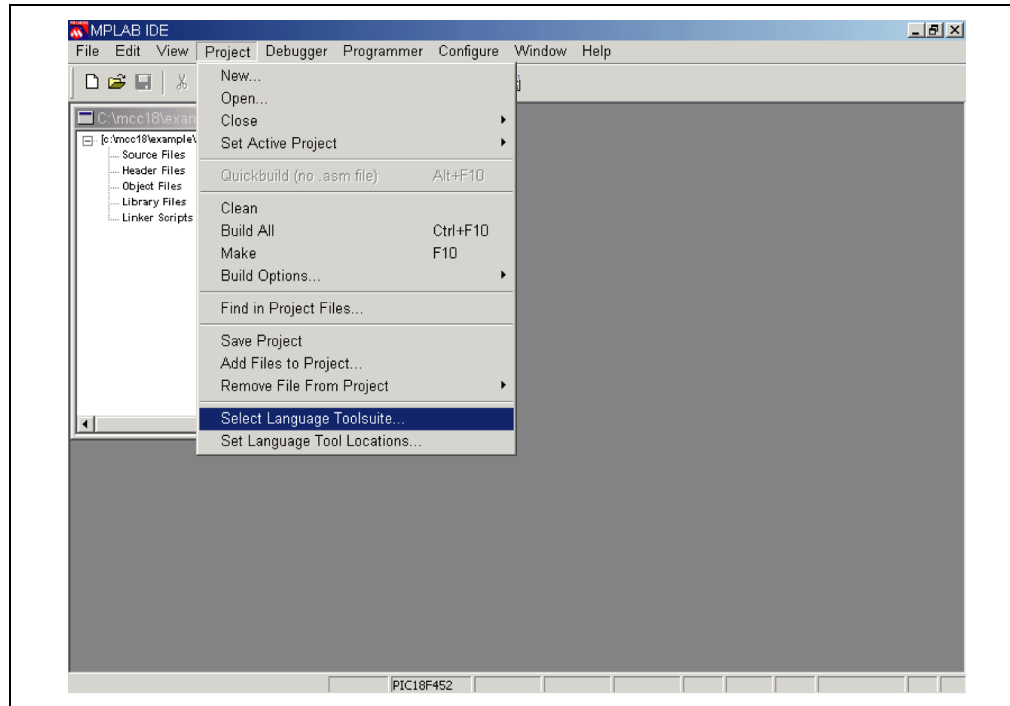
For this example, the PIC18F452 device will be used. Select the device and press **OK**.



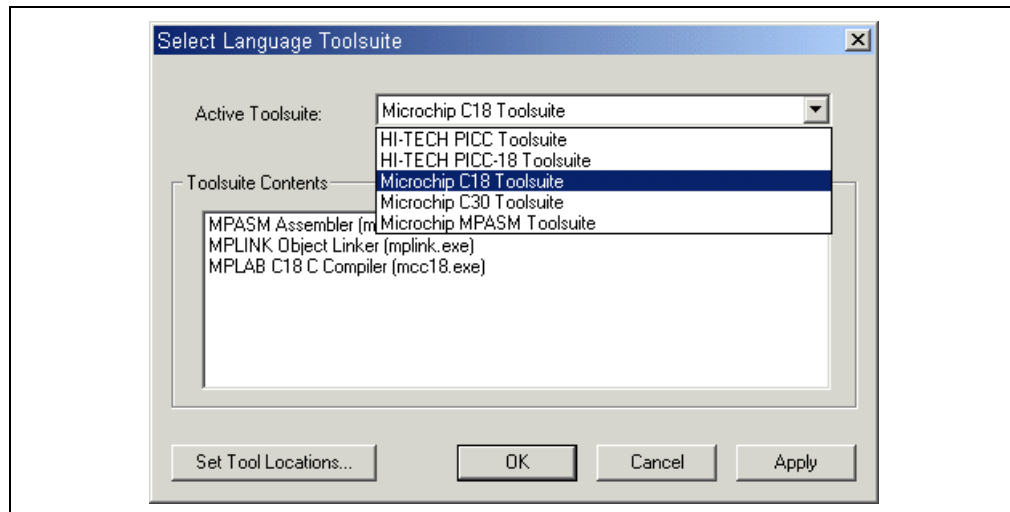
MPLAB® C18 C Compiler Getting Started

3.3.3 Select Project Settings

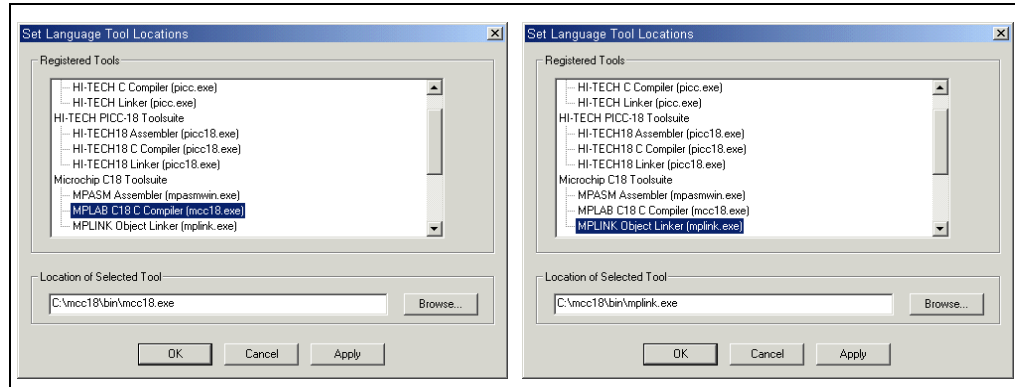
The MPLAB IDE needs to know which compiler and linker to use. To select MPLAB C18 and the MPLINK linker, first choose *Project>Select Language Toolsuite*.



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. Then click **Set Tool Locations**.

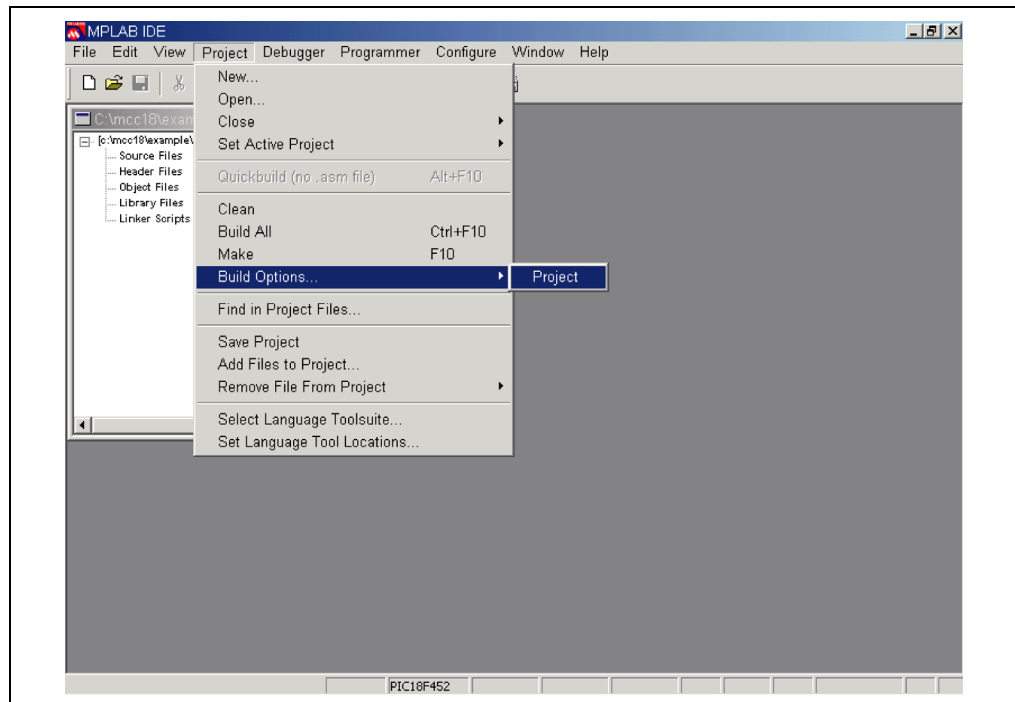


Select "MPLAB C18 C Compiler" and "MPLINK Object Linker", and make sure that the paths are to the newly installed versions of the executables, `mcc18.exe` and `mplink.exe`, respectively. Press **OK**.



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.

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

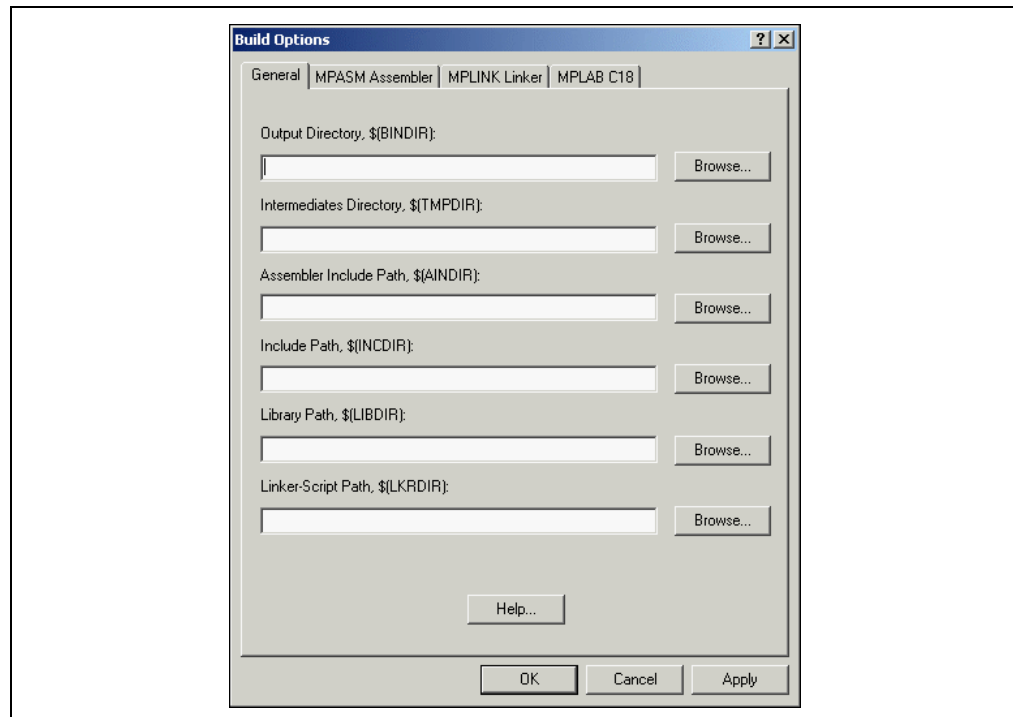


MPLAB® C18 C Compiler Getting Started

Enter the paths of the header file and library subdirectories of the MPLAB C18 installation directory on the "General" tab. 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.

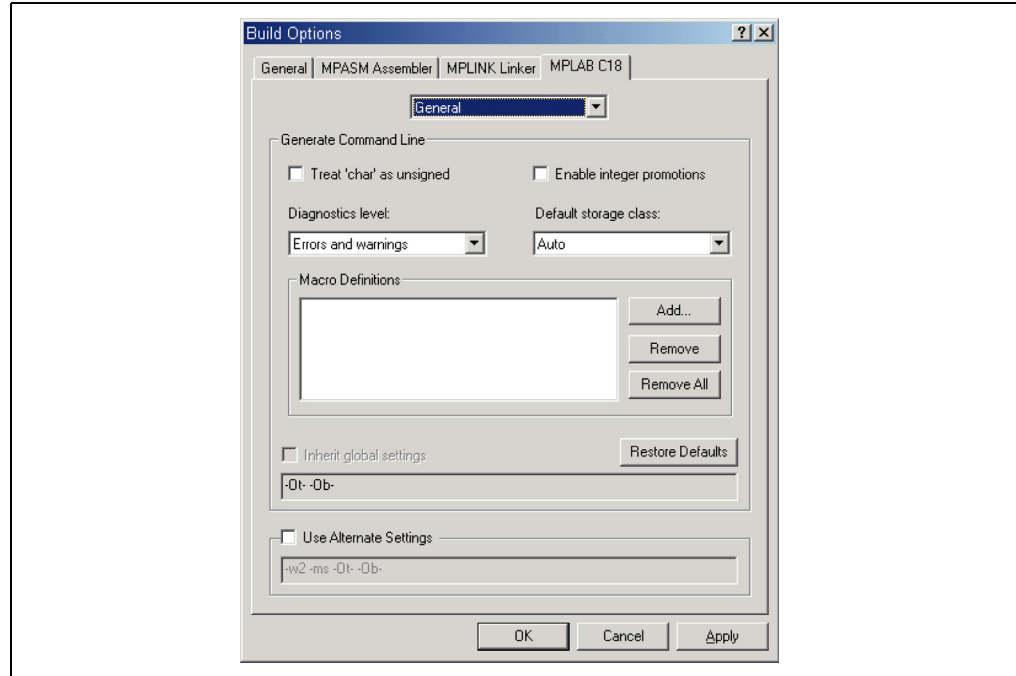
"Intermediates Directory" is where the object files produced by the compiler will be placed. Leave this entry blank as well; as a result, the object file (`example1.o`) will be placed in the same directory as the source file.

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.

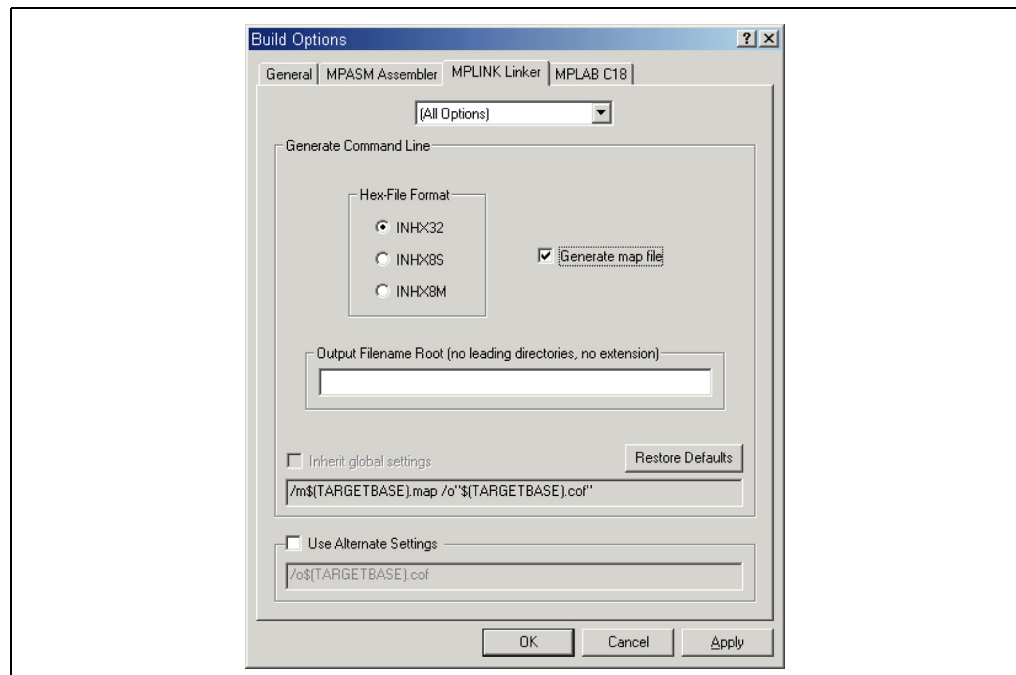


3.3.4 Select Compiler and Linker Settings

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.



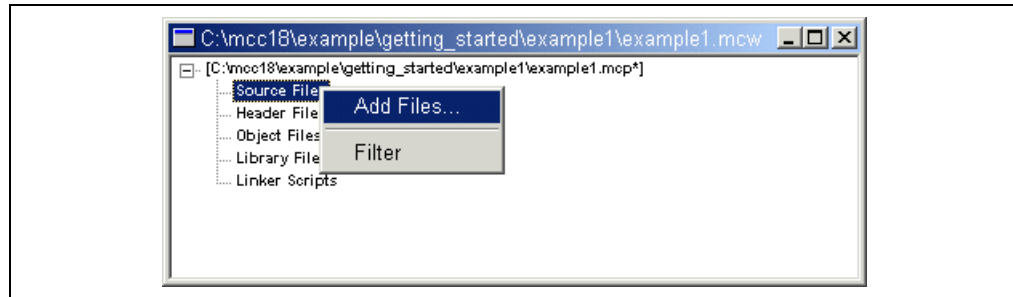
By default, when the MPLINK linker is run from the MPLAB IDE, it will not generate a map (example1.map) file. To change this, select "Generate map file" on the "MPLINK Linker" tab. Press **OK**. The default settings will be used for the remainder of the command-line options.



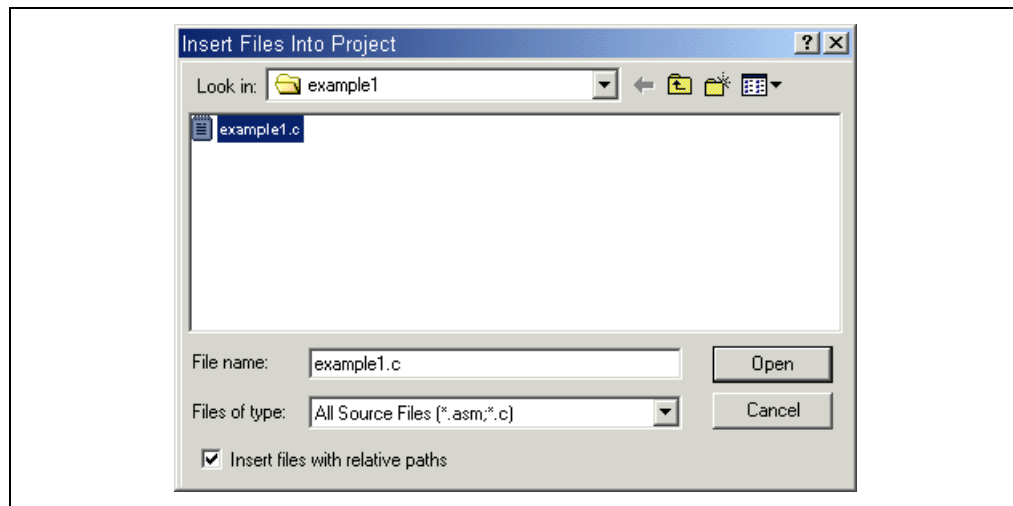
MPLAB® C18 C Compiler Getting Started

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".

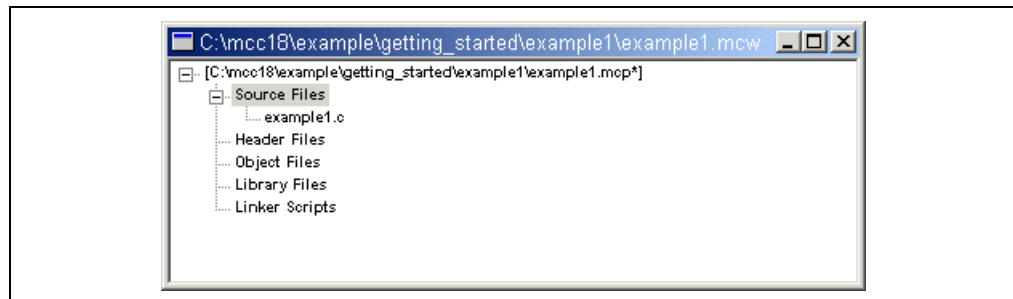


If `c:\mcc18\example\getting_started\example1` was chosen as the project directory, the source file `example1.c` already exists there. Browse to this directory and select the file `example1.c`. Press **Open** to add the file to the project.



Note: With "Insert files with relative paths" checked, the file's location will be stored as a path relative to the project directory. This allows the project to be built if it is moved to another directory, so long as all relative file paths remain the same.

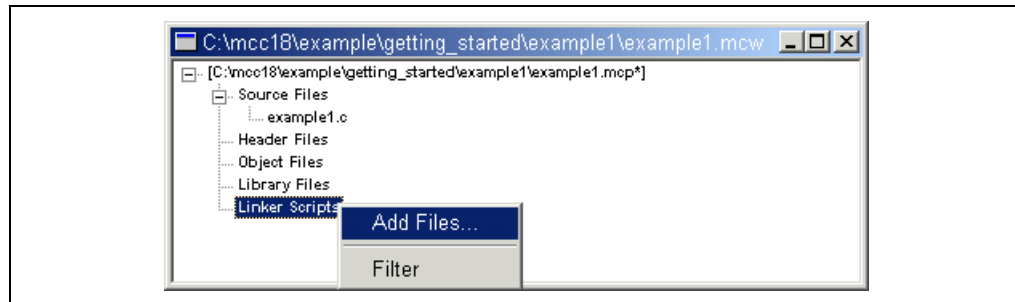
The source file should appear in the project tree.



Examples of Use

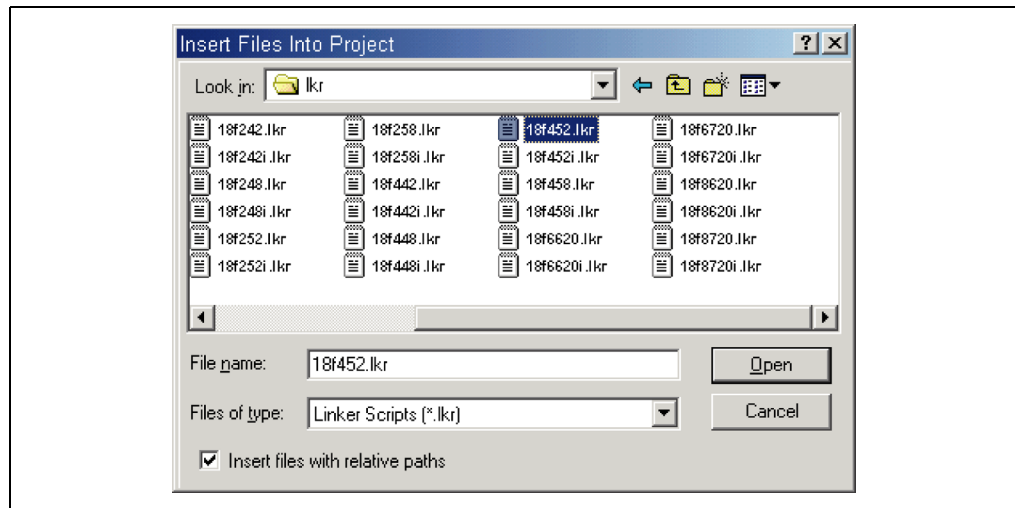
The header file is specified in the C source file; therefore no file needs to be added to "Header Files" in the project tree. A header file may be added to "Header Files" for convenient viewing of the file, but it is only required that the header file be included in the C source code to build the project. 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**.



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

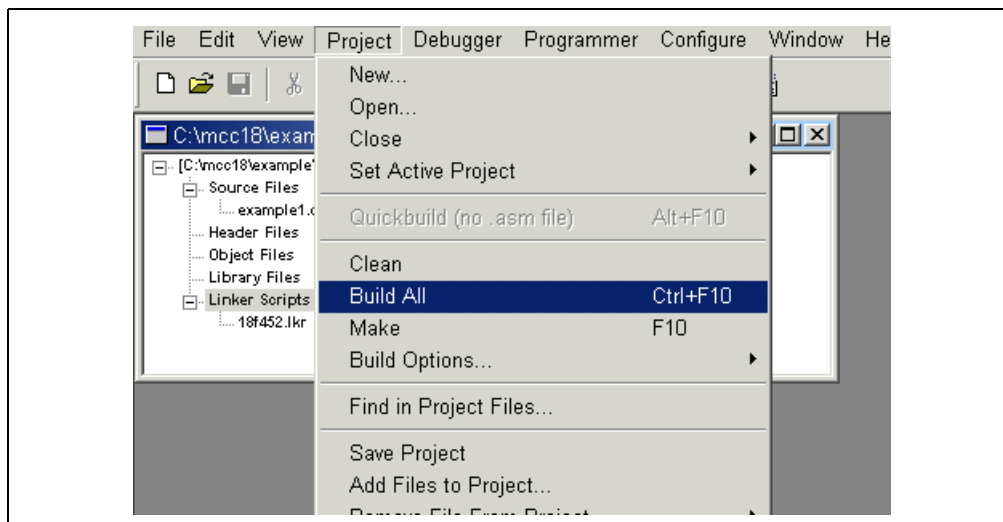
Press **Open** to add the file to the project tree.



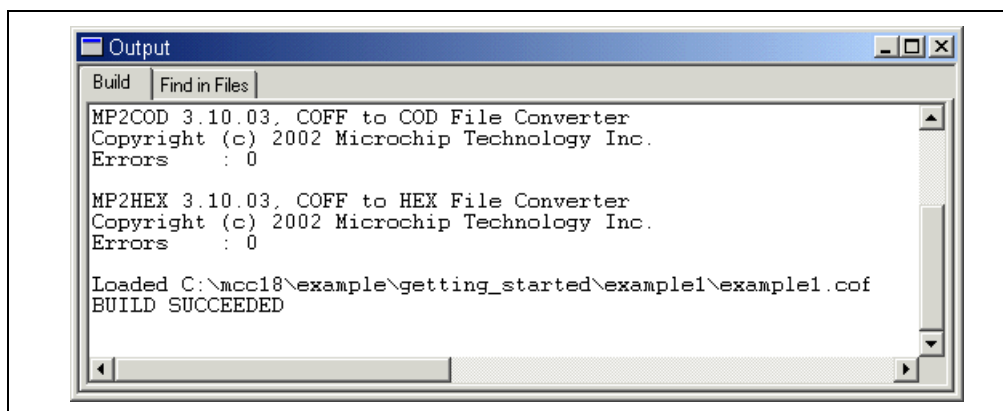
MPLAB® C18 C Compiler Getting Started

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.

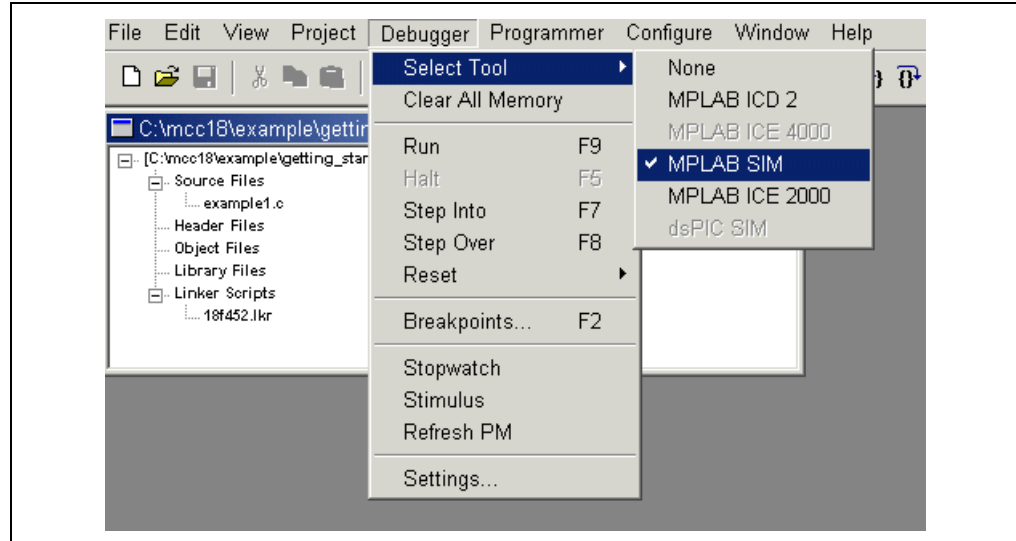


For this example, the output window should display no errors and a message stating the output file was successfully built should be visible. If there were any errors, check to see that the content of the source file matches the program text displayed at the beginning of 3.3 “**Example 1**”.

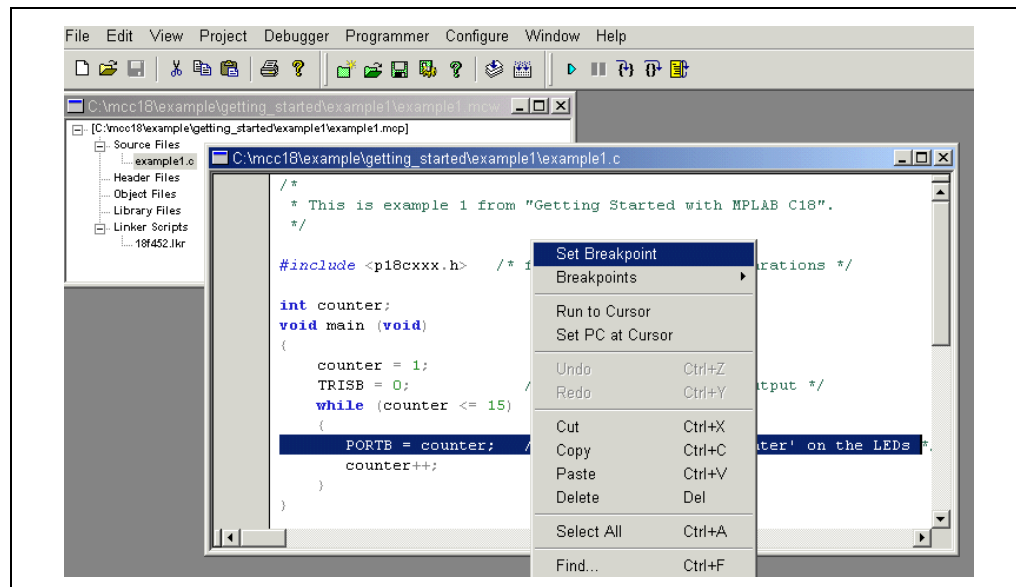


3.3.7 Debugging with the MPLAB SIM Simulator

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*.

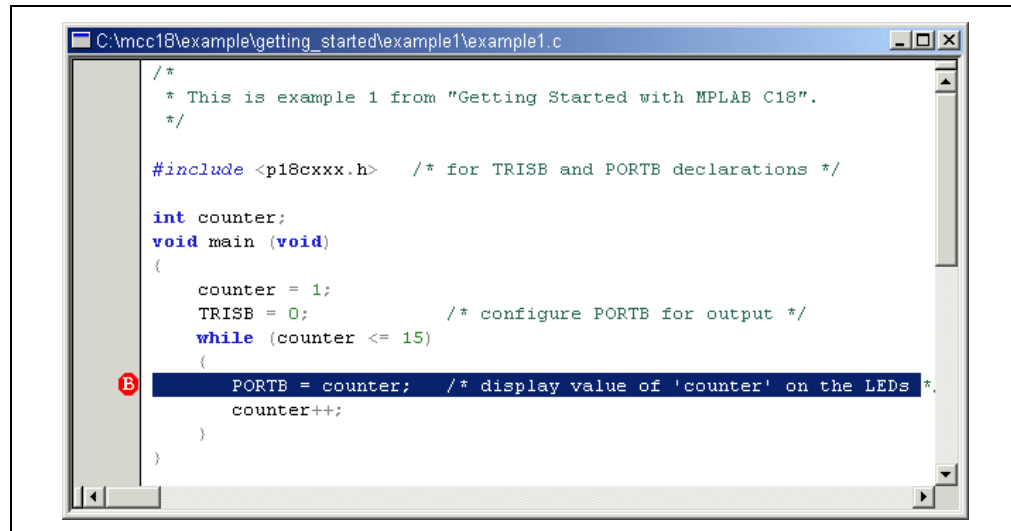


Open the source file by double-clicking on it in the project tree. In the source file, place the cursor over the line where the breakpoint is desired to be set, and click the right mouse button. Select "Set Breakpoint".

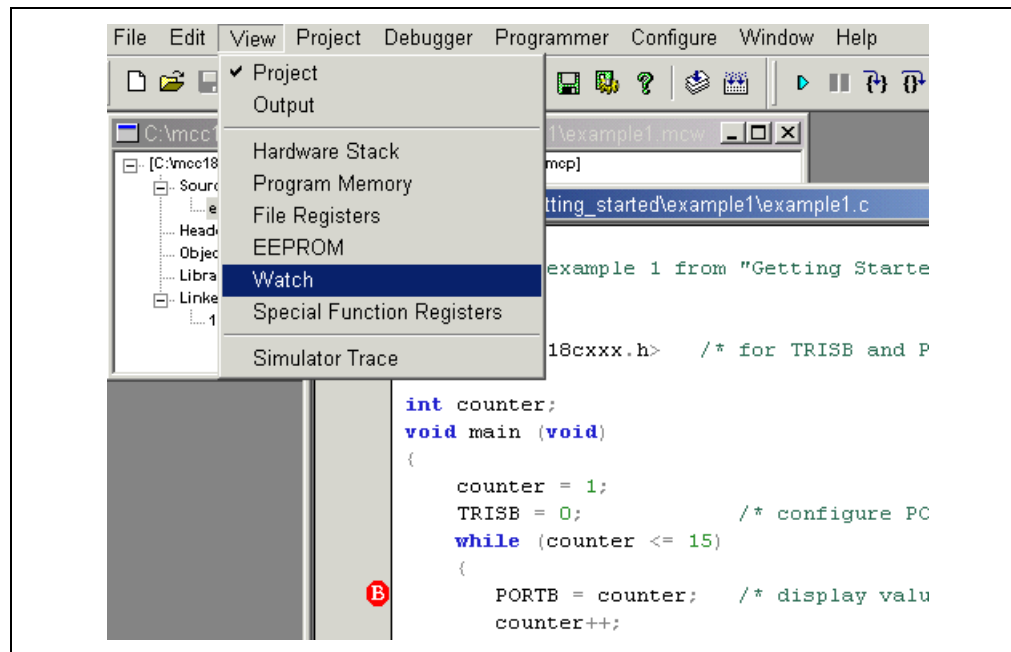


MPLAB® C18 C Compiler Getting Started

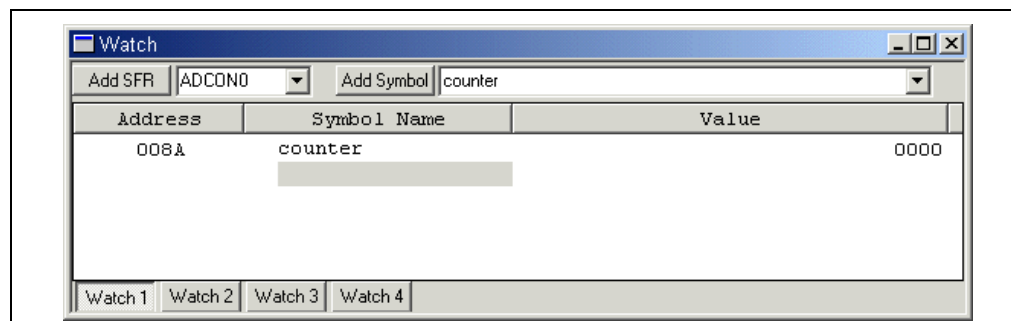
The red dot in the gutter along the side of the source window indicates that the breakpoint has been set and is enabled.



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

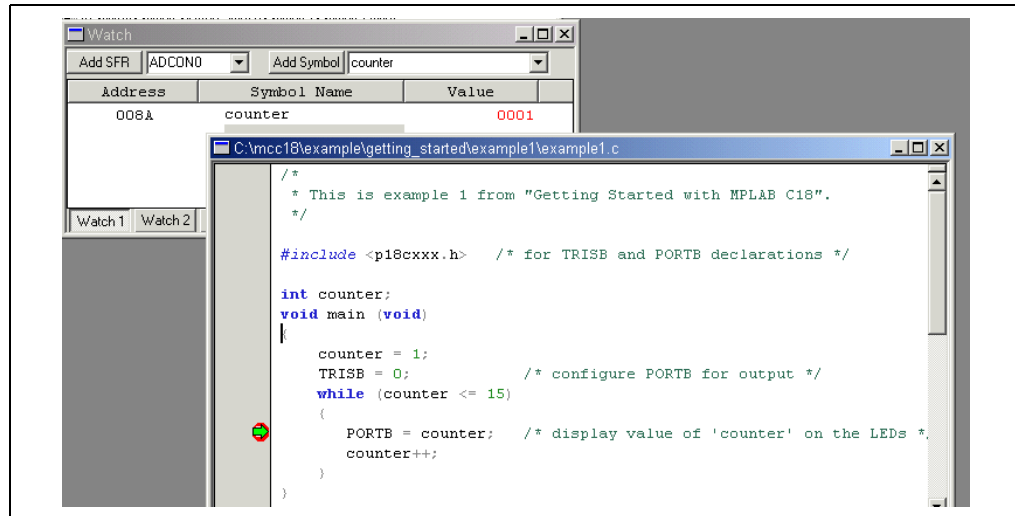


Select counter from the menu next to **Add Symbol**, and press **Add Symbol**.

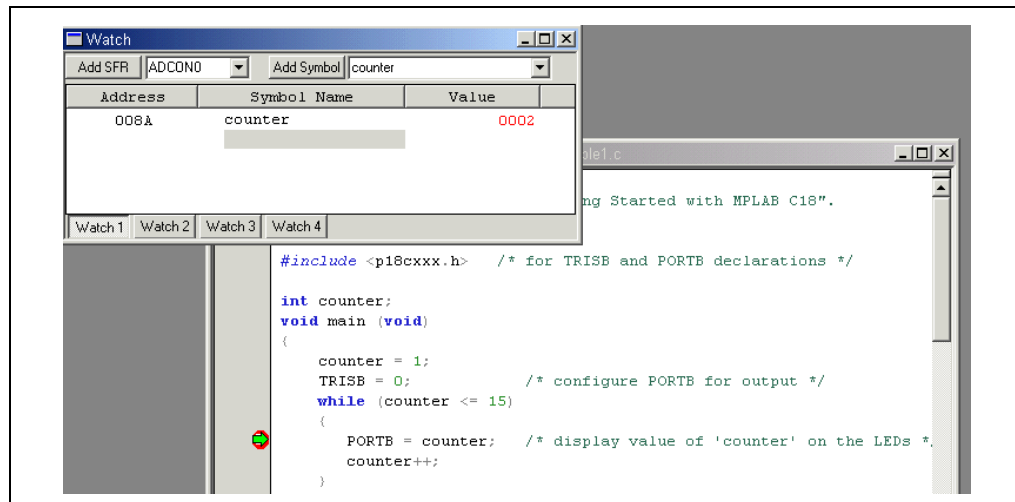


Press **Run** on the toolbar  to run the program.

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.



Press **Run** again to continue the program. Execution should halt again at the breakpoint. The watch window should show `counter` with a value of 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 pressing **Halt** on the toolbar .

MPLAB® C18 C Compiler Getting Started

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.

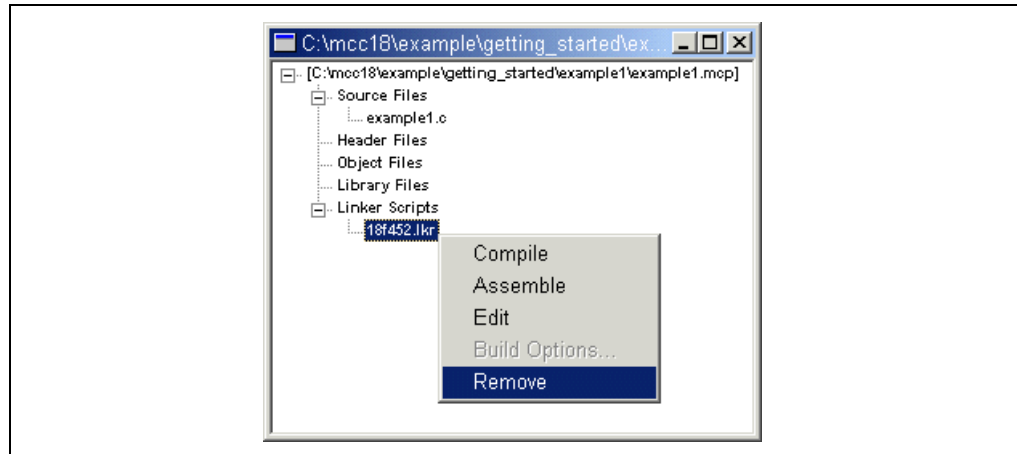
EXAMPLE 3-2: LISTING FILE

```

Address  Value  Disassembly      Source
-----
                                                    #include <p18cxxx.h> /* for
TRISB and PORTB declarations */
                                                    int counter;
                                                    void main (void)
                                                    {
0000e2  0e01  MOVLW 0x1          counter = 1;
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  aee8  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          }
Debugging with the MPLAB ICD 2
```

Examples of Use

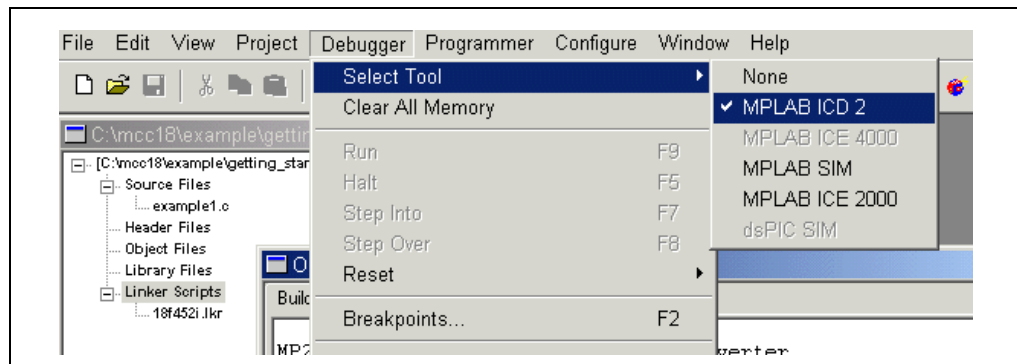
The MPLAB ICD 2 can be used to actually program the device and step through the application. To do this, the project must be rebuilt with a linker script designed for use with the MPLAB ICD 2. In the project window, click the right mouse button on the file `18f452.lkr` under "Linker Scripts", and press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



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.

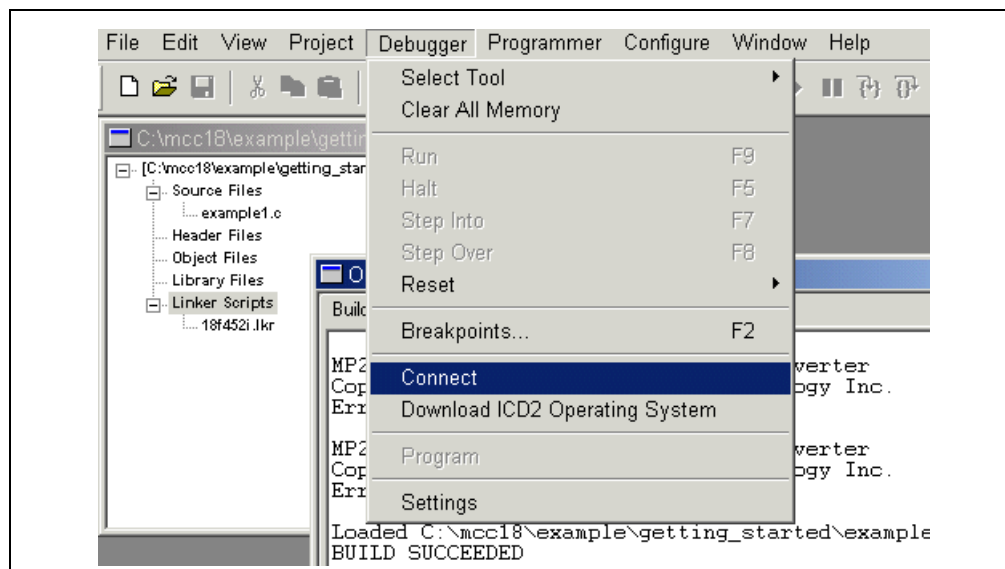
Rebuild the project by selecting *Project>Build All*.

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

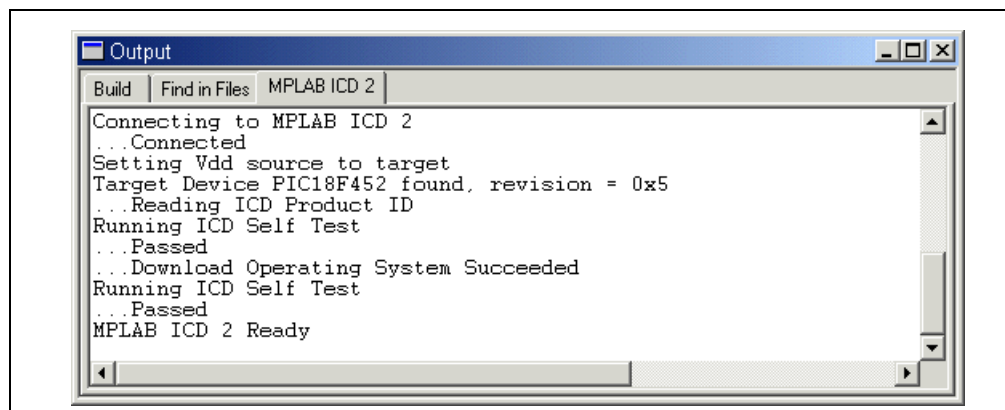


MPLAB® C18 C Compiler Getting Started

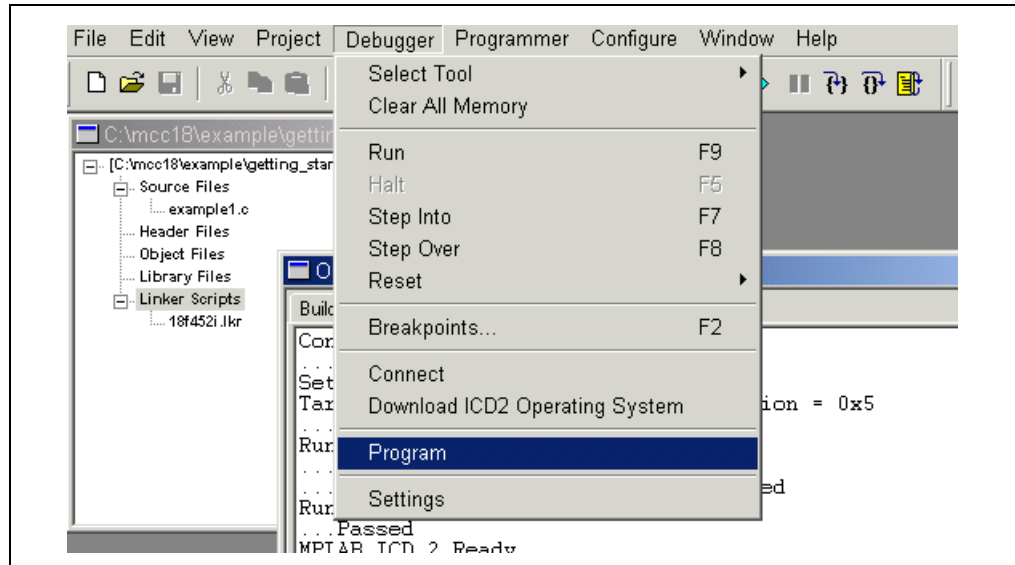
To connect the MPLAB ICD 2, choose *Debugger>Connect*. See the MPLAB ICD 2 documentation for information on how to configure the MPLAB ICD 2 connection settings.



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.

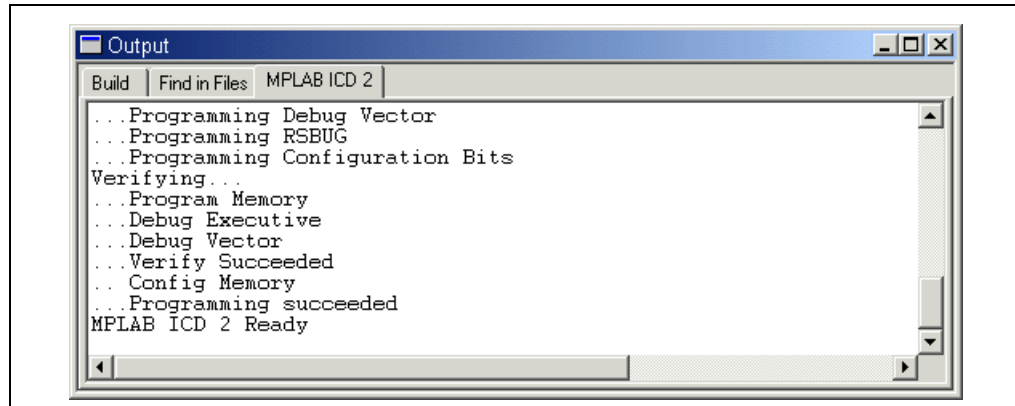


To program the device, select *Debugger>Program*.



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.

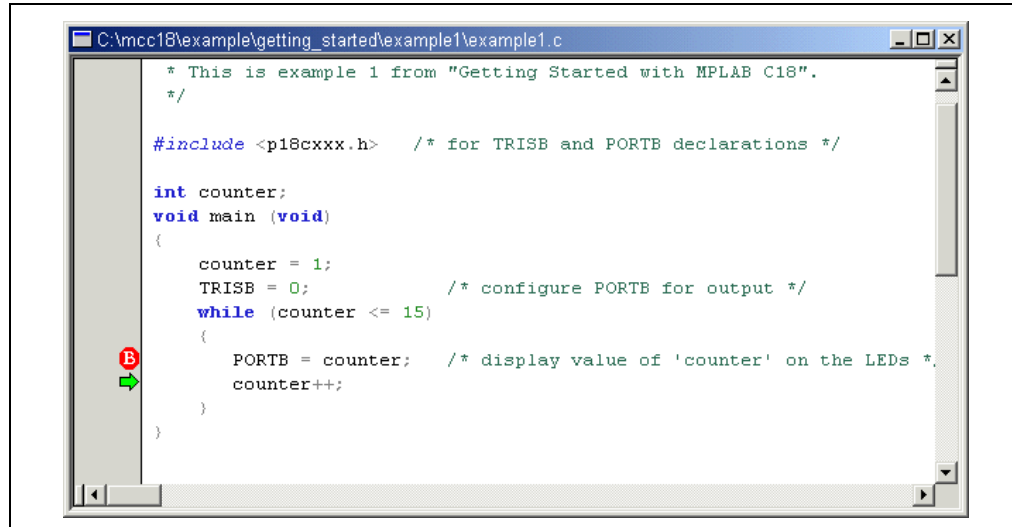
The output window should show that the programming operation succeeded.



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 pressed, 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 at the breakpoint is executed. The green arrow points to the next statement to be executed.



```
C:\mcc18\example\getting_started\example1\example1.c
* This is example 1 from "Getting Started with MPLAB C18".
*/
#include <p18cxxx.h> /* for TRISB and PORTB declarations */

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++;
    }
}
```

The screenshot shows a text editor window with the above C code. A red 'B' icon with a green arrow points to the line `PORTB = counter;` in the `while` loop.

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 pressed, execution halts after the assignment to `PORTB` and the value on the LEDs should reflect the incremented value of `counter`.

3.4 EXAMPLE 2

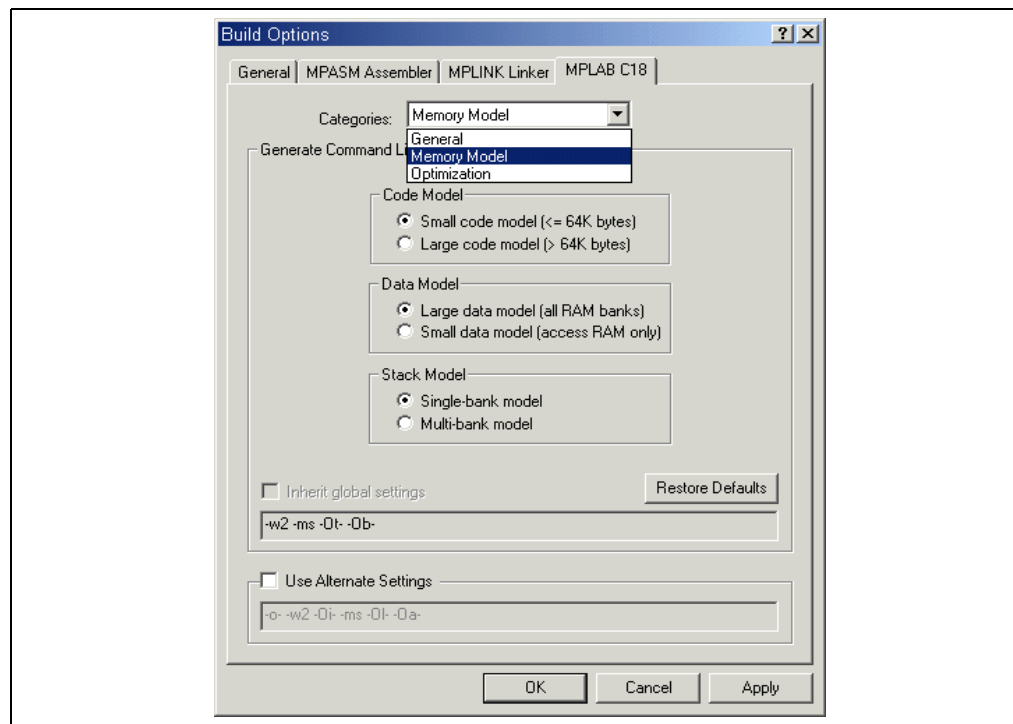
This example is designed for use with the MPLAB IDE v6.xx, the MPLAB ICD 2, the PICDEM 2 Plus demo board and the PIC18F452 device. It demonstrates the use of the MPLAB C18 peripheral libraries and the C standard library. It also demonstrates the allocation of variables into program memory. The program 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.

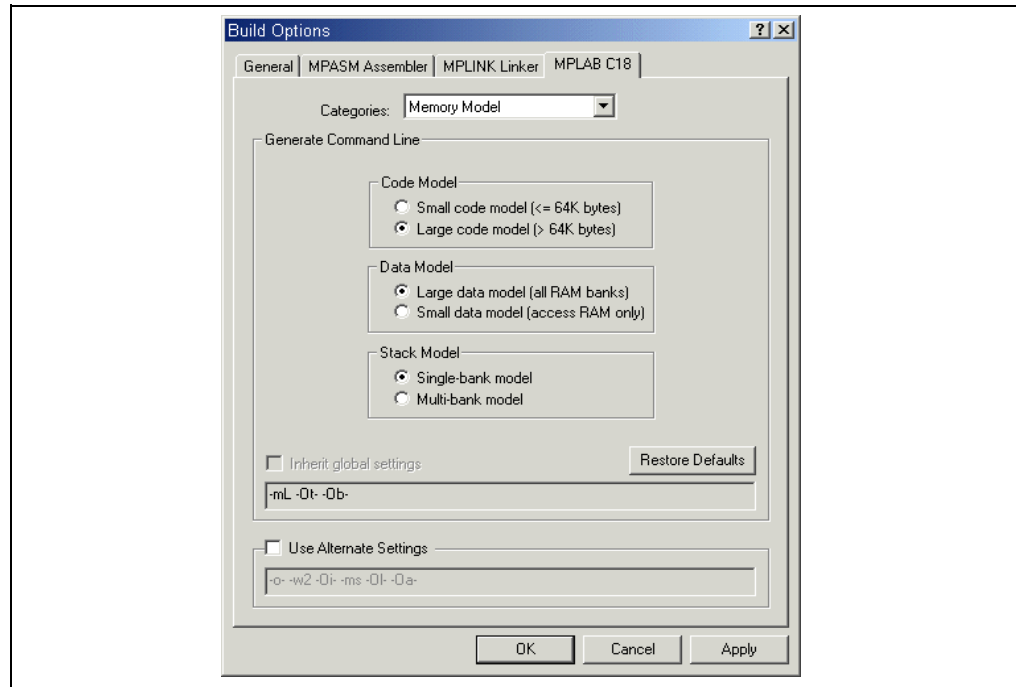
- MPLAB C18 places string literals in program memory; therefore, the `rom` keyword is required in the declaration of the array `string_table`. The `const` keyword alone will not place the data in program memory; the `rom` keyword is required. Since program memory in general cannot be modified without additional specialized code, the `const` keyword is appropriate.
- The 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.





```
#include <string.h>    /* for 'strcpygm2ram'    */
#include <stdlib.h>    /* for 'atoi'      */
#include <delays.h>    /* for 'Delay10KTCYx' */
#include <p18f452.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"
};

void main (void)
{
    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'
*/
        strcpygm2ram (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) */
    }
}
```

MPLAB® C18 C Compiler Getting Started

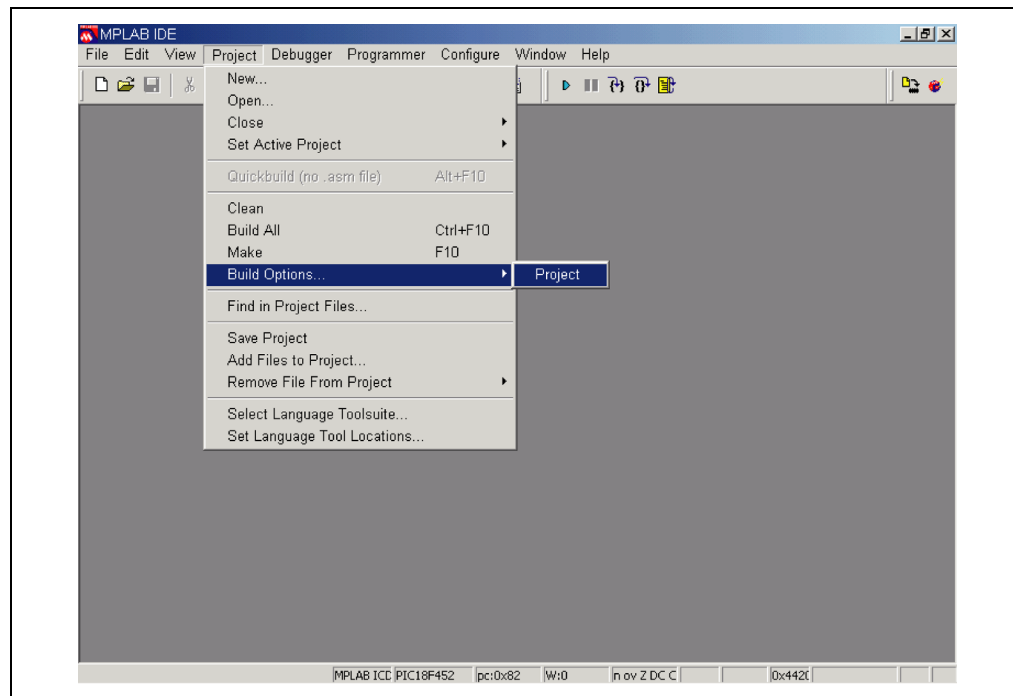
3.5 EXAMPLE 3

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.

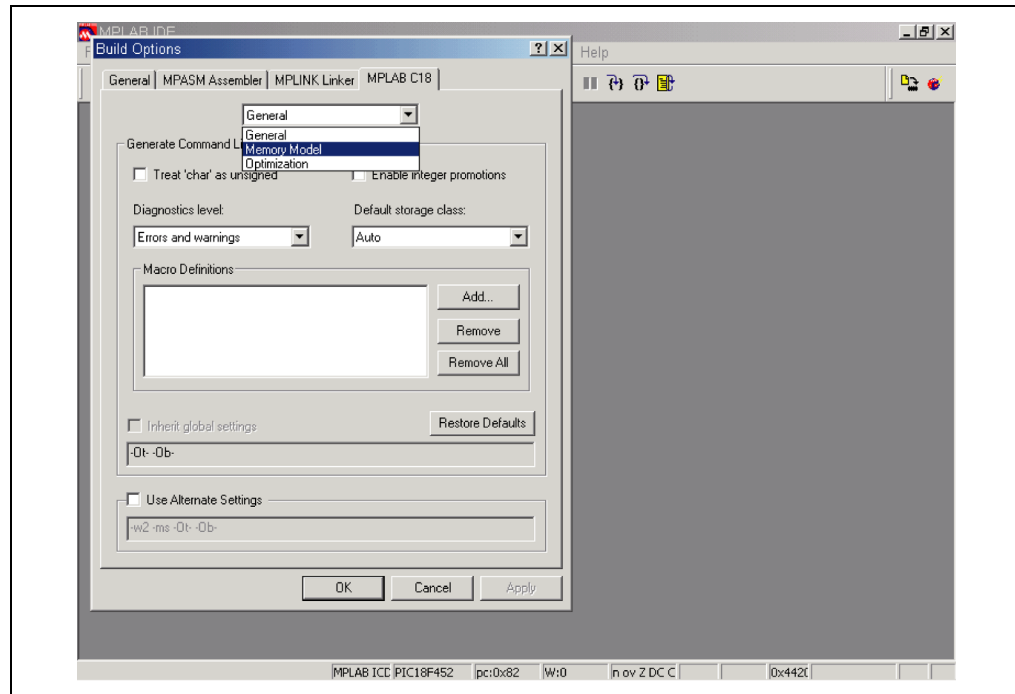
- The square root function is implemented as a table in program memory. This has several advantages. If the table were in data memory, it would need to be copied from program memory to data memory at program initialization. Locating the table in program memory also saves data memory space. Finally, the 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.

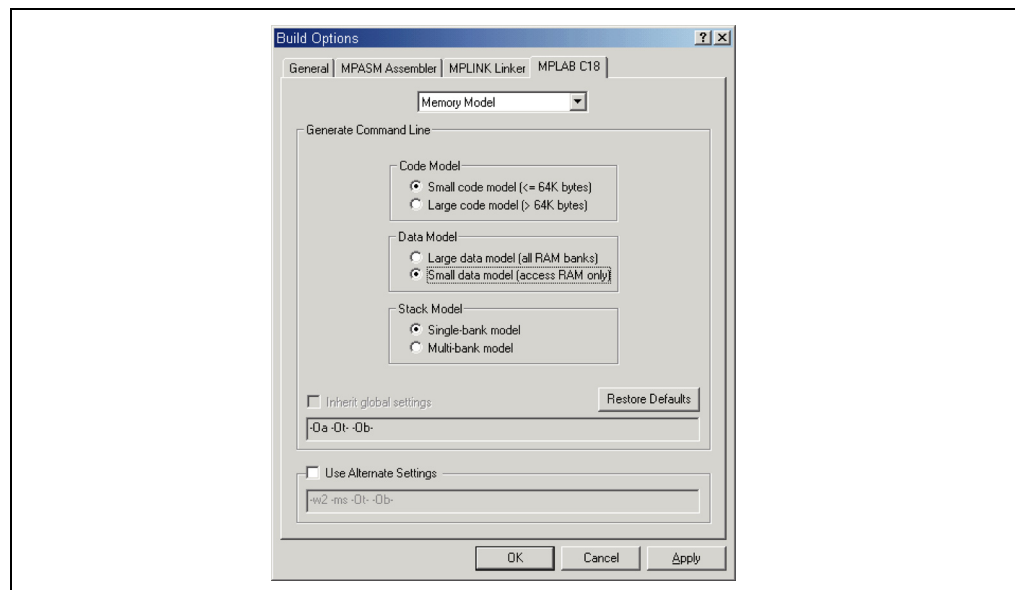
MPLAB C18, by default, assumes all statically allocated data objects, unless explicitly specified with the `near` type qualifier, reside in banked (non-access) RAM. This default behavior can be changed with the command-line option `-Oa+`. To make this change, choose *Project>Build Options>Project*.



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



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. Press **OK**.



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, 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.

3.6 EXAMPLE 4

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. At this address is located the interrupt service routine (*high_ISR*) which branches to the procedure that turns the buzzer either off or on.
- *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 */

/*
 * 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 */
}
```

MPLAB® C18 C Compiler Getting Started

```
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

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) is 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.

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). *See recursive.*

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.

runtime 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

A		N	
Access RAM	1, 19, 40	New Project	20
Add Files to Project	26	P	
B		Paths	24
Breakpoint	29, 31, 36	Project Paths	24
Build Options	25	Project Settings	22
Build Project	28	Project Tree	21
C		R	
COD	9	README File	3, 12
COFF	9	References	3
Customer Support	4	S	
D		Select Device	21
Directory	12	Select Language Toolsuite	22
Directory Contents	8	Set Tool Locations	22
Documentation	14	Support	
Conventions	2	Customer	4
Layout	1	System Requirements	7
Numbering Conventions	2	T	
Updates	2	Troubleshooting	3
E		U	
Examples	8, 14, 20, 37, 40, 43	Uninstalling MPLAB C18	17
H		User's Guide	3
HEX	9	W	
I		Watch Window	30, 31
Installing MPLAB C18	11		
interrupt service routine	1, 19, 43, 47		
L			
Language Tools	9, 22		
Libraries	3, 8, 14		
Linker Scripts	8, 14, 27, 33		
Listing Files	32		
little endian	47		
M			
Map Files	32		
MCC_INCLUDE	16		
Memory Model	41		
MPLAB C18 C Compiler Libraries	3		
MPLAB C18 C Compiler User's Guide	3		
MPLAB ICD 2	32		
MPLAB SIM Simulator	29		



MICROCHIP

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>

Rocky Mountain

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-4338

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, Indiana 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

Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto

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

ASIA/PACIFIC

Australia

Microchip Technology Australia Pty Ltd
Marketing Support Division
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing

Microchip Technology Consulting (Shanghai)
Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu

Microchip Technology Consulting (Shanghai)
Co., Ltd., Chengdu Liaison Office
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

Microchip Technology Consulting (Shanghai)
Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Hong Kong SAR

Microchip Technology Hongkong Ltd.
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

Microchip Technology Consulting (Shanghai)
Co., Ltd.
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

Microchip Technology Consulting (Shanghai)
Co., Ltd., Shenzhen Liaison Office
Rm. 1812, 18/F, Building A, United Plaza
No. 5022 Binhe Road, Futian District
Shenzhen 518033, China
Tel: 86-755-82901380 Fax: 86-755-82966626

China - Qingdao

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

India

Microchip Technology Inc.
India Liaison Office
Marketing Support Division
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaughnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan

Microchip Technology Japan K.K.
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

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

Singapore

Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan

Microchip Technology (Barbados) Inc.,
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

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

Denmark

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

France

Microchip Technology SARL
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

Microchip Technology GmbH
Steinheilstrasse 10
D-85737 Ismaning, Germany
Tel: 49-089-627-144-100
Fax: 49-089-627-144-44

Italy

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

United Kingdom

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

02/12/03