

# Getting Started Guide For the PIC<sup>®</sup> MCU

## Command-Line Compilers for C

January 2008

**Includes device programming with the Mach X and  
ICD plus MPLAB<sup>®</sup> integration instructions.**



---

Custom Computer Services, Inc.  
Brookfield, Wisconsin, USA  
262-522-6500

Copyright © 2008 Custom Computer Services, Inc.

All rights reserved worldwide. No part of this work may be reproduced or copied in any form by any means—electronic, graphic or mechanical, including photocopying, recording, taping or information retrieval systems—without written permission.

PIC<sup>®</sup>, PICmicro<sup>®</sup> and MPLAB<sup>®</sup> are registered trademarks of Microchip Technology Inc. in the USA and in other countries.



Custom Computer Services, Inc. proudly supports the Microchip brand with highly optimized C compilers and embedded software development tools.

# TABLE OF CONTENTS

## **Chapter 1**

Installing and Setup for Command-Line Compilers

## **Chapter 2**

Compiling a Program

## **Chapter 3**

Command Line Options and Files Produced

## **Chapter 4**

Introduction to the Mach X Device Programmer

## **Chapter 5**

Mach X Production Use and Advanced Features

## **Chapter 6**

In-Circuit Serial Programming with an ICD Unit.

## **Chapter 7**

The In-Circuit Programming/Debugging Interface

## **Chapter 8**

MPLAB® Integration

# 1

## INSTALLING AND SETUP FOR COMMAND LINE COMPILER

- ❑ Insert the CD-ROM and wait for the install program to start. If your computer is not set up to auto-run CDs, then select **Start>Run** and enter **D:\SETUP1.EXE** where D: is the drive letter for your CD drive.
- ❑ Select the compiler. Click on **Start Install** and use the default settings for all subsequent prompts by clicking NEXT, OK, CONTINUE...as required.



- ❑ Identify a directory to be used for the programs in this booklet. The install program will have created a directory **c:\program files\picc** that may be used for this purpose. Check to see if the file **CCSC.ini** exists. If the file does not exist, then open Notepad or another text editor and enter in the following line:  

```
I="C:\Program Files\PICC\Devices;C:\Program Files\PICC\Drivers"
```
- ❑ Save this file in the same directory as **CCSC.exe**, which is default is in **c:\program files\picc**. Make sure this file is saved as an **CCSC.ini**

# 2

## COMPILING A PROGRAM

- In the PICC directory, create a new folder called Exercises.
- In the Exercises folder, create a file called **ex2.c** using any text editor, such as NotePad.
- Type in the following program and save the file.

```
#include <16f877A.h>

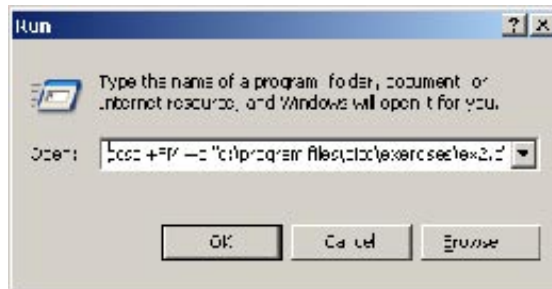
#fuses HS,NOLVP,NOWDT,NOPROTECT
#use delay (clock=20000000)

#define GREEN_LED PIN_A5

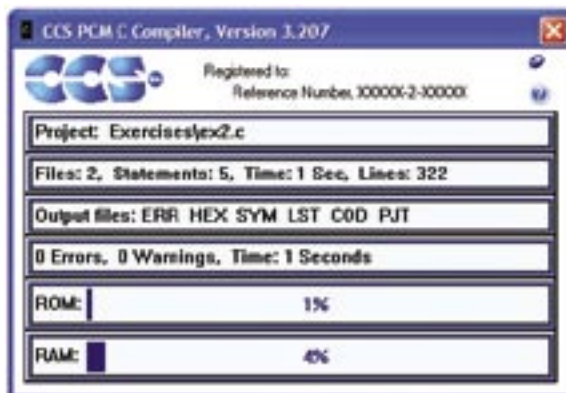
void main () {
    while (TRUE) {
        output_low(GREEN_LED);
        delay_ms(1000);
        output_high(GREEN_LED);
        delay_ms(1000);
    }
}
```

- The line `#include <16f877A.h>` in the above example assumes a 14-Bit compiler. For PIC18 change the first line to `#include <18F452.h>`
- Use +FH instead of +FM, as +FM is for 14-Bit parts, +FH is for PIC18 parts and +FB is for 12-bit parts.
- Next, open a command line prompt by clicking **Start>Run**.
- Type the following line into the command prompt and press the Enter key:  

```
ccsc +FM +p "C:\Program Files\PICC\Exercises\ex2.c"
```
- The "+p" option tells the compiler to keep the compiler window open. This allows the user to read the compiler window.



- If correct, the following screen should appear:



- Change the "While(TRUE)" to "While(TRU)" in the ex2.c file and compile again.
- There should now be a compilation error. The error message identifies the line with the error.

# 3

## COMMAND-LINE OPTIONS AND FILES PRODUCED

The previous chapter showed you how to compile a program. This chapter will introduce some of the other command line options available and detail the output files.

- ❑ Enter the following command (From **Start>Run**) for a full list of Command Line Options:

```
CCSC +?
```

- ❑ The more commonly used options are:

- +PE Only leave the compiler window up if there is an error.

- +T Create a call tree file (.TRE).

- D Do not create a debug file.

- +EA Put all errors/warnings in the .err file. (By default only the first error is entered).

- ❑ To find the versions of the installed compilers:

```
CCSC +V
```

- ❑ To find the chips supported by this version:

```
CCSC +Q
```

- ❑ The CCS C Compiler also has some complex options that may appear on the command line. For example:

```
CCSC +FM #Debug="TRUE" MyFile.C
```

- ❑ The above is the same as if the following were at the TOP of MyFile.C

```
#define Debug TRUE
```

- ❑ If included files in the source code are not fully specified, the compiler will search for the files. The directories searched can be listed using the I= option. For example:

```
I="C:\PIC\MYINCLUDES;C:\PIC\GROUPINCLUDES"
```

- ❑ Before processing the options on a command line, the compiler reads options from CCSC.ini. This file is in the same directory as CCSC.exe. It is common practice to put I= in that file so it does not need to be retyped on every occasion.

- ❑ After a compilation the compiler creates several files in the same directory as the main project file. These files have the same name as the main project file and the following extensions:

.HEX	The final object code image that can be loaded into the target chip.
.LST	A mixed C/Assembly text file that shows the C source line and the corresponding assembly.
.SYM	A text file that shows where in memory all user symbols have been located. This file also has a summary of all compilation activities at the end.
.COD or .COF	A file used by debuggers that contain debugging information like symbols, line numbers, etc.
.ERR	A text file with errors and warnings.
.TRE	An optional file that shows the program call tree.
.PJT	A project file used by the PCW and MPLAB® IDEs to maintain project-specific settings.

## NOTES

- For more help use:  
**Start>Programs>PICC>PIC C HELP**

## 4

# INTRODUCTION TO THE MACH X DEVICE PROGRAMMER

Mach X is used to burn and run a .hex file on a PIC® chip. This section will give you an overview of how to use the Mach X software. For more help click on the Help icon in the upper right of any Mach X window.

- Install the Mach X software from the CCS website ([www.ccsinfo.com/download.shtml](http://www.ccsinfo.com/download.shtml)) or from the installation CD.
- Connect the Mach X to the PC and start the software.
- The main window is the Program Window that has options to burn, erase and read the target chips. You can also review and edit the hex file.
- Place the DIP chip to be programmed into the DIP socket or connect the target board using the ICD jack and modular cable.
- Click on browse and select the **ex2.hex** file. The chip will automatically be selected in the device box.
- Choose the ZIF socket or ICD jack option.
- Click on **Burn Chip** to program the target.
- The setup button edits the programming and erase settings.




- The review/edit section allows for viewing and editing the hex files.



# 5

## MACH X PRODUCTION USE AND ADVANCED FEATURES

Click on the windows icon  (near the help icon) to switch between windows.

### SELECTOR WINDOW

The selector window lists all the hex files in the currently selected directory and the user can select one to program the chip.



- Comments on the HEX file (the first comment line) may appear on the line of this window, allowing product names or versions to be shown.
- Additional operator instructions like serial numbering may be included in the HEX file as additional comment lines. A pop-up screen will appear before the chip is burned.

### Burning a Chip

- Click on the folder icon.  
Navigate to C:\Program Files\PICC\Exercises\.
- Click OK
- Select ex2.HEX
- Click **Burn Chip**

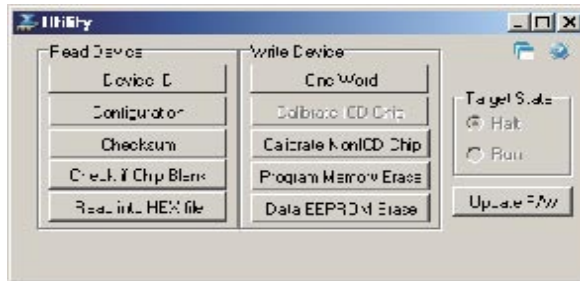


# 5

## MACH X PRODUCTION USE AND ADVANCED FEATURES (CONT.)

### ❑ UTILITY WINDOW

The utility window provides advanced features like updating the Mach X firmware, checking the device I.D., fuses, checksum and calibrating special chips.



### ❑ DIAGNOSTIC WINDOW

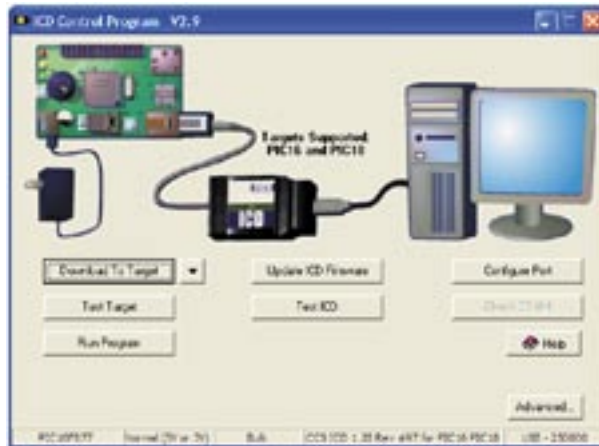
The diagnostics window can be used to test the hardware setup and to check firmware and software version details.



# 6

## ICSP WITH AN ICD UNIT

- ❑ Install the ICD software from the CCS website ([www.ccsinfo.com/download.shtml](http://www.ccsinfo.com/download.shtml)) or from the installation CD.
- ❑ Compile the file in Chapter 2 and ensure you have the **EX2.HEX** file.

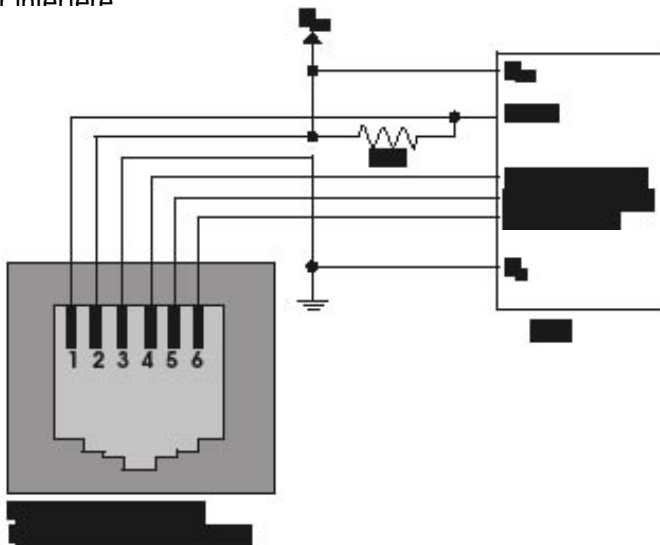


- ❑ Connect the ICD to the PC and the target board and power up the target board. Verify the LED on the ICD is on.
- ❑ Start the ICD program by going to **Start>Program Files>PICC>ICD**.
- ❑ Start by clicking **Test ICD**, then **Test Target**, to make sure that all the equipment is ready.
- ❑ Once all the checks and tests have passed, click on Download To Target. Open **c:\Program Files\PICC\Exercises>EX2.HEX**.
- ❑ After the progress bar appears and shows that the program has been loaded to the target, click **Run Program**. Pin A5 should toggle once every second. Click the **Stop Program** when finished.
- ❑ The **Update ICD Firmware** button allows firmware to be updated to the newest version.
- ❑ See Chapter 7 for target board electrical connections.
- ❑ The **Advanced** button brings up the advanced window that provides options to read, calibrate, erase, verify, change the programming and erase settings and test the hardware setup.
- ❑ Refer to the ICD help file for more details.

## 7

# THE IN-CIRCUIT PROGRAMMING/ DEBUGGING INTERFACE

- The ICD-S, ICD-U and Mach X have an RJ12 connector, and provide an RJ12 cable to connect to prototyping boards. Below is the pin-out of the RJ12 connector.
  - The ICD-U40 and Mach X is powered by the USB.
  - The ICD-S40 requires mA. If the target power is not to be used, the connection from 5-2 may be cut and an external 5V power supply can be used. This technique may also be used to power both the ICD and the target through the ICD connectors.
  - To program and/or debug in-circuit, two I/O pins (B6,B7) are reserved. If debugging is not to be done, then these pins may also be used in the target circuit. However, take care to ensure the target circuit has high impedance during programming.
  - The ICD does not use the Low Voltage Programming mode. C programs should set the NOLVP Fuse.
  - The target chip oscillator must be running for the ICD to work with a debugger. Programming can be done without an oscillator.
  - The B3 pin is optional and is not used for programming. However, the Monitor feature of the debugger does use B3. It is possible to program and debug (without monitor) and allocate B3 to the target hardware. In this case, do not connect B3 to the ICD connector. If the monitor feature is not used, userstream can be disabled in the configure tab and the connection from 1-6 does not matter. In the old version of the software, where you cannot disable the userstream using the configure tab, the pin needs to be pulled high at all times. Although B3 is recommended, any PIC pin can be used for this feature.
  - The MCLR pin is used for programming and debugging. Note that during programming the voltage is 13V. The 47K resistor to 5V is sufficient isolation for the 13V. However, if anything else is connected to the MCLR pin, be sure the 13V will not damage or interfere



8. Note the ICD to target cable reverses the pins so the MCLR signal is ICD pin 6 and connects to the target pin 1.

Example of a few chips that do not use B6,B7

<i>Chip</i>	<i>Instead of B6</i>	<i>Instead of B7</i>
<i>PIC12F629</i>	<i>GP1</i>	<i>GP0</i>
<i>PIC12F675</i>	<i>GP1</i>	<i>GP0</i>
<i>PIC12F683</i>	<i>GP1</i>	<i>GP0</i>
<i>PIC16F630</i>	<i>RA1</i>	<i>RA0</i>
<i>PIC16F676</i>	<i>RA1</i>	<i>RA0</i>
<i>PIC16F684</i>	<i>RA1</i>	<i>RA0</i>
<i>PIC16F688</i>	<i>RA1</i>	<i>RA0</i>

- The following chips are examples that do NOT have debugging capability in the standard version of the part. A specific ICD version of the chip is needed for debugging. The ICD chip will have more pins.

<i>ICD-Chip</i>	<i>Pins</i>
<i>PIC12F629</i>	<i>14</i>
<i>PIC12F675</i>	<i>14</i>
<i>PIC12F683</i>	<i>14</i>
<i>PIC16F630</i>	<i>20</i>
<i>PIC16F676</i>	<i>20</i>
<i>PIC16F627A</i>	<i>28</i>
<i>PIC16F628A</i>	<i>28</i>
<i>PIC16F648A</i>	<i>28</i>
<i>PIC16F684</i>	<i>20</i>
<i>PIC16F688</i>	<i>20</i>

## CCS Plug-In Installation

- ❑ This plug-in allows MPLAB® 6.x/7.x IDE users to use CCS C Compiler to build their projects. The plug-in can be downloaded and installed from:

<http://www.ccsinfo.com/mplab-ccs-plugin-install.exe>

Note: This plug-in works with CCS C Compiler versions 3.168 and up.

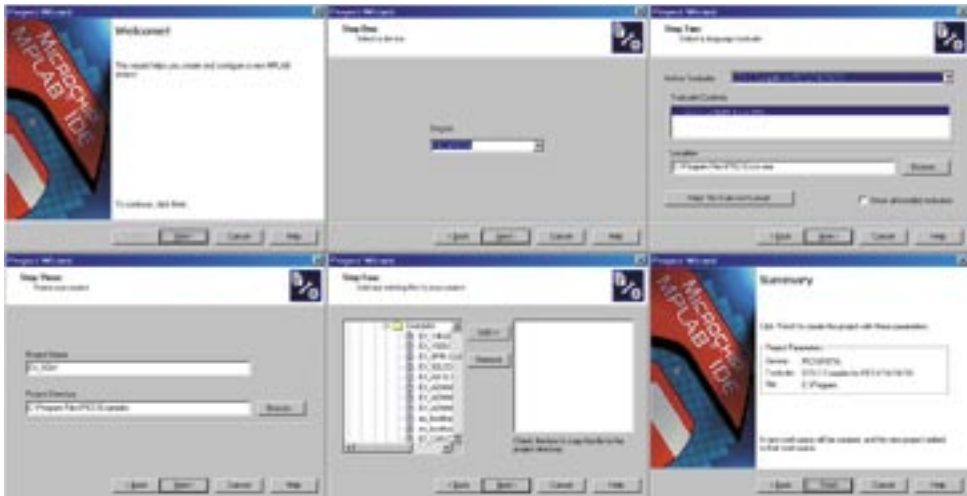
## MPLAB® IDE Installation

- ❑ You can download and install MPLAB® IDE from Microchip's website ([www.microchip.com](http://www.microchip.com)). MPLAB® IDE and additional plug-ins may be installed in any order. Use only MPLAB® IDE versions 6.40 and up.

## Getting Started

### Starting a MPLAB® IDE Project From an Existing Source

- ❑ To start a new project, you can use either the Wizard or create it manually. To create a CCS project using the Wizard, follow these steps:



- ❑ Select the “Project” Menu.
- ❑ Choose “Project Wizard” to make the Project Wizard window appear.
- ❑ Click “Next” on the Welcome Page.
- ❑ Step One, choose “PIC16F877A” from the drop-down list and click “Next.”

- ❑ Step Two, select “CCS C Compiler for PIC12/14/16/18” from the drop-down list. Click “Next.”  
Note: If the plug-in is not listed, make sure that “Show all installed tool suite” is checked. If it still does not appear, then the CCS Plug-in may not be installed correctly. You will need to re-install.
- ❑ Step Three, type the project name “EX\_SQW” and type or select the project directory “C:\Program Files\PICC\Examples.” Click “Next.”
- ❑ Step Four, select file “Ex\_SQW.C.” Click “Add.” Click “Next.”
- ❑ View Summary, click “Finish.”

## Start a New Project Manually

```
#include<16F877a.h>
#fuses HS, NOLVP, NOWDT, NOPROTECT
#use delay(clock=2000000)

void main() {
    int8 a, b, c;
    a = 2;
    b = 3;

    c = a + b;
    c = a * b;
    c = b - a;
}
```

- ❑ Enter the following program into MPLAB® IDE as “EX8.c” in the “Exercises Directory.”
- ❑ Go to *Project>New*.
- ❑ Name the project “EX8” and choose the “Exercises Directory.”
- ❑ Click “Add File to the Project” in the “Project” menu and select the “ex8.c” file.
- ❑ Click “Select Language Tool Suite” and make sure “CCS C Compiler for PIC12/14/16/18” is selected.
- ❑ Go to *Configure>Select Device* and select “PIC16F877A,” then click “OK.” MPLAB® IDE will then choose the right CCS C Compiler and options for the device.

## Compiling

- ❑ Go to Project>Compile.
- ❑ If the output window is open, the status of the compiler is displayed here.
- ❑ You can view the CCS C Compiler options that MPLAB® IDE is using by going to Project>Build Options>Project. If you wish to change these options, click on the “Use Alternative Settings” checkbox to edit the options that you want.
- ❑ The output window will show if the code contains any warnings or errors. The line number, which document, and a brief description of the error will be listed for each error or warning. To correct an error or warning go to the line of the specified document and check where the error or warning description occurred on that line. Go through each error and warning to get the project to successfully compile.

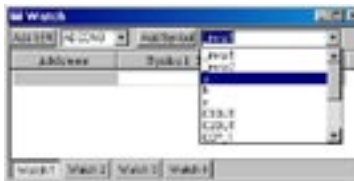
## Debugging



- ❑ Follow the steps of the MPLAB® ICD 2 USB Device Driver First Time Installation document found in “C:\Program Files\Microchip\MPLAB IDE\ICD2\Drivers\ddicd2.htm.” This will install the MPLAB® ICD 2 device drivers. Be sure to install the drivers for the MPLAB® ICD 2 before connecting the device to the PC via a USB cable.
- ❑ Connect the MPLAB® ICD 2 the PC to the target board, and power up the target board.
- ❑ Go to Debugger>Select Tool>MPLAB® ICD 2.



- ❑ The MPLAB® ICD 2 Setup Wizard will open or can be opened by going to Debugger>MPLAB® ICD 2 Setup Wizard. Click “Next” in the Welcome Page.
- ❑ Step One, select “USB.”
- ❑ Step Two, check “Target Has Own Power Supply.” Click “Next.”
- ❑ Step Three, select “MPLAB® IDE Automatically Connects to the MPLAB® ICD 2.” Click “Next.”
- ❑ Step Four, select “MPLAB® ICD 2 Automatically Downloads the Required Operating System.” Click “Next.”
- ❑ View Summary, click “Finish.”
- ❑ Go to Debugger>Connect. This makes sure that the MPLAB® ICD 2 is ready.
- ❑ Click Debugger>Program. This will download the program to the target board.
- ❑ Once the program is downloaded you can control the operation of the program with the buttons on the toolbar. You can just run the program or step through the program one line of code at a time. You can also step into function calls.
- ❑ Go to View>Watch.



- ❑ In the Watch Window use the drop-down box on the right to choose “a” and click “Add Symbol.” Do this for both “b” and “c.”
- ❑ Now step through the program using the “Step Over” button on the toolbar and watch the values change for “a”, “b”, and “c.”
- ❑ A breakpoint can be added to a line of code by double clicking on it.
- ❑ The “View Menu” has several other windows, which can be useful.



# REAL TIME EMULATION USING AN ICEPIC III

The ICEPIC III is an in-circuit emulator for PIC microcontrollers. The software for the ICEPIC III is an add-on to MPLAB, which implies that MPLAB is a requirement for using an ICEPIC III. The ICEPIC III connects to the host PC through a USB connection and to the PIC microcontroller board via a socket adapter. These connections can be seen below. The ICEPIC III supports unlimited breakpoints, total control of stepping through a program, full speed emulation, and low voltage emulation.

## Installation of ICEPIC III Add-on

- This add-on allows MPLAB 6.x/7.x IDE users to emulate their projects using an ICEPIC III. The add-on can be downloaded from: <http://www.rfsolutions.co.uk/acatalog/Downloads/IP3MP-v15212.exe>
- Install the MPLAB add-on in the same folder as the MPLAB installation. To install the drivers for the ICEPIC III, follow the instructions found here: <http://www.rfsolutions.co.uk/acatalog/Downloads/DriverInstallManual.htm>

## Getting Started

- Create a new project named ICEPICIII and a C source file named EX\_ICEPICIII.C as was done in the previous chapter. We can reuse the code from the previous chapter to test the functionality of the ICEPIC III by copying and pasting the code into EX\_ICEPICIII.C. Change the first line of code to `#include<18F452.h>` or any other PIC supported by the ICEPIC III. Be sure to select the target microcontroller by going to **CONFIGURE>SELECT DEVICE**, and choose the appropriate PIC microcontroller.
- Select ICEPIC III as the debugger by going to **DEBUGGER>SELECT TOOL>ICEPIC-III**.
- If everything is connected and installed correctly, the output window will display ICEPIC Ready, signifying that the ICEPIC III can now be used for emulation.

## Compiling and Debugging

- Compile the program and be sure there are no reported errors. The program can now be run and debugged in the same way as the previous program was. Open the watch window and step through the code to verify that the ICEPIC III is correctly emulating the selected PIC microcontroller. Also verify that the ICEPIC III functions correctly with breakpoints by adding a breakpoint to the code and running the program. The program should stop on the breakpoint so that further debugging can be performed.

# APPENDIX A: BUILD OPTIONS

## General

List File: Determines the format of the output list file from the compiler. There are three different formats:

Normal CCS Format:	Generate the list file in regular CCS format. This format displays PC address and Assembly instructions.
MPASM™ Assembler Format:	Generate the list file in MPASM™ Assembler list file formats. This format displays PC address, Opcode, line number, and Assembly instructions.
Symbolic Format:	This is the same as Normal CCS Format, but displays symbols instead of addresses in the list file. This format is more readable than the others.

Debug: Determines the debugging output file.

None:	Generates a regular COD file without any additional debugging information.
Expanded COD format:	Generates a COD file with expanded CCS debugging information. The expanded information is used inside the CCS PCW IDE Compiler, but not in MPLAB® IDE.
COFF format:	Generates a COFF file. This format is the preferable format to use in MPLAB® IDE.

Other Files: Specify which files that the compiler generates.

Call Tree:	Create call tree. This file contains a tree of the calls in the code.
Statistics File:	Create a file with statistics information about the memory usage and segments.
Symbol File:	Create a file with all the symbols and addresses. It also includes some compiler settings.
Show Warnings:	Force the compiler to generate warnings when compiling the source code.

Device Family: To select the device family, use *Configure->Select Device*.

Compile for use with ICD: Generates code that is ready to run with ICD debugger. This is similar to putting #device ICD=TRUE in the source code.

# APPENDIX A: BUILD OPTIONS

## Optimization

Set the optimization level for the compiler. A higher optimization level produces more efficient code.

## Extra Defines

This category allows defining additional global defines that are going to be used when compiling the code.

- |                           |  |
|---------------------------|--|
| Adding a Define:          | Put the define name in the ID field. Put the value in the Value field. Press the “+” button to add the define. |
| Modifying a Define:       | Select a definition from the list box. Change the “ID” field and/or the “Value” field, and click “Modify.”     |
| Removing a Define:        | Select a definition from the list box. Press the “-“ button to remove it.                                      |
| Removing all Definitions: | Click “Remove All” to remove all the definitions in the list box.  |