

Multisim™ MCU Module

User Guide

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

Worldwide Offices

Australia 1800 300 800, Austria 43 0 662 45 79 90 0, Belgium 32 0 2 757 00 20, Brazil 55 11 3262 3599, Canada 800 433 3488, China 86 21 6555 7838, Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 385 0 9 725 725 11, France 33 0 1 48 14 24 24, Germany 49 0 89 741 31 30, India 91 80 41190000, Israel 972 0 3 6393737, Italy 39 02 413091, Japan 81 3 5472 2970, Korea 82 02 3451 3400, Lebanon 961 0 1 33 28 28, Malaysia 1800 887710, Mexico 01 800 010 0793, Netherlands 31 0 348 433 466, New Zealand 0800 553 322, Norway 47 0 66 90 76 60, Poland 48 22 3390150, Portugal 351 210 311 210, Russia 7 495 783 68 51, Singapore 1800 226 5886, Slovenia 386 3 425 42 00, South Africa 27 0 11 805 8197, Spain 34 91 640 0085, Sweden 46 0 8 587 895 00, Switzerland 41 56 200 51 51, Taiwan 886 02 2377 2222, Thailand 662 278 6777, United Kingdom 44 0 1635 523545

For further support information, refer to Appendix B, “Technical Support and Professional Services”. To comment on National Instruments documentation, refer to the National Instruments Web site at ni.com/info and enter the info code feedback.

© 2007 National Instruments Corporation. All rights reserved.

Important Information

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THEREOF PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

National Instruments respects the intellectual property of others, and we ask our users to do the same. NI software is protected by copyright and other intellectual property laws. Where NI software may be used to reproduce software or other materials belonging to others, you may use NI software only to reproduce materials that you may reproduce in accordance with the terms of any applicable license or other legal restriction.

BSIM3 and BSIM4 are developed by the Device Research Group of the Department of Electrical Engineering and Computer Science, University of California, Berkeley and copyrighted by the University of California. The ASM51 cross assembler bundled with Multisim MCU is a copyrighted product of MetalLink Corp. (www.metaice.com). MPASM™ macro assembler and related documentation and literature is reproduced and distributed by Electronics Workbench under license from Microchip Technology Inc. All rights reserved by Microchip Technology Inc. MICROCHIP SOFTWARE OR FIRMWARE AND LITERATURE IS PROVIDED "AS IS," WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL MICROCHIP BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY ARISING OUT OF OR IN CONNECTION WITH THE SOFTWARE OR FIRMWARE OR THE USE OF OTHER DEALINGS IN THE SOFTWARE OR FIRMWARE.

Trademarks

National Instruments, NI, ni.com, and LabVIEW are trademarks of National Instruments Corporation. Refer to the *Terms of Use* section on ni.com/legal for more information about National Instruments trademarks. Ultiboard is a registered trademark and Multisim and Electronics Workbench are trademarks of Electronics Workbench. Other product and company names mentioned herein are trademarks or trade names of their respective companies.

Members of the National Instruments Alliance Partner Program are business entities independent from National Instruments and have no agency, partnership, or joint-venture relationship with National Instruments.

Patents

For patents covering National Instruments products, refer to ni.com/patents.

Some portions of this product are protected under United States Patent No. 6,560,572.

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Documentation Conventions

When this guide refers to a toolbar button, an image of the button appears in the left column.

This guide uses the convention **Menu/Item** to indicate menu commands. For example, “**File/Open**” means choose the **Open** command from the **File** menu.

This guide uses the convention of an arrow (➤) to indicate the start of procedural information.

This guide uses the construction CTRL-KEY and ALT-KEY to indicate when you need to hold down the “Ctrl” or “Alt” key on your keyboard and press another key.

The Multisim Documentation Set

Multisim documentation consists of a *User Guide*, the *Component Reference Guide* and online help. All Multisim users receive PDF versions of the *User Guide* and the *Component Reference Guide*.

The *Multisim MCU Module User Guide* describes the functions specific to Multisim’s MCU module.

You should also refer to *Getting Started with NI Circuit Design Suite*.

Online Help

Multisim offers a full helpfile system to support your use of the product.

Choose **Help/Multisim Help** to display the helpfile that explains the Multisim program in detail, or choose **Help/Component Reference** to display the helpfile that contains details on the components families provided with Multisim. Both are compiled HTML helpfiles, offering a table of contents and index.

In addition, you can display context-sensitive help by pressing F1 from any command or window, or by clicking the **Help** button on any dialog box that offers it.

Table of Contents

1. Introduction

- 1.1 Overview 1-1
- 1.2 Multisim MCU Basics 1-2
- 1.3 Debugging Tools 1-2
- 1.4 Multisim MCU Module Source Code Editor 1-3
- 1.5 MCU Memory View 1-3
- 1.6 Advanced Features 1-3
- 1.7 Peripheral Devices 1-3

2. Multisim MCU Module File Management

- 2.1 MCU Design Overview 2-1
- 2.2 MCU Wizard 2-3
- 2.3 Adding and Removing Projects, Folders and Files 2-4

3. Multisim MCU Module Code Manager

- 3.1 MCU Code Manager Overview 3-1
- 3.2 Adding and Removing Projects, Folders and Files in the MCU Code Manager 3-3
- 3.3 MCU Project Build Settings 3-3
 - 3.3.1 8051 Workspace Build Settings Example 3-5
 - 3.3.2 Loading an External Hex File Project 3-10
- 3.4 Building an MCU Workspace 3-11
 - 3.4.1 Errors and Warnings 3-12
 - 3.4.2 Simulation of Machine Code File 3-13

4. Multisim MCU Module Source Code Editor

4.1	Opening a Source Code File	4-1
4.2	Building Source Code Files	4-1
4.3	Source Code View	4-2

5. Multisim MCU Module Debugging Features

5.1	Definitions	5-1
5.2	Opening a Debug View	5-2
5.3	Debug Window Settings	5-3
5.4	Simulation Markers	5-4
5.5	Breakpoints	5-4
5.6	Stepping and Breaking	5-6
5.7	Memory View	5-7

Appendix A - Multisim MCU Module Parts

A.1	8051/8052 Microcontroller Units	A-1
A.2	PIC16F84/16F84A Microcontroller Units	A-3
A.3	RAM	A-4
A.4	ROM	A-5

Appendix B - Support and Services

B.1	Technical Support and Professional Services	B-1
-----	---	-----

Chapter 1

Introduction

The following are found in this chapter.

Subject	Page No.
Overview	1-1
Multisim MCU Basics	1-2
Debugging Tools	1-2
Multisim MCU Module Source Code Editor	1-3
MCU Memory View	1-3
Advanced Features	1-3
Peripheral Devices	1-3

1.1 Overview

The Multisim MCU module is an add-on to Multisim, the schematic capture and simulation application of National Instruments Circuit Design Suite.

Microcontroller (MCU) components are useful for many circuit designs. A modern microcontroller typically combines a CPU, data memory, program memory, and peripheral devices on a single physical chip. The integration of these essential elements of a computer into a single chip reduces component counts and board size resulting in higher reliability with more capabilities. The Multisim MCU module's co-simulation system provides software development features for writing and debugging code for embedded devices.

Embedded software development can be a challenging process for even the best programmers. Multisim MCU helps you produce high quality code more quickly and easily. The MCU development interfaces allow you to pause a simulation, inspect the internal memory and registers of the MCU, set code breakpoints and single step through your code.

Note Refer to *Getting Started with NI Circuit Design Suite* for a tutorial covering Multisim MCU's main functions.

1.2 Multisim MCU Basics

This and subsequent sections give a brief overview of the Multisim MCU module's functionality.

- To place an MCU:



1. Select **Place/Component** to display the **Select a Component** dialog box.
2. Navigate to the MCU Module **Group** and select the **Family** containing the desired MCU (e.g., 805x, PIC).
3. Select the desired MCU, click **OK** and click again to place the component on the workspace. The **MCU Wizard** dialog appears. The **MCU Wizard** helps you get started by creating an MCU Workspace, default project and source file.
4. Complete the steps in the wizard as detailed in “2.2 MCU Wizard” on page 2-3.
5. Use the **MCU Code Manager** to manage the MCU files in your MCU workspace and to set the build settings for your MCU project. See “3.1 MCU Code Manager Overview” on page 3-1 for details.

1.3 Debugging Tools

The Multisim MCU Module's debugging tools give you the ability to control execution at the instruction level while also providing views of the memory and registers within the MCU.

After an MCU's project has been built (without errors) using the **Build** command, the debug window becomes accessible.

- To open the debug window for a particular MCU, select **MCU/<MCU name, e.g., MCU 8051 UI>/Debug View**.

This opens a new window in Multisim with a drop-down list at the top used to select between the different views.

There is one view available for each source code file, as well as one for the disassembly of the entire project. By default, opening a **Debug View** opens the project disassembly view which is a complete disassembly of the ROM for the relevant MCU. In the project disassembly view, every memory address is shown from address zero to the maximum address of the ROM. The project disassembly view always shows the disassembly (not the listing assembly) because it shows memory addresses which may not have a corresponding listing (such as the opcodes from statically linked libraries).

For details on debugging, see Multisim MCU Module Debugging Features on page 5-1.

1.4 Multisim MCU Module Source Code Editor

The source code view shows the assembly or C source code for the MCU program.

The numbers on the far left are the program memory addresses and the hexadecimal codes to their immediate right are the assembled codes for each mnemonic assembly instruction. The column of numbers in the middle shows the line number in the original assembly or C source. The remainder of the line to the right shows the assembly source and comments.

1.5 MCU Memory View

The contents of the **MCU Memory View** change depending on the type of MCU. It may, for example, contain internal memory information, register views and configuration information.

For details on specific MCUs, see “A.1 8051/8052 Microcontroller Units” on page A-1 and “A.2 PIC16F84/16F84A Microcontroller Units” on page A-3.

1.6 Advanced Features

The Multisim MCU Module provides advanced debugging tools to make it easy to pause your circuit and explore the internal data and state of the MCU controlling your circuit. You can set breakpoints and single step through assembly code while validating that the register contents are changing as expected. You can also manually edit most memory views while debugging.

An example of the Multisim MCU Module’s advanced debugging features is found in *Getting Started with NI Circuit Design Suite*.

1.7 Peripheral Devices

Along with its selection of MCUs, Multisim MCU contains a number of peripheral devices.

The MCU Module **Group** contains RAM and ROM devices that are designed to function specifically with the MCUs. For details on these components, see “A.3 RAM” on page A-4 and “A.4 ROM” on page A-5.

The Advanced Peripherals **Group** contains a selection of Keypads, LCDs, Terminals and Miscellaneous Peripherals like the Liquid Holding Tank.

For details, refer to the *Multisim Component Reference Guide*, or the component helpfile.

Chapter 2

Multisim MCU Module File Management

This chapter describes the file management features used in the Multisim MCU module. The following are described in this chapter.

Subject	Page No.
MCU Design Overview	2-1
MCU Wizard	2-3
Adding and Removing Projects, Folders and Files	2-4

2.1 MCU Design Overview

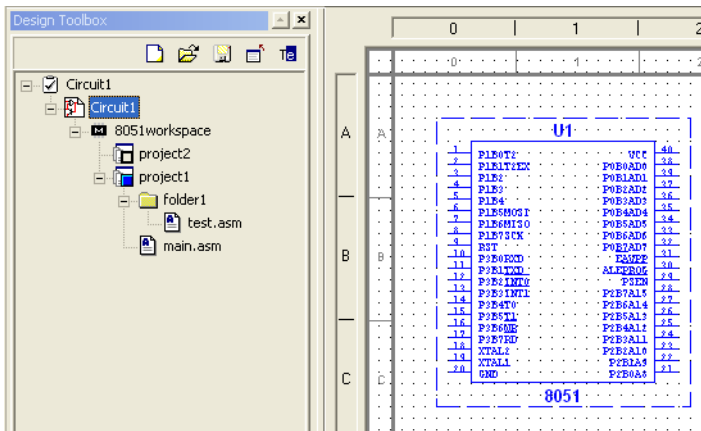
Every MCU component that is placed on the circuit schematic is associated with an MCU design consisting of a set of MCU files that exist on your hard disk separate from the Multisim circuit file (* .ms10):

MCU File Type	File Extension	Description
MCU workspace file	.mcuws	Maintains information on projects contained in the MCU workspace.
MCU project file	.mcuprj	Maintains information on files contained in the MCU project.
Assembly source file	.asm	Assembly code
Assembly include file	.inc	Assembly code
C source file	.c	C code
C header file	.h	C code

Each MCU workspace file is contained inside a workspace folder with the same name as the workspace file. All the other MCU-related files for this particular MCU component are stored inside the MCU workspace file in a hierarchical manner.

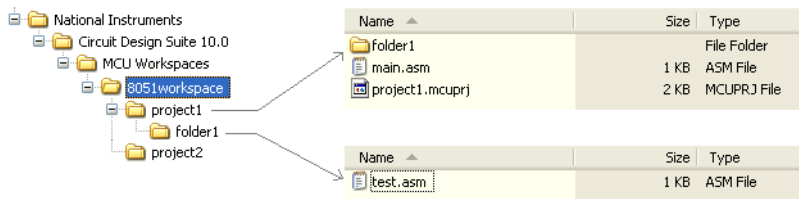
Each MCU project has its own project folder inside the workspace folder. Each MCU project file resides inside the project folder and the name of the project folder is the same as the project file. The assembly and C files are contained inside the project folder and also can be stored inside additional folders nested inside the project folder.

Below is an example of a circuit (`Circuit1`) in Multisim. It contains an 8051 MCU component (`U1`) that has an MCU design in the MCU workspace `8051workspace` as shown in the **Design Toolbox** view. The MCU workspace contains two projects, `project1` and `project2`. `Project1` has some MCU files, but `project2` is empty.



MCU component U1 with associated MCU workspace shown in the Design Toolbox in the left pane

The MCU files for the above circuit are organized on disk in the following file structure:



The arrows point to the contents contained in the `project1` and `folder1` folders.

Multisim stores the path to the MCU workspace file inside the circuit file (`*.ms10`). Every time `Circuit1.ms10` is opened, the MCU workspace must be found on disk in order for

Multisim to load the MCU design and display it in the **Design Toolbox** or perform any actions on the MCU workspace.

2.2 MCU Wizard

The **MCU Wizard** helps you get started by creating an MCU Workspace, default project and source file.

➤ To invoke the **MCU Wizard**:



1. Select **Place/Component** to display the **Select a Component** dialog box.
2. Navigate to the MCU Module **Group** and select the **Family** containing the desired MCU (e.g., 805x, PIC).
3. Select the desired MCU, click **OK** and click again to place the component on the workspace. Step 1 of the **MCU Wizard** dialog appears. This is where you specify the MCU workspace information.
4. Complete the following as desired:
 - **...workspace path for this MCU** — specifies where the workspace folder will be created. Or, click **Browse** and navigate to the desired location.
 - **...workspace name** — the name of the folder and the name of the MCU workspace file inside the workspace folder.
5. Click **Next** to display step 2 of the **MCU Wizard**. This is where you specify the project information.
6. Complete the following as desired:
 - **Project type** — choose one of: Standard - this project type contains source code files that needs to be built in order to create a machine code (intel hex) file to load into the MCU at the start of simulation; Load External Hex File - this project type contains no files and does not need a build step.
 - **Programming language** — active for standard **Project type** only. Select C or Assembly.
 - **Assembler/compiler tool** — active for standard **Project type** only.
 - **Project name** — the name of the project file and the project folder that will be created inside the MCU workspace folder.
7. Click **Next** to display step 3 of the **MCU Wizard**. This is where you specify the source file that you want to create for the project.
8. Complete the following as desired:
 - **Create empty project** — select if you do not wish to create a source file.
 - **Add source file** — enable if you wish to create a source file.
 - **Enter the name of the source file** — becomes active when **Add source file** is enabled. Can be an assembly or C file.

- Click **Finish** to close the **MCU Wizard** and create the MCU related files for the MCU component.

Note If you select **Cancel** at any time, no MCU workspace is created and the MCU component is removed from the circuit schematic.

2.3 Adding and Removing Projects, Folders and Files

After you place the MCU component and create the MCU workspace, project and file, you can add or remove any projects and/or MCU files to the MCU design.

- To manipulate items in the MCU design:
 - Right-click on an MCU item in the **Design Toolbox**. The contents of the pop-up menu that appears vary depending on the MCU item type that you clicked, as detailed below:

MCU Item Type	Right-click Context Menu Items
MCU workspace	Add MCU Project...
MCU project	Add New MCU Source File... Add Existing MCU Source File... Add MCU Source Folder... Remove MCU Project Set Active MCU Project
MCU source file	Remove MCU Source File From Project
MCU source folder	Add MCU Source File... Add Existing MCU Source File... Add MCU Source Folder... Remove MCU Source Folder

- Select the desired menu item

Add MCU Project — displays a dialog box where you specify the **Project Type** that you want to create (standard/load external hex file) and the **Project Name**. The new project is created in the MCU Workspace folder of the MCU workspace that you right-clicked in the **Design Toolbox**.

Add New MCU Source File — displays a dialog box where you select the **File type** (assembly file (.asm), assembly include file (.inc), C source file (.c), or C header file (.h)), and enter the **Filename**. The file is created in the MCU project folder or MCU source folder that you right-clicked in the **Design Toolbox**.

Add Existing MCU Source File — displays a file browser where you select an existing source file. The selected file is copied to the MCU project folder or source folder that was selected in the **Design Toolbox**.

Add MCU Source Folder — adds an MCU folder to the MCU project folder or source folder that you right-clicked in the **Design Toolbox**.

Remove MCU Project — removes the selected MCU project from the MCU workspace.

Remove MCU Source File From Project — removes the selected source file (*.asm, *.inc, *.c, *.h) from the MCU design.

Remove MCU Source Folder — removes the selected folder from the MCU design.

Set Active MCU Project — sets the selected MCU Project to be the active project. The active project in the **Design Toolbox** for an MCU workspace, has a blue colored square in the icon next to it. Any build operation applied to an MCU instance uses this project's files. The results of the build on the active project are also used for simulating the circuit.

Note Each file manipulation command that appears in the right-click context menus also appears in the **MCU Code Manager** (see “3.1 MCU Code Manager Overview” on page 3-1 for details).

Chapter 3

Multisim MCU Module Code Manager

This chapter describes the Multisim MCU module's **MCU Code Manager**. The following are described in this chapter.

Subject	Page No.
MCU Code Manager Overview	3-1
Adding and Removing Projects, Folders and Files in the MCU Code Manager	3-3
MCU Project Build Settings	3-3
8051 Workspace Build Settings Example	3-5
Loading an External Hex File Project	3-10
Building an MCU Workspace	3-11
Errors and Warnings	3-12
Simulation of Machine Code File	3-13

3.1 MCU Code Manager Overview

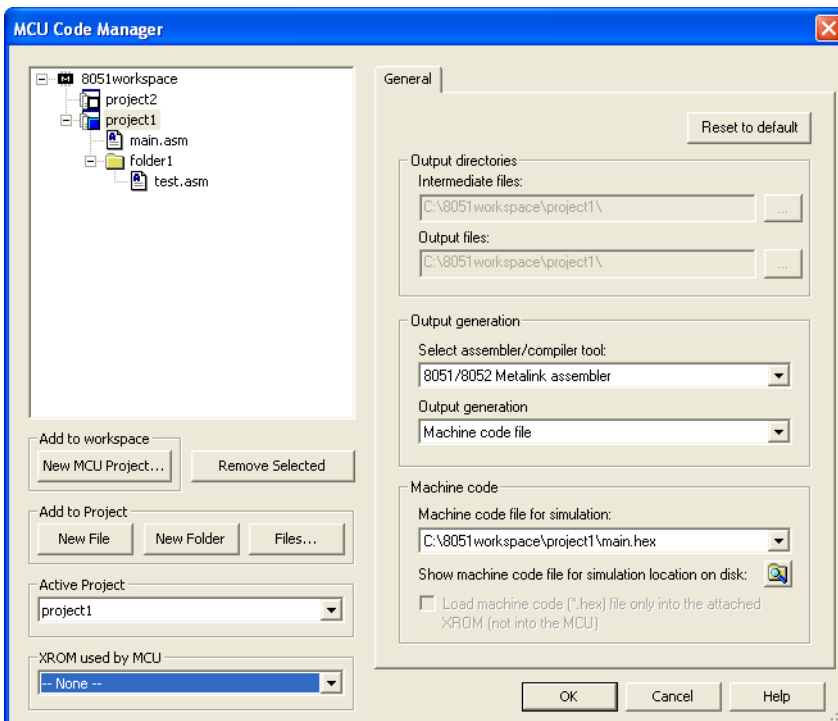
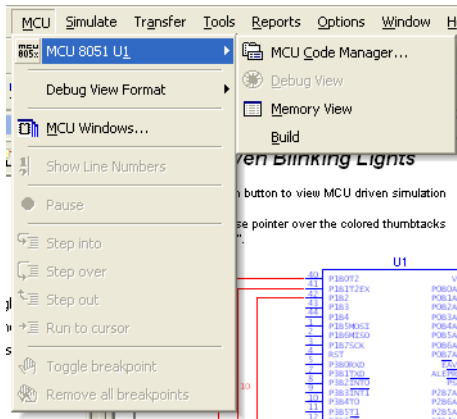
The **MCU Code Manager** dialog box lets you manage the MCU files in an MCU workspace and enter the build settings for each MCU project. Each MCU project has a number of build settings that consist of assembler/compiler tool information, the command line options to be used with the selected tool, and additional information on the intermediate/output files to generate during a build, and where the tools/files are located.

- To access the **MCU Code Manager** for a specific MCU:
 1. Select **MCU/<MCU name, e.g., MCU 8051 U1>/MCU Code Manager**.

Or

Right-click on the MCU workspace in the **Design Toolbox** and select **MCU Code Manager** from the pop-up that displays.

Note The **MCU** menu displays all instances of MCUs in a circuit design. In the example below, there is only one MCU “U1”.



The above shows the contents of the **MCU Code Manager** for MCU U1 from the previous screen capture. `Project1` is an 8051 Metalink assembler project that has only one tab (**General**) in its project build settings.

3.2 Adding and Removing Projects, Folders and Files in the MCU Code Manager

After you place the MCU component and create the MCU workspace, project and file, you can add or remove any projects or MCU files in the MCU design.

The buttons in the **MCU Code Manager** located below the tree display of the MCU workspace correspond to the file manipulation commands that display in the right-click context menus of MCU items in the **Design Toolbox**.

- To manipulate items in the MCU design from the **MCU Code Manager**:
 1. Click the **New MCU Project**, **New File**, **New Folder** or **Files** buttons to display the same dialog boxes as the right-click context menu commands **Add MCU Project**, **Add New MCU Source File**, **Add MCU Source Folder**, and **Add Existing MCU Source File**. For details, see “2.3 Adding and Removing Projects, Folders and Files” on page 2-4.
- To remove an item from the MCU workspace tree view:
 1. Highlight the desired item in the tree and click **Remove Selected**.
Note You cannot remove the MCU workspace.
- To select a different active project:
 1. Select another project from the **Active Project** drop-down list.
- To select the external ROM into which to load the MCU’s machine code:
 1. Select the desired ROM from the **XROM used by MCU** drop-down list.
Note There are no XROMs in the example in “3.1 MCU Code Manager Overview” on page 3-1, but if there was and an XROM part was selected, the **Load machine code (*.hex) file only...** checkbox would be enabled to let you could control how to load the machine code into the XROM for simulation.

3.3 MCU Project Build Settings

Each MCU project in an MCU design has its own build settings that can be different from other projects inside the same workspace. The build settings available are different for each assembler or compiler tool that is supported by Multisim. The tools supported by Multisim

are the 8051 Metalink Assembler, the Microchip PIC Assembler, the Hi-Tech 8051 C Lite compiler and the Hi-Tech PICC Lite compiler.

- To configure the build settings, use the tabs that appear on the right of the **MCU Code Manager** dialog box. The tabs that appear, and their content, correspond to the project that is currently selected in MCU workspace tree on the left.

The following table lists the assembler/compiler families of tools that are supported by Multisim and the **MCU Code Manager** tabs that are available for configuring those tools:

Assembler/compiler Family	MCU Code Manager tabs
Metalink - 8051/8052 Cross Assembler	General
Microchip (PIC) - MPASM absolute and relocatable assembler - MPLINK linker - MPLIB librarian	General C/Assembly Link Library
Hi-Tech - 8051 C Lite compiler - HLINK linker - LIBR librarian	General C/Assembly Library
Hi-Tech - PICC Lite compiler - LINK linker - LIBR librarian	General C/Assembly Library

Documentation for Supported Tools

User documentation for the supported tools is found at the following:

- **Metalink assembler** — ... \Program Files\National Instruments\Circuit Design Suite 10.0\documentation\MetaLink 8051 Cross Assembler Guide.pdf.
- **Microchip MPASM assembler** — ... \Program Files\National Instruments\Circuit Design Suite 10.0\documentation\DS-33014J.pdf.
- **HI-TECH 8051 C compiler** — ... \Program Files\HI-TECH Software\HC51\9.60\docs>manual.pdf.
- **HI-TECH PICC Lite compiler** — go to www.htsoft.com and download HI-TECH PICC (V9.50PL2).

3.3.1 8051 Workspace Build Settings Example

In the example shown in “3.1 MCU Code Manager Overview” on page 3-1, `project1` is selected. The Metalink assembler is selected for the project. Since this is an absolute assembler, the build settings for it are very simple. Consequently, only the **General** tab is available to configure the build settings. The **Intermediate files** and **Output files** paths are disabled because the Metalink assembler does not allow you to redirect your output.

After the output machine code file (*.hex) is generated, the contents must be loaded into the Multisim simulation engine in order to simulate the circuit.

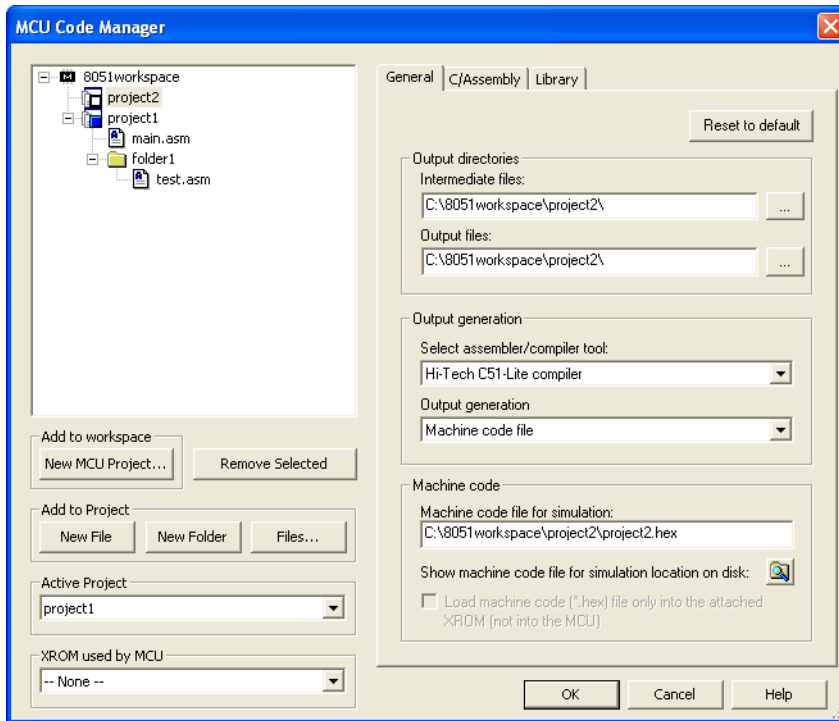
- To set the output *.hex file generated during a build step, select the file from the **Machine code file for simulation** drop-down list.
- To see the whole path name of output files:



1. Click **Show machine code for simulation location on disk**. Windows Explorer opens in the folder where the output file will be located.

In the screen capture below, `project2` of `8051workspace` is selected in the **MCU Code Manager**. `Project2` has the “Hi-tech C51-Lite compiler” selected as the tool. This is a C compiler which compiles and links code in order to produce an output machine code file

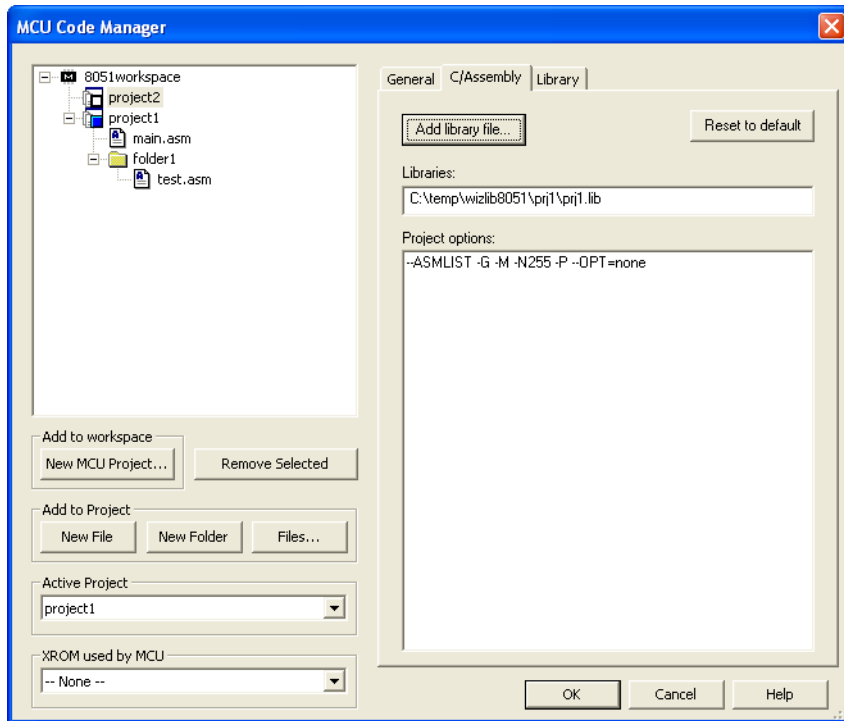
(* .hex), and is more advanced than the Metalink assembler. It has three tabs (**General**, **C/Assembly**, and **Library**) to configure its build settings.



The assembler or compiler type determines the type of output target files that you can generate. In the screen capture above, the Hi-Tech C51 Lite compiler is chosen. Consequently, there are two types of target files that you can create: a machine code file and a library file. If you choose a library file, the compiler generates object files from the source files and places them in the specified library file instead of creating a machine code file. The library file is configured in the **Library** tab.

There is a **C/Assembly** tab (see below) in addition to the **Library** tab available to allow you to configure the Hi-Tech tools further. One of the advanced features of the Hi-Tech tools is that

intermediate and output files generated by the Hi-Tech C51 Lite can be redirected - therefore the **Intermediate files** and **Output files** paths are enabled in the **General** tab.



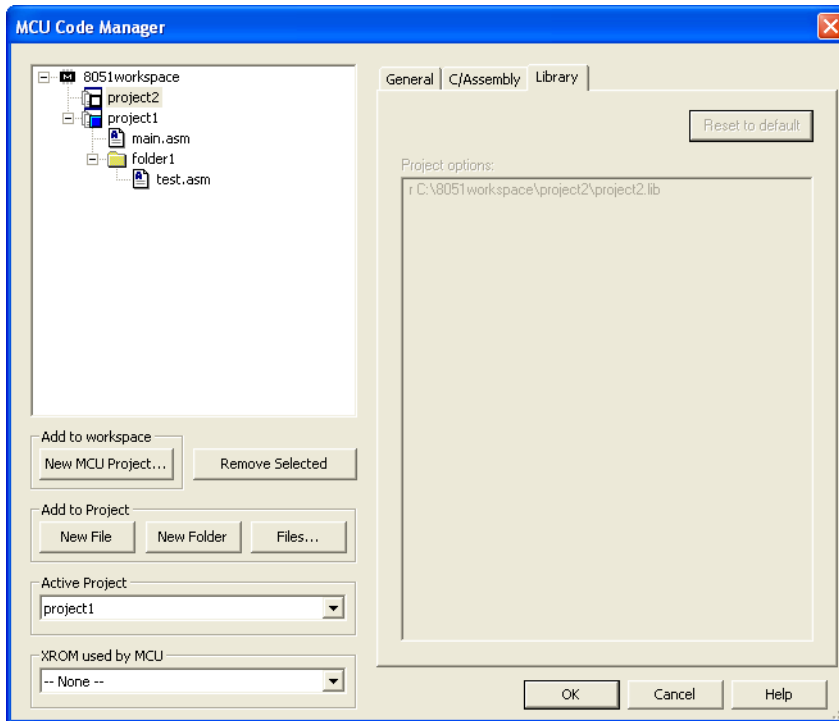
The above shows the **MCU Code Manager** for MCU U1 in “3.1 MCU Code Manager Overview” on page 3-1. The **C/Assembly** tab of the Hi-Tech C51 Lite compiler for `project2` is displayed.

The **Project options** edit box contains the compiler line options for the C51 Lite compiler. The default settings are normally sufficient to build an 8051 Hi-Tech C program but you can customize the options based on the Hi-Tech C51 Lite documentation (see “Documentation for Supported Tools” on page 3-4).

The **Libraries** edit box is where you can specify pre-existing libraries to be linked into the final build. In the above screen capture, one library file will be linked into the final build. Use the **Add library file** button to display a file browse window to find the library that should be linked into the build and automatically add the library file path to the **Libraries** edit box.

The **Library** tab, as shown below, is enabled only if the output target selected in the **General** tab is a library. The **Project options** edit box in the **Library** tab displays the command line options used when the Hi-Tech tools generate the library. The screen capture below shows that the **Library** options are disabled since the output selected for `project2` is a hex file.

Refer to the Hi-Tech compiler documentation for more details on its command line options (see “Documentation for Supported Tools” on page 3-4).



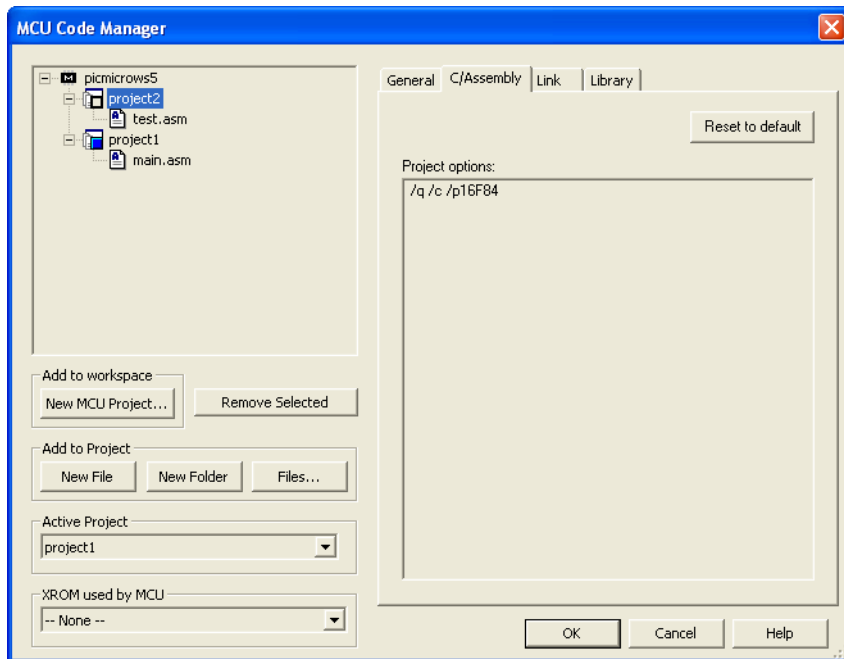
The above screen capture shows the **MCU Code Manager** for MCU U1 in “3.1 MCU Code Manager Overview” on page 3-1. The **Library** tab of Hi-Tech C51 Lite compiler for `project2` is displayed.

The tabs for the other assembler/compiler files are all similar except that the command line options in the **C/Assembly** and **Library** tabs are different for each tool. Refer to the assembler/compiler documentation for each tool for more information on the options that can be used with them (see “Documentation for Supported Tools” on page 3-4).

The tab that was not shown in the above example is the **Link** tab. The Hi-Tech compilers have the ability to pass on the command line options to their linker tool from the compiler. This means that the command line options for both the compiler and the linker can be specified all at once in the command line options for the compiler, which is when no **Link** tab is required.

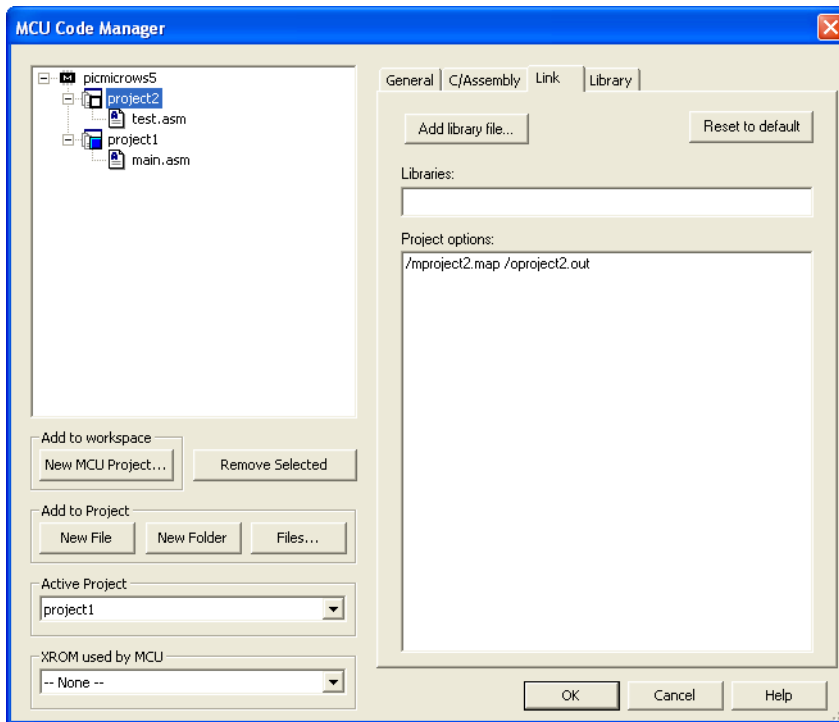
The Microchip relocatable assembler requires the **Link** tab to configure the command line options for MPLINK, its linker. The **C/Assembly** tab for the Microchip tools displays the

options for the compiler, MPASM. The figure below shows an example of the **C/Assembly** tab for the Microchip assembler.



Use the **Link** tab for the Microchip assembler to add any libraries that need to be linked into the final build and the linker options that have to be passed to MPLINK. The next screen capture shows an example of the **Link** tab.

Note The Microchip assembler has both absolute and relocatable assembler. To use MPASM in relocatable assembler mode, you must add the `/o` option to the **Project options** edit box in the **C/Assembly** tab shown above, to generate the object files (`*.o`) used in the linking stage. Otherwise the object files do not generate and only the machine code file (`*.hex`) will be generated as the output file. You must also write your code in a relocatable fashion using the `CODE` directive before the code can be assembled in relocatable mode.

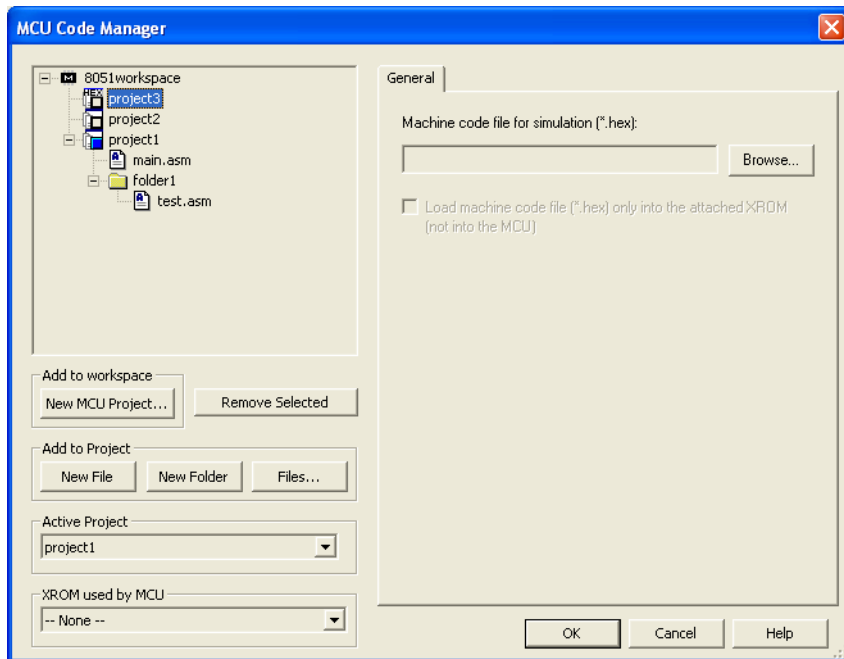


The **Link** tab for the Microchip MPLINK tool displays command line options in the **Project options** edit box and provides a way of specifying the libraries that should be linked into the final build.

3.3.2 Loading an External Hex File Project

The build settings described in the previous section are for standard projects that have an assembler/compiler tool associated with them. The second type of project, (load external file project) requires no tools. Only the machine code file path must be configured. The screenshot below shows the **General** tab for a load external hex file project in the **MCU Code Manager**. During simulation, the **Machine code file** specified in the edit box containing the

opcodes is loaded into the simulation and run. It is also possible to load the code into an XROM part associated with the MCU as is done for standard projects.



3.4 Building an MCU Workspace

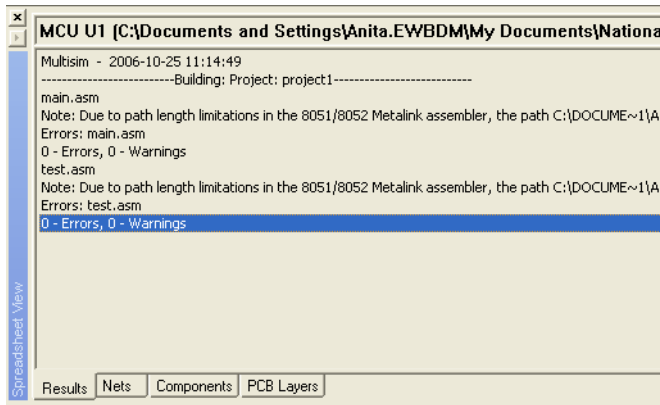
To build an MCU workspace, the proper build settings must be configured in the **MCU Code Manager** as described in the previous sections and all necessary source files must be present in the active MCU project. The assembler/compiler tools selected in the **General** tab of the **MCU Code Manager** for the active project, are invoked during the build to generate the appropriate files. These files can be machine code files (*.hex), object files, libraries, listing files and any other intermediate or output files described in the tool's documentation (see "Documentation for Supported Tools" on page 3-4).

- To build the MCU Workspace:
 1. From the MCU menu, navigate to the submenu of the MCU that you wish to build and select **Build**.

Or

Right-click on the workspace or active project item in the **Design Toolbox** to display the context menu and select **Build**.

The results of the build are displayed in the **Results** tab of the **Spreadsheet View**. An example is shown below. If the build is successful, you will have either a *.hex or *.lib file in the output path where your output files should go. You can verify this by using Windows Explorer to view the files in that subdirectory.

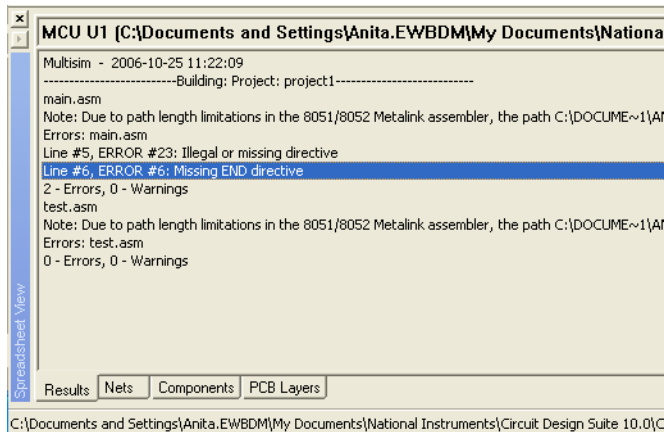


3.4.1 Errors and Warnings

If there are errors or warnings in your build, they will be displayed in the **Results** tab.

- To find an error:
 1. Double-click on the line displaying the error in the **Results** tab. The file that contains the error comes into focus with the cursor placed on the line corresponding to the error.
 2. In the example below, there are two errors generated by the Metalink assembler listed: line #5 and line #6. Double-click on either of these lines to open `main.asm` and place the cursor on lines 5 and 6 respectively.

- Correct the errors in the source file and then build the MCU workspace again, continuing in the same manner until the build is error-free.



3.4.2 Simulation of Machine Code File

Standard Projects

After building the MCU workspace successfully, simulate the circuit containing the MCU component by selecting **Simulate/Run**. If you try to run the simulation without building the MCU workspace first, Multisim prompts you to build the MCU.

When you build the MCU workspace, listing files (*.lst) are generated that are loaded into Multisim for debugging purposes. Depending on the assembler/compiler tool, the listing files contain a combination of data such as opcodes, where the opcodes should reside in the MCU internal ROM, error information, assembly instructions corresponding to the opcodes and the matching line of source code for each assembly instruction. If the listing file is missing, or cannot be loaded into Multisim, then no instruction level debugging can be performed on the MCU.

The intel hex format machine code file (*.hex) that is generated by the build is also loaded into Multisim at the start of simulation so that the MCU component's internal ROM representation can be loaded with the opcodes that should be executed during simulation. This machine code file path is specified in the **General** tab of the **MCU Code Manager**. You can check that the correct machine code file exists in the machine code file path if Multisim has trouble loading it.

Load external hex file projects

Standard MCU projects generate the *.hex file during the build step. However the *.hex file does not have to be generated for load external hex file projects. The machine code file specified in the **General** tab of the **MCU Code Manager** is loaded at the start of simulation. It is still possible to debug through disassembled assembly instructions for the MCU during simulation in the **Debug view** since the opcodes are disassembled when the *.hex file is loaded.

Simulating MCU with XROM

The **General** tab of the **MCU Code Manager** allows an XROM (external ROM) component to be used with the MCU. There are three scenarios that can happen relating to XROM:

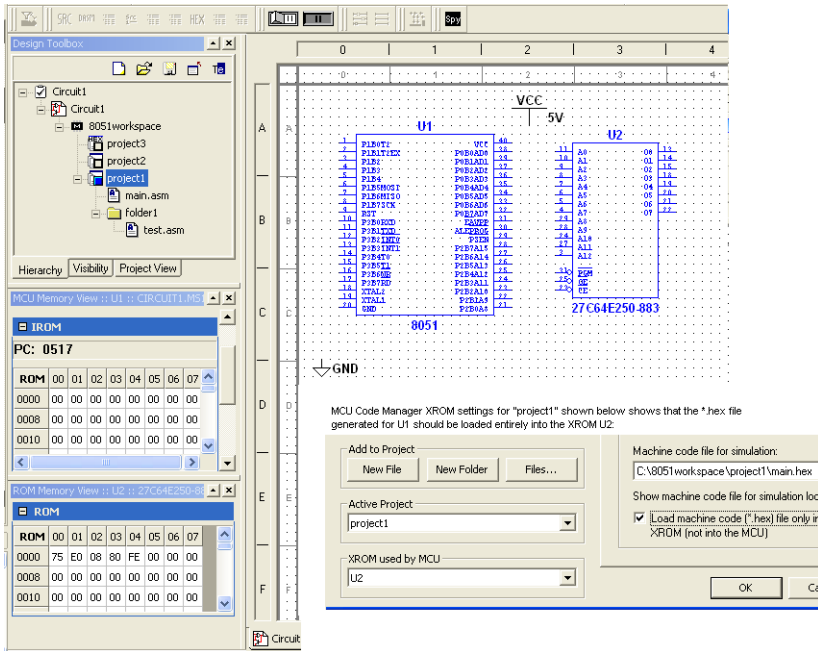
- If no XROM is selected, the machine code (*.hex) file opcodes are loaded entirely into the MCU as described in all of the other examples above for standard and load external hex file projects.
- If an XROM is selected and the **Load machine code (*.hex) file into XROM** checkbox is selected, the machine code file generated by the MCU build is loaded entirely into the XROM. None of the opcodes are loaded into the MCU; the internal ROM of the MCU will be empty (filled with no-ops).
- If an XROM is selected and the **Load machine code into XROM** checkbox is not selected, the machine code file generated by the MCU build is loaded into the MCU's internal ROM, but any opcodes that should be loaded into memory address extending beyond the physical size of the internal ROM are loaded into the XROM at those very high memory addresses.

The example below shows a circuit containing an MCU (U1) and an XROM (U2) where the opcodes in the *.hex file generated during the build step for a standard project `project1` is loaded entirely into the XROM (U2). The simulation is paused and the **XROM Memory View** shows that the opcodes are indeed loaded into U2 and that there is nothing in U1 as shown in the **MCU Memory View**. The bottom right side is a snippet from the **MCU Code Manager** settings for `project1` that shows the XROM settings. The *.hex file is generated using the 8051 Metalink assembler from the `main.asm` source code:

```
$MOD51      ;This includes 8051 definitions for the metalink assembler

MOV ACC,#08H
LOOP:
  JMP LOOP

END
```



The circuit containing MCU U1 and XROM U2 is configured so that during simulation, the opcodes generated from the source code for project1 are loaded entirely into the XROM as shown in the XROM Memory View.

Chapter 4

Multisim MCU Module Source Code Editor

This chapter describes Multisim MCU's Source Code Editor. The following are described in this chapter.

Subject	Page No.
Opening a Source Code File	4-1
Building Source Code Files	4-1
Source Code View	4-2

4.1 Opening a Source Code File

- To open a source code file in the editor:
 1. Double-click on the name of the source code file (for example, `main.c` or `main.asm`) in the **Design Toolbox**.
 2. Optionally, select **MCU/Show Line Numbers** if you wish to display line numbers in the source code view.

4.2 Building Source Code Files

Source code files must be built to create a machine code (intel hex) file to load into the MCU at the start of simulation. Also, any changes in source code files will not take effect until the MCU project is built.

➤ To build source code files:

1. Select **MCU/<MCU name, e.g., MCU 8051 U1>/Build**.

Or

Right-click on the desired source code file in the **Design Toolbox** and select **Build**.

The **Build** command invokes the appropriate compilers, assemblers and linkers, and then loads the error, listing and hex files that they generate into Multisim. Errors, warnings and messages display in the **Results** tab of the **Spreadsheet View**.

If the build is successful, the hex file is loaded into the appropriate MCU or external ROM. If it is unsuccessful, the ROM is filled with zeros.

➤ To go to the line of source code associated with an error:

1. Double-click on the error in the **Results** tab.

4.3 Source Code View

Sections of the Multisim MCU Module's Source Code Editor are color-coded to denote the text type:

- Blue text — key words
- Green text — notes
- Black text — all other text, including data, errors.

Note If you type text with an error (spelling, incorrect syntax) into a key word section of the code, the text changes color from blue to black.

```

CMD_GRAREA    EQU 43H    ;SET GR AREA
CMD_OFFSET    EQU 22H    ;SET OFFSET ADD
CMD_ADPSET    EQU 24H    ;SET ADD PTR
CMD_SETDATA_INC EQU 0COH  ;WRITE DATA AND INCREASE ADP
CMD_AWRON     EQU 0BOH   ;SET AUTO WRITE MODE
CMD_AWROFF    EQU 0B2H   ;RESET AUTO WRITE MODE

GOTO START

ORG    0x10

; DATA
DATA_NUM EQU 23H
TXPRT    ; Text data "Grapical LCD T6963C for Multisim"
ADDWF   PCL, 1
RETLW   0x27
RETLW   0x52
RETLW   0x41
RETLW   0x50
RETLW   0x48
RETLW   0x49
RETLW   0x43
RETLW   0x41
RETLW   0x4c
RETLW   0x00
RETLW   0x2C
RETL    0x23
RETLW   0x24

```

- To save the source code file, select **File/Save** from the source code view.
- To print the source code file, select **File/Print** from the source code view.
- To find a text string in the source code:
 1. Select **Edit/Find** to display the **Find** dialog box.
 2. Enter the desired string and click **Find Next**.

Chapter 5

Multisim MCU Module Debugging Features

This chapter describes Multisim MCU's Debugging Features.

We also recommend that you read the Multisim MCU Module Tutorial chapter of *Getting Started with NI Circuit Design Suite*.

The following are described in this chapter.

Subject	Page No.
Definitions	5-1
Opening a Debug View	5-2
Debug Window Settings	5-3
Simulation Markers	5-4
Breakpoints	5-4
Stepping and Breaking	5-6
Memory View	5-7

5.1 Definitions

After a MCU project has been compiled/assembled into a hex file and loaded by the Multisim MCU Module several different pieces of information are available:

- **source code** — the original text that was entered by you and passed to the compilers.
- **listing assembly** — the assembly text that is output by the compiler into the listing files. This form assembly has variable names and other extra notations from the compiler which is useful when debugging or analyzing the output of a C program. If the original source code was assembly the listing assembly is often identical; only when macros are used does it differ.

- **disassembly** — created by the Multisim MCU Module from the raw hex opcodes. This is often similar to the listing assembly, however, it usually contains absolute values instead of variable names, except for the jump/goto instructions which use the named labels.

The Multisim MCU Module includes code and debugging windows in addition to Multisim's standard circuit windows. Inside the debugging windows a dropdown list allows you to pick different debugging views. The views show different subsets of the same information, and the debug window settings let you further filter and customize what the information that is presented in these views.

5.2 Opening a Debug View

After an MCU's project has been built (without errors) using the **Build** command the debug window becomes accessible.

- To open the debug window for a particular MCU, select **MCU/<MCU name, e.g., MCU 8051 U1>/Debug View**.

This opens a new window in Multisim with a drop-down list at the top that you use to select between the different views.

There is one view available for each source code file, as well as one for the disassembly of the entire project. By default, when you open a **Debug View** the project disassembly view which is a complete disassembly of the ROM for the relevant MCU opens. In the project disassembly view, every memory address displays from address zero to the maximum address of the ROM. The project disassembly view always shows the disassembly (not the listing assembly) because it shows memory addresses which may not have a corresponding listing (such as the opcodes from statically linked libraries).

- To switch to another of the available views, left-click on the drop-down list.

The source file debug listing view is similar to the listing text that a compiler generates when it builds a single file. It contains a subset of the information available in the complete ROM disassembly, but the information is sorted into the order of the original source code, rather than the memory address.

Note The debug file view can also be launched directly — right-click the appropriate file in the **Design Toolbox** and select **Debug View**.

5.3 Debug Window Settings

➤ To select the information to view in the **Debug view**:

1. Select **MCU/Debug View Format** and select the desired option.

Or







Right-click on the **Debug view** and enable the desired option from the **Debug View Format** menu item.




Or

Enable the desired button in the **MCU toolbar**.

These settings are saved as part of each MCU component, thus if you have two components they can have different settings, however all the views inside the debugging window will have the same settings.

MCU toolbar buttons:

Button	Description
	Source Code button. Enable to debug through the original source code or the assembly from the compilers listing files/multisim disassembler. Applies to the file debugging views, not to the project disassembly.
	Disassembly button. Determines whether you see the listing assembly text or the version from the disassembler. Applies to the file debugging views, not to the project disassembly.
	Show Secondary Language as Comments button. Enables/disables one of two things. If you are looking at the project disassembly view or if you are looking at the listing assembly or disassembly of a file it will add the original source code corresponding to those lines as comments above or beside the assembly. If you are viewing the source code debug listing then this will show the corresponding listing assembly/disassembly as comments below the source code.
	Show Line Numbers button. Displays the line numbers in the Debug View .
	Show Memory Addresses in Debug View button. Enables/disables the showing of memory addresses for any text that shows source code in any of the debug listing views.
	Show Memory Addresses in Assembly Code button. Enables/disables the showing of memory addresses for any text that shows listing assembly or disassembly in any of the debug listing views.

Button	Description
	<p>Show Hex Opcodes in Assembly Code button. Enables/disables the showing of the hex values of opcodes for any text that shows listing assembly or disassembly in any of the debug listing views.</p>
	<p>Show Jump/Goto Labels button. Enables/disables the showing of the jump/goto labels for any text that shows listing assembly or disassembly in any of the debug listing views.</p>
	<p>Show Headings Above Code button. Enables/disables the showing of the heading comments above each block of source or assembly code which labels the different fields being shown in the text.</p>

5.4 Simulation Markers

The margin that runs down the left side of the MCU debug windows and **Source Code View** is where breakpoints and simulation markers are shown. When you pause the simulation, a yellow arrow in the bar on the left side of the source code indicates the current line of execution which corresponds to the current value of the program counter (PC) in that MCU. Since each line of source code may actually refer to many opcodes, the current program counter may actually be located somewhere in the middle of the opcodes for a line when you pause the simulation.

The simulation marker also appears in the debug view, which is where most of the debugging is done. If your debug view is currently set to show source code, it behaves the same way the source code window does. If you are viewing the disassembly or the listing assembly, the indicator is located on the line that contains the memory address that corresponds to the program counter. Since the debug view corresponds to a single MCU it has a maximum of one simulation marker. However, the source code view may have more than one indicator if there are two MCU instances which share that MCU project (this can only happen if you have MCU components inside subcircuits or hierarchical blocks).

5.5 Breakpoints

You can place breakpoints in either source code or debugging views. Breakpoints can only be added to lines that correspond to real memory addresses, therefore they cannot be placed on comments or other lines that do not map to any real opcodes. Before breakpoints can be placed the project must be built (see “4.2 Building Source Code Files” on page 4-1).

- To place a breakpoint:
 1. Double-click on the margin at the left edge of the source code or debugging window. Breakpoints are displayed as a red dot in the left margin beside any line of source code or assembly.

Or

Select the line of source code or debug listing text that the breakpoint should be placed on and choose the **Toggle Breakpoint** button in the **Simulation** toolbar

Or

Right-click on the line of source code or debugging text and choose **Toggle Breakpoint** from the context menu.
- To remove a breakpoint, repeat any of the above.

A breakpoint placed on a source code window will also be visible in the debug view. These are simply two different ways of looking at the same information. Breakpoints that are placed on listing assembly or disassembly views that are associated with a line of source code, even if they are in the 'middle' of the opcodes for that source code, show up in any source code views. Breakpoints placed on source code correspond to the first memory address associated with that line of source code in the disassembly or listing assembly.

After each build the breakpoints may be adjusted based on the changes to the source code and may shift slightly in both views. This happens because breakpoints placed in the source code view will remain 'locked' to the line of source code on which they were placed even if changes result in the compiler moving the memory address of the opcodes that correspond to the source code. Breakpoints that are placed on the listing assembly or the disassembly remain 'locked' to the exact memory address, so if the source code is modified and then built they may shift their position in the source code view to the source code that now corresponds to that memory address.

If MCU components are placed in hierarchical blocks or subcircuits it becomes possible to have two different 'instances' of the same component in your circuit. In this case both MCU components share the same MCU project and source code files, and therefore a breakpoint will be placed in each of MCU components. Toggling the breakpoint on or off in the source code toggles them all on or off, however toggling them on or off in the debug view only removes the breakpoint for the corresponding component. The breakpoint marker in the source code view only disappears when all the breakpoints in all the MCU's debug windows are removed.

5.6 Stepping and Breaking

The controls for simulation stepping and breaking into an MCU are located in the **Simulation** toolbar, and also in the **MCU** menu.

When you select the **Pause Simulation** toolbar button menu item during a simulation, the simulator stops at the next SPICE timestep, possibly leaving the CPU simulator in mid-instruction. If you press the **Pause Simulation at Next MCU Instruction Boundary** button, the simulation waits slightly longer before breaking. If the currently activated window and view is displaying listing assembly or disassembly then the simulation break does not cause the simulation to stop until it reaches the beginning of a new MCU opcode. If the current window and view is displaying source code then the simulator continues until it reaches the start of an opcode which is associated with the start of the next line of source code.

The stepping functions are **Step into**, **Step over**, **Step out** and **Run to cursor**. All of these buttons work by starting a simulation and then stopping when a certain point is reached. The behavior depends on whether the view that is active is showing source code (as the primary language) or listing assembly/disassembly.

If you click the **Step into** toolbar button, the simulator starts and runs until it reaches the next line of source code or listing assembly/disassembly. If the current opcode is the start of a function call, the **Step into** command follows the call to its new location and breaks at the next opcode.

If you click the **Step over** button, the simulator starts and runs until it reaches the next line of source code or listing assembly/disassembly. If the current opcode is the start of a function call then **Step over** does not follow the call, but rather breaks at the first opcode after the functions return. Another way of thinking of this is that the simulator breaks at the start of the first opcode where the call stack depth is less than or equal to the current value.

If you click the **Step out** button, the simulator starts and runs until it reaches the line of source code or listing assembly/disassembly that immediately follows the return from the current function. Another way of thinking of this is that the simulator breaks at the start of the first opcode where the call stack depth is less than the current value.

For a demonstration of these functions, refer to the Multisim MCU Module Tutorial chapter of *Getting Started with NI Circuit Design Suite*.

5.7 Memory View

- To display an MCU's memory view, select **MCU/<MCU name, e.g., MCU 8051 U1>/Memory View**. A dockable window opens which contains several spreadsheet-like grids of data.

These grids contain data only while the simulation is running and in a paused state. When paused, they represent a snapshot of the internal registers and memory banks for that component at a point in simulation time. Different MCU's have different subsets of these grids depending on the features of that microchip. While the simulation is paused these values can be used to examine the internal state of the component and also to manually modify them.

- To modify the values, click on the corresponding grid entry and type in new values in the same format as the previous values.

Note For details on the memory view, refer to “A.1 8051/8052 Microcontroller Units” on page A-1 and “A.2 PIC16F84/16F84A Microcontroller Units” on page A-3

External RAM and ROM components also have memory views. When the simulation is paused, the you can view the contents of the memory for these parts. See “A.3 RAM” on page A-4 and “A.4 ROM” on page A-5 for details.

Appendix A - Multisim MCU Module Parts

This appendix contains information on components that are specific to the Multisim MCU module.

A.1 8051/8052 Microcontroller Units

40	P1B0T2	VCC	38	40	P1B0T2	VCC	38
41	P1B1T2EX	P0B0AD0	32	41	P1B1T2EX	P0B0AD0	32
42	P1B2	P0B1AD1	36	42	P1B2	P0B1AD1	36
43	P1B3	P0B2AD2	35	43	P1B3	P0B2AD2	35
44	P1B4	P0B3AD3	34	44	P1B4	P0B3AD3	34
1	P1B5M0SI	P0B4AD4	33	1	P1B5M0SI	P0B4AD4	33
2	P1B6M1SO	P0B5AD5	37	2	P1B6M1SO	P0B5AD5	37
3	P1B7SCK	P0B6AD6	31	3	P1B7SCK	P0B6AD6	31
4	RST	P0B7AD7	30	4	RST	P0B7AD7	30
5	P3B0R<D	EAVPP	22	5	P3B0R<D	EAVPP	22
7	P3B1T<D	ALFRFG	27	7	P3B1T<D	ALFRFG	27
8	P3B2INT0	PSEN	26	8	P3B2INT0	PSEN	26
9	P3B3INT1	P2B7A15	25	9	P3B3INT1	P2B7A15	25
10	P3B4T0	P2B6A14	24	10	P3B4T0	P2B6A14	24
11	P3B5T1	P2B5A13	23	11	P3B5T1	P2B5A13	23
12	P3B6WR	P2B4A12	22	12	P3B6WR	P2B4A12	22
13	P3B7RD	P2B3A11	21	13	P3B7RD	P2B3A11	21
14	XTAL2	P2B2A10	20	14	XTAL2	P2B2A10	20
15	XTAL1	P2B1A9	19	15	XTAL1	P2B1A9	19
16	GND	P2B0A8	18	16	GND	P2B0A8	18

The 8051 and 8052 microcontrollers combine a CPU, data memory, program memory, and built-in external memory on a single chip.

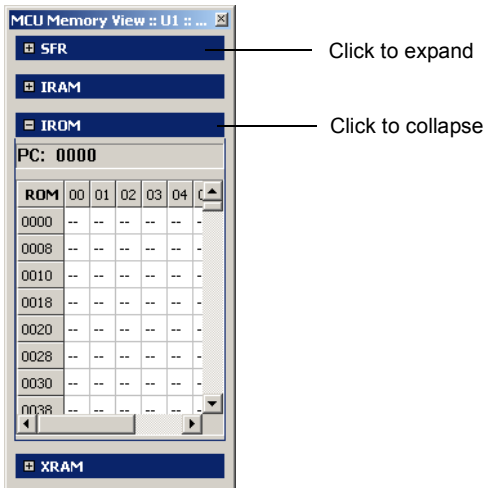
- To show/hide the elements of the 805x:
 1. Select MCU/<MCU name, e.g., MCU 8051 U1>/Debug View to display the MCU's debug view. (You must build the file first).
 2. Select MCU/<MCU name, e.g., MCU 8051 U1>/Memory View to display the MCU's memory view.
- To change the values for the placed 805x:
 1. Double-click on the placed MCU to display its properties dialog box and click on the Value tab.
 2. Change the values as desired:
 - **Built-in External RAM** — the external on-chip RAM for the MCU, as displayed in the XRAM section of the MCU Memory View.
 - **ROM Size** — the ROM for the MCU, as displayed in the IROM section of the MCU

Memory View.

- **Clock Speed** — the speed of the MCU's internal clock.
3. Click **OK** to close the dialog and accept the changes.

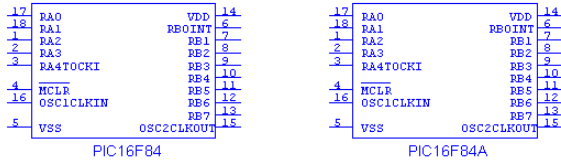
MCU Memory View

You can collapse/expand the fields in the **MCU Memory View** as shown below.



- **SFR** is the MCU's Special Function Registers.
- **IRAM** is the internal RAM (Random Access Memory) of the MCU. Shows the data that is inside the MCU's memory and is modified as the program runs. The green shaded area shows the four "R" register banks; the currently selected one is brighter than the other three.
- **IROM** is the internal ROM (Read Only Memory) of the MCU. Shows the program memory code in hexadecimal format. These are the actual machine instructions that the simulation uses when it is activated. The left column shows the memory address and the header row shows the offset from the address on the left.
- **XRAM** is the MCU's external on-chip RAM.

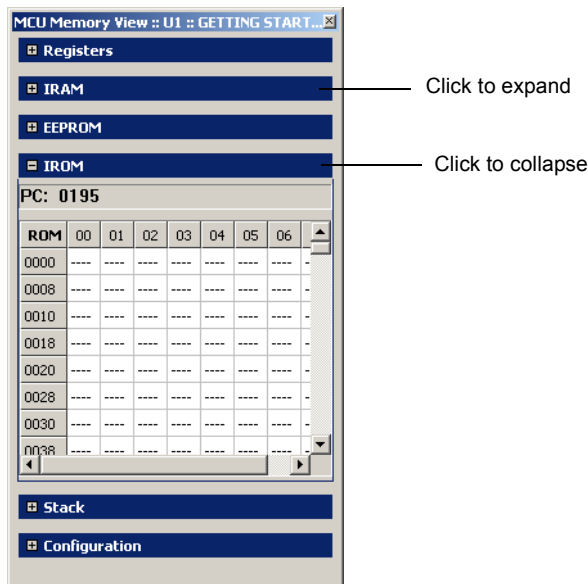
A.2 PIC16F84/16F84A Microcontroller Units



- To show/hide the **MCU Memory View**:
 1. Select **MCU/<MCU name, e.g., MCU PIC16F84A U1>/Memory View**.
- To change the speed of the internal clock for a placed PIC16F84/A:
 1. Double-click on the placed MCU to display its properties dialog box and click on the **Value** tab.
 2. Change the value in the **Clock Speed** field as desired.
 3. Click **OK** to close the dialog and accept the changes.

MCU Memory View

You can collapse/expand the fields in the **MCU Memory View** as shown below.



- **Registers** contains the SFRs.

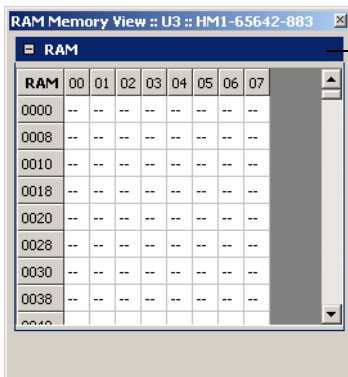
- **IRAM** contains the internal RAM (Random Access Memory) of the MCU. Shows the data that is inside the MCU's memory and is modified as the program runs. The colored areas represent SFRs (Special Function Registers), GPRs (General Purpose Registers) and different memory banks.
- **EEPROM** (Electrically Erasable Programmable Read Only Memory) contains the data EEPROM.
- **IROM** is the internal ROM (program memory) of the MCU. Shows the program memory code in hexadecimal format. These are the actual machine instructions that the simulation uses when it is activated. The left column shows the memory address and the header row shows the offset from the address on the left.
- **Stack** contains the processor stack view.
- **Configuration** contains the PIC configuration bits.

A.3 RAM

The RAM Family in the MCU Module Group contains a number of RAM devices for use with the Multisim MCU Module devices.

10	A0	D00	11
9	A1	D01	12
8	A2	D02	13
7	A3	D03	14
6	A4	D04	15
5	A5	D05	16
4	A6	D06	17
3	A7	D07	18
25	A8		19
24	A9		
21	A10		
23	A11		
2	A12		
27	W		
22	C		
26	E1		
24	E2		

HM1-65642-883



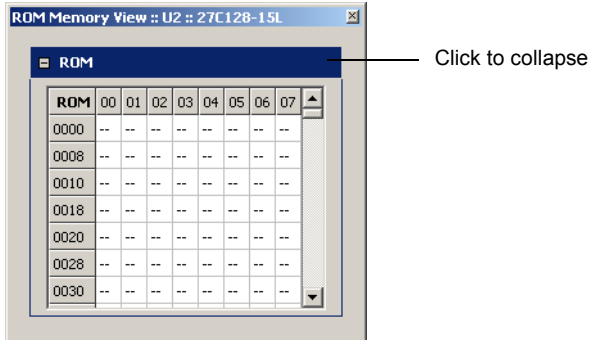
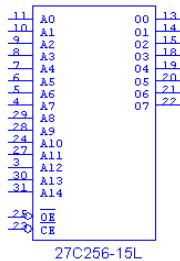
Click to collapse

The **RAM Memory View** is used to view the contents of the RAM chip while debugging.

- To show/hide the **RAM Memory View**:
 1. Select **MCU/<RAM name, e.g., RAM HM1-65642-883 U2>/Memory View**.

A.4 ROM

The **ROM Family** in the **MCU Module Group** contains a number of ROM devices for use with the Multisim MCU Module devices.



The **ROM Memory View** is used to view the contents of the ROM chip while debugging.

- To show/hide the **ROM Memory View**:
 1. Select **MCU/<ROM name, e.g., ROM 27C128-12L U3>/Memory View**.

Appendix B - Support and Services

B.1 Technical Support and Professional Services

Visit the following sections of the National Instruments web site at ni.com for technical support and professional services:

- **Support** — online technical support resources at ni.com/support include the following:
 - **Self-Help Resources** — For answers and solutions, visit the award-winning National Instruments web site for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
 - **Free Technical Support** — All registered users receive free Basic Service, which includes access to hundreds of Application Engineers worldwide in the NI Discussion Forums at ni.com/forums. National Instruments Application Engineers make sure every question receives an answer. For information about other technical support options in your area, visit ni.com/services or contact your local office at ni.com/contact.
- **Training and Certification** — Visit ni.com/training for self-paced training, eLearning, virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration** — If you have time constraints, limited in-house technical resources, or other project challenges, National Instruments Alliance Partner members can help. To learn more, call your local NI office or visit ni.com/alliance.

If you searched ni.com and could not find the answers you need, contact your local office or NI corporate headquarters. Phone numbers for our worldwide offices are listed in the front of this manual. You can also visit the Worldwide Offices section of ni.com/niglobal to access the branch office web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

Index

Numerics

8051 MCU A-1

8052 MCU A-1

B

breaking 5-6

breakpoints 5-4

build errors and warnings 3-12

building MCU projects 4-1

D

debug window settings 5-3

L

loading external hex file project 3-10

M

machine code file simulation 3-13

managing files in MCU workspace 2-4

MCU Code Manager

file management 3-3

MCU Code Manager overview 3-1

MCU design overview 2-1

MCU project build settings 3-3

MCU Wizard 2-3

MCU workspace 3-11

memory view 5-7

O

opening debug view 5-2

S

simulation markers 5-4

source code file 4-1

stepping 5-6