nohau

EMUL–MICROBLAZE–PC™

Getting Started Guide

Edition 4

Contents

Warranty Information v

European CE Requirements vi

User Responsibility vi

Special Measures for Electromagnetic Emission Requirements vi

System Requirements vi

Warnings vii

About This Guide viii

1 Overview of the EMUL–MICROBLAZE–PC Emulator System 1

Hardware 1

Software 1

2 Installing the Hardware 2

USB Driver 2

Power Supply 2

Installation Instructions 3

3 Installing the Seehau Software 5

Installing the Seehau Software From the CD 5

Installing the MicroBlaze EDK Software From the Xilinx EDK CD 5

Downloading and Installing the Seehau Software From the Internet 6

4 Configuring the Seehau Software 7

Selecting to Automatically Start the Seehau Configuration Program 7

Configuring the Communications Interface 8

Connect Tab 8

Hardware Configuration Tab 10

Configuring the Emulator Options From Within Seehau 11

Buttons Common to All Tabs 11

Hardware Configuration Tab 11

Miscellaneous Setup Tab 13

Starting Seehau 13

Demo Mode 13

Non-Demo Mode 13

5 Running Program Examples 15

Using The Trace 17

Buttons Common to All Tabs 17

Events Tab 17

Trace Display Window 18

6 Using the Nohau Project Manager 19

Overview 19

Accessing the Nohau Project Manager 19

Working with Workspaces 19

Adding a New Workspace 19

Opening a Workspace 20

Saving a Workspace 20

Working with Projects 20

Adding a New Project 20

Removing a Project 21

Adding a File to the Project 21

Removing a File from the Project 21

Changing the Workspace Settings 21

Changing the File Settings 22

Building a Project 23

Building a Single Project 23

Building All Projects 23

Forcing a Build of a Single Project 23

Forcing a Build of All Projects 23

Forcing a Build of a Selected File 23

7 JTAG TROUBLESHOOTER 24

JTAG TROUBLESHOOTER – WHEN SEEHAU DOES NOT START 24

Start JTAG TROUBLESHOOTER 25

Starting Seehau with the Right Data 26

First time from Config: 26

8 Using MicroBlaze Trace 27

Events Tab 27

Trigger Conditions 28

Program Address Trigger Condition 28

Register Write Trigger Condition 28

Probe Pins Channel A [7..0] Trigger Condition 28

Probe Pins Channel B[31..0] Trigger Condition 28

Post Trigger Count 29

Break on Trace Stop 29

Buttons Common to All Tabs 29

Trace Display Window 29

9 BlazeGen for EDK 6.2 Projects 33

BlazeGen 33

Compiling Programs from XPS for use with Nohau Tools 39

10 Using BlazeGen to Added DebugTrace Module 45

Working with Platform Studio 6.1 & BlazeGen 45

Make a Quick Project Using Xilinx's Platform Studio and Nohau's BlazeGen 45

Working with Platform Studio 6.2 & BlazeGen 51

Make a Quick Project Using Xilinx's **Platform Studio** and Nohau's **BlazeGen** 51

Index 58

Product Notes

Warranty Information

The pod board is sold with a one-year warranty starting from the date of purchase. The defective component under warranty will either be repaired or replaced at Nohau's discretion.

Nohau's Seehau software is sold with no warranty, but upgrades can be obtained to all customers at the Nohau Web site: http://www.icetech.com.

Nohau makes no other warranties, express or implied, including, but not limited to the implied warranties of merchantability and fitness for a particular purpose. In no event will Nohau be liable for consequential damages. Third-party software sold by Nohau carries the manufacturer's warranty.

European CE Requirements

Nohau has included the following information in order to comply with European CE requirements.

User Responsibility

The in-circuit debugger application, as well as all other unprotected circuits need special mitigation to ensure Electromagnetic Compatibility (EMC).

The user has the responsibility to take required measures in the environment to prevent other activities from disturbances from the debugger application according to the user and installation manual.

If the debugger is used in a harsh environment (field service applications for example), it is the user's responsibility to control that other activities cannot be disturbed in such a way that there might be risk for personal hazard/injuries.

Special Measures for Electromagnetic Emission Requirements

To reduce the disturbances to meet conducted emission requirements it is necessary to place a ground plane on the table under the pod cable and the connected processor board. The ground plane shall have a low impedance ground connection to the host computer frame. The insulation sheet between the ground plane and circuit boards shall not exceed 1mm of thickness.

System Requirements

CAUTION

Like all Windows applications, the Seehau software requires a minimum amount of free operating system resources. The recommended amount is at least 40%. (This is only a guideline. This percentage might vary depending on your PC.) If your resources are dangerously low, Seehau might become slow, unresponsive or even unstable. If you encounter any of these conditions, check your free resources. If they are below 40%, reboot and limit the number of concurrently running applications. If you are unable to free at least 40% of your operating system resources, contact your system administrator or Nohau Technical Support at support@icetech.com.

The following are minimum system requirements:

- Pentium 1Ghz (Pentium III or faster is recommended)
- Single-Processor System
- USB Port
- Windows 2000 or XP
- Random Access Memory (RAM)
  - For Windows 2000/XP: 256 MB

Warnings

To avoid damage to the pod or to your target, do not connect the pod to your target when the target power is on.

Do not apply power to your system unless you are sure the target adapter is correctly oriented. Failing to do so can cause damage to your target.

When using the pod with a target, disable all pod resources that are duplicated on the target. Failure to disable the pod's resources can damage the pod or the target or both.

When installing a controller into a pod, never press on the chip body. Press only on the carrier or cover. Pressing on the chip might bend pins and cause short circuits.

About This Guide

The EMUL–MICROBLAZE–PC is a PC-based hardware debugger for the MicroBlaze™ Core, available for the Spartan™-II, Spartan™-II/E, Spartan™-III and Virtex™-II FPGA platforms from Xilinx®. This guide helps you to get started with the basics of setting up, configuring, and running the Seehau software and debugger.

The *EMUL–MICROBLAZE–PC Getting Started Guide* is intended for both novice and advanced users. This guide introduces the following tasks:

- Installing and configuring the Seehau software
- Installing the debugger hardware
- Starting the hardware and Seehau software
- Shutting down Seehau

To download an electronic version of this guide, do the following:

1. Open Nohau's home page at www.icetech.com.
2. Click **Publications/Documents**.
3. Click **Nohau Manuals**.
4. Scroll down to EMUL–MICROBLAZE–PC. Then select EMUL–MICROBLAZE–PC to download a PDF version of this guide.

1 Overview of the EMUL–MICROBLAZE–PC Emulator System

Hardware

The basic hardware for the EMUL–MICROBLAZE debugger system is the EMUL–PC/USB–JTAG. At present, SeehauBLAZE supports the Spartan-II, Spartan-II/E and the Virtex-II FPGA platforms.

Refer to Chapter 2, "Installing the Hardware" for detailed hardware information overview.

Software

The debugger is configured and operated by the SeehauBLAZE user interface. Seehau is a high-level language user interface that allows you to perform many useful tasks, for example:

- Editing code in Source Window and building projects using the Nohau Project Manager and the supplied MicroBlaze GNU tools. The mb-gcc compiler, mb-as assembler and mb-ld loader/linker platform supplied by Nohau.
- Loading code into RAM, running, and stopping programs based on the MicroBlaze GNU tools. Additional documentation for programming the MicroBlaze core is available from Xilinx. This special version of the assembler outputs in ELF format specifically for the Nohau debugger. Contact Nohau Technical Support if you have questions.
- Modifying and viewing memory contents including general-purpose registers.
- Setting multiple software breakpoints that are placed in RAM.
- Set a trigger and display the trace.

2 Installing the Hardware

USB Driver

When installing the USB device, you must install the Seehau software first before connecting the Nohau hardware (Refer to Chapter 3, "Installing the Seehau Software."). This allows the computer to recognize the proper driver for the hardware. The USB option is not supported by Windows 95 or NT.

The USB drivers are loaded as part of the Seehau software installation. The driver is located in the root directory of the installation CD. After installation, the driver is also located in the SeehauBLAZE subdirectory on your hard drive (C:\Nohau\SeehauBLAZE\). The system software should find and load the USB driver without your intervention.

Power Supply

The EMUL–PC/USB–JTAG uses power supplied by the USB cable and the target board. The amount of power used by the pod from the target board is less than 100 µA. When drawing power from the target board, the pod must draw from a 2.3V to 3.6V range. The power supplied by the target board drives the signal from the buffer on the pod through the 10-pin serial debug connector to the target board. The pod draws power from the target board through Pin 9 (Vdd).

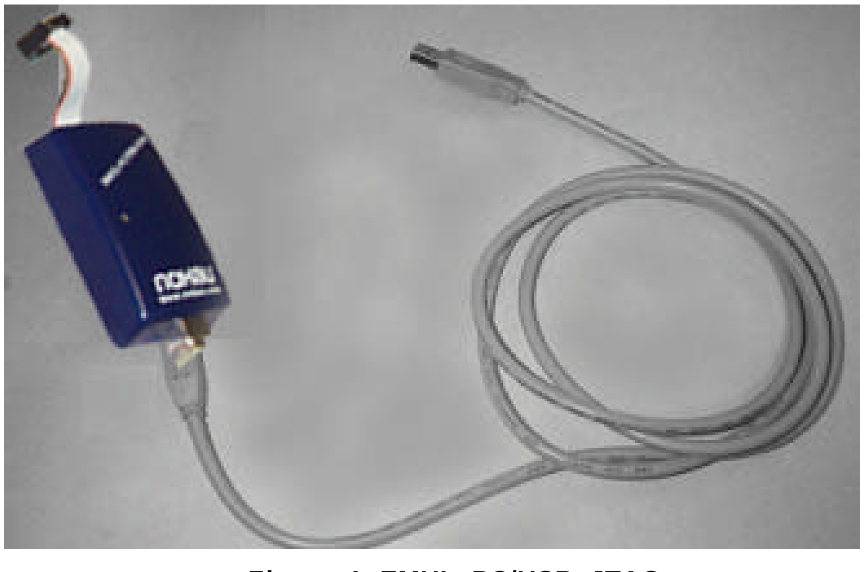


Figure 1. EMUL–PC/USB–JTAG

Note

The EMUL–PC/USB–JTAG is powered from the target. This allows the hardware to operate with a wide range of target voltages.

The following table shows the signal layout of the connector:

Pin	Name	Description
1	JTAG RST	JTAG Reset (not used for MicroBlaze) (Drive = Push/Pull)
2	/TMS	Target slave select
3	GND	Ground
4	TCK	Target data clock
5	Reserved	Reserved
6	Reserved	Reserved
7	/SRST	System reset (Drive = Open-Drain)
8	TDI	Target serial input
9	Vdd	Power
10	TDO	Target serial output

Installation Instructions

Refer to Figure 2 for a diagram of the connectors while following the installation instructions.

1. Make sure your target is powered off.

2. Plug the 10-pin EMUL–PC/USB–JTAG serial debug connector onto your target’s 10-header. This connector, supplies the signals needed to communicate with the EMUL–PC/USB–JTAG.

3. Power on your target.

Note

Exit Seehau and shut down the power to the target when the target is not in use.

The red wire indicates pin one on the 10-pin serial debug connector.

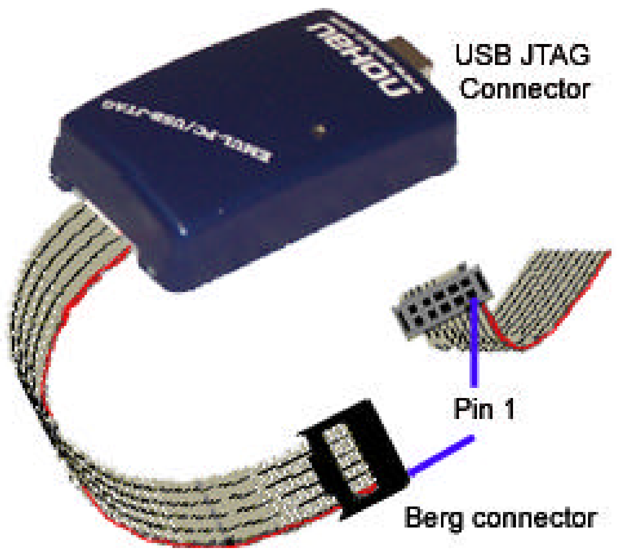


Figure 2. EMUL–MicroBlaze–PC/USB–JTAG with the 10-Pin Serial Debug Connector

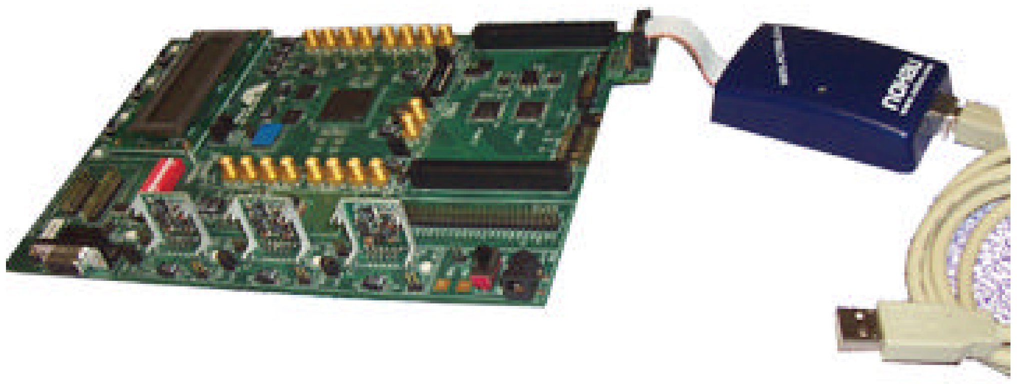


Figure 3. Connections for the USB Cable and the Red Wire Pin 1

3 Installing the Seehau Software

Installing the Seehau Software From the CD

To install the Seehau software, do the following:

1. Locate your Seehau CD and insert the CD into your CD ROM drive. The installation process will start automatically.
2. Click **Install Seehau Interface for EMUL-MicroBlaze-PC** and follow the instructions that appear on your screen.

If the installation does not start automatically, you might have your Windows Autorun feature disabled. You will then need to do one of three things:

- Use Windows Explorer and navigate to the CD root directory. Double-click **Autorun.exe**. The Windows Install Shield will start the installation process.
- Right-click on the CD ROM symbol while running Windows Explorer and select **AutoPlay** to start the installation process.
- From the taskbar, select **Start**, then **Settings**. Click on **Control Panel**, then **Add/Remove Programs**, and then **Install**. The installation process will start when you select the correct path to the CD.

Installing the MicroBlaze EDK Software From the Xilinx EDK CD

In order to build programs using the Nohau Project Manager you need to install the MicroBlaze EDK software using the following:

1. Locate your EDK CD and insert the CD into your CD ROM drive. The installation process will start automatically.
2. Click **Install the MicroBlaze EDK** and follow the instructions that appear on your screen.

If the installation does not start automatically, you might have your Windows Autorun feature disabled. You will then need to do one of three things:

- Use Windows Explorer and navigate to the CD root directory. Double-click **Autorun.exe**. The Windows Install Shield will start the installation process.
- Right-click on the CD ROM symbol while running Windows Explorer and select **AutoPlay** to start the installation process.

- From the taskbar, select **Start**, then **Settings**. Click on **Control Panel**, then **Add/Remove Programs**, and then **Install**. The installation process will start when you select the correct path to the CD.

Downloading and Installing the Seehau Software From the Internet

1. Go to the Nohau web site (http://www.icetech.com/). Click **Downloads**. The Nohau Software Downloads page opens.

2. Click **Current Seehau Software**. The Seehau Software Status page opens.

3. Locate the EMUL–MICROBLAZE–PC product listing. There will be two listings, one for documentation and one for software.

4. Click **Information and Download (Seehau)**.

5. Review the information on the page.

6. Click **Yes I Want to Download**. A Customer Information Form page opens. Complete this form, then click **Proceed**. (You have the option to download more than one product.) A verification page opens with the information you have just entered. If all information is correct, select **SEND** at the bottom of the page. A message will open that verifies your information has been sent.

7. Click **Go to Download**. The **Available Download Areas** page opens.

8. Click either option for a download site. The Nohau Software Updates page opens.

9. Click the EMUL–MICROBLAZE–PC link.

10. Click the MicroBlaze.exe link. The application will start downloading. Remember which directory has this downloaded file.

11. Following the download, go to the directory, which has the downloaded file. Click the MicroBlaze.exe file and follow the installation instructions.

After installing the Seehau software, the **Setup Complete** dialog boxes appears that allows you to view the Readme.txt file and/or launch the SeehauBLAZE configuration.

Note

You must launch the SeehauBLAZE configuration before running the Seehau software.

4 Configuring the Seehau Software

Selecting to Automatically Start the Seehau Configuration Program

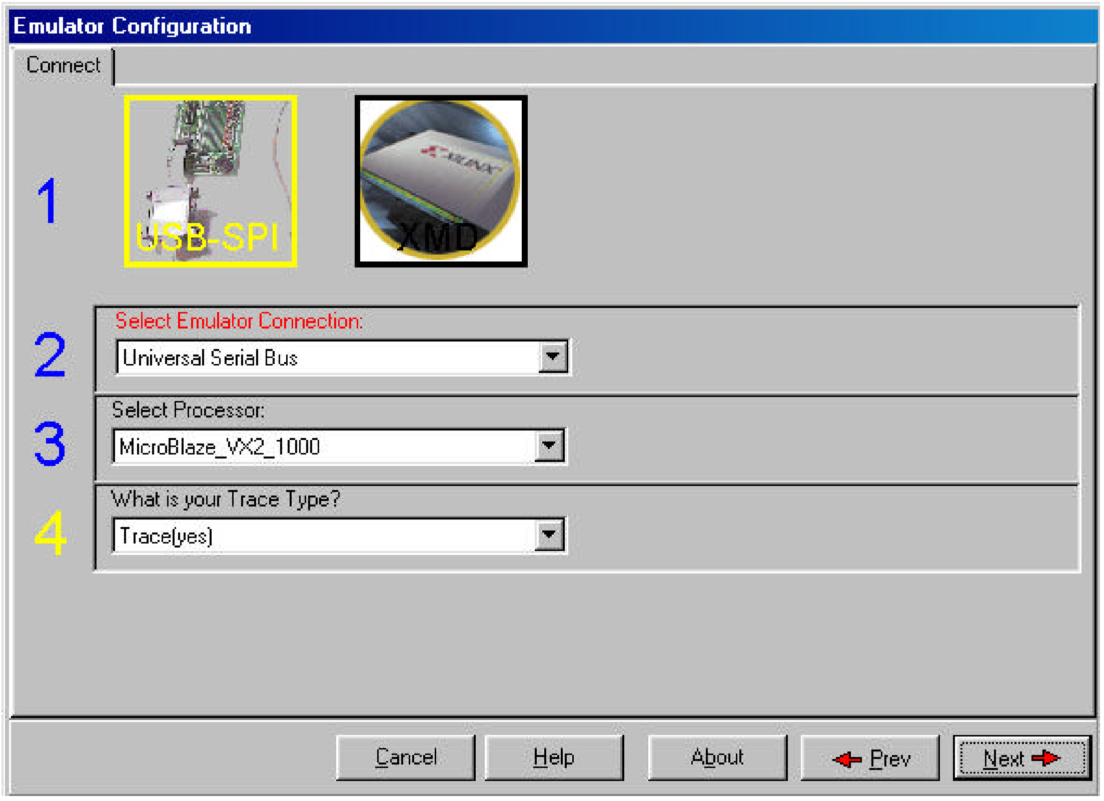
After installing Seehau, it is recommended that you automatically start the Seehau Configuration program. Do the following steps before starting Seehau:

1. From the **Setup Complete** dialog box, select **Launch SeehauBLAZE Configuration**.
2. Click **Finish**.

If you do not select to automatically start the Seehau Configuration Program, do the following:

From the **Start** menu, select **Programs**.

Select **SeehauBLAZE**. Then click **Config** to open the **Emulator Configuration** window displaying the **Connect** tab (Figure 4).



Select Emulator Connection: Universal Serial Bus

Select Processor: MicroBlaze_VX2_1000

What is your Trace Type? Trace(yes)

Figure 4. Emulator Configuration Window Displaying the Connect Tab

Note

You do not need the hardware connected at this time

Configuring the Communications Interface

Connect Tab

The graphical user interface for this tab is divided into four regions. Do the following in each region:

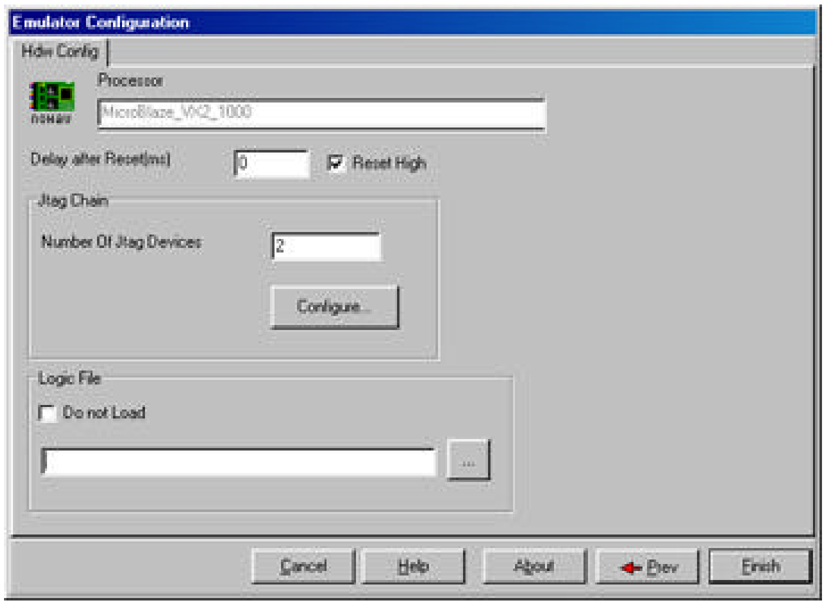
1. Region 1—**Communications Interface**:
   Displays the USB-SPI communications interface for the MicroBlaze pod or the Xilinx Parallel Cable option.

2. Region 2—**Select Emulator Connection**:
   No action required. Default is **Universal Serial Bus**.

3. Region 3—**Select Processor**:
   Default is **MicroBlaze_SPARTAN_2E_200** for the Spartan-II/E on the Nohau MicroBlaze evaluation board.
   Select **MicroBlaze-VX2-1000** for the Virtex-II or the Virtex-II PRO. Depending on your version of software you may have to select the this option also for the Spartan-III. Otherwise, Select Spartan-III for the Spartan-III.

4. Region 4—**What is your Trace Type?**:
   Select **Trace (Yes)**. Default is **Yes**.

5. Click **Next**. The **Hdw Config** tab opens (Figure 5).


Emulator Configuration

Hdw Config

Processor: MicroBlaze_VX2_1000

Delay after Reset(ms): 0

☑ Reset High

Jtag Chain

Number Of Jtag Devices: 2

Logic File

☐ Do not Load

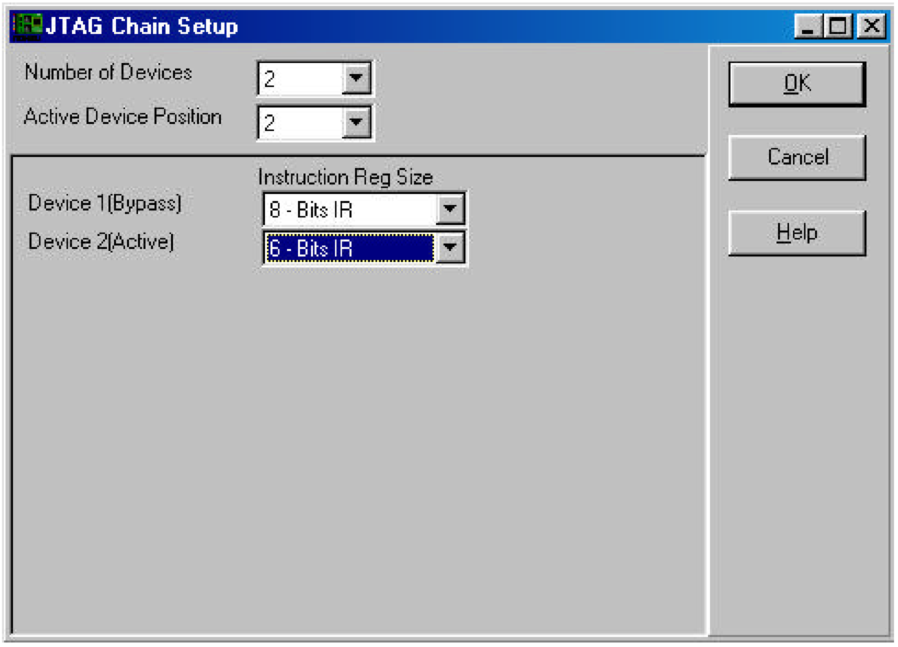
Figure 5. Emulator Configuration Window Displaying the Hdw Config Tab

Hardware Configuration Tab

The initial configuration settings are read from the Startup.bas file when the debugger initializes. After the initial startup and configuration of Seehau, the configuration parameters are saved in the Startup.bas file (located in the Macro subdirectory). The next time the emulator is started, the configuration parameters are read in from the Startup.bas file and compared with the default parameters on the chip.

- **Processor**: Shown for reference only. This is the selected processor. If you need to change the pod type, click **Prev**.
- **Delay after Reset(ms)**: Used to set the length of time Seehau asserts reset.
- **Reset High**: Used to set the type of reset configured in the MicroBlaze core. The default is active high.
- **JTAG Chain**: Used to configure the JTAG chain setup so that Seehau can talk to the MicroBlaze core in the current scan chain. (See Figure 6). With versions of Seehau that have been built after May 20, 2004 the software will automatically try to detect the devices in the scan chain and automatically fill in these values.
- **Logic File**: Used to configure the initial bit stream to be loaded into the FPGA.

When you click **Finish**, Seehau starts to load. For more information about starting Seehau, see the “Starting Seehau” section at the end of this chapter.



Number of Devices: 2
Active Device Position: 2
Instruction Reg Size
Device 1(Bypass): 8 - Bits IR
Device 2(Active): 6 - Bits IR

Figure 6. Emulator Configuration Window Configure JTAG Chain Button

- **Number of Devices**: Used to set the total number of devices in the scan chain
- **Active Device Position**: Used to set where the MicroBlaze core is in scan chain relationship.

- **Device (x) Bypass**: Used to set the “Instruction Register length” of each device in the chain.

Configuring the Emulator Options From Within Seehau

From Seehau, open the Emulator Configuration window. From the **Config** menu select **Emulator**.

The Emulator Configuration window contains two tabs. When selected, each tab allows you to set the following options:

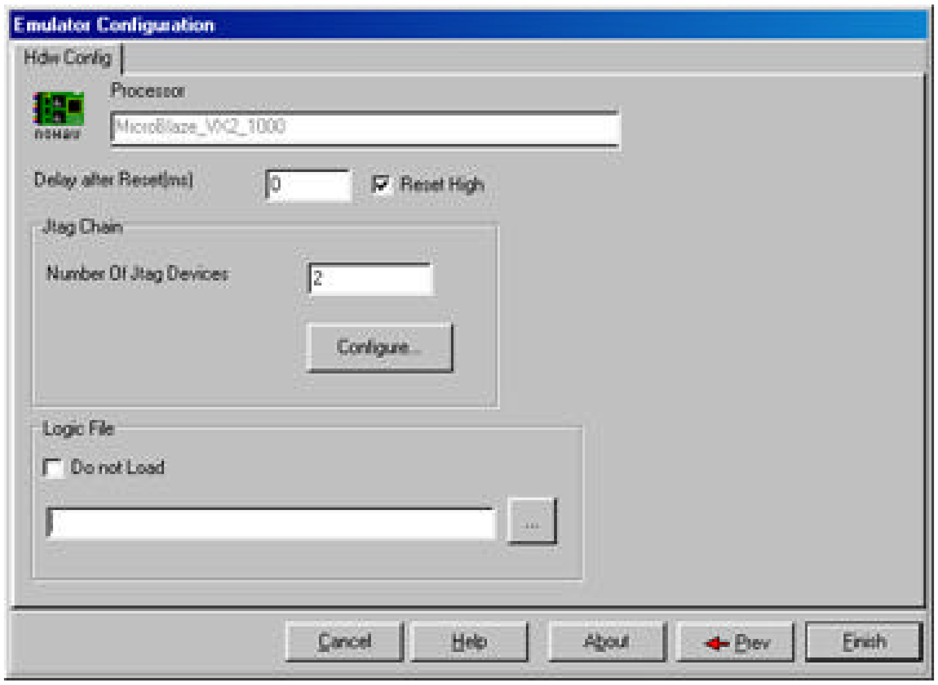
Hdw Cfg:	Set up emulator hardware options.
Misc Setup:	Select reset options.

Buttons Common to All Tabs

- **OK**: Saves the settings for the tab and exits the dialog box.
- **Apply**: Saves the settings for the tab.
- **Cancel**: Exits without saving the settings for the dialog box.
- **Help**: Displays the Seehau Help file.
- **Refresh**: Allows you to retrieve and view the current emulator hardware configuration settings.

Hardware Configuration Tab

- **Processor**: Shown for reference only. This is the selected processor. If you need to change the pod type, click **Prev**.
- **Delay after Reset(ms)**: Used to set the length of time Seehau asserts reset.
- **Reset High**: Used to set the type of reset configured in the MicroBlaze core. The default is active high.
- **JTAG Chain**: Used to configure the JTAG chain setup so that Seehau can talk to the MicroBlaze core in the current scan chain. (See Figure 6).
- **Logic File**: Used to configure the initial bit stream to be loaded into the FPGA.



Processor: MicroBlaze_VX2_1000

Delay after Reset(ms): 0

☑ Reset High

Jtag Chain

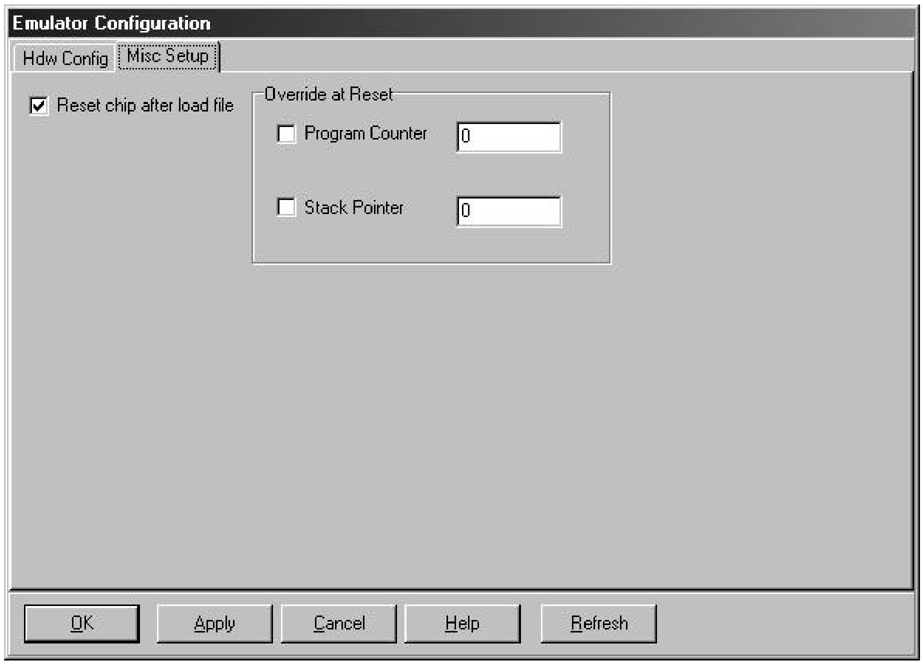
Number Of Jtag Devices: 2

Configure...

Logic File

☐ Do not Load

Figure 7. Emulator Configuration Window Displaying the Hdw Config Tab



☑ Reset chip after load file

Override at Reset

☐ Program Counter: 0

☐ Stack Pointer: 0

Figure 8. Emulator Configuration Window Displaying the Misc Setup Tab

Miscellaneous Setup Tab

The **Misc Setup** tab () is accessible only after the initial software configuration.

- **Reset chip after load file**: Sets the MicroBlaze core to issue a reset after the code is loaded.
- **Override at Reset**
  - The **Program Counter** option selects the value that the program counter will be set to after a reset. Enter the program counter value in the box.
  - The **Stack Pointer** option selects the value that the stack pointer will be set to after a reset. Enter the stack pointer value in the box.

WARNING

When the target is powered on, the EMUL–PC/USB–JTAG pod is powered by the target through a diode.

When powering down, turn off the target first and then the pod (the pod receives its power through the USB cable so the PC must be powered down to power off the pod.)

Starting Seehau

Demo Mode

1. Double-click the Demo icon. The Seehau main window opens (Figure 9). Seehau will load its configuration from the Startup.bas file. Notice that the word macro is displayed in red at the bottom of the main window while Startup.bas is running.

2. When the software startup is complete, you can position and resize the main window to your preference. At this time, you will need to load code.

3. To open new windows, from the **New** menu click a window option.

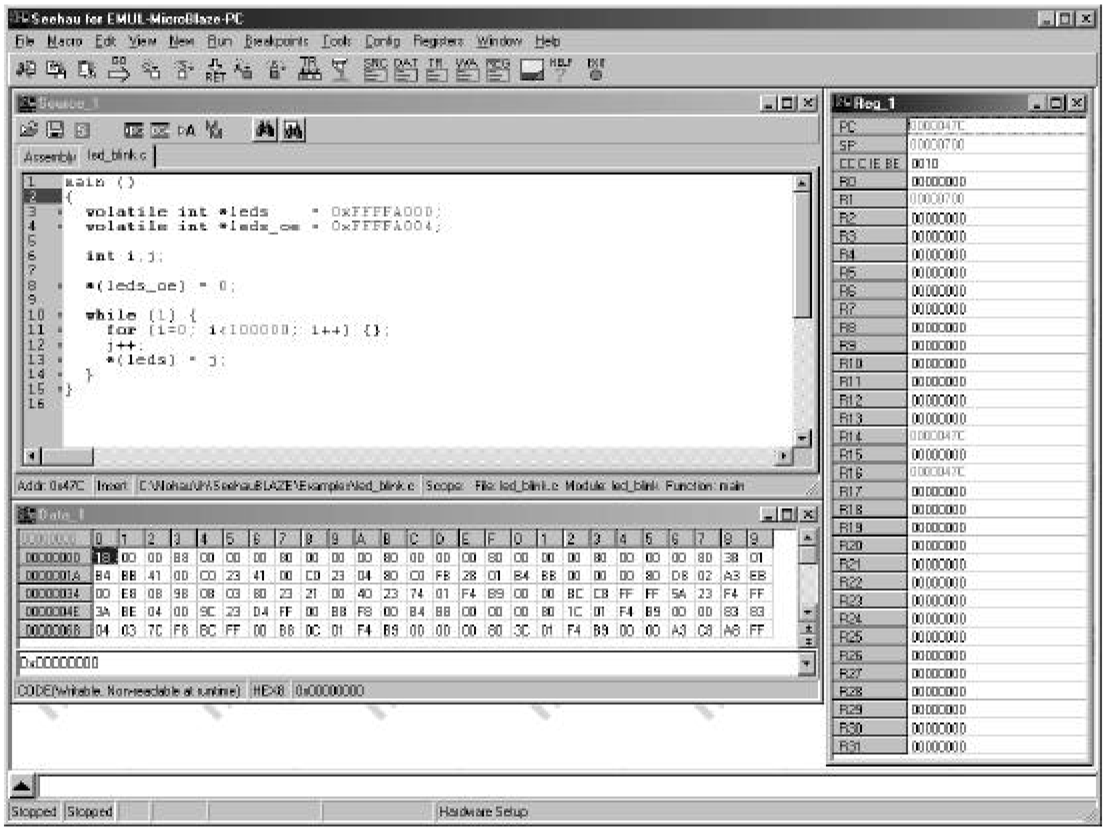
Non-Demo Mode

To start Seehau, in the non-demo mode do the following:

1. Double-click the SeehauBLAZE icon. The Seehau main window opens (Figure 9). Seehau will load its configuration from the Startup.bas file. Notice that the word macro is displayed in red at the bottom of the main window while Startup.bas is running.

2. When the software startup is complete, you can position and resize the main window to your preference. At this time, you will need to load code.

3. To open new windows, from the **New** menu click a window option.


Seehau for EMUL-MicroBlaze-PC
File Macro Edit View New Run Breakpoints Tools Config Registers Window Help
Source_1
Assembly
led_blink.c
Data_1
Reg_1
Stopped
Stopped
Hardware Setup

```
main ()
{
  volatile int *leds    = 0xFFFFA000;
  volatile int *leds_oe = 0xFFFFA004;

  int i,j;

  *(leds_oe) = 0;

  while (1) {
    for (i=0; i<100000; i++) {};
    j++;
    *(leds) = j;
  }
}
```

```
Addr: 0x47C | Insert | C:\Nohau\UI\SeehauBLAZE\Examples\led_blink.c | Scope: File: led_blink.c Module: led_blink Function: main
```

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0  1  2  3  4  5  6  7  8  9
00000000  18 00 00 B8 C0 C0 00 80 00 00 00 80 00 00 C0 80 C0 00 00 80 00 00 00 80 38 01
0000001A  B4 BB 41 00 C0 23 41 00 C0 23 04 80 C0 F8 28 01 B4 BB 00 00 00 80 D8 02 A3 EB
00000034  00 E8 08 98 08 03 80 23 21 00 40 23 74 01 F4 B9 C0 00 BC C8 FF FF 5A 23 F4 FF
0000004E  3A BE 04 00 9C 23 D4 FF 00 BB F8 00 B4 BB C0 C0 C0 80 1C 01 F4 B9 00 00 83 83
00000068  04 03 7C F8 BC FF 00 B8 0C 01 F4 B9 00 00 C0 80 3C 01 F4 B9 00 00 A3 C8 A8 FF
0x00000000
CODE(Writable, Non-readable at runtime) | HEX8 | 0x00000000
```

```
PC         0000047C
SP         00000700
CC C IE BE 0010
R0         00000000
R1         00000700
R2         00000000
R3         00000000
R4         00000000
R5         00000000
R6         00000000
R7         00000000
R8         00000000
R9         00000000
R10        00000000
R11        00000000
R12        00000000
R13        00000000
R14        0000047C
R15        00000000
R16        0000047C
R17        00000000
R18        00000000
R19        00000000
R20        00000000
R21        00000000
R22        00000000
R23        00000000
R24        00000000
R25        00000000
R26        00000000
R27        00000000
R28        00000000
R29        00000000
R30        00000000
R31        00000000
```

Figure 9. Seehau for EMUL–MICROBLAZE–PC

5 Running Program Examples

Nohau provides a C program example called led_blink located at:
C:\Nohau\SeehauBLAZE\Examples\led_blink.out)

Note

The C programs do not have a jump to main when XMD_STUB is used. Using the **Misc Setup** tab of the Emulator Configuration window, set the PC to **main** and the stack pointer to **700h** before stepping into the C programs. Once you have done this, step into the C source code by pressing the Source Step Into button on the Seehau Speedbar.

Start Seehau by following the instructions in the “Starting Seehau” section in Chapter 4. Then do the following to run the test program:

1. Resize the windows on your screen, but do not add the Watch window.
2. From the **File** menu select **Load Code**. The **Open** dialog box opens (Figure 10).
3. Select one of the program example files for your controller.

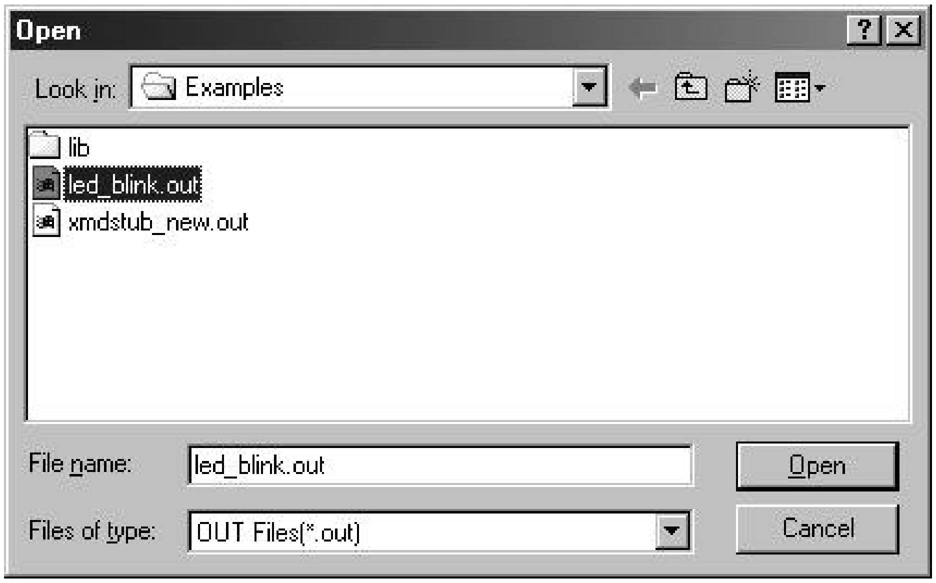
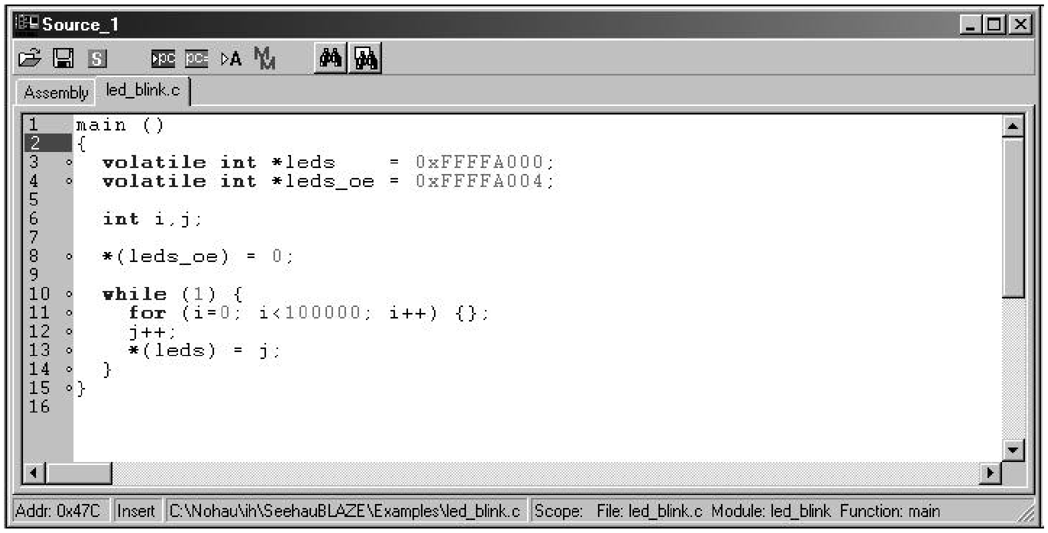


Figure 10. Open Dialog Box Displaying the Program Example Directories



```
main ()
{
  volatile int *leds    = 0xFFFFA000;
  volatile int *leds_oe = 0xFFFFA004;

  int i,j;

  *(leds_oe) = 0;

  while (1) {
    for (i=0; i<100000; i++) {};
    j++;
    *(leds) = j;
  }
}
```

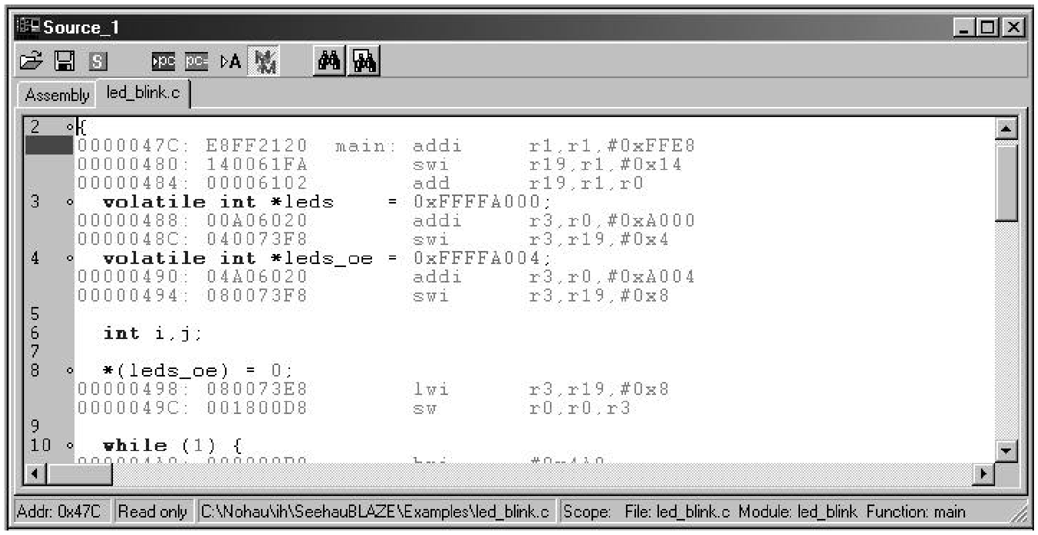
Addr: 0x47C | Insert | C:\Nohau\ih\SeehauBLAZE\Examples\led_blink.c | Scope: File: led_blink.c Module: led_blink Function: main

Figure 11. Program Example in Source Only

4. Highlight the program example file and click **Open**. The source file automatically displays the program example in source only (Figure 11).

5. To single-step in mixed mode, click the Assembly Step Into or Step Over button. You will see the assembly code mixed in with the associated source lines (Figure 12).

6. To single-step in source only mode, first click the Source window. Select the **led_blink.c** tab. Right-click to verify that mixed mode is cleared. Then point to the **Setting** sub-menu item, and select **Lock Tabs**.



```
{
0000047C: E8FF2120  main: addi     r1,r1,#0xFFE8
00000480: 140061FA        swi      r19,r1,#0x14
00000484: 00006102        add      r19,r1,r0
  volatile int *leds    = 0xFFFFA000;
00000488: 00A06020        addi     r3,r0,#0xA000
0000048C: 040073F8        swi      r3,r19,#0x4
  volatile int *leds_oe = 0xFFFFA004;
00000490: 04A06020        addi     r3,r0,#0xA004
00000494: 080073F8        swi      r3,r19,#0x8

  int i,j;

  *(leds_oe) = 0;
00000498: 080073E8        lwi      r3,r19,#0x8
0000049C: 001800D8        sw       r0,r0,r3

  while (1) {
```

Addr: 0x47C | Read only | C:\Nohau\ih\SeehauBLAZE\Examples\led_blink.c | Scope: File: led_blink.c Module: led_blink Function: main

Figure 12. Program Example in Mixed Mode

Using The Trace

The EMUL-MICROBLAZE-PC-Pro provides a 128K deep trace with a trigger, post trigger count and break control.

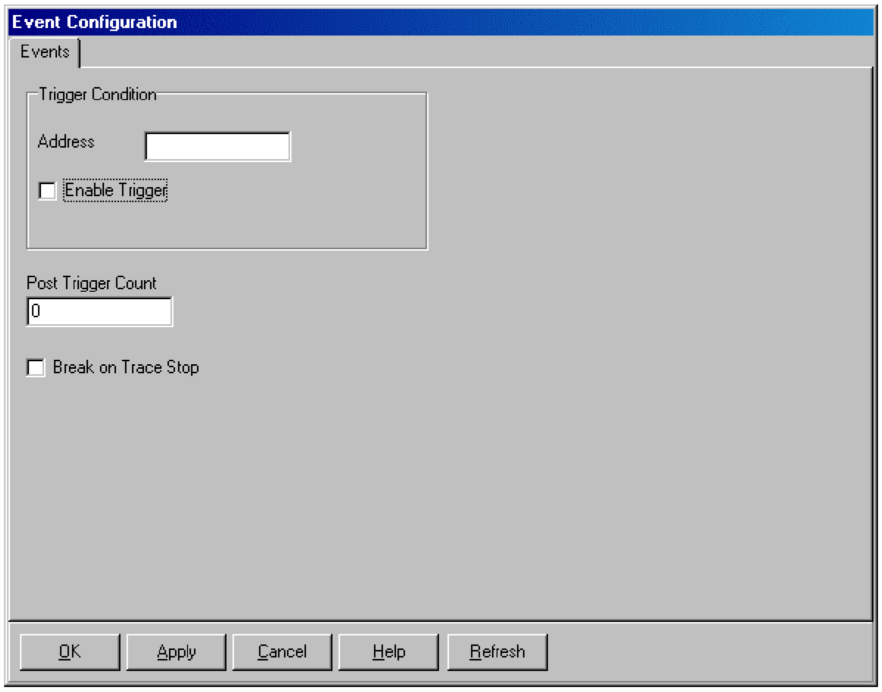
The **Event Configuration** window is selected from the **Config** menu select **Trace**. The Event Configuration window opens displaying the **Events** tab (Figure 13).

Buttons Common to All Tabs

- **OK**: Saves the settings for the tab and exits the dialog box.
- **Apply**: Saves the settings for the tab.
- **Cancel**: Exits without saving the settings for the dialog box.
- **Help**: Displays the Seehau Help file.
- **Refresh**: Allows you to retrieve and view the current event configuration settings.

Events Tab

- **Trigger Condition**
  - The value in the **Address** is the value that trigger is set to.
  - The **Trigger Enable** activates the trigger.
- **Post Trigger Count**: The frames to capture after the trigger condition has occurred. When the Post Trigger Count is zero, this will indicate the trigger has occurred.
- **Break on Trace Stop**: When checked will cause emulation to break when the trace has stopped.



Event Configuration

Events

Trigger Condition

Address

☐ Enable Trigger

Post Trigger Count

0

☐ Break on Trace Stop

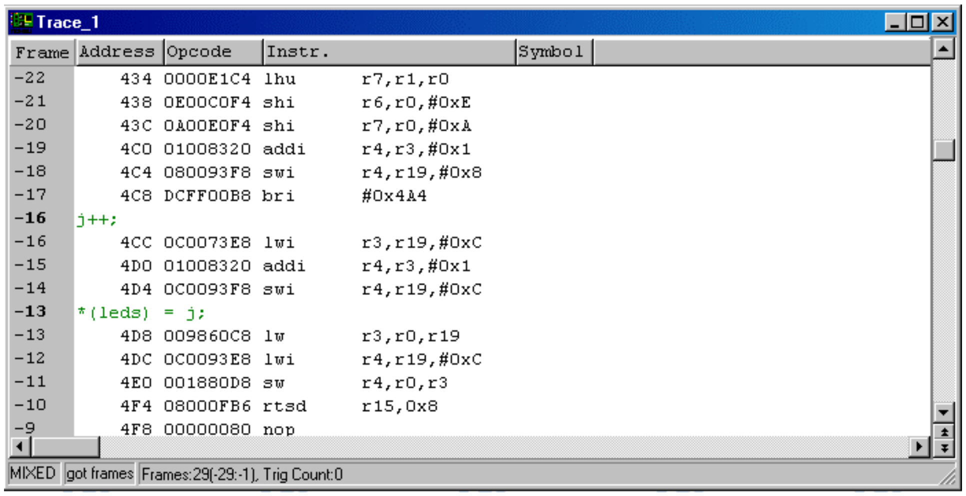
OK | Apply | Cancel | Help | Refresh

Figure 13. Event Configuration Window Displaying the Events Tab

Trace Display Window

The **Trace Display** window is used to display the results of the trace capture. The default fields displayed from right to left are the **Frame, Address, Opcode, Instruction** and **Symbol**.

The **Trace Display** window can display trace only (assembly), mixed (source and assembly) or source only. The mixed mode is shown in (Figure 14).



Trace_1

```
Frame Address Opcode   Instr.            Symbol
-22       434 0000E1C4 lhu      r7,r1,r0
-21       438 0E00C0F4 shi      r6,r0,#0xE
-20       43C 0A00E0F4 shi      r7,r0,#0xA
-19       4C0 01008320 addi     r4,r3,#0x1
-18       4C4 080093F8 swi      r4,r19,#0x8
-17       4C8 DCFF00B8 bri      #0x4A4
-16   j++;
-16       4CC 0C0073E8 lwi      r3,r19,#0xC
-15       4D0 01008320 addi     r4,r3,#0x1
-14       4D4 0C0093F8 swi      r4,r19,#0xC
-13   *(leds) = j;
-13       4D8 009860C8 lw       r3,r0,r19
-12       4DC 0C0093E8 lwi      r4,r19,#0xC
-11       4E0 001880D8 sw       r4,r0,r3
-10       4F4 08000FB6 rtsd     r15,0x8
-9        4F8 00000080 nop
```

MIXED | got frames | Frames:29(-29:-1), Trig Count:0

Figure 14. Trace Display in Mixed Mode

6 Using the Nohau Project Manager

Overview

The Nohau Project Manager is an external application that allows you to launch compilers and linkers for processing source files and linker command files as specified in the workspace. The Project Manager gives you different options of how to build the source files and automatically load the final object file into the emulator. This chapter describes how to:

- Create a workspace.
- Add a project to the workspace.
- Add files to the project.
- Specify file settings for each of the source files (can be done as a group or individually).
- Test the project settings.
- Open the source file in the Seehau Source window.

Accessing the Nohau Project Manager

Accessing the Nohau Project Manager can be done in one of the following ways:

- Start the Windows Explorer utility and go to C:\Nohau\SeehauBLAZE. Double-click NPrjMngr.exe.
- Within Seehau, from the **Tools** menu, select **Open Project Manager**.

The Nohau Project Manager window opens.

Working with Workspaces

Adding a New Workspace

1. From the **File** menu, select **New Workspace**. (You can also click the New Workspace button on the toolbar). This opens the **New Workspace** dialog box.
2. Type in the name of the new workspace in the **Workspace name** text field.
3. Click **OK**. The new workspace name appears in the left side of the Nohau Project Manager window. (In Figure 15, the example is **My Workspace**.)

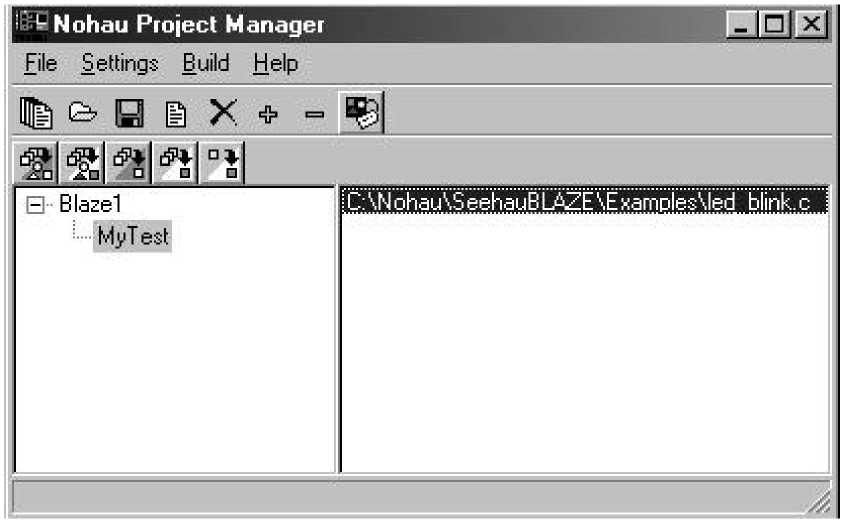


Figure 15. Nohau Project Manager Window

Opening a Workspace

1. To open a workspace from the **File** menu, select **Open Workspace**. (You can also click the Open Workspace button on the toolbar). This opens the **Open** dialog box.

2. Select the file **C:\Nohau\SeehauBLAZE\Examples\Blaze1.npr**, click **Open**. The name of the selected file appears in the left side of the Nohau Project Manager window.

Saving a Workspace

1. To save a workspace from the **File** menu, select **Save Workspace**. (You can also click the Save Workspace button on the toolbar.) This opens the **Select Directory** dialog box.

2. When you have placed your workspace file where you want it, click **OK** to save the file and exit the dialog box.

Working with Projects

Adding a New Project

1. Click on the workspace name in the left side of the Nohau Project Manager window. (In Figure 15, the example is **Blaze1**.)

2. From the **File** menu, select **Add New Project**. (You can also click the New Project button on the toolbar). This opens the **New Project** dialog box.

3. Type in the name of the new project in the **Project Name** text field.

4. Click **OK**. The new project name appears in the left side of the Nohau Project Manager window. (In Figure 15, the example is **My Test**.)

Removing a Project

1. Select (highlight) the project name.

2. From the **File** menu select **Remove Project**. (You can also click the Remove Project button on the toolbar).

3. A message box opens asking you **Do you want to remove project (file name)?** Click **Yes** to remove the selected file, or **Cancel** to close this box.

Adding a File to the Project

1. Select (highlight) the project name.

2. From the **File** menu select **Add File to Project**. (You can also click the Add File to Project button on the toolbar). This opens the **Open** dialog box.

3. When you have located the .c file you want to add, click **Open**. The name of your selected file appears in the right side of the Nohau Project Manager window.
(In Figure 15, the example is **C:\Nohau\SeehauBLAZE\Examples\led_blink.c**.)

Removing a File from the Project

1. Select (highlight) the file name. (In Figure 15, the example is **C:\Nohau\SeehauBLAZE\Examples\led_blink.c**.)

2. From the **File** menu select **Remove File from Project**. (You can also click the Remove File from Project button on the toolbar).

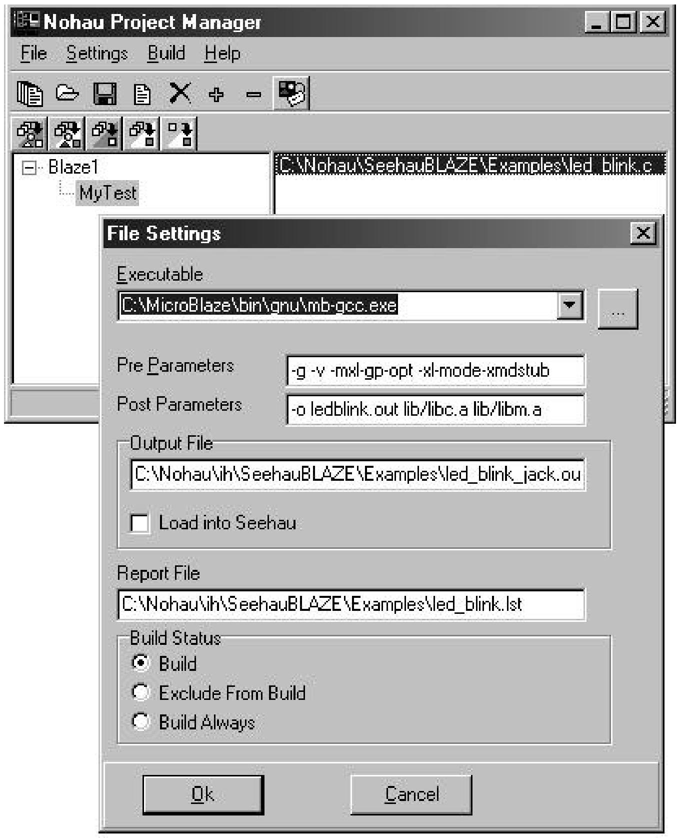
3. A message box opens asking you **Do you want to remove file (file name)?** Click **Yes** to remove the selected file, or **Cancel** to close this box.

Changing the Workspace Settings

1. From the **Settings** menu select **Workspace Settings**. The **Workspace Settings** dialog box opens.

2. Select one of the compiler-specific tools, this is mb-gcc.exe for MicroBlaze. This file translates the errors generated by the compiler into a format that allows you to view the errors in the Source window. If your compiler does not have a Nohau error-translating tool, contact Nohau Technical Support at support@icetech.com.

You can also write your own error-translating tool. The .exe file that you create reads the file as a command line parameter when you launch the tool. This file name is displayed in the **Report File** text field in the **File Settings** dialog box (Figure 16). To open this dialog box from the **Settings** menu, select **File Settings**.



Executable: C:\MicroBlaze\bin\gnu\mb-gcc.exe

Pre Parameters: -g -v -mxl-gp-opt -xl-mode-xmdstub

Post Parameters: -o ledblink.out lib/libc.a lib/libm.a

Output File: C:\Nohau\ih\SeehauBLAZE\Examples\led_blink_jack.ou

☐ Load into Seehau

Report File: C:\Nohau\ih\SeehauBLAZE\Examples\led_blink.lst

Build Status: [✓ Build] Exclude From Build Build Always

Figure 16. File Settings Dialog Box

The error-translating tool parses the file for error messages. It then appends them to the Cmperror.log file for Seehau to read it. The .log file should be in the following format:
Filename#linenumber#the original error text

With this convention, the Nohau tool supports most compilers and message outputs.

Changing the File Settings

1. Select (highlight) the file name. (In Figure 16, the example is **C:\Nohau\SeehauBLAZE\Examples\led_blink.c**.)

2. From the **Settings** menu select **File Settings**. (You can also click the File Settings button on the toolbar.) This opens the **File Settings** dialog box (Figure 16).

You can now choose the functions you wish to perform, such as entering post parameters, creating an executable path to process the file, creating an output file name and a report file name. You can also choose to load the output file into Seehau, exclude the file from the build, or specify if it should be the default file. Selecting the **Build** option ensures the source file will be built if the date of the output file is older than the date of the source file.

For example, if you want to compile a .c file that is called Test.c, the executable field should contain the path to the compiler. The **Pre Parameter** field contains command line parameters to the compiler that appear before the file name Test.c. The **Post Parameter** field specifies command line parameters that appear after the file name Test.c. The **Output File** filename is Test.obj. The **Report File** filename is Test.lst.

If the file settings are specified for the linker command file, the **Output File** name is Test.omf or Test.hex and the **Load Into Seehau** option should be selected. This instructs Seehau to load the linked output file into the emulator.

Building a Project

Building a Single Project

1. Select (highlight) the project.
2. From the **Build** menu select **Build Selected Project (Make)**. (You can also click the Build Selected Project(Make) button on the toolbar.)

Building All Projects

From the **Build** menu select **Build All Projects (Make)**. (You can also click the Build All button on the toolbar.)

Forcing a Build of a Single Project

1. Select (highlight) the project.
2. From the **Build** menu select **Rebuild Selected Project**. (You can also click the Rebuild Selected Project button on the toolbar.)

Forcing a Build of All Projects

From the **Build** menu select **Rebuild All Projects**. (You can also click the Rebuild All button on the toolbar.)

Forcing a Build of a Selected File

1. Select (highlight) the file.
2. From the **Build** menu select **Build Selected File**. (You can also click the Build Selected File button on the toolbar.)

7 JTAG TROUBLESHOOTER

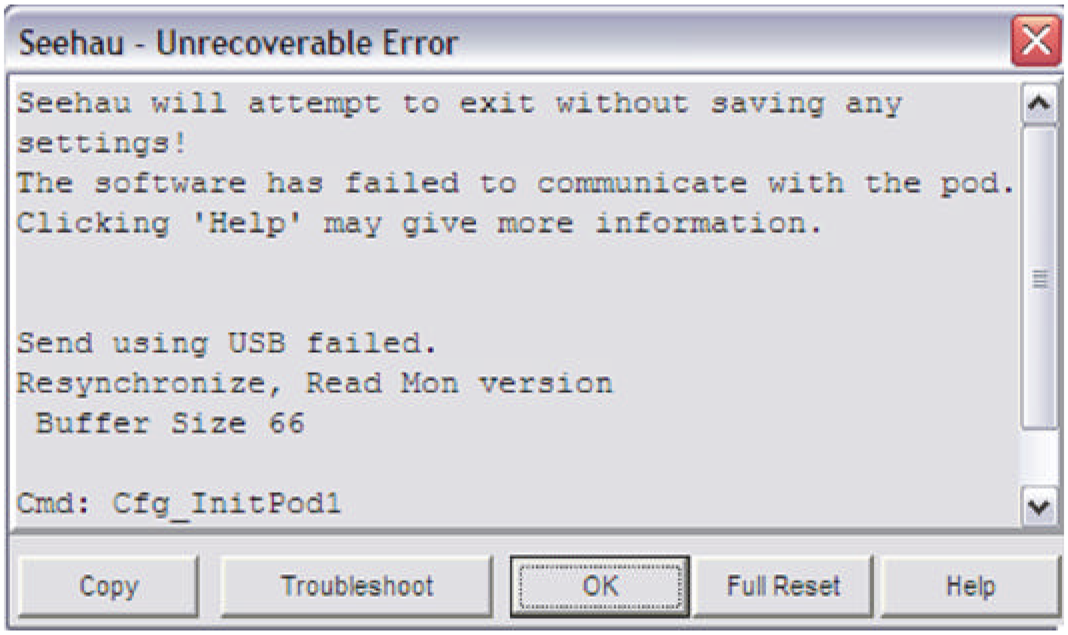
WARNING

When powering up, connect the USB cable and then power on the target.

ERROR CONDITION: If you ever note that the led indicator (Blue or Green) on the Pod is continuously on when you are not running Seehau or when you are not actively interacting with the target, it is an error condition likely caused by an erroneous power sequence. To correct the situation, power down the target then disconnect the USB cable to the pod wait 20 seconds. Then reconnect the USB cable followed by power up of the board. If there has been no damage to the pod, the pod led indicator will be off.

JTAG TROUBLESHOOTER – WHEN SEEHAU DOES NOT START

After a period of about a minute of trying to load, one of several messages like the one below will appear. It is an error reporting that communication to the FPGA was not established with the POD.



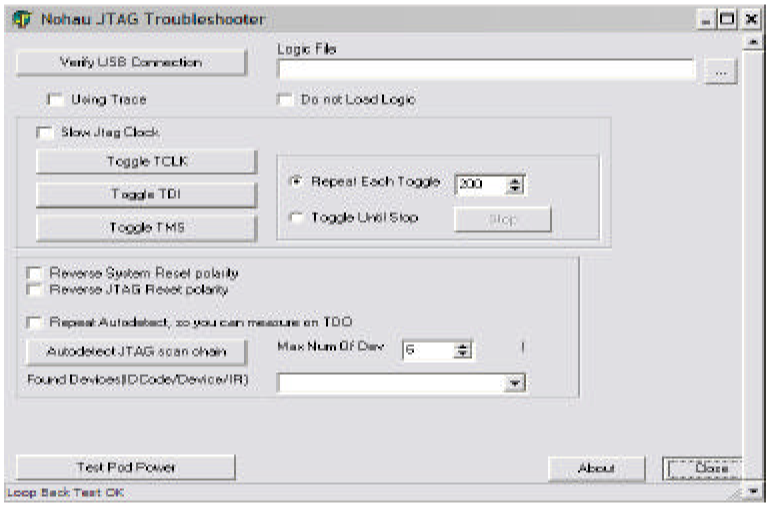
```
Seehau will attempt to exit without saving any
settings!
The software has failed to communicate with the pod.
Clicking 'Help' may give more information.


Send using USB failed.
Resynchronize, Read Mon version
 Buffer Size 66

Cmd: Cfg_InitPod1
```

Figure 17. Typical Error Message when Communication Fails

The message is a direct result of our not being sure what to specify in the JTAG scan chain. However on new boards, solder shorts or any number of conditions can cause this failure.

Start JTAG TROUBLESHOOTER

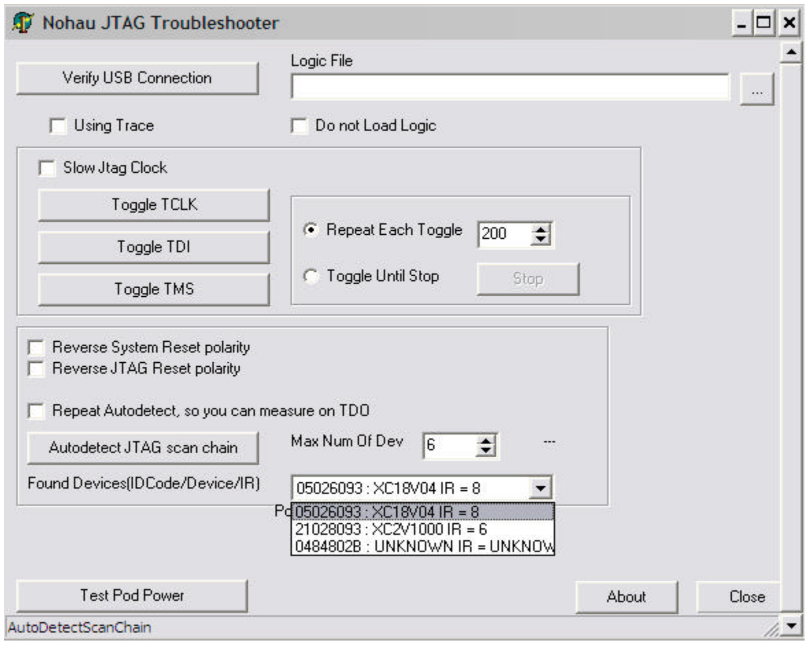
Select the **Troubleshoot** button to bring up the dialog box below:



Using Trace [ ]
Do not Load Logic [ ]
Slow Jtag Clock [ ]
Repeat Each Toggle [✓] 200
Toggle Until Stop [ ]
Reverse System Reset polarity [ ]
Reverse JTAG Reset polarity [ ]
Repeat Autodetect, so you can measure on TDO [ ]
Max Num Of Dev 6
Found Devices(IDCode/Device/IR)
Loop Back Test OK

Figure 18. TROUBLESHOOTER Control

1. Select **Verify USB** and note the result displayed on the bottom bar
2. Select **Toggle TCLK** 400 times and note result.
3. Select **Test Pod Power** and note result, which should be ok.
4. Select **Autodetect JTAG Scan Chain** and pull down the drop down box after it completes as shown in Figure 19.



Using Trace [ ]
Do not Load Logic [ ]
Slow Jtag Clock [ ]
Repeat Each Toggle [✓] 200
Toggle Until Stop [ ]
Reverse System Reset polarity [ ]
Reverse JTAG Reset polarity [ ]
Repeat Autodetect, so you can measure on TDO [ ]
Max Num Of Dev 6
Found Devices(IDCode/Device/IR) 05026093 : XC18V04 IR = 8

```
05026093 : XC18V04 IR = 8
21028093 : XC2V1000 IR = 6
0484802B : UNKNOWN IR = UNKNOW
```

AutoDetectScanChain

Figure 19. TROUBLESHOOTER Control Data

Take out a piece of paper and write down the device ID, the number of bits in the instruction register and their relative position in the scan chain. For unknown devices, you must look them up on your schematics and consult data sheets from manufactures to determine their Instruction Register Lengths.

Close the troubleshooter program and return to programs ->SeehauBlaze->config. Repeat the procedure to start the Seehau Configuration Program, only this time with the correct data.

If you have a problem with this, ask for help so it doesn't take up too much of your lab time.

Starting Seehau with the Right Data

First time from Config:

When you click **Finish** from **Config**, Seehau prompts you for a selection to start the emulator.

Select **Yes** and Seehau starts to load and program the FPGA configuration from the specified .bit file. When you exit Seehau the first time, you may choose to save settings which will place all settings in the **Startup.bas** file for easy configuration and entry as described next.

When loading and programming is complete, the Seehau main window opens (Figure 10).

Seehau will load its configuration from the settings specified in the **Config** program.

As shown in Figure 10, the initial Seehau screen defaults to address 400 in the source display, all registers are displayed in the window at right with the PC set to 400 by default. A data window is shown at the bottom with the memory contents at address 400. When a program is loaded, it will be loaded at the address specified in the .elf file from the compiler and the initial PC in the debugger will be set to the value specified in the **Config emulator-> Misc Setup** menu (which we have not covered yet).

8 Using MicroBlaze Trace

The EMUL-MICROBLAZE-PC provides a 512 or 2K deep trace with a trigger, post trigger count and break control. Probe Pins may be either 8 or 40-bits wide. It will display data connected to it as specified in the MHS file of Xilinx Platform Studio (XPS).

The Trace is triggered by an event specified in the **Event Configuration** window which is selected from the **Config** menu by selecting **Trace**. The **Event Configuration** window opens displaying the Events tab. (Figure 22).

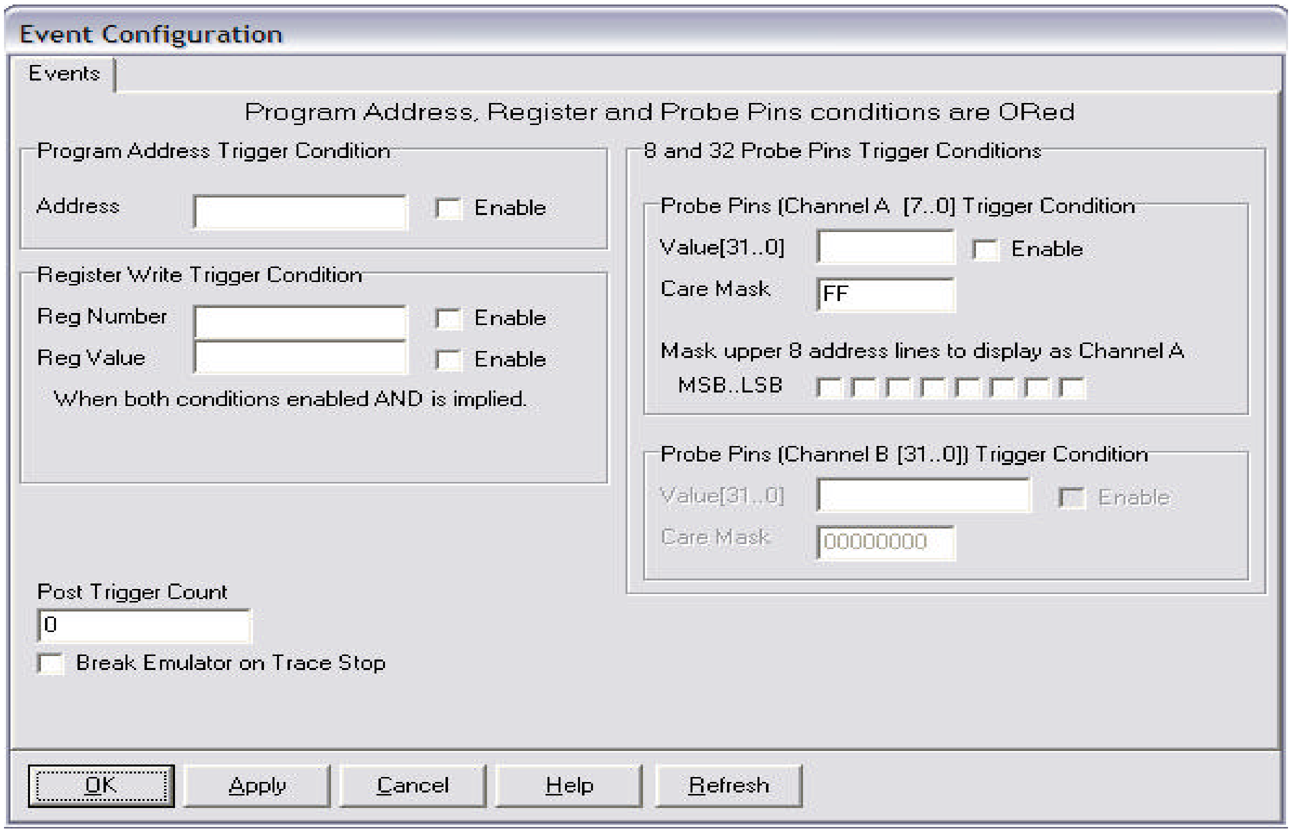

Event Configuration
Events
Program Address, Register and Probe Pins conditions are ORed
Program Address Trigger Condition
Address
Enable
Register Write Trigger Condition
Reg Number
Enable
Reg Value
Enable
When both conditions enabled AND is implied.
8 and 32 Probe Pins Trigger Conditions
Probe Pins (Channel A [7..0] Trigger Condition
Value[31..0]
Enable
Care Mask
FF
Mask upper 8 address lines to display as Channel A
MSB..LSB
Probe Pins (Channel B [31..0]) Trigger Condition
Value[31..0]
Enable
Care Mask
00000000
Post Trigger Count
0
Break Emulator on Trace Stop
OK
Apply
Cancel
Help
Refresh

Figure 22. Event Configuration Window Displaying the Events Tab

Events Tab

Events to trigger the trace may be specified on the basis of execution addresses, register accesses "anded" with the register value, or data values sensed by the "logic probe". The logic probe is an abstraction used to refer to signals connected to the Nohau debug core as specified in the MHS file of your project. The analogy to the probe pins of a logic analyzer is complete and Seehau refers to "probe pins" as if they were physically arranged in a row, even though they are actually signals in the FPGA fabric connected to the Nohau debug IP.

Trigger Conditions

On the display, trigger conditions that are specified in separate outline “gray boxes” on the display are logically or’ed together and conditions within the same box are logically anded together. Some examples will be given later to help clarify the various conditions.

Program Address Trigger Condition

The value in the **Address** is the value that trigger is set to.

The **Enable check box** makes the trigger active.

If only a simple address is specified, the occurrence of that address will trigger the trace. If for example a register write trigger condition is specified also, then an “or” condition is implied which results in a trigger from the condition that occurs first.

Register Write Trigger Condition

Register Number is a number from 0 to 31 representing the 32 registers in MicroBlaze.

Register Value is a 32-bit Hex value contained in a register.

If both values are specified and enabled, an “and” is implied which means the trigger will happen when that specific register is written with the value specified.

Probe Pins Channel A [7..0] Trigger Condition

Value [31 .. 0] is any 32-bit data value containing the 8 bit value we wish to trigger on.

Care Mask is an 8 bit mask that allows you to specify which of the 8 bit in the 32-bit word you wish to trigger on.

Mask upper address lines to display is a mask to specify if we want to use the upper 8 bits of the address field of the internal BRAM for the 8 bit probe. This is a mechanism to conserve BRAM on board the FPGA and is normally not used.

For example, if the value is specified as 12345678 and the care mask is 000000FF, then the trigger will be on the value 78 in the most significant bits of the data. For a mask of 0000FF00, the value would be 56 . Remember MicroBlaze is a Big Endian machine.

Probe Pins Channel B[31..0] Trigger Condition

These pins are optionally present because the additional channels require more on board BRAM for operation. It requires 4 on board ram for the trace but results in a trace that is 40-bits wide and 2K deep. It is included in a project by specifying the 32-bit version of the Nohau IP in the XPS pcores directory of your project. This is normally accomplished by simply renaming the debugtraceblaze file and the debugtraceblaze32 file. Whenever the BRAM is available, it is recommended that the additional 32-bit probe be included.

Value[31..0] is any 32-bit data value you would like to trigger on when enabled.

Care Mask is a 32-bit mask specified in hexadecimal, where a 1 implies a care bit and a 0 a don't care. This provides a powerful capability to trigger on a single bit or combination of bits in a 40-bit field.

If both 8 bit and 32-bit conditions are specified, then there is an implied "and" which means all 40-bits must be true to cause a trigger.

Post Trigger Count

The frames to capture after the trigger condition has occurred are specified here. When the Post Trigger Count is zero, the trace will stop. When the post trigger count is 100, you will record an additional 100 frames after the trigger.

Break on Trace Stop

When checked, will cause emulation to break when the trace has stopped. This breakpoint has the additional feature of utilizing on-chip hardware breakpoints, which allows operation of trace breakpoints in ROM or Flash sections of memory.

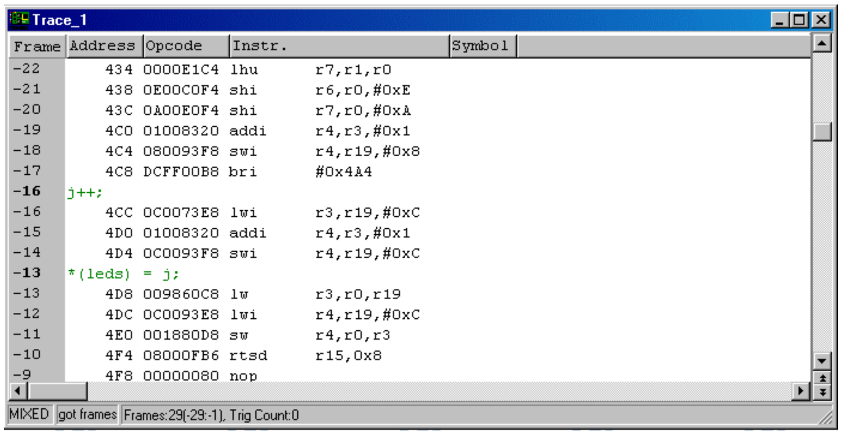
Buttons Common to All Tabs

- **OK**: Saves the settings for the tab and exits the dialog box.
- **Apply**: Saves the settings for the tab.
- **Cancel**: Exits without saving the settings for the dialog box.
- **Help**: Displays the Seehau Help file.
- **Refresh**: Allows you to retrieve and view the current event configuration settings.

Trace Display Window

The **Trace Display** window is used to display data from trace capture. The default fields displayed from left to right are the **Frame, Address, Opcode, Instruction** and **Symbol**. The display can be activated by clicking on the blue button marked **TR** on the upper tool bar or via the **New** menu by selecting **New Trace Window**. There is no theoretical limit to the number of windows allowed.

The **Trace Display** window can display trace only (assembly), mixed (source and assembly) or source only. The mixed mode is shown in Figure 23.



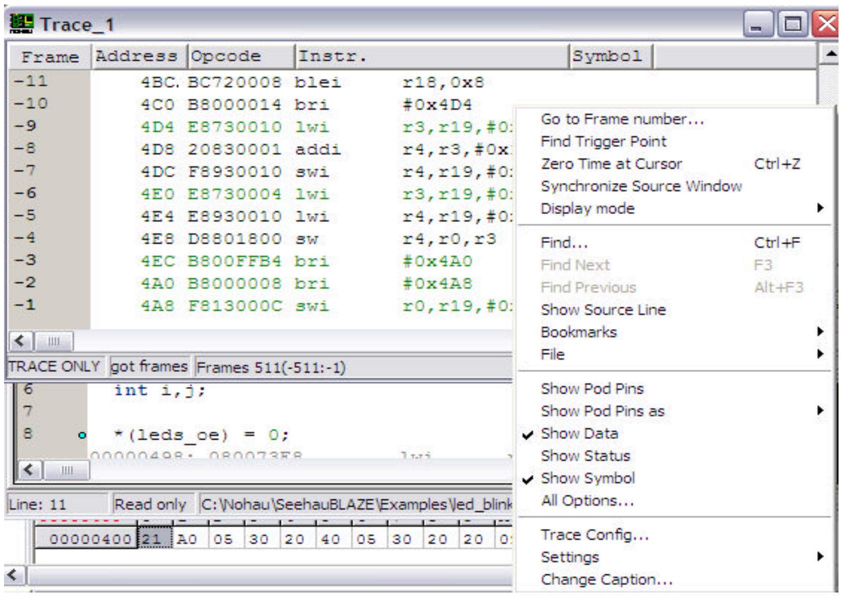
```
Trace_1
Frame Address Opcode   Instr.            Symbol
-22       434 0000E1C4 lhu      r7,r1,r0
-21       438 0E00C0F4 shi      r6,r0,#0xE
-20       43C 0A00E0F4 shi      r7,r0,#0xA
-19       4C0 01008320 addi     r4,r3,#0x1
-18       4C4 080093F8 swi      r4,r19,#0x8
-17       4C8 DCFF00B8 bri      #0x4A4
-16  j++;
-16       4CC 0C0073E8 lwi      r3,r19,#0xC
-15       4D0 01008320 addi     r4,r3,#0x1
-14       4D4 0C0093F8 swi      r4,r19,#0xC
-13  *(leds) = j;
-13       4D8 009860C8 lw       r3,r0,r19
-12       4DC 0C0093E8 lwi      r4,r19,#0xC
-11       4E0 001880D8 sw       r4,r0,r3
-10       4F4 08000FB6 rtsd     r15,0x8
-9        4F8 00000080 nop
MIXED  got frames  Frames:29(-29:-1), Trig Count:0
```

Figure 23. Trace Display in Mixed Mode

Trace display options are displayed by right clicking in the trace window to yield the display in Figure 24 below.

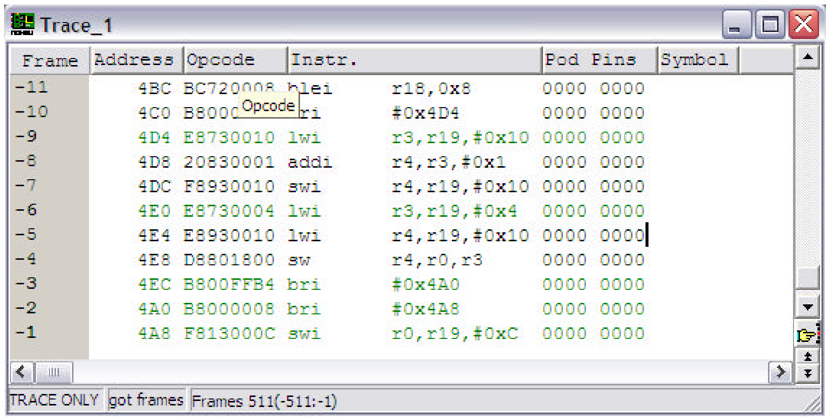
In the options list, select **Show Pod Pins** to display the probe signals in the next column of the display as shown for an 8-bit probe in Figure 24 below.

In the options list, select **Show Pod Pins** as to display the probe signals in the next column of the display as shown for a 32-bit probe in Figure 25 below. You may choose to display in Hex and the MSB first or LSB first.



```
Trace_1
Frame Address Opcode   Instr.          Symbol
-11       4BC. BC720008 blei  r18,0x8
-10       4C0 B8000014 bri   #0x4D4
-9        4D4 E8730010 lwi   r3,r19,#0:
-8        4D8 20830001 addi  r4,r3,#0x
-7        4DC F8930010 swi   r4,r19,#0:
-6        4E0 E8730004 lwi   r3,r19,#0:
-5        4E4 E8930010 lwi   r4,r19,#0:
-4        4E8 D8801800 sw    r4,r0,r3
-3        4EC B800FFB4 bri   #0x4A0
-2        4A0 B8000008 bri   #0x4A8
-1        4A8 F813000C swi   r0,r19,#0:
TRACE ONLY  got frames  Frames 511(-511:-1)

6        int i,j;
7
8    o   *(leds_oe) = 0;
   00000498: 080073F8     lwi
Line: 11  Read only  C:\Nohau\SeehauBLAZE\Examples\led_blink
00000400 21 A0 05 30 20 40 05 30 20 20 0
```

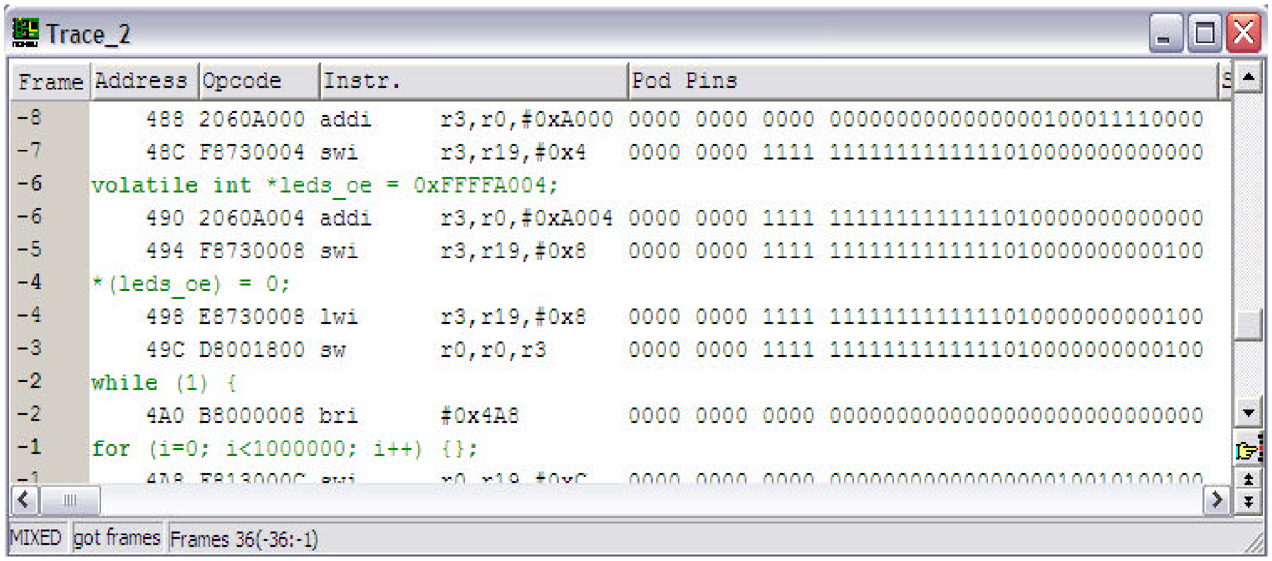
Figure 24. Trace Display Pop-Up Window



```
Frame Address Opcode   Instr.                   Pod Pins   Symbol
-11       4BC BC720008 blei     r18,0x8         0000 0000
-10       4C0 B8000    ri       #0x4D4          0000 0000
-9        4D4 E8730010 lwi      r3,r19,#0x10    0000 0000
-8        4D8 20830001 addi     r4,r3,#0x1      0000 0000
-7        4DC F8930010 swi      r4,r19,#0x10    0000 0000
-6        4E0 E8730004 lwi      r3,r19,#0x4     0000 0000
-5        4E4 E8930010 lwi      r4,r19,#0x10    0000 0000
-4        4E8 D8801800 sw       r4,r0,r3        0000 0000
-3        4EC B800FFB4 bri      #0x4A0          0000 0000
-2        4A0 B8000008 bri      #0x4A8          0000 0000
-1        4A8 F813000C swi      r0,r19,#0xC     0000 0000
```

TRACE ONLY | got frames | Frames 511(-511:-1)

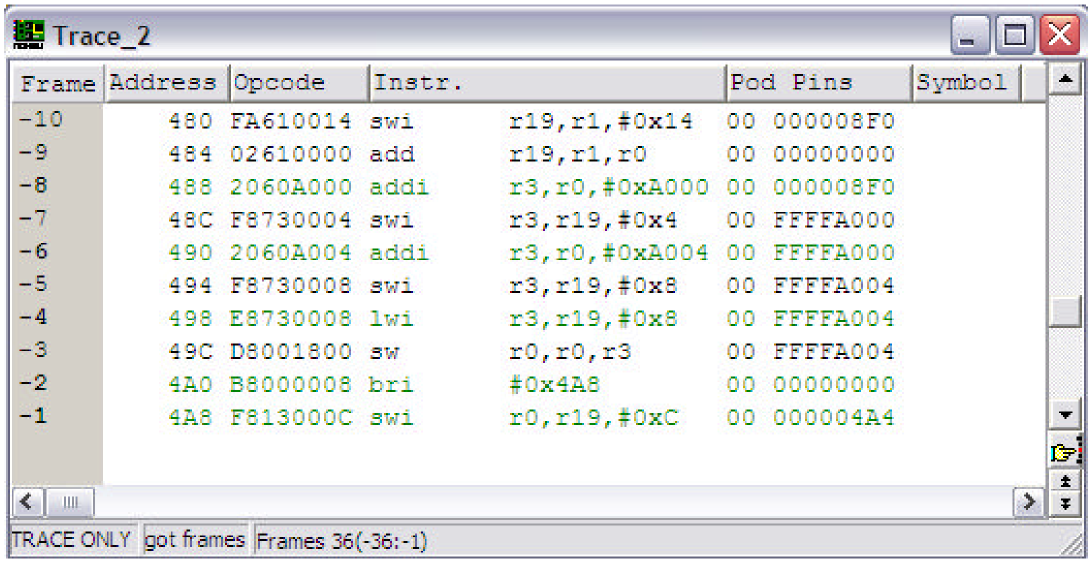
Figure 25. Trace Display with 8 bit Probe in Binary



```
Frame Address Opcode   Instr.                 Pod Pins
-8        488 2060A000 addi     r3,r0,#0xA000 0000 0000 0000 0000000000000000100011110000
-7        48C F8730004 swi      r3,r19,#0x4   0000 0000 1111 1111111111111010000000000000
-6    volatile int *leds_oe = 0xFFFFA004;
-6        490 2060A004 addi     r3,r0,#0xA004 0000 0000 1111 1111111111111010000000000000
-5        494 F8730008 swi      r3,r19,#0x8   0000 0000 1111 1111111111111010000000000100
-4    *(leds_oe) = 0;
-4        498 E8730008 lwi      r3,r19,#0x8   0000 0000 1111 1111111111111010000000000100
-3        49C D8001800 sw       r0,r0,r3      0000 0000 1111 1111111111111010000000000100
-2    while (1) {
-2        4A0 B8000008 bri      #0x4A8        0000 0000 0000 0000000000000000000000000000
-1    for (i=0; i<1000000; i++) {};
-1        4A8 F813000C swi      r0,r19,#0xC   0000 0000 0000 0000000000000000001001010100
```

MIXED | got frames | Frames 36(-36:-1)

Figure 26. Trace 40 bit Mixed Mode Display in Binary



Frame	Address	Opcode	Instr.		Pod Pins		Symbol
-10	480	FA610014	swi	r19,r1,#0x14	00	000008F0	
-9	484	02610000	add	r19,r1,r0	00	00000000	
-8	488	2060A000	addi	r3,r0,#0xA000	00	000008F0	
-7	48C	F8730004	swi	r3,r19,#0x4	00	FFFFA000	
-6	490	2060A004	addi	r3,r0,#0xA004	00	FFFFA000	
-5	494	F8730008	swi	r3,r19,#0x8	00	FFFFA004	
-4	498	E8730008	lwi	r3,r19,#0x8	00	FFFFA004	
-3	49C	D8001800	sw	r0,r0,r3	00	FFFFA004	
-2	4A0	B8000008	bri	#0x4A8	00	00000000	
-1	4A8	F813000C	swi	r0,r19,#0xC	00	000004A4	

TRACE ONLY | got frames | Frames 36(-36:-1)

Figure 27. Trace 40 bit Display in Hexadecimal

9 BlazeGen for EDK 6.2 Projects

BlazeGen

BlazeGen is a platform generator tool designed to accomplish two goals:

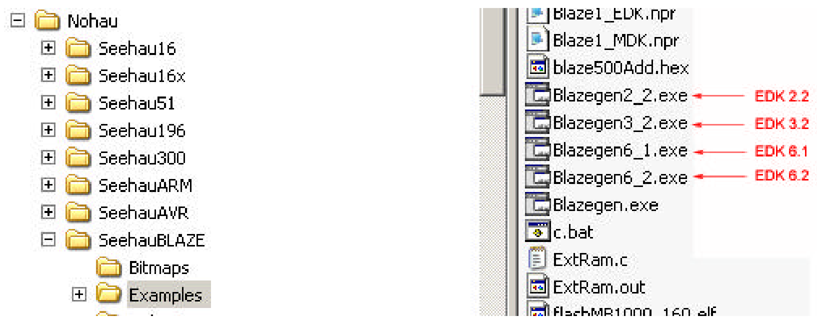
1. Build small known good projects for use in board power up and getting started exercises.
2. Adding Nohau debug tools to existing projects without hand editing numerous text files and re-compiling numerous times. A procedure is provided in the next chapter to simplify the process.

The procedure for building a project with BlazeGen will be covered here. For using Base System Builder, see the procedure in the next section.

Invoke BlazeGen by Double-Clicking the correct .exe File in the Nohau Examples Directory

For 6.2 select BlazeGen6_2.exe.

The application is located under the installed SeehauBlaze directory. Normally located on the hard disk at c:\nohau\seehaublaze\examples (Figure 28). You will need to select the correct version for the PLATFORM STUDIO version that you are using.



```
⊟ Nohau
   ⊞ Seehau16
   ⊞ Seehau16x
   ⊞ Seehau51
   ⊞ Seehau196
   ⊞ Seehau300
   ⊞ SeehauARM
   ⊞ SeehauAVR
   ⊟ SeehauBLAZE
        Bitmaps
      ⊞ Examples
```

```
Blaze1_EDK.npr
Blaze1_MDK.npr
blaze500Add.hex
Blazegen2_2.exe  ←—— EDK 2.2
Blazegen3_2.exe  ←—— EDK 3.2
Blazegen6_1.exe  ←—— EDK 6.1
Blazegen6_2.exe  ←—— EDK 6.2
Blazegen.exe
c.bat
ExtRam.c
ExtRam.out
flashMB1000_160.elf
```

Figure 28. Directory Location

Then, use the **Edit -> CoresetUp** option in BlazeGen to begin your build (Figure 29).

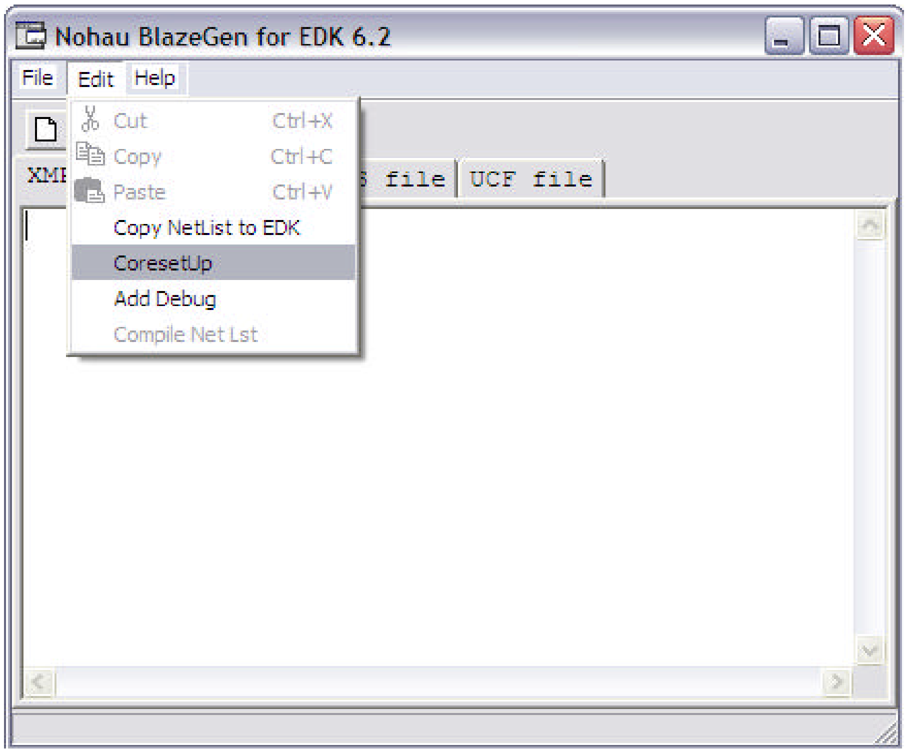
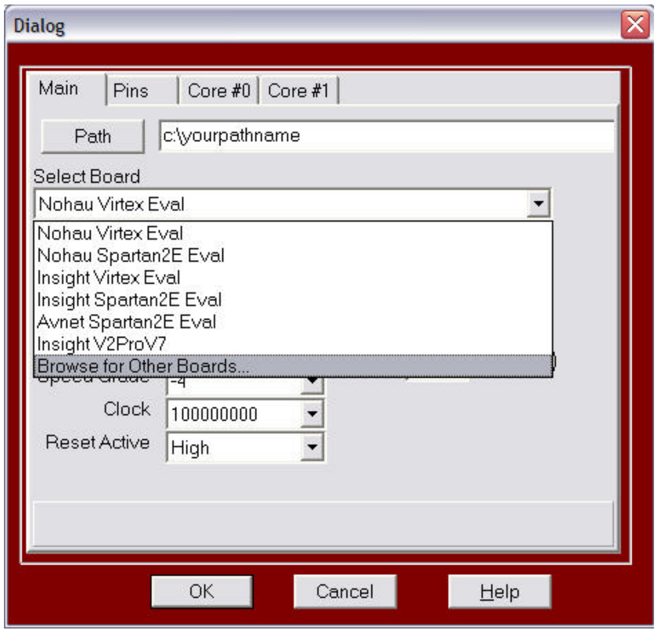


Figure 29. Core Setup via BlazeGen

From the dialog in Figure 30, select as follows for your first small project.



Path: c:\yourpathname

Select Board: Nohau Virtex Eval

- Nohau Virtex Eval
- Nohau Spartan2E Eval
- Insight Virtex Eval
- Insight Spartan2E Eval
- Avnet Spartan2E Eval
- Insight V2ProV7
- Browse for Other Boards...

Speed Grade: -4

Clock: 100000000

Reset Active: High

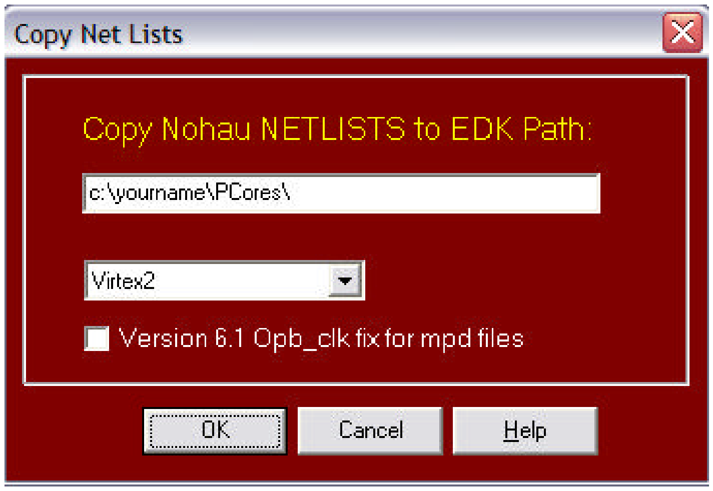
Figure 30. BlazeGen Main Setup

Enter a path for your directory or a new one 'yourname' as shown. Select your board from the list or browse to a list of all boards supported in EDK. For this example,
EDK -> Board ->Memec_Design ->Boards -> Memec_Design_V2P4
_FG456_Rev4_P160_Comm.

Family	Virtex2P
Device	XC2VP4
Speed	Grade 6
Clock	100Mhz
Reset	Active LOW
Cores	1

Select OK

Figure 31 below will pop up.


Copy Net Lists

Copy Nohau NETLISTS to EDK Path:

c:\yourname\PCores\

Virtex2

Version 6.1 Opb_clk fix for mpd files

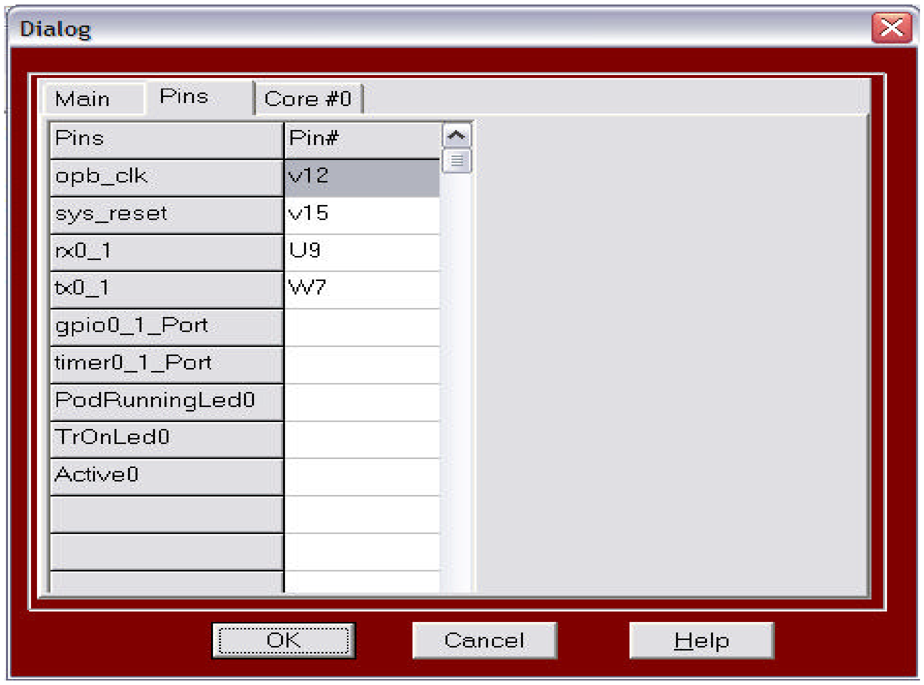
Figure 31. Netlist Direction

Select **Virtex2** for the family and leave PCores in yourname directory.
The checkbox is an artifact when upgrading older files to 6.1.

Click on **OK**.

Re-enter **Coresetup** (**Edit ->coresetup**)
The purpose is to allow BlazeGen to determine pinouts on the FPGA and options on the board.

Select the **Pins** tab to produce the display of Figure 32 below.

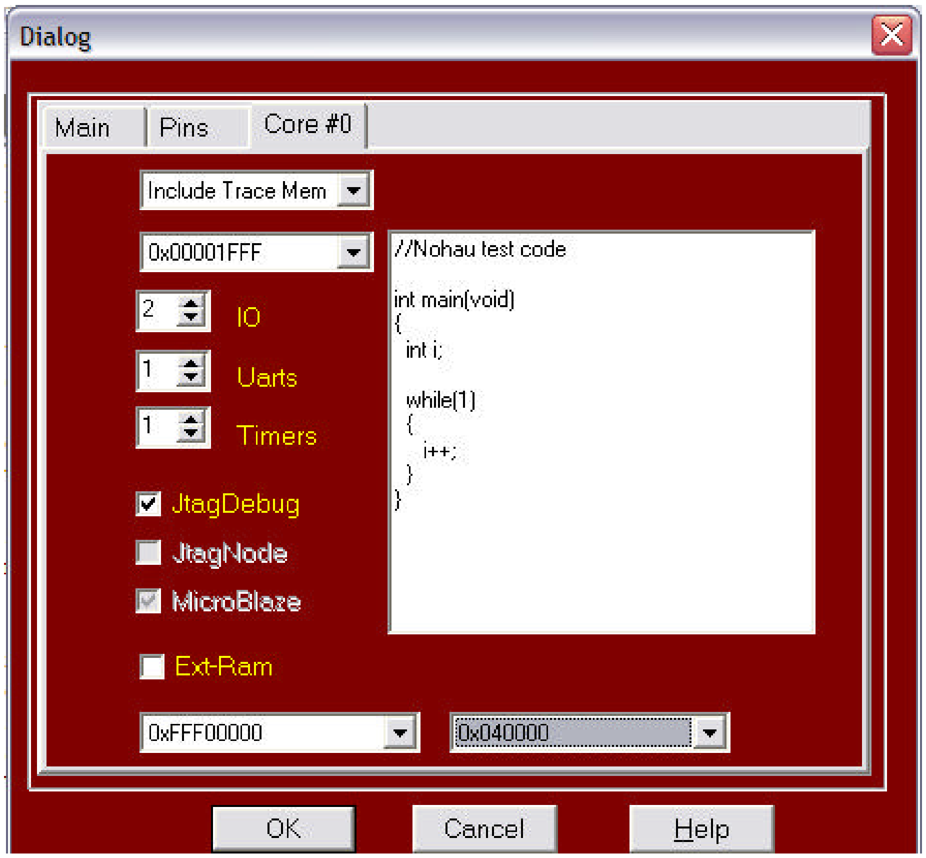


Pins	Pin#
opb_clk	v12
sys_reset	v15
rx0_1	U9
tx0_1	W7
gpio0_1_Port	
timer0_1_Port	
PodRunningLed0	
TrOnLed0	
Active0	

Figure 32. Minimum Pinouts

The Pins display in Figure 32 specifies the minimum set of pins to run the board specified. In this case, only the opb_clk, sys_reset, rx0_1 and tx0_1 need be specified with the FPGA pin location shown. For all of these simple systems, these are the only pins included. For more complex systems you may use Base System Builder as described in the next section or add additional pins manually using a text editor such as Wordpad.

Select the TAB labeled Core #0 to produce the display of Figure 33.



Include Trace Mem

0x00001FFF

2 IO

1 Uarts

1 Timers

[✓] JtagDebug

[ ] JtagNode

[✓] MicroBlaze

[ ] Ext-Ram

0xFFF00000

0x040000

```
//Nohau test code

int main(void)
{
  int i;

  while(1)
  {
    i++;
  }
}
```

Figure 33. FPGA Simple Core Specification

Select as follows:

Include Trace Memory

Memory 0x00001FFF specifies the end address of BRAM used in trace

Specify 2 GPIO IP – any number 0- 10 allowed

Specify 1 UART IP– any number 0- 10 allowed

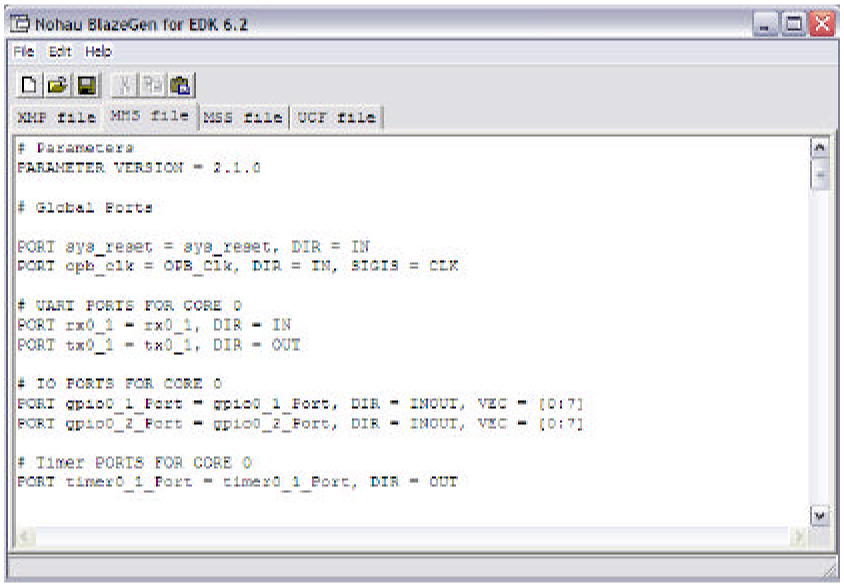
Specify 1 timer– any number 0- 10 allowed

Specify JTAG Debug

Do not specify external RAM nor the RAM start address and Length. On newer systems specifica-tion of more complex systems should be done using the procedure using Base System Builder, which is described below.

The code snippet shown in the List box is fixed and will be produced with each BlazeGen target.

Select OK to produce the display in Figure 34.



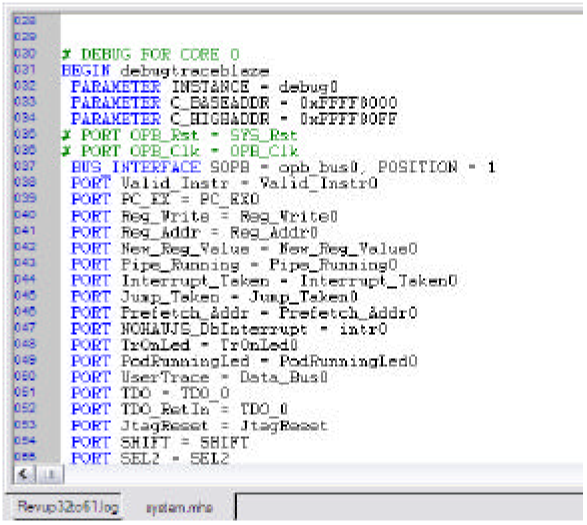
```
# Parameters
PARAMETER VERSION = 2.1.0

# Global Ports

PORT sys_reset = sys_reset, DIR = IN
PORT opb_clk = OPB_Clk, DIR = IN, SIGIS = CLK

# UART PORTS FOR CORE 0
PORT rx0_1 = rx0_1, DIR = IN
PORT tx0_1 = tx0_1, DIR = OUT

# IO PORTS FOR CORE 0
PORT gpio0_1_Port = gpio0_1_Port, DIR = INOUT, VEC = [0:7]
PORT gpio0_2_Port = gpio0_2_Port, DIR = INOUT, VEC = [0:7]

# Timer PORTS FOR CORE 0
PORT timer0_1_Port = timer0_1_Port, DIR = OUT
```

Figure 34. XMP, MHS, MSS, UCF Files Produced by BlazeGen

At this point, BlazeGen has placed all 4 of the files produced in the yourname directory, and BlazeGen may be closed. The next step is to build the project specified in the system.xps file in yourname directory.

In the yourname directory, examine the PCORES directory and observe BlazeGen has placed the Nohau debug core there for use during platform building. Also, a code folder has been created that contains the code snippet called main.c, which was seen above in BlazeGen.

Under **Windows Explorer**, double-click the **system.xmp** file which will launch XPS and load the project. The project files to become familiar with them but do not change them. On entry a bunch or error and revup error messages are displayed. Ignore them for now. Select and display the system.mhs file and scroll down to debugtraceblaze core to display Figure 35.



```
# DEBUG FOR CORE 0
BEGIN debugtraceblaze
 PARAMETER INSTANCE = debug0
 PARAMETER C_BASEADDR = 0xFFFF8000
 PARAMETER C_HIGHADDR = 0xFFFF80FF
# PORT OPB_Rst = SYS_Rst
# PORT OPB_Clk = OPB_Clk
 BUS_INTERFACE SOPB = opb_bus0, POSITION = 1
 PORT Valid_Instr = Valid_Instr0
 PORT PC_EX = PC_EX0
 PORT Reg_Write = Reg_Write0
 PORT Reg_Addr = Reg_Addr0
 PORT New_Reg_Value = New_Reg_Value0
 PORT Pipe_Running = Pipe_Running0
 PORT Interrupt_Taken = Interrupt_Taken0
 PORT Jump_Taken = Jump_Taken0
 PORT Prefetch_Addr = Prefetch_Addr0
 PORT NOHAUJS_DbInterrupt = intr0
 PORT TrOnLed = TrOnLed0
 PORT PodRunningLed = PodRunningLed0
 PORT UserTrace = Data_Bus0
 PORT TDO = TDO_0
 PORT TDO_RetIn = TDO_0
 PORT JtagReset = JtagReset
 PORT SHIFT = SHIFT
 PORT SEL2 = SEL2
```

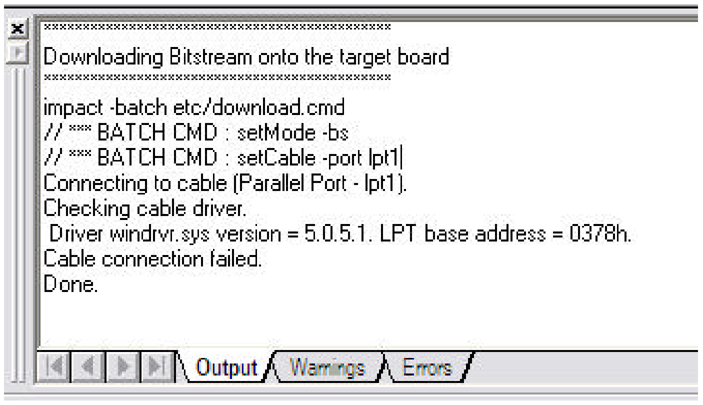
Figure 35. System.mhs Showing Debug Core

In **XPS** select **Tools -> clean -> all** to clear out older files.

Then select **Tools -> Download** and wait several minutes for completion. Monitor the console window for errors. The build will produce the download.bit file under the implementation directory in yourname directory. For more information on the build under XPS, please see the Xilinx EDK DOC directory.

Note the project could be built one section at a time throughout place and route and net-list creation. Selecting the **Download** button or from the **Tools** menu is a fast shortcut.

After a successful build of a .bit file, an error message is shown in the console as illustrated in Figure 36.



```
*********************************************
Downloading Bitstream onto the target board
*********************************************
impact -batch etc/download.cmd
// *** BATCH CMD : setMode -bs
// *** BATCH CMD : setCable -port lpt1
Connecting to cable (Parallel Port - lpt1).
Checking cable driver.
 Driver windrvr.sys version = 5.0.5.1. LPT base address = 0378h.
Cable connection failed.
Done.
```

Figure 36. Loaded "Normal" Error on Successful Completions

This error message appears since the download command assumes a file will be built that can be loaded by a Xilinx tool. Because no target Xilinx device is connected, the load fails; however the correct file is built and stored safely away.

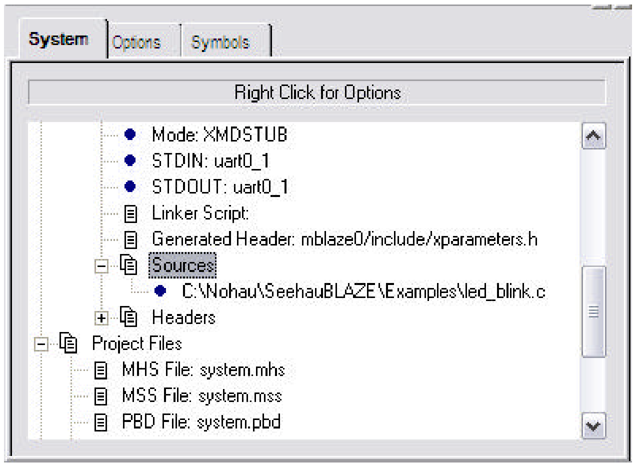
The .bit file may be loaded into Seehau to program your FPGA as described above.

Compiling Programs from XPS for use with Nohau Tools

Now that we have successfully built a small project in XPS, the elements of compiling Source programs for execution and producing the .elf files can be described. To prepare a program file and compile it, the following steps are required:

- Prepare source files under an external editor or under XPS
- Add the source files to the project by adding them to the XPS project
- Specify compile with XMD-STUB
- Output directories and filenames

- Linker Script (if Any)
- Under the System Tab on the left of the XPS display scroll down to cpu-Mblaze and to the sources element. Right click sources and select ADD. Select the file led_blink.c under the No-hau Examples directory.


System | Options | Symbols

Right Click for Options

```
        ● Mode: XMDSTUB
        ● STDIN: uart0_1
        ● STDOUT: uart0_1
        Linker Script:
        Generated Header: mblaze0/include/xparameters.h
      ⊟ Sources
          ● C:\Nohau\SeehauBLAZE\Examples\led_blink.c
      ⊞ Headers
  ⊟ Project Files
      MHS File: system.mhs
      MSS File: system.mss
      PBD File: system.pbd
```

Figure 37. Adding source files to a project

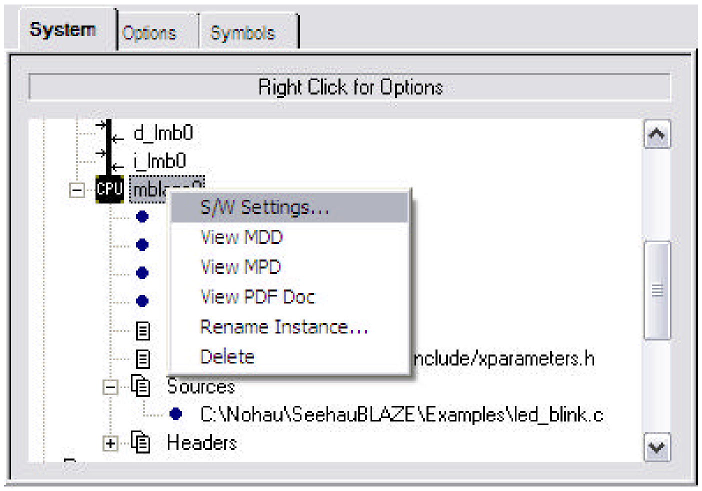
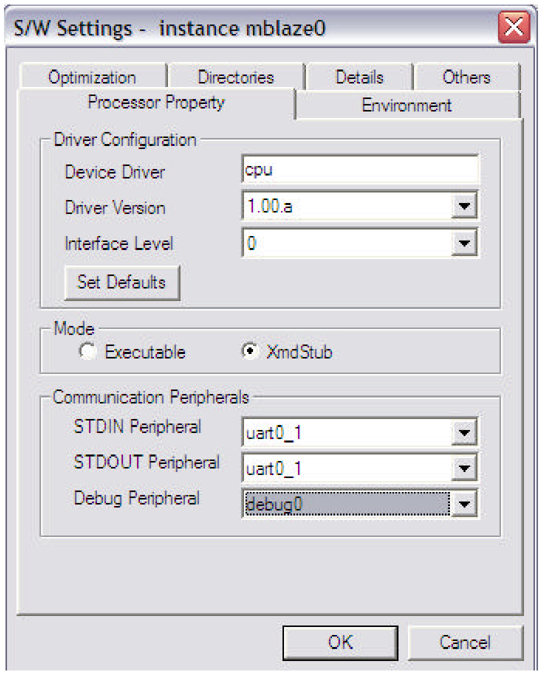


Figure 38. Select Software Settings

Select compiler options by right-clicking on the MicroBlaze instance as shown below, and selecting S/W Settings (Figure 38):


S/W Settings - instance mblaze0

Driver Configuration
Device Driver: cpu
Driver Version: 1.00.a
Interface Level: 0

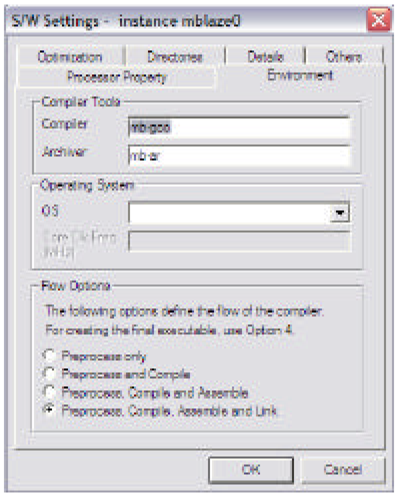
Mode
Executable [✓ XmdStub]

Communication Peripherals
STDIN Peripheral: uart0_1
STDOUT Peripheral: uart0_1
Debug Peripheral: debug0

Figure 39. Software Settings Dialog

The defaults shown for Processor properties are OK. Be sure to select compile with XMD-STUB, which is a Xilinx-generated small monitor that resides in the first 1K of MB memory space. The STDIN and STDOUT provide a path for the compiler to send messages to the UART specified. Also, be sure when compiling for Nohau operation that the debug peripheral is debug0 and it matches the instance of the debugtraceblaze core in your MHS file.

Figure 40 shows the compiler specification. Since GNU is the only compiler available it is static for the present time.


S/W Settings - instance mblaze0

Compiler Tools
Compiler: mb-gcc
Archiver: mb-ar

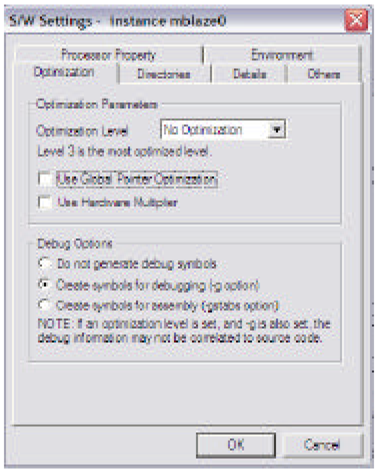
Operating System
OS:

Flow Options
The following options define the flow of the compiler.
For creating the final executable, use Option 4.

Preprocess only
Preprocess and Compile
Preprocess, Compile and Assemble
[✓ Preprocess, Compile, Assemble and Link]

Figure 40. Compiler Environment

Figure 41 shows the **Optimization** dialog. Optimization should be turned completely off and debug symbols turned on with the -g option.



S/W Settings - instance mblaze0

Optimization Parameters

Optimization Level: No Optimization

Level 3 is the most optimized level.

☐ Use Global Pointer Optimization

☐ Use Hardware Multiplier

Debug Options

○ Do not generate debug symbols

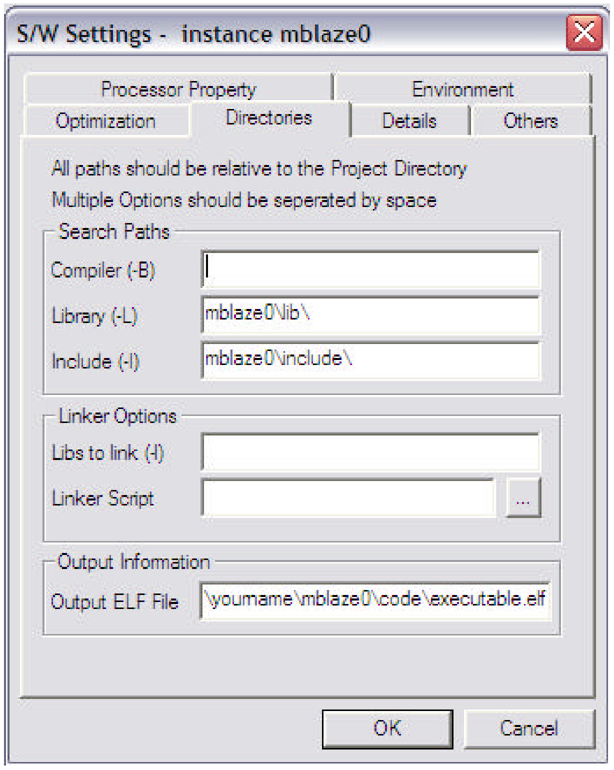
● Create symbols for debugging (-g option)

○ Create symbols for assembly (-gstabs option)

NOTE: If an optimization level is set, and -g is also set, the debug information may not be correlated to source code.

Figure 41. Optimization Dialog

Figure 42 shows the **Directories** specifications. See the XPS documents for details. The important thing to note is that the library and include paths must be correct to find referenced sources.



S/W Settings - instance mblaze0

All paths should be relative to the Project Directory

Multiple Options should be seperated by space

Search Paths

Compiler (-B):

Library (-L): mblaze0\lib\

Include (-I): mblaze0\include\

Linker Options

Libs to link (-l):

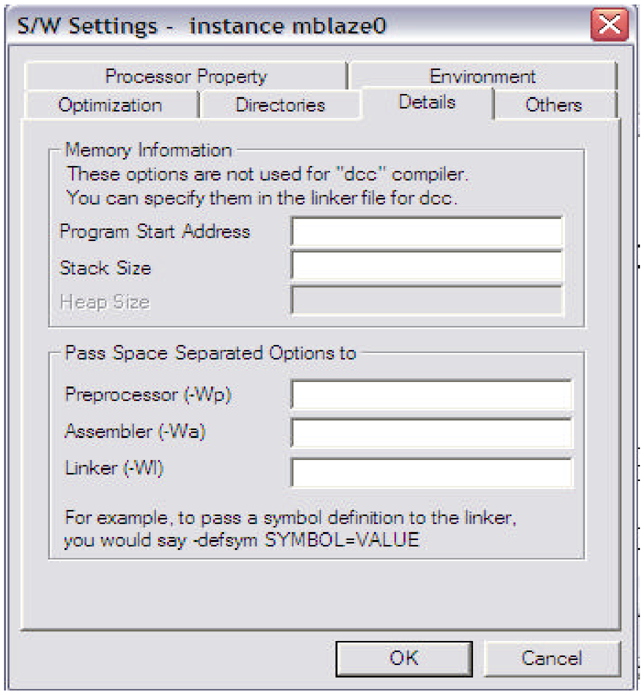
Linker Script:

Output Information

Output ELF File: \yourname\mblaze0\code\executable.elf

Figure 42. Directory Specifications for Output Files

The output .elf file is specified as shown in Figure 42. By convention, it is normally called executable .elf and located in the MBlaze directory, but you may choose to put it anywhere.



Memory Information
These options are not used for "dcc" compiler.
You can specify them in the linker file for dcc.

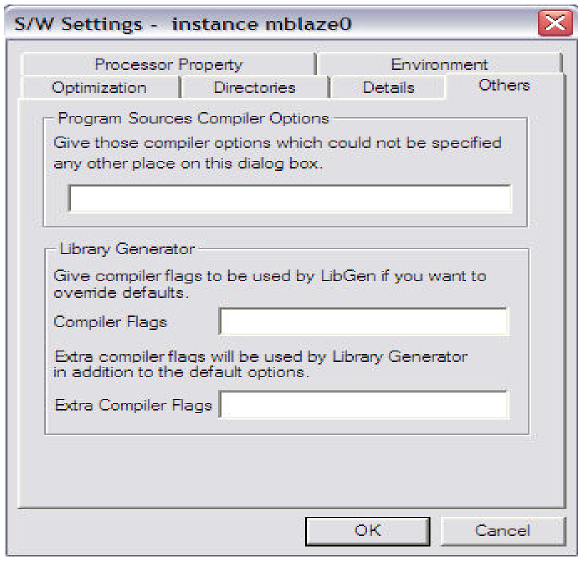
Program Start Address
Stack Size
Heap Size

Pass Space Separated Options to
Preprocessor (-Wp)
Assembler (-Wa)
Linker (-Wl)

For example, to pass a symbol definition to the linker,
you would say -defsym SYMBOL=VALUE

Figure 43. Start Address and Pass Specifications

Figure 43 shows the **Detail** specifications for the compiler and linker. The important point to note is that the program start address is specified here. Enter d100000 for a compile to be run at d100000h. Linker scripts may be specified here to scatter programs and data into desired memory MAPs.



Program Sources Compiler Options
Give those compiler options which could not be specified any other place on this dialog box.

Library Generator
Give compiler flags to be used by LibGen if you want to override defaults.
Compiler Flags
Extra compiler flags will be used by Library Generator in addition to the default options.
Extra Compiler Flags

Figure 44. Specifying Extras

The final compiler tab, **Other**, is a catch-all to specify other program generation options and is included for completeness. It is used when special compiler considerations must be specified for special drivers such as for the eMAC cores.

10 Using BlazeGen to Added DebugTrace Module

Working with Platform Studio 6.1 & BlazeGen

Make a Quick Project Using Xilinx's Platform Studio and Nohau's BlazeGen

Start the Xilinx **Platform Studio** program. Select the option **File | New Project | Base System Builder** (figure 45).

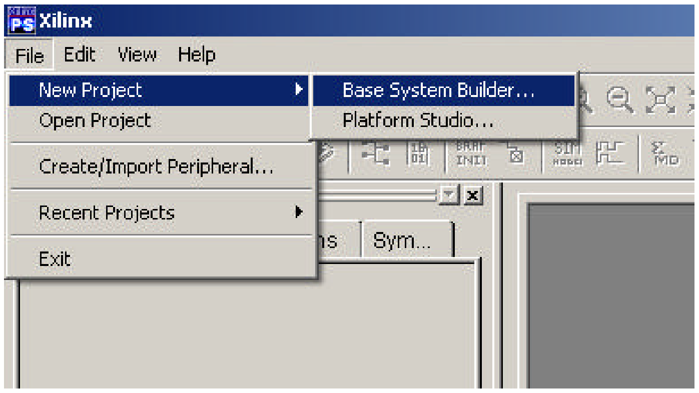
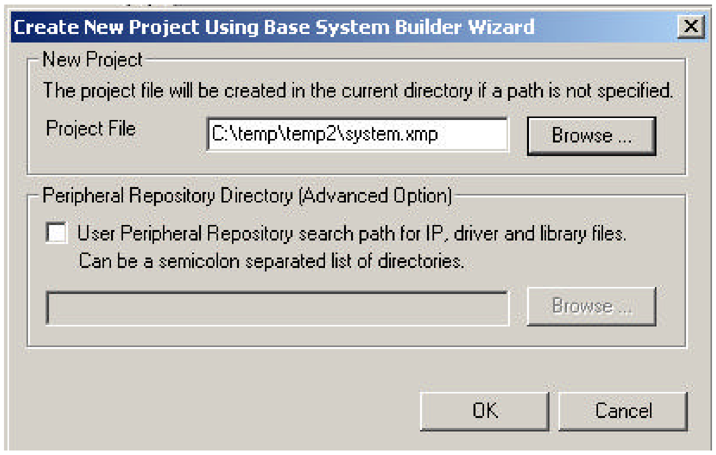


Figure 45. Xilinx Platform Studio

The **Create New Project** dialog box should appear. You will need to browse to a location on your hard disk to save and build this project, then click **OK** (Figure 46).


Create New Project Using Base System Builder Wizard

New Project

The project file will be created in the current directory if a path is not specified.

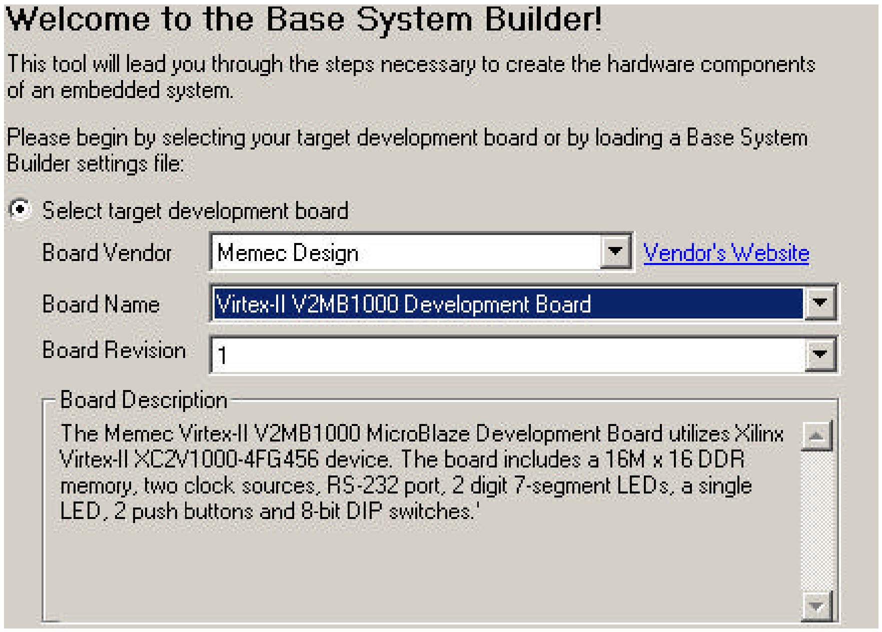
Project File: C:\temp\temp2\system.xmp

Peripheral Repository Directory (Advanced Option)

☐ User Peripheral Repository search path for IP, driver and library files. Can be a semicolon separated list of directories.

Figure 46. Create New Project with Base System Builder Wizard

Now select a vendor of your evaluation board, if any, then click **NEXT** (Figure 47).



Welcome to the Base System Builder!

This tool will lead you through the steps necessary to create the hardware components of an embedded system.

Please begin by selecting your target development board or by loading a Base System Builder settings file:

[● Select target development board]

Board Vendor: Memec Design

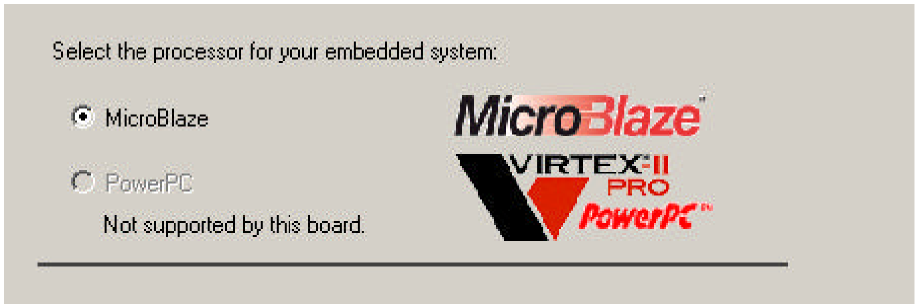
Board Name: Virtex-II V2MB1000 Development Board

Board Revision: 1

Board Description: The Memec Virtex-II V2MB1000 MicroBlaze Development Board utilizes Xilinx Virtex-II XC2V1000-4FG456 device. The board includes a 16M x 16 DDR memory, two clock sources, RS-232 port, 2 digit 7-segment LEDs, a single LED, 2 push buttons and 8-bit DIP switches.'

Figure 47. Vendor Evaluation Board Selection

On the next screen, select the option for **MicroBlaze**.



Select the processor for your embedded system:

[● MicroBlaze]

PowerPC — Not supported by this board.

Figure 48. Processor Selection

On the next screen, you will be setting the clock frequency and various options for Debug, Cache, and RAM. Please select the correct frequency, No Cache and make sure the RAM block size is correct for the FPGA version that you are using. You will also want to select the **No Debug** option (see Figure 49), then you can click on the **NEXT** button. The option for **No Debug** is selected so that the tool set will **not** add the default MDM links to your project. The MDM links will conflict with the Nohau Debug IP that you will be adding later with the BlazeGen application.

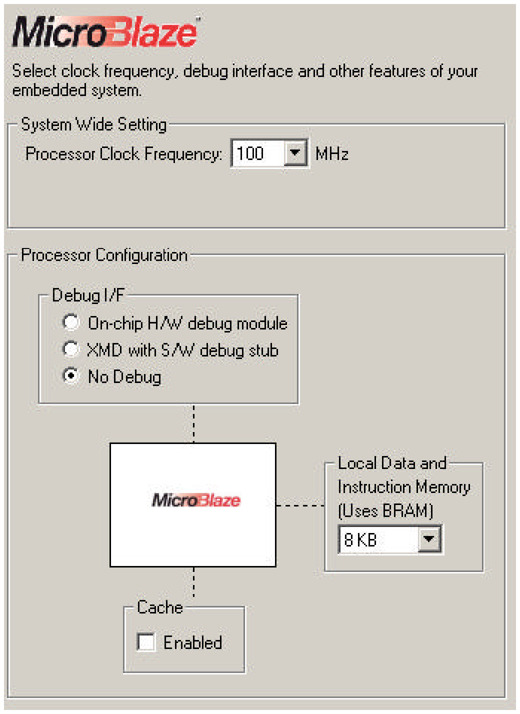

MicroBlaze
Select clock frequency, debug interface and other features of your embedded system.
System Wide Setting
Processor Clock Frequency: 100 MHz
Processor Configuration
Debug I/F
( ) On-chip H/W debug module
( ) XMD with S/W debug stub
(●) No Debug
MicroBlaze
Local Data and Instruction Memory (Uses BRAM)
8 KB
Cache
☐ Enabled

Figure 49. Setting Clock Frequency and other Options

You may continue on to the following screens. It is advisable that you limit the option of the peripheral IPs that you want.

Once the project has been generated, you will want to use the Nohau **BlazeGen** application to add the Nohau Debug IP.

The application is located under the installed SeehauBlaze directory, normally located on the hard disk at c:\nohau\seehaublaze\examples (Figure 50). You will need to select the correct **Platform Studio** version that you are using.

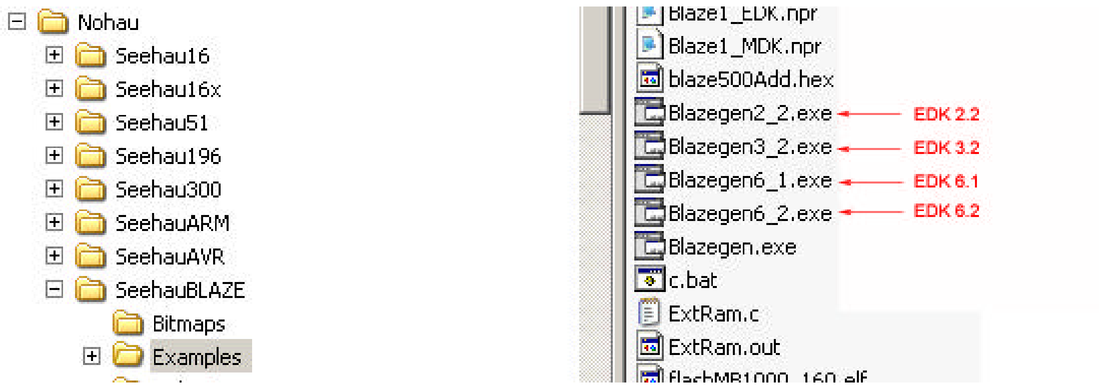


Figure 50. Application Location in SeehauBlaze Directory

You will then use the **File / Open** option in Blazegen to select the system.mhs file from the project you just created (Figure 51).

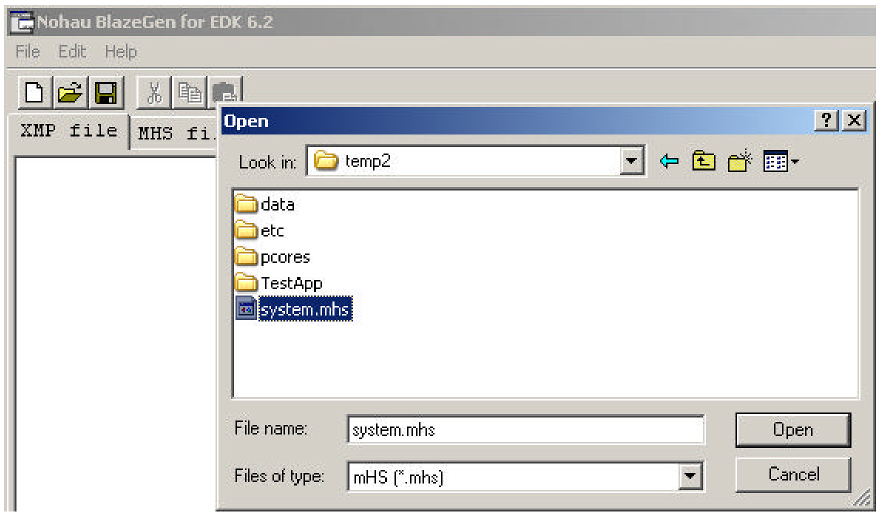
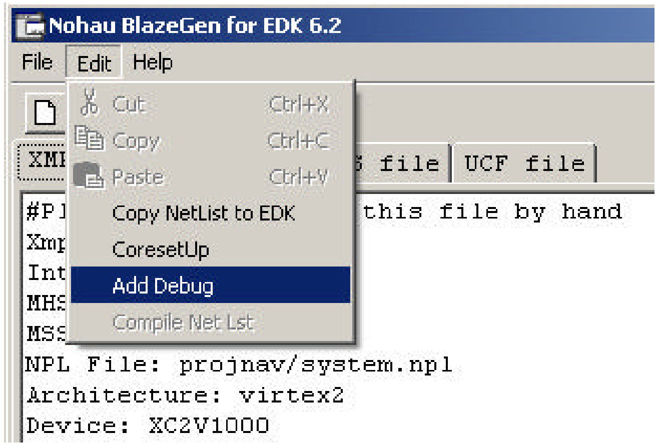


Figure 51. Select the system.mhs File

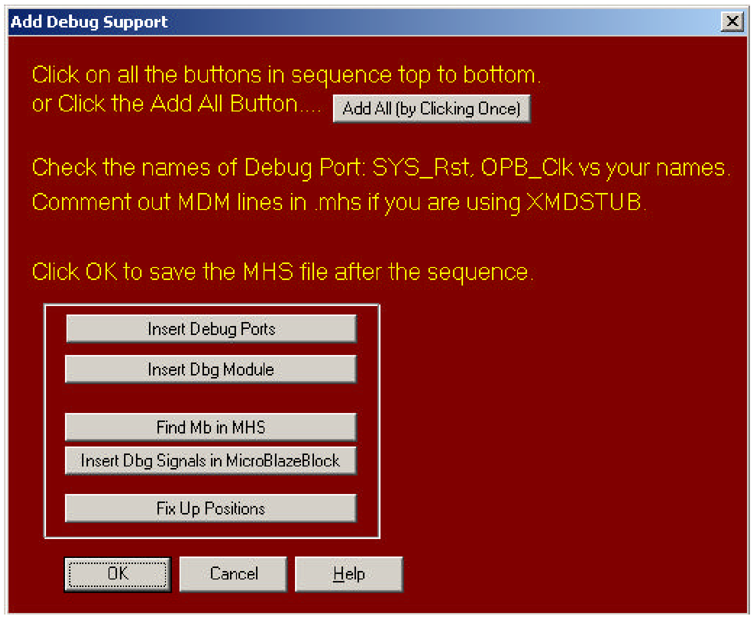
Once the file is open, select **Edit | Add Debug** (Figure 52).



```
#P                         this file by hand
Xmp
Int
MHS
MSS
NPL File: projnav/system.npl
Architecture: virtex2
Device: XC2V1000
```

Figure 52. Add/Debug Menu Item

The following dialog box should appear. You will then select (by clicking once) the **Add all** button (see Figure 53).



Add Debug Support

Click on all the buttons in sequence top to bottom.
or Click the Add All Button.... Add All (by Clicking Once)

Check the names of Debug Port: SYS_Rst, OPB_Clk vs your names.
Comment out MDM lines in .mhs if you are using XMDSTUB.

Click OK to save the MHS file after the sequence.

Figure 53. Add Debug Support

There will be a pop-up that will state ‘X’ number of locations have been fixed up. Click **OK** to acknowledge the pop-up and click **OK** again to save the file after the sequence.

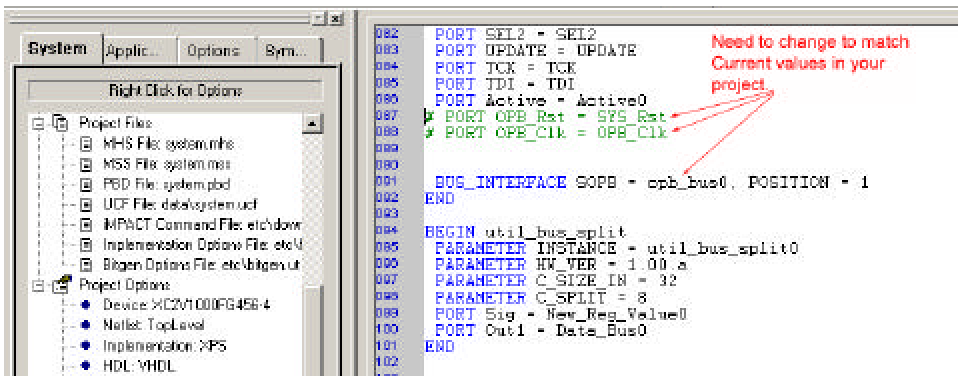
Now make sure you select the correct micro type, so that the correct version of the debug IP will be added to your project.

Virtex-2 / -2 Pro and Spartan3 = select Virtex2

Spartan 2 / 2E = select the Spartan2/2E option.

Once you have done this, you may close the BlazeGen software and return to the **Platform Studio** and open your project. We will need to make a few changes in the project at this time.

First, open the system.mhs file by double-licking on it in the project tree. Now, make the changes to ensure that the debug core will connect to the Microblaze instance in the project as highlighted in Figure 54.



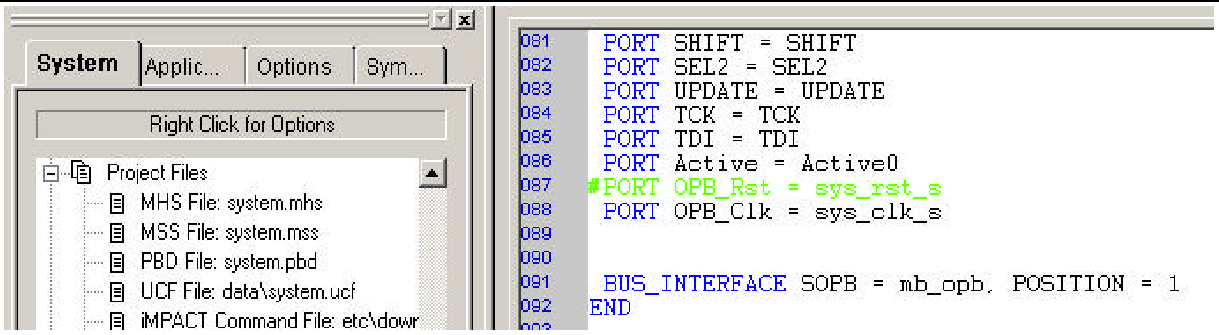
```
 PORT SEL2 = SEL2
 PORT UPDATE = UPDATE
 PORT TCK = TCK
 PORT TDI = TDI
 PORT Active = Active0
# PORT OPB_Rst = SYS_Rst
# PORT OPB_Clk = OPB_Clk


 BUS_INTERFACE SOPB = opb_bus0, POSITION = 1
END

BEGIN util_bus_split
 PARAMETER INSTANCE = util_bus_split0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_SIZE_IN = 32
 PARAMETER C_SPLIT = 8
 PORT Sig = New_Reg_Value0
 PORT Out1 = Data_Bus0
END
```

Need to change to match Current values in your project.

Figure 54. Project Tree Changes

You can locate these settings in the project's MHS file and replace the items accordingly. Remove the "#" for the PORT OPB_Rst and PORT OPB_Clk lines so they are no longer referred to as comment entries. See Figure 55.

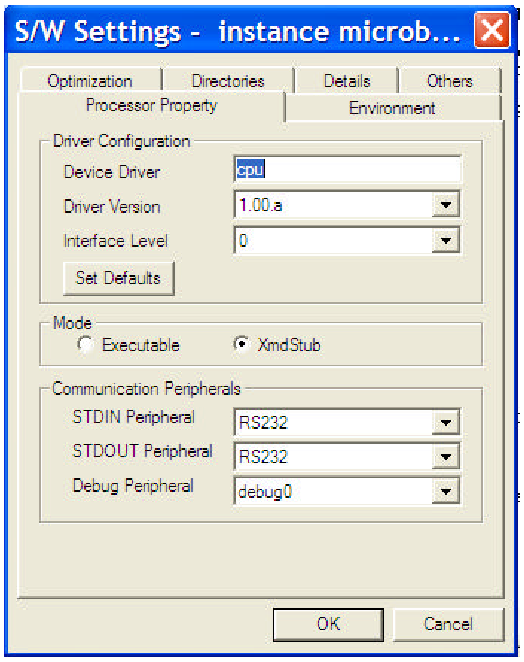
Note: Do not enable the PORT OPB_Rst; this will work by default.



```
 PORT SHIFT = SHIFT
 PORT SEL2 = SEL2
 PORT UPDATE = UPDATE
 PORT TCK = TCK
 PORT TDI = TDI
 PORT Active = Active0
#PORT OPB_Rst = sys_rst_s
 PORT OPB_Clk = sys_clk_s


 BUS_INTERFACE SOPB = mb_opb, POSITION = 1
END
```

Figure 55. Reconfigure Comment Entries

Double-click on the MicroBlaze instance in the project tree, and change the settings for the build mode to be XmdStub, Debug Peripheral to debug0. This will place the XmdStub in the lower 1K of memory before your application, as shown in Figure 56.



Driver Configuration
Device Driver: cpu
Driver Version: 1.00.a
Interface Level: 0

Mode
( ) Executable [✓ XmdStub]

Communication Peripherals
STDIN Peripheral: RS232
STDOUT Peripheral: RS232
Debug Peripheral: debug0

Figure 56. Change MicroBlaze Instance Settings

Now click on the Download button on the tool bar to completely build your application. There will be an error at the end because you will not have the Xilinx parallel cable connected to the target, and the Xilinx tools will not download via Nohau’s USB-JTAG interface.

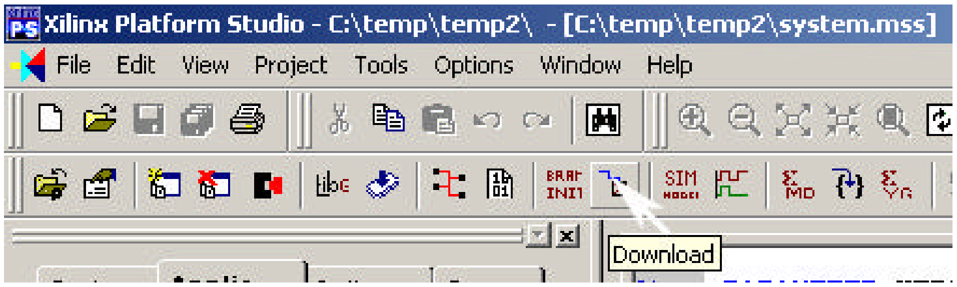

Xilinx Platform Studio - C:\temp\temp2\ - [C:\temp\temp2\system.mss]
File Edit View Project Tools Options Window Help
Download

Figure 57. Download Icon

Working with Platform Studio 6.2 & BlazeGen

Make a Quick Project Using Xilinx's **Platform Studio** and Nohau's **BlazeGen**

Start the Xilinx **Platform Studio** program.

Now select the option **File | New Project | Base System Builder** (Figure 58).

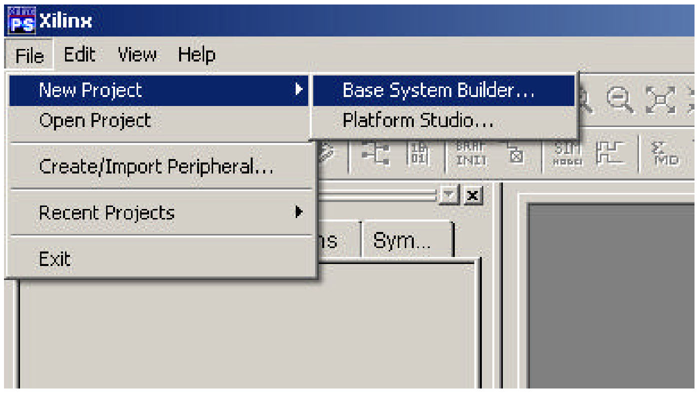
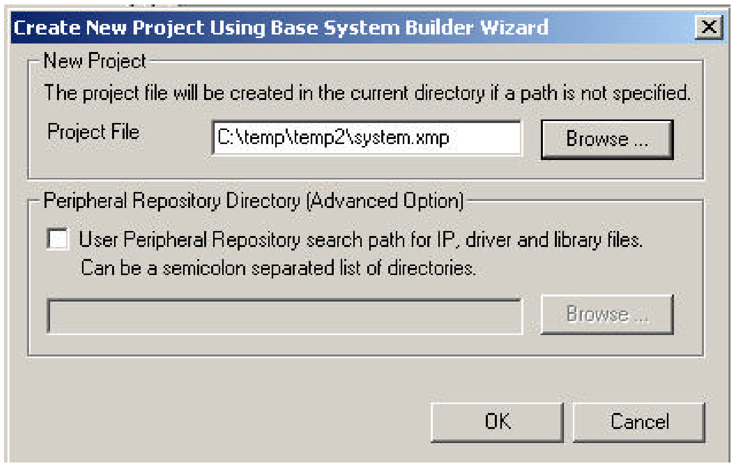


Figure 58. Base System Builder Option

The **Create New Project** dialog box should appear. You will need to browse to a location on your hard disk to save and build this project,then click **OK** (Figure 59).


Create New Project Using Base System Builder Wizard
New Project
Browse ...
Peripheral Repository Directory (Advanced Option)
Browse ...
OK
Cancel

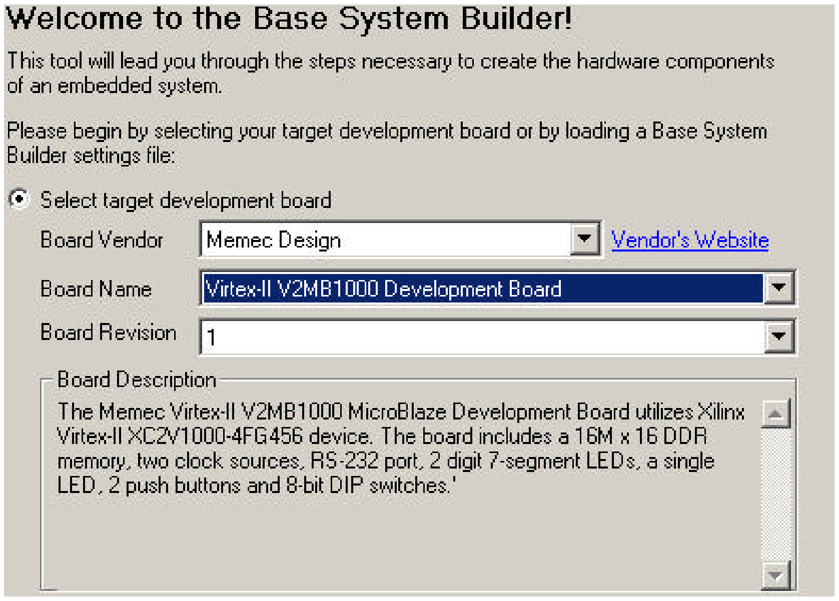
The project file will be created in the current directory if a path is not specified.

Project File: C:\temp\temp2\system.xmp

☐ User Peripheral Repository search path for IP, driver and library files. Can be a semicolon separated list of directories.

Figure 59. Base System Builder Create New Project Screen

Now select the vendor of your evaluation board, if any, then click **NEXT** (Figure 60).



This tool will lead you through the steps necessary to create the hardware components of an embedded system.

Please begin by selecting your target development board or by loading a Base System Builder settings file:

● Select target development board

Board Vendor: Memec Design

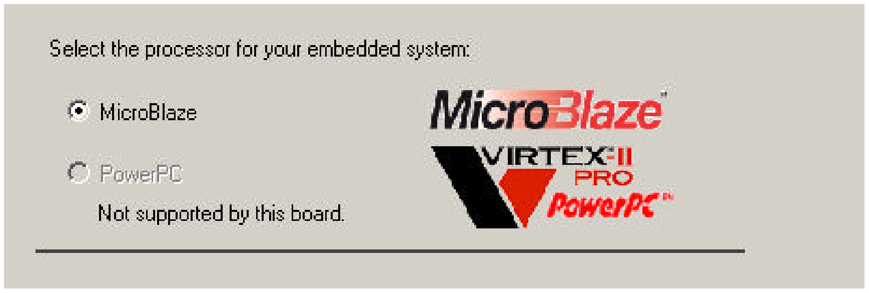
Board Name: Virtex-II V2MB1000 Development Board

Board Revision: 1

The Memec Virtex-II V2MB1000 MicroBlaze Development Board utilizes Xilinx Virtex-II XC2V1000-4FG456 device. The board includes a 16M x 16 DDR memory, two clock sources, RS-232 port, 2 digit 7-segment LEDs, a single LED, 2 push buttons and 8-bit DIP switches.'

Figure 60. Selecting Evaluation Board Vendor

On the next screen, select the option for MicroBlaze.



Select the processor for your embedded system:

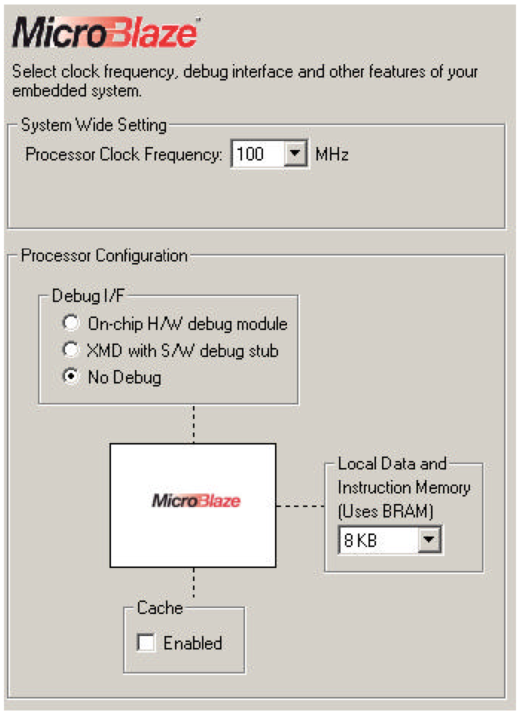
● MicroBlaze

○ PowerPC

Not supported by this board.

Figure 61. Processor Selection

On the next screen, you will be setting the clock frequency and various options for Debug, Cache, and RAM. Please select the correct frequency, No Cache, and make sure the RAM block size is correct for the FPGA version you are using. You will also want to select the **No Debug** option (see Figure 62), and then click on the **NEXT** button. The option for **No Debug** is selected so that the tool set will **not** add the default MDM links to your project. The MDM links will conflict with the Nohau Debug IP you will be adding later with the BlazeGen application.


Select clock frequency, debug interface and other features of your embedded system.

System Wide Setting
Processor Clock Frequency: 100 MHz

Processor Configuration
Debug I/F
On-chip H/W debug module
XMD with S/W debug stub
[✓ No Debug]

Local Data and Instruction Memory (Uses BRAM): 8 KB

Cache
Enabled [ ]

Figure 61. Setting Clock Frequency and Other Options

You may continue to click "next" on the following screens. It is advisable that you limit the option of the peripheral IPs that you want. Once the project has been generated, use the Nohau **BlazeGen** application to add the Nohau Debug IP.

The application is located under the installed SeehauBlaze directory, normally located on the harddisk at c:\nohau\seehaublaze\examples (Figure 63). You will need to select the correct **Platform Studio** version that you are using.

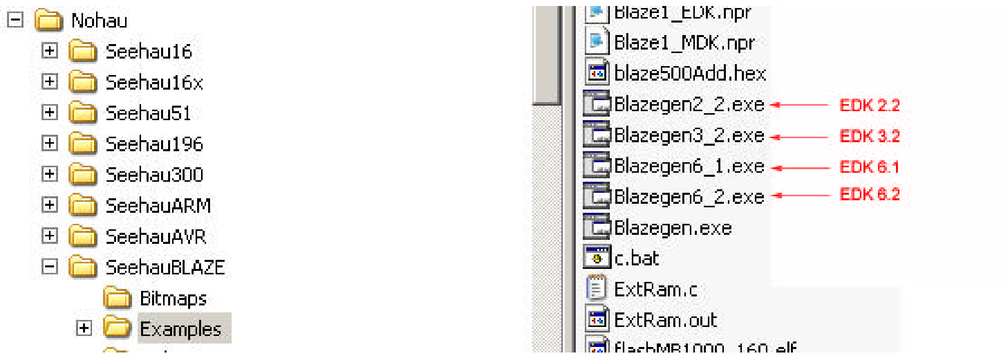
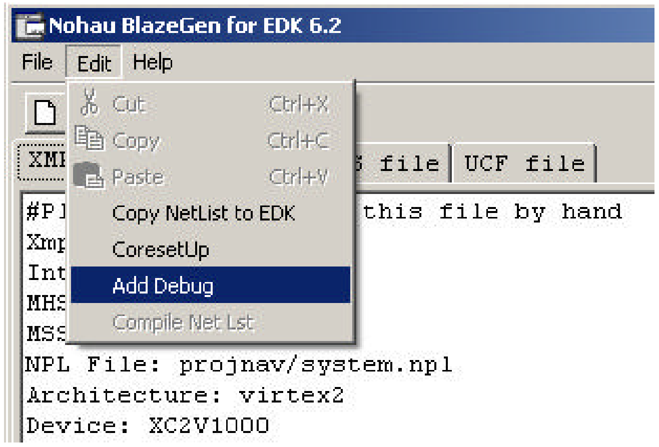


Figure 63. Selecting the Correct Platform Studio Version

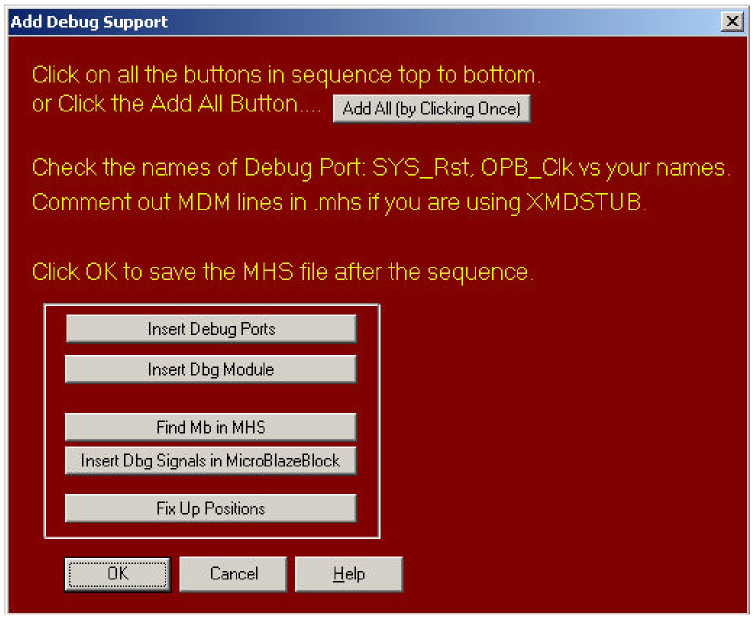
Use the **File / Open** option in Blazegen to select the system.mhs file from the project you just created. Once the file is open, select **Edit | Add Debug** (Figure 64).



```
#P                         this file by hand
Xmp
Int
MHS
MSS
NPL File: projnav/system.npl
Architecture: virtex2
Device: XC2V1000
```

Figure 64. Edit / Add Debug Option

The following dialog box should appear. Click once on the **Add all** button (see Figure 65).



Click on all the buttons in sequence top to bottom.
or Click the Add All Button....

Check the names of Debug Port: SYS_Rst, OPB_Clk vs your names.
Comment out MDM lines in .mhs if you are using XMDSTUB.

Click OK to save the MHS file after the sequence.

Figure 65. Add All Function

There will be a pop-up that will state 'X' number of locations have been fixed up. Click **OK** to acknowledge the pop-up and click **OK** again to save the file after the sequence.

Make sure you select the correct micro type, so that the correct version of the debug IP will be added to your project.

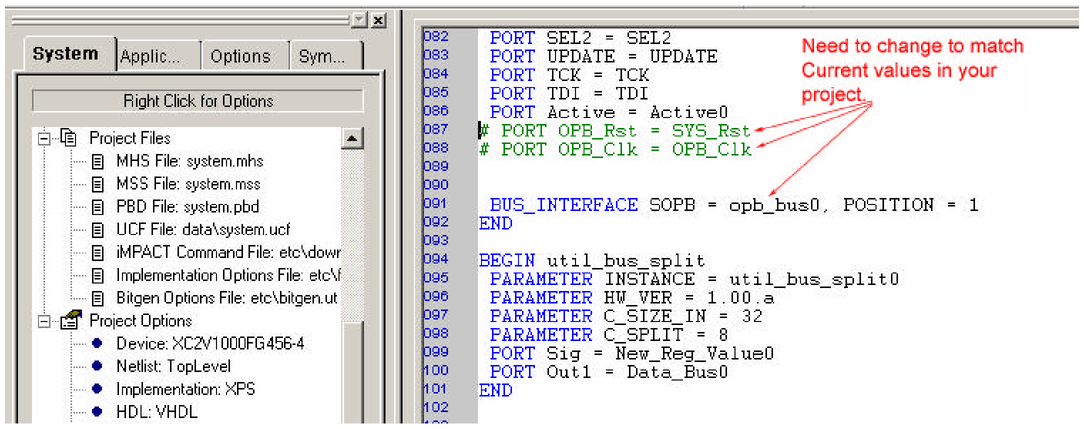
Virtex-2 / -2 Pro and Spartan3 = select Virtex2

Spartan 2 / 2E = select the Spartan2/2E option.

Once you have done this, you may close the BlazeGen software and return to **Platform Studio** and open your project. You will need to make a few changes in the project at this time.

First, open the system.mhs file by double-clicking on it in the project tree.

Now, make the changes to ensure that the debug core will connect to the Microblaze instance in the project as highlighted in Figure 66.

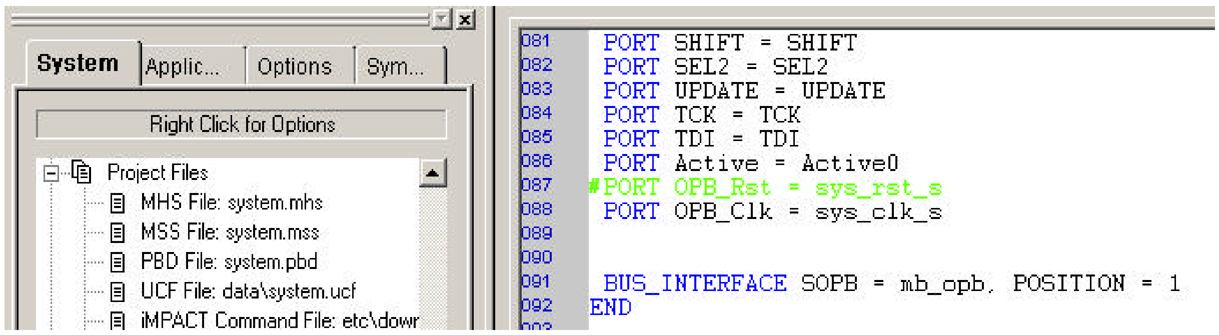


```
 PORT SEL2 = SEL2
 PORT UPDATE = UPDATE
 PORT TCK = TCK
 PORT TDI = TDI
 PORT Active = Active0
# PORT OPB_Rst = SYS_Rst
# PORT OPB_Clk = OPB_Clk


 BUS_INTERFACE SOPB = opb_bus0, POSITION = 1
END

BEGIN util_bus_split
 PARAMETER INSTANCE = util_bus_split0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_SIZE_IN = 32
 PARAMETER C_SPLIT = 8
 PORT Sig = New_Reg_Value0
 PORT Out1 = Data_Bus0
END
```

Figure 66. Changes to Match Current Values

You can locate these settings in the project's MHS file and replace the items accordingly. Remove the "#" for the PORT OPB_Rst and PORT OPB_Clk lines so they are no longer referred to as comment entries. See Figure 67.

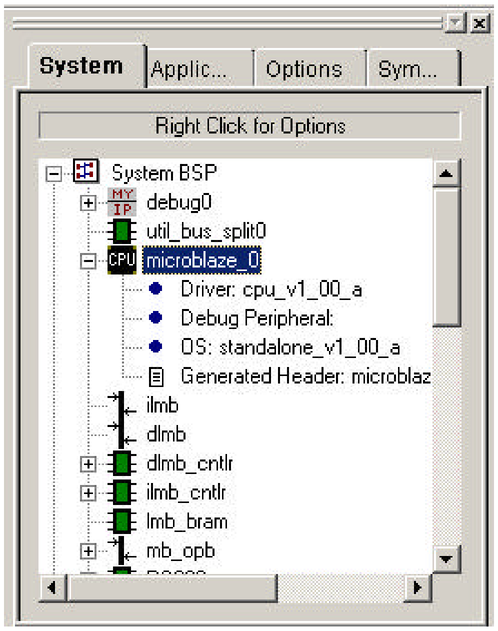
Note: Do not enable the PORT OPB_Rst; this will work by default.



```
 PORT SHIFT = SHIFT
 PORT SEL2 = SEL2
 PORT UPDATE = UPDATE
 PORT TCK = TCK
 PORT TDI = TDI
 PORT Active = Active0
#PORT OPB_Rst = sys_rst_s
 PORT OPB_Clk = sys_clk_s


 BUS_INTERFACE SOPB = mb_opb, POSITION = 1
END
```

Figure 67. Reconfiguring Comment Lines

Click on the microblaze cpu entry in the project tree (Figures 68 & 69), then select the **Application** tab and set it to initialize the BRAM. This will allow the Xmdstub to be placed in memory in the lower 1K before your application.



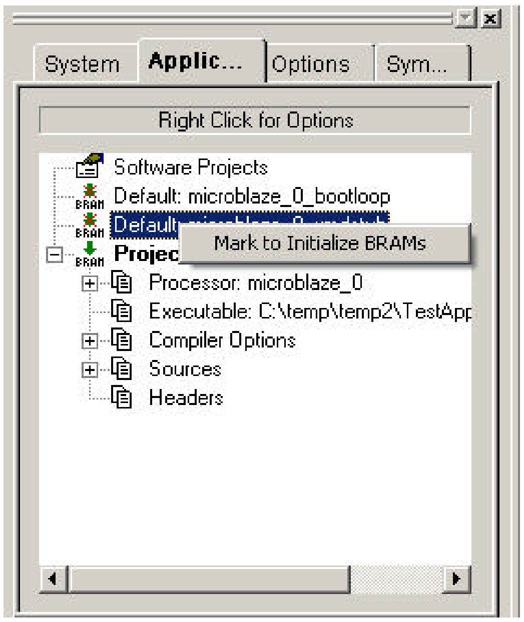
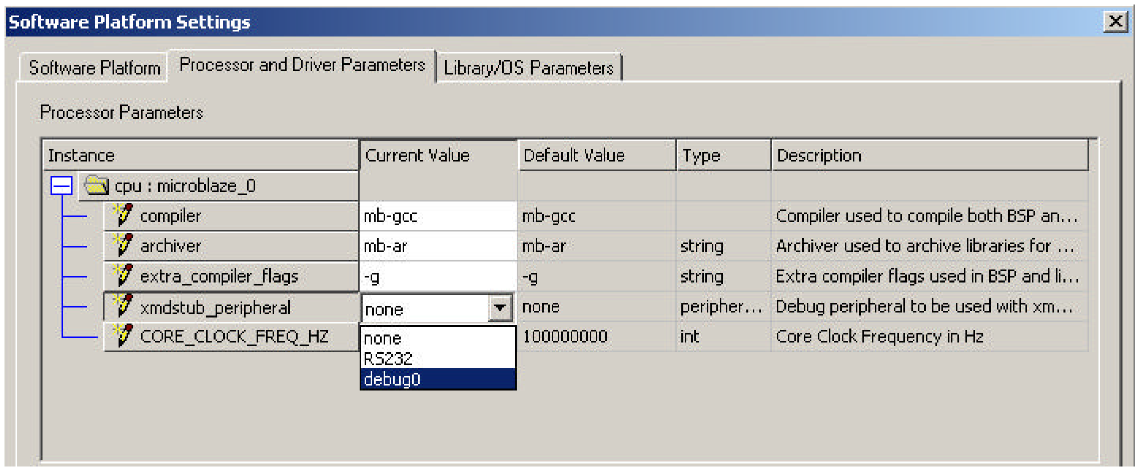


Figure 68. MicroBlaze CPU Entry

Figure 69. Initialize BRAM

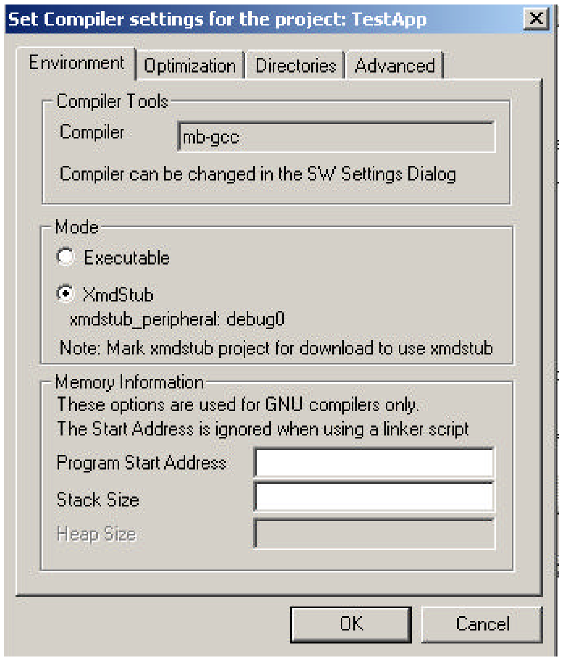
Next, double-click on the microblaze instance in the project tree, and change the option for the debug peripheral (see Figure 70).



Instance	Current Value	Default Value	Type	Description
cpu : microblaze_0				
compiler	mb-gcc	mb-gcc		Compiler used to compile both BSP an...
archiver	mb-ar	mb-ar	string	Archiver used to archive libraries for ...
extra_compiler_flags	-g	-g	string	Extra compiler flags used in BSP and li...
xmdstub_peripheral	none	none	peripher...	Debug peripheral to be used with xm...
CORE_CLOCK_FREQ_HZ		100000000	int	Core Clock Frequency in Hz

Figure 70. Change Debug Peripheral Option

Select the microblaze instance again, and then click on the **Applications** tab above. To make changes to the compiler options, double-click on the compiler options and change the build mode from Executable to XmdStub as shown in Figure 71.



Set Compiler settings for the project: TestApp

Environment | Optimization | Directories | Advanced

Compiler Tools
Compiler: mb-gcc
Compiler can be changed in the SW Settings Dialog

Mode
( ) Executable
(●) XmdStub
xmdstub_peripheral: debug0
Note: Mark xmdstub project for download to use xmdstub

Memory Information
These options are used for GNU compilers only.
The Start Address is ignored when using a linker script
Program Start Address:
Stack Size:
Heap Size:

OK | Cancel

Figure 71. Compiler Settings

Click on the **Directories** tab and clear the entry in the option for the linker Script. If you don't do this, you will need to alter the linker script because the defaults will cause memory overlap errors and the project will not build.

Now click on the **Download** button on the toolbar to completely build your application. There will be an error at the end because you will not have the Xilinx parallel cable connected to the target, and the Xilinx tools will not download via Nohau's USB-JTAG interface.

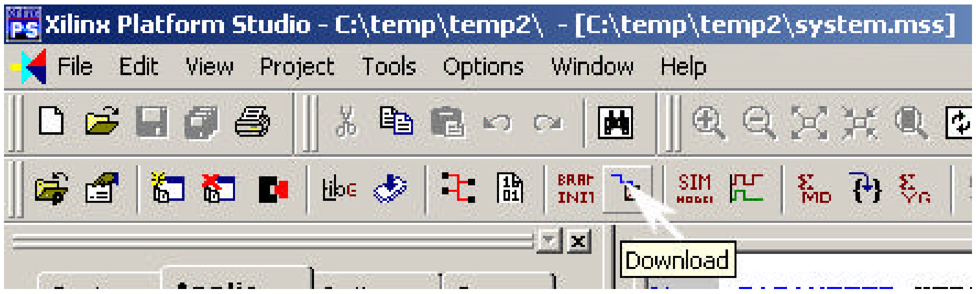

Xilinx Platform Studio - C:\temp\temp2\ - [C:\temp\temp2\system.mss]
File Edit View Project Tools Options Window Help
Download

Figure 72. Download Icon

Index

A

accessing the Nohau Project Manager · 24
adding a file to the project · 27
adding a new project · 26
adding a new workspace · 25
Apply · 14, 21, 35
automatically starting the Seehau Configuration Program · 9

B

Break on Trace · 22
building a single project · 29
building all projects · 29
Buttons Common to All Tabs · 14, 21, 35

C

C program example · 19
Cancel · 14, 21, 35
chip groups · 1
Clock(MHz) · 13, 14
Communications Interface · 11
configuring
  communications interface · 11–10
  emulator options · 11–10
Configuring the Emulator Options From Within Seehau · 14
Connect tab · 10
Connect Tab · 11
Connections for the USB Cable and the Red Wire Pin 1 (photo) · 5

D

damage to your target · vii
debugger · 1
dialog boxes
  File Settings · 27–26
  New Project · 25
  New Workspace · 24
  Open · 25
  Select Directory · 25
  Setup Complete · 7
  Workspace Settings · 27–26
disable all pod resources · vii
do not connect the pod to your target when the target power is on · vii
downloading
  electronic version of this guide · viii
  Seehau · vi
downloading an electronic version of this guide · viii

E

Electromagnetic Emission Requirements · vi
ELF format · 1
Emulator Configuration window · 9, 14, 19
Emulator Configuration Window Displaying the Connect Tab (screen shot) · 10
Emulator Configuration Window Displaying the Hdw Config Tab (screen shot) · 12, 15
Emulator Configuration Window Displaying the Misc Setup Tab (screen shot) · 16
Emulator Connection · 11
EMUL–MicroBlaze–PC/USB–JTAG with the 10-Pin Serial Debug Connector (diagram) · 5
EMUL–PC/USB–JTAG (photo) · 3
EMUL–PC/USB–JTAG, installation instructions · 4
European CE Requirements · vi
  Special Measures · vi
  User Responsibility · vi
Event Configuration Tab · 22

F

File Settings dialog box · 27, 28
File Settings Dialog Box (screen shot) · 28
file settings, changing · 28
forcing a build of a selected file · 29
forcing a build of a single project · 29

forcing a build of all projects · 29

H

Hdw Config tab · 11, 21, 33
Help · 14, 21, 35

I

installing
  power supply · 2
  Seehau
    from the CD · 6
    from the internet · 6
  USB driver · 2

L

led_blink.c tab · 21

M

mb-as assembler · 1
mb-gcc compiler · 1
mb-ld loader/linker platform · 1
MDK · 6
Misc Setup tab · 17, 19
Miscellaneous Setup Tab · 17

N

New Project dialog box · 26
New Workspace dialog box · 25
Nohau Project Manager
  accessing · 24
  changing settings
    file · 28
    workspace · 27–26
  Overview · 24
  projects
    building · 29–28
    working with · 25
  workspaces · 24
Nohau Project Manager window · 24, 25, 26
Nohau Project Manager Window (screen shot) · 25
non-demo mode · 17

O

OK · 14, 21, 35
Open dialog box · 19, 25, 27
Open Dialog Box Displaying the Program Example Directories (screen shot) · 20
opening a workspace · 25
Override at Reset · 17, 22, 34
overview
  hardware · 1
  Nohau Project Manager · 24
  software · 1

P

port · 11
Post Trigger Count · 22, 35
Power Supply · 2
Pressing on the chip · vii
Processor · 13, 14
Program Counter · 17, 22, 34
Program Example in Mixed Mode (screen shot) · 21
Program Example in Source Only (screen shot) · 20
program examples
  loading code · 19
  single-stepping in mixed mode · 20
  single-stepping in source only mode · 20

R

Refresh · 14, 21, 35
removing a file from the project · 27
removing a project · 27
Reset chip after load file · 17
running the test program · 19

S

saving a workspace · 26
Seehau for EMUL–MICROBLAZE–PC (screen shot) · 18
Seehau main window · 17, 32
Seehau, configuring
  Communications Interface
    Connect Tab · 11–10
    Hdw Config Tab · 13–12
  emulator options
    Hdw Config Tab · 13–12
    Misc Setup Tab · 17–16
Seehau, installing
  from the CD · 6
  from the internet · 6
Seehau, starting
  in demo mode · 17
  in non-demo mode · 17
Select Directory dialog box · 26
Select Processor · 11
Setup Complete dialog box · 7, 9
signal layout of the connector · 4
single-stepping in mixed mode · 20
single-stepping in source only mode · 21
Source window · 21, 27
Stack Pointer · 17
Startup.bas · 17
System Requirements · vi

T

tabs
  Connect · 10
  Hdw Config · 11–10
  led_blink.c · 20
  Misc Setup · 14
Trace Type · 11
Trigger Condition · 22, 34
Trigger Enable · 22, 34

U

USB Driver · 2
User Responsibility · vi

W

Warnings · vii
Warranty Information · v
Watch window · 19
web site (http://www.icetech.com/) · 7
windows
  Emulator Configuration · 10
  New · 17–16
  Nohau Project Manager · 24
  Seehau main · 17–16
  Source · 20
  Watch · 19
Windows Autorun feature · 6
Working with Projects
  adding · 26
  adding a file to the project · 27–26
  removing · 27–26
  removing a file from the project · 27–26
Working with Workspaces
  adding · 25
  opening · 25
  saving · 25
Workspace Settings dialog box · 27
workspace settings, changing · 27