# Hardware Debugger Guide

# About This Manual

This manual describes Xilinx’s Hardware Debugger program, a tool used for configuring and debugging FPGA devices.

Before using this manual, you should be familiar with the operations that are common to all Xilinx software tools: how to bring up the system, select a tool for use, specify operations, and manage design data. These topics are covered in the *Quick Start Guide*.

Other publications you can consult for related information are *The Programmable Logic Data Book*, *Development System Reference Guide*, and *Hardware User Guide*.

## Additional Resources

For additional information, go to http://support.xilinx.com. The following table lists some of the resources you can access from this page. You can also directly access some of these resources using the provided URLs.

| Resource | Description/URL |
|---|---|
| Tutorial | Tutorials covering Xilinx design flows, from design entry to verification and debugging<br>http://support.xilinx.com/support/techsup/tutorials/index.htm |
| Answers Database | Current listing of solution records for the Xilinx software tools<br>Search this database using the search function at<br>http://support.xilinx.com/support/searchtd.htm |
| Application Notes | Descriptions of device-specific design techniques and approaches<br>http://support.xilinx.com/apps/appsweb.htm |

| Resource | Description/URL |
|---|---|
| Data Book | Pages from *The Programmable Logic Data Book*, which describe device-specific information on Xilinx device characteristics, including read-back, boundary scan, configuration, length count, and debugging http://support.xilinx.com/partinfo/databook.htm |
| Xcell Journals | Quarterly journals for Xilinx programmable logic users http://support.xilinx.com/xcell/xcell.htm |
| Tech Tips | Latest news, design tips, and patch information on the Xilinx design environment http://support.xilinx.com/support/techsup/journals/index.htm |

## Manual Contents

This manual covers the following topics.

- Chapter 1, "Introduction," defines the Hardware Debugger and its operation.
- Chapter 2, "Getting Started," briefly describes the interface and summarizes the design and hardware requirements for running the Hardware Debugger.
- Chapter 3, "Design Preparation," is a step-by-step explanation of how to prepare your design and create a configuration data file for use with the Hardware Debugger.
- Chapter 4, "Connecting Your Cable," describes the different cables available for use with the Hardware Debugger and how to connect them to your computer and target device. It also describes the different software settings used to implement the programming and debugging functions and explains how to test the MultiLINX™ Cable and the XChecker™ Cable hardware.
- Chapter 5, "Programming a Device or a Daisy Chain," explains how to download a design to a single device and verify that the design was sent in its entirety. It also explains how to configure multiple devices.
- Chapter 6, "Debugging a Device," explains how to display the states of specific signals in a device that was programmed with a design.

- Chapter 7, “Customizing the Interface,” explains how to use macros, how to control the waveform display parameters, and how to use the additional features to optimize your use of the Hardware Debugger.
- Appendix A, “Glossary of Terms,” defines common terms used in the context of the Hardware Debugger.
- Appendix B, “Console Commands,” interprets the syntax of the commands that are displayed in the Console window each time you execute a menu command. You can execute a command directly by typing the appropriate syntax in the Console window command bar. Console commands also constitute the building blocks of macros.

# Conventions

## Typographical

This manual uses the following conventions. An example illustrates each convention.

- `Courier font` indicates messages, prompts, and program files that the system displays.

  ```
  speed grade: -100
  ```

- **`Courier bold`** indicates literal commands that you enter in a syntactical statement.

  **`rpt_del_net=`**

  **`Courier bold`** also indicates commands that you select from a menu.

  **`File`** → **`Open`**

- *Italic font* denotes the following items.
  - Variables in a syntax statement for which you must supply values

    **`edif2ngd`** *design_name*

  - References to other manuals

    See the *Development System Reference Guide* for more information.

  - Emphasis in text

    If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.

- Square brackets “[ ]” indicate an optional entry or parameter. However, in bus specifications, such as bus [7:0], they are required.

  `edif2ngd` [*option_name*] *design_name*

  Square brackets also enclose footnotes in tables that are printed out as hardcopy in DynaText®.

- Braces “{ }” enclose a list of items from which you choose one or more.

  **`lowpwr ={on|off}`**

- A vertical bar “ | ” separates items in a list of choices.

  `symbol` *editor_name* `[bus|pins]`

- A vertical ellipsis indicates repetitive material that has been omitted.

  ```
  IOB #1: Name = QOUT’
  IOB #2: Name = CLKIN’
  .
  .
  .
  ```

- A horizontal ellipsis “. . .” indicates that an item can be repeated one or more times.

  `allow block` *block_name loc1 loc2* `...` *locn;*

## Online Document

Xilinx has created several conventions for use within the DynaText online documents.

- Red-underlined text indicates an interbook link, which is a cross-reference to another book. Click on the red-underlined text to open the specified cross-reference.
- Blue-underlined text indicates an intrabook link, which is a cross-reference within a book. Click on the blue-underlined text to open the specified cross-reference.
- There are several types of icons.

  Iconized figures are identified by the figure icon.

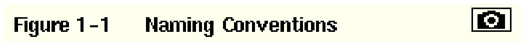


Figure 1-1 Naming Conventions

Iconized tables are identified by the table icon.

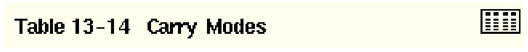
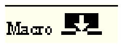


Table 13-14 Carry Modes

The Copyright icon displays in the upper left corner on the first page of every Xilinx online document.

The DynaText footnote icon displays next to the footnoted text.



Macro

Double-click on these icons to display figures, tables, copyright information, or footnotes in a separate window.

- Inline figures display within the text of a document. You can display these figures in a separate window by clicking on the figure.

# Chapter 1

# Introduction

This chapter provides an overview of the Hardware Debugger, and includes the following sections.

- “Overview”
- “Architectures”
- “Hardware Debugger Features”
- “Design and Hardware Considerations”

## Overview

The Hardware Debugger is a graphical interface that allows you to download a design to a device, verify the downloaded configuration, and display the internal states of the programmed device. Use the program to perform the following tasks.

- Download a BIT file to an FPGA, or a PROM file to a daisy chain of FPGAs.
- Verify the configuration data of a single device using the Multi-LINX™ Cable or the XChecker Cable.
- Debug the internal logic states of a configured device using an XChecker Cable.

The graphical based Hardware Debugger replaces the command line based XChecker program.

Refer to the “Programming a Device or a Daisy Chain” chapter for information on how to open, download, and verify your design. Refer to the “Debugging a Device” chapter for information on how to read back and debug your configured device.

**Note:** Before invoking the Hardware Debugger, connect the configuration cable to your computer and target board; you should also be aware of the design and hardware issues described in the “Design and Hardware Considerations” section in this chapter.

## Architectures

You can use the Hardware Debugger with the following Xilinx devices:

- XC3000A / L™
- XC3100A / L™
- XC4000E / L™
- XC4000EX™ / XL™ / XV™ / XLA™
- XC5200™
- Spartan™ / XL™
- Spartan2™
- Virtex™
- VirtexE ™

The Spartan, Virtex, XC3000, XC4000, and XC5200 families compose the FPGA families. The Hardware Debugger does not support CPLDs.

## Hardware Debugger Features

You can use the Hardware Debugger to perform the following functions.

- Configure one or more devices
- Verify configuration data for single devices
- Debug a single device in synchronous or asynchronous mode
- Generate multiple graphical and textual waveforms for probe points
- Group signals for waveforms for later use
- Reuse debug settings from one session to another

- Create macros (command scripts) by copying commands from the Console window into a macro window and saving them as macro files
- Examine real-time digital values of the XChecker pins
- Use the XChecker Cable to probe and display the values of internal nodes of an FPGA device as listed in the table.

**Table 1-1 Probe Points in FPGA Devices**

| XC3000 | XC4000 | XC5200 |
|---|---|---|
| CLB outputs | CLB outputs | — |
| IOB outputs | IOB outputs | — |
| — | RAM / ROM bits | internal ROM bits |
| Flip-flop outputs | Flip-flop outputs | Flip-flop outputs |

## Design and Hardware Considerations

Before using the Hardware Debugger, review the requirements for the design entry phase in the following table. These requirements are described in detail in the “Design Preparation” chapter and the “Connecting Your Cable” chapter.

**Table 1-2 Requirements for Hardware Debugger Operations**

| | Downloading | Verification | Debugging |
|---|---|---|---|
| **File Types** | BIT, RBT, PROM | BIT, LL | |
| **Design Symbols for Virtex/ Spartan2 devices** | No symbols required | MSK, BIT, LL | |
| **Design Symbols for XC4000/ XC5200 devices** | No symbols required | READBACK STARTUP[a] | |

**Table 1-2 Requirements for Hardware Debugger Operations**

| | **Downloading** | **Verification** | **Debugging** |
|---|---|---|---|
| **Configuration options for XC3000 devices** | Enable pull-up resistor on DONE/PROG pin | - Enable pull-up resistor on DONE/PROG pin<br>- Set Readback Mode to Once or On Command or use the -l option in BitGen | |
| **Configuration options for XC4000/XC5200 devices** | Enable pull-up on DONE pin | - Set readback clock to CClk<br>- Enable the Enable Bitstream Verification and In-Circuit Hardware Debugging option or use the -l option in BitGen | |
| **Cable Type** | MultiLINX, XChecker, serial, or parallel cables | MultiLINX™, XChecker Cable | |
| **Pins Used** | VCC<br>GND<br>CCLK<br>D/P (DONE)<br>DIN<br>PROG (XC4000/XC5200)<br>INIT<br>RST | VCC<br>GND<br>CCLK<br>RT<br>RD | VCC<br>GND<br>CCLK<br>RT<br>RD<br>TRIG[a]<br>CLKI [b]<br>CLKO[b] |
| **Configuration Mode** | Slave (SelectMAP mode is supported in the Hardware Debugger with the MultiLINX Cable.) | N/A | N/A |

a. Optional. Use the STARTUP symbol to connect the design's global reset to the GSR (XC4000) or GR (XC5200) pin on the XChecker Reset. An inverter must be inserted in front of the GSR or GR pin to interface the active-Low XChecker reset with the active-High GSR/ GR reset.
b. Optional

## Design Entry and Bitstream Generation

You need access to certain signals when you download and debug your designs.

For downloading, use any of the three configuration data file types (BIT, RBT, or PROM) and any download cable. You do not need any special symbols or configuration options to generate the configuration data.

**Note:** To create a PROM file using multiple bitstreams, use the PROM File Formatter. For more information, see the *PROM File Formatter Guide*.

For single device verification or debugging, use a BIT file, enable readback to generate an LL file in your design directory, and for XC4000 / XC5200 devices. For Virtex, Spartan2, generate a MSK file for verification. Include the READBACK symbol in your design and optionally, the STARTUP symbol. In addition, you must use an XChecker Cable.

## Hardware Considerations

This section describes some hardware issues you should be aware of before using the Hardware Debugger.

### Configuration Mode

When using the MultiLINX Cable the default mode is already set to slave serial. For Virtex and Spartan II the configuration mode dialog box will automatically pop up and ask the user for the proper configuration mode. This configuration mode dialog is only enabled for the MultiLINX Cable.

When using other download cables to configure a device or daisy chain of devices, you must set the device configuration mode to slave serial. You must set M0, M1, and M2 to VCC, and if you intend to use them as user I / Os, use 4.7 kilohm pull-ups. Refer to the *Development System Guide* or *The Programmable Logic Data Book* for information on setting the mode pins.

### Target Board Selection

Your target board can be either a Xilinx FPGA demonstration board or your own board. You can use the demonstration boards to test most designs.

If you are using an FPGA demonstration board, set up the board as explained in the *Hardware Debugger Watch Tutorial*. The *Watch Tuto-*

*rials* including the *Hardware Debugger* are located on the support.xilinx.com web page.

(Use the following URL: http://support.xilinx.com/support/techsup/tutorials/index.htm).

You must set the board switches and connect the appropriate cable. If you are using your own board, you must configure the board and connect the appropriate cable. Refer to the “Connecting Your Cable” chapter for information.

### Single or Multiple Device Configuration

You can configure one device or a daisy chain. See the “XChecker Operation Mode Connections” table in the “Connecting Your Cable” chapter for information on pin connections.

The MultiLINX Cable must be configured only for download and verify. Refer to the “MultiLINX™ Cable” chapter of the *Hardware User Guide* for information on the MultiLINX pin connections.

To configure a single device, connect the pins as specified for that particular device type. For a daisy chain of devices, connect the pins of the lead device for downloading. Connect the pins of the slave devices to the lead device as described in the *The Programmable Logic Data Book*.

## Cable Connections

You can perform three main operations using the Hardware Debugger. Each operation requires a specific setup and, in the case of verification and debugging, a particular download cable. Refer to the “XChecker Operation Mode Connections” table in the “Connecting Your Cable” chapter for cable connections information for each operation.

### Downloading

Connect the cable header to the pin assembly on your target board that is connected to the FPGA configuration pins. If you have an XChecker Cable, connect a cable header connector to the outermost slot of the XChecker Cable assembly.

### Verification

For this operation, you must use an XChecker Cable. Connect both header connectors to the pin assembly on your target board.

### Debugging

For this operation, you must use an XChecker Cable. Connect both header connectors to the pin assembly on your target board. If you are debugging in synchronous mode, you must connect the CLKI and CLKO pins; you can disconnect them for asynchronous mode debugging.

**Note:** The RST pin is only required for XC3000 downloading and debugging. You do not need to connect it for XC4000 or XC5200 downloading. Optionally, you can use it as a system design reset for XC4000 or XC5200 debugging.

## Hardware Debugger Cable Support

The Hardware Debugger supports the following cables.

**Table 1-3 Cable Support**

| Cable Name | Download | Readback & Verify | Debug |
|---|---|---|---|
| MultiLINX[a] | X | X | |
| XChecker[b] | X | X | X |
| Parallel | X | | |
| Serial | X | | |

a. MultiLINX supports Slave Serial and SelectMAP modes.
b. XChecker supports only Slave Serial Mode. XChecker can support devices up to 256K bits.

## Hardware Debugger Platform Support

The Hardware Debugger supports the following platforms.

**Table 1-4 Platform Support**

| Cable | Platforms | Ports |
|---|---|---|
| MultiLINX | Sol / HP / WinNT / Win95 / Win98 | RS-232,USB[a] |
| XChecker | Sol / HP / WinNT / Win95 / Win98 | RS-232 |
| Parallel | WinNT / Win95 / Win98 | Parallel |

a. Only on Win 98 / 95C which has USB support.

# Chapter 2

# Getting Started

This chapter explains how to start the Hardware Debugger. It describes the main screen and the three major tasks you can complete: download a design, verify configuration data, and debug the states of a configured device. In addition, it outlines the design and hardware requirements for using the Hardware Debugger. This chapter contains the following sections.

- “Starting the Hardware Debugger”
- “Exiting the Hardware Debugger”
- “Using the Interface”
- “Using Help”

## Starting the Hardware Debugger

**Note:** Before using the Hardware Debugger, familiarize yourself with the design and hardware issues described in the “Design and Hardware Considerations” section of the “Introduction” chapter.

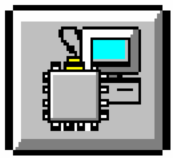
The Hardware Debugger runs on PCs and workstations. The following table summarizes the various ways of starting this tool.

| Starting the Hardware Debugger | Workstation | PC |
|---|---|---|
| From the Design Manager | Yes | Yes |
| From the Command Line | Yes (UNIX) | Yes (DOS) |
| As a Standalone Tool | No | No |

## From the Design Manager

To start the Hardware Debugger from the Design Manager, follow these steps.

1. To run the Hardware Debugger on a specific design, select the desired revision in the Design Manager project view.
2. In the Design Manager toolbox, click the Hardware Debugger button, shown in the following figure.



## Standalone Tool

If you installed the Hardware Debugger as a standalone tool on a PC, click the Hardware Debugger icon (shown in the previous figure) on the Windows desktop or select hwdebugr.exe from the Windows 95®, Windows 98®, or Windows NT® Start button.

## From the Command Line

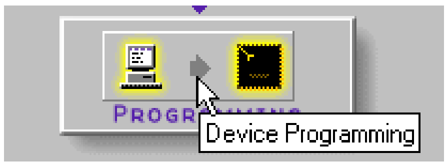
To start the Hardware Debugger from the UNIX® or DOS™ command prompt, enter the following command.

```
hwdebugr
```

## From the Foundation Project Manager

To launch the Hardware Debugger from the Foundation Project Manager, follow these steps.

1. Select the Device Programming icon in the Programming box on the Project Manger’s Flow tab.



2. From the Select Program box, choose the Hardware Debugger.

For more information about accessing and using the Hardware Debugger from the Foundation Project Manager, see the “Verification and Programming” chapter of the *Foundation Series Guide*.

## Exiting the Hardware Debugger

To exit the Hardware Debugger, select **`File`** → **`Exit`**. If you have an open waveform window, you are asked whether you want to save the data before quitting the application.

## Using the Interface

This section describes the Hardware Debugger interface.

### Main Window

The main window is shown in the following figure. By default, this window displays a title bar, menu bar, toolbar, and status bar. The Debug Control Panel dialog box appears when your design is generated for verification. To hide the toolbar, status bar, or control panel, select the appropriate commands from the View menu.

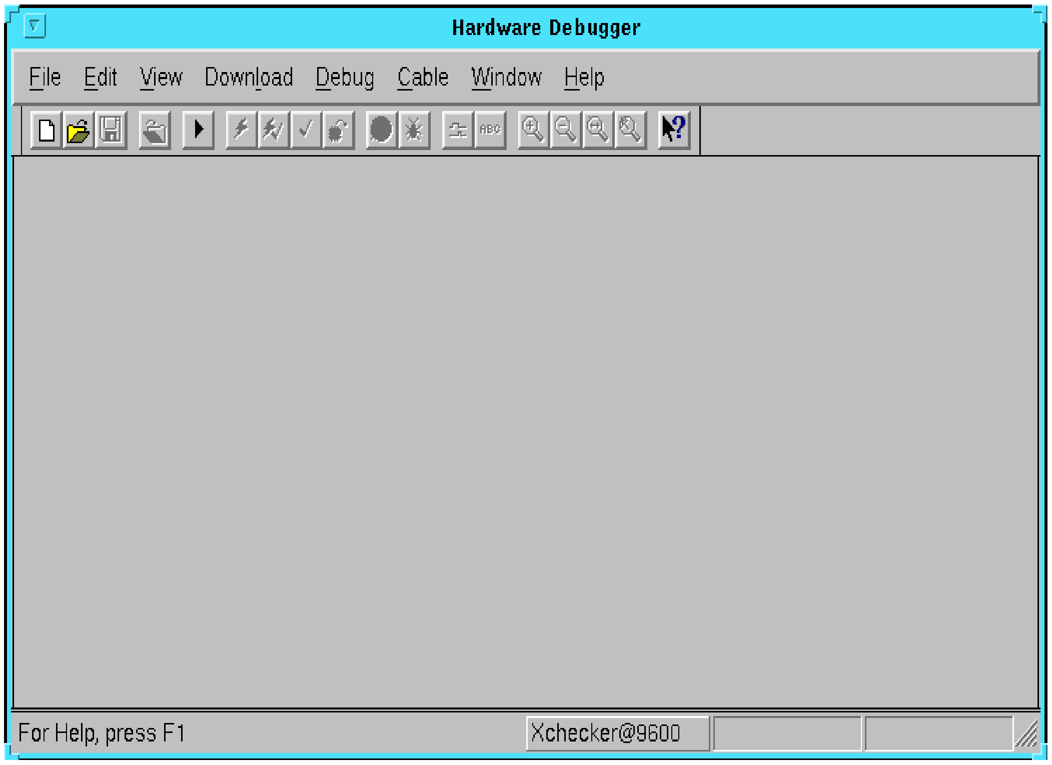


**Figure 2-1 Hardware Debugger Window**

## Title Bar

The title bar displays the program name followed by the name of the currently loaded design.

## Menu Bar

The menu bar, located at the top of the window, includes the File, Edit, View, Download, Debug, Cable, Window, and Help menus. You can also select menu commands by typing the letter underlined in the menu name while holding down the Alt key.

### Toolbar

The toolbar, located below the menu bar, consists of buttons that you can use to execute commands. Place the mouse pointer over each button to display the command associated with the button. The command name appears as a “tool tip” and the status bar provides more descriptive information.

### Status Bar

The status bar, located at the bottom of the Hardware Debugger window, provides command and processing information. When you select a menu command, a brief description of the command’s function appears in the status bar. As the software processes, status messages are dynamically updated and displayed.

### Control Panel

The Debug Control Panel is a dialog box that appears when your design is generated for verification or readback. Use the commands in the Control Panel to control aspects of the debugging session, such as readback snapshots, signal and signal groups displayed, design clocking, and design readback.

## Commands and Dialog Boxes

You communicate with the Hardware Debugger by selecting commands from the menus, the toolbar, or the Debug Control Panel. Alternatively, you can run commands from the Console window. Most commands display dialog boxes in which you specify information and options.

### Common Fields

The fields shown in the following table are common to most dialog boxes.

**Table 2-1 Common Dialog Box Fields**

| Dialog Box Field | Function |
|---|---|
| OK | Closes the dialog box and implements the intended action according to the settings in the dialog box |
| Cancel | Closes the dialog box without effecting any action |
| Help | Displays information on that particular dialog box |

## File Open/File Save Dialog Boxes

The standard file open and file save dialog boxes allow you to load a project file, a saved waveform, or a saved macro. They also allow you to save a waveform or a macro. This type of dialog box includes a file browser.

## Filter Dialog Boxes

Use Filter dialog boxes to specify criteria to select signals and groups for debugging, as described in the following steps.

1. Specify the pattern of the signal names to include in your display list by typing the characters in the field located in the Filter For Signals group. The characters can be alphanumeric or blank characters.
2. Include one or more wildcards (*) to perform a global search on the specified string.

   Precede the character string with a wildcard to retrieve all signal names that end with the string of specified characters.

   Append the wildcard to the character string to retrieve all signal names that start with the specified character string.
3. Click **`Apply`** after specifying the filter criteria.

   The available signals list displays only the signals that match the selection criteria.

4. To clear the filter, click **`Clear`** or backspace over the information in the filter text box.

### Selection Dialog Boxes

Use Selection dialog boxes to specify specific values and selections.

## Selecting Commands and Dialog Box Options

To choose a menu item, a toolbar button, or a dialog box option, you can use the mouse or the keyboard.

### Using the Mouse

1. Move the mouse cursor over the object to select, and click the left mouse button to select the object.

   If you clicked on a toolbar button, a list box or a dialog box appears. If you clicked on a menu, menu options are displayed.

2. To exit a dialog box without making a selection, click **`Cancel`** or double-click the close box in the upper left corner of the dialog box.
3. To obtain help, click **`Help`** in the dialog box.

### Using the Keyboard

You can use the keyboard to select objects on your screen, such as a dialog box button or a menu option.

1. To select a dialog box option, use the **`Tab`** key to position the cursor on that object and highlight it. Press the **`Enter`** key to process the selection.

   To exit a dialog box without making a selection, press the **`Esc`** key.

2. To choose a menu and display its commands, press the **`Alt`** key and the appropriate underlined letter key corresponding to the menu you want. For example, press **`Alt F`** to select the File menu.
3. Use the arrow keys to scroll down the list of commands in a menu or the options in a list box. Press **`Enter`** when the selection you want to use is highlighted or, in the case of a menu item,

press the underlined letter corresponding to the menu command. For example, press the **N** key to select the New command of the File menu.

## Using Help

The Hardware Debugger includes a context-sensitive help and a Help menu. You can obtain help on commands and procedures through the Help menus or by selecting the Help toolbar button. In addition, the dialog boxes associated with many commands have a Help button that you can click to obtain context-sensitive help.

### Help Menu

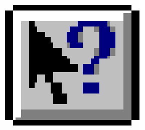
Use the following Help menu commands to get help.

- The Help Topics command opens the online help and lists the various topics available for the Hardware Debugger. From the Help Topics page, you can jump to command information or step-by-step instructions for using the Hardware Debugger. When you want to return to the help topic list, click the Help Topics button.
- The Online Documentation command provides access to the online documentation.
- The About Hardware Debugger command displays a popup window that shows the version number of the Hardware Debugger software and a copyright notice.

### Toolbar Help Button

You can access context-sensitive help as follows.

1. Click the Help button in the toolbar.



The cursor changes to a question mark.

2. Click once with the left mouse button on the menu item or toolbar button for which you want help.

   The Hardware Debugger displays help for the selected command or option.

**Note:** You can also press **`Shift F1`** to obtain context-sensitive help.

## F1 Key

Pressing the F1 key on a dialog box displays help on that dialog box. Pressing the F1 key is the same as selecting Help → Help Topics, if no dialog boxes are displayed.

## Help Button in Dialog Boxes

Many of the dialog boxes in the Hardware Debugger have a Help button that you can click to get help on the dialog box options. You can also press **`Alt H`** or **`F1`** on your keyboard with the cursor positioned over the dialog box to access the online help.

Chapter 3

# Design Preparation

This chapter describes how to prepare a design for use with the Hardware Debugger and how to generate the proper configuration files. The first section covers the special components needed to perform some of the Hardware Debugger operations. The second section discusses the various file types used by the Hardware Debugger and the options that must be specified to create these files correctly. This chapter includes the following sections.

- "Creating a Design"
- "Generating Configuration Data Files"

## Creating a Design

To create a design for use with the Hardware Debugger, follow the design generation instructions for the type of operation you want to complete for your specific device.

This section provides instructions on starting the Hardware Debugger on the workstation or the PC.

### Preparing a Design for Downloading

When using the Hardware Debugger to download a design only, you do not need any special symbols in the design. You can use the default values for all options.

### Preparing a Design for Verification and Debugging

Readback is the process of reading a bitstream from an FPGA. The bitstream contains configuration information and information about the state of the design. You can use the readback bitstream to verify the configuration and probe the internal states of your design.

### XC3000

An XC3000 design does not need to be modified at the schematic or HDL level in order to perform readback. For XC3000 devices, a readback is initiated when a Low to High transition is applied to the M0/RTRIG pin. After the readback begins, the serial readback data is presented on the M1(RDATA) pin.

### XC4000/XC5200

To provide more flexibility, the XC4000/XC5200 readback signals (RTRIG and RDATA) can be assigned to any of the user programmable device pins as well as the M0 and M1 pins. Because the readback signals are user programmable, you must use the READBACK component in the design when using the Hardware Debugger to verify or debug an XC4000 or an XC5200 device.

To prepare the design for verifying or debugging, use the following steps.

1. Include the READBACK symbol in the design schematic.
2. Connect IPAD, OPAD, IBUF, and OBUF primitives to the TRIG and DATA pins of the READBACK macro as shown in the "READBACK Symbol" figure. You can then lock the IPAD and OPAD components to any of the user-programmable I/O locations.

**Note:** If you want TRIG and DATA to correspond to the mode pins M0 and M1, replace the IPAD and OPAD primitives with the special primitives MD0 and MD1.

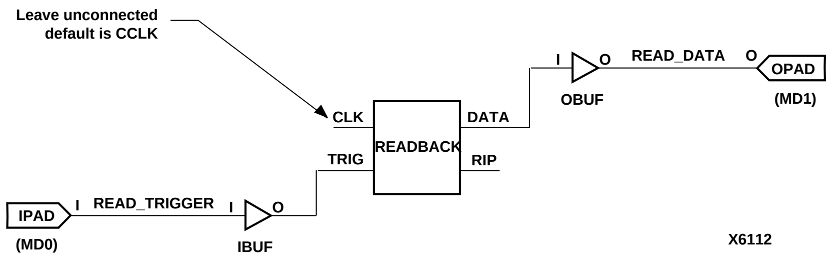


**Figure 3-1 READBACK Symbol**

3. If you plan to have the Hardware Debugger reset the flip-flops in the design, include the STARTUP symbol and connect the GSR (XC4000) or GR (XC5200) input pin to an unused input of the target device. You will later connect the input to the RESET pin on the XChecker cable.

**Note:** Because the XC4000 / XC5200 reset is active-High and the Hardware Debugger assumes active-Low, the signal sent from the Hardware Debugger through the XChecker Cable or MultiLINX Cable must be inverted as shown in the following two figures.

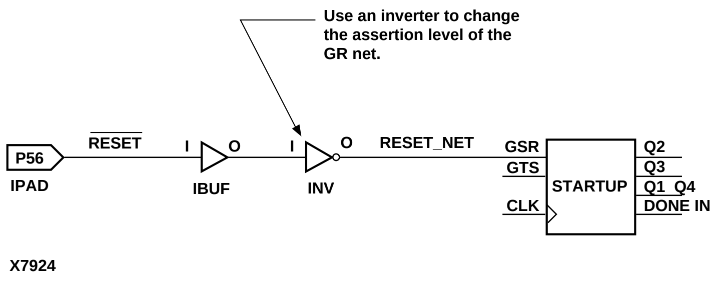

Use an inverter to change
the assertion level of the
GR net.
P56
IPAD
$\overline{\text{RESET}}$
I
O
IBUF
I
O
INV
RESET_NET
GSR
GTS
CLK
STARTUP
Q2
Q3
Q1 Q4
DONE IN
X7924

**Figure 3-2 XC4000 STARTUP Symbol**

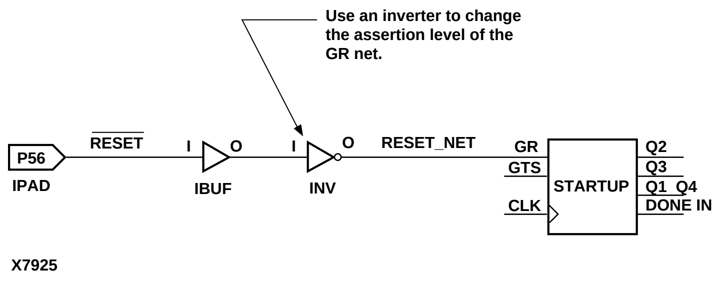


**Figure 3-3 XC5200 STARTUP Symbol**

The inverter preceding the STARTUP symbol implements the active-Low reset asserted by the Hardware Debugger.

### Spartan

The Spartan device uses an internal configuration clock, CCLK, when configuring in a master mode. The configuration rate option allows you to select the rate for this clock. The default is Slow. Refer to the “Spartan Configuration Options” section of the *Design Manager/Flow Engine Guide* more information.

## Generating Configuration Data Files

After the design is placed and routed, you must generate the configuration data files. The Hardware Debugger can download to a chain of multiple devices, known as a daisy chain, as well as to an individual device. The Hardware Debugger can only verify the configuration data and probe the internal states of a single device at a time. It is possible to verify and debug a device configured on a daisy chain only if you connect the XChecker cable directly to it. This process of probing a configured device is also known as read capture.

To generate a bitstream, open the implementation revision from the Design Manager. If you do not have a project for your design, use the Design Manager to create a project for that design. If you have a project for your design but the project has not been updated, go ahead and implement the designs using the **`Design`** → **`New Version`** command as explained in the “Implementing a Design from the Design Manager” section of the *Design Manager/Flow Engine Guide*.

For detailed information on the various implementation and configuration options, see the "Implementation Flow Options" chapter of the *Design Manager/Flow Engine Guide*.

## Creating Files for a Single XC3000 Device

To configure an XC3000 device, you need a bitstream. The Hardware Debugger accepts a bitstream as a BIT file, an RBT file, or a PROM file. To verify and / or debug a design, you must use a BIT file and have a logic allocation file (*design_name*.ll) in your design directory. The Design Manager generates these files by default.

### Creating Downloadable Files (XC3000)

Follow these steps to prepare your XC3000 designs for downloading.

1. From the Design Manager, click the left mouse button on the implementation revision, as shown in the following figure.

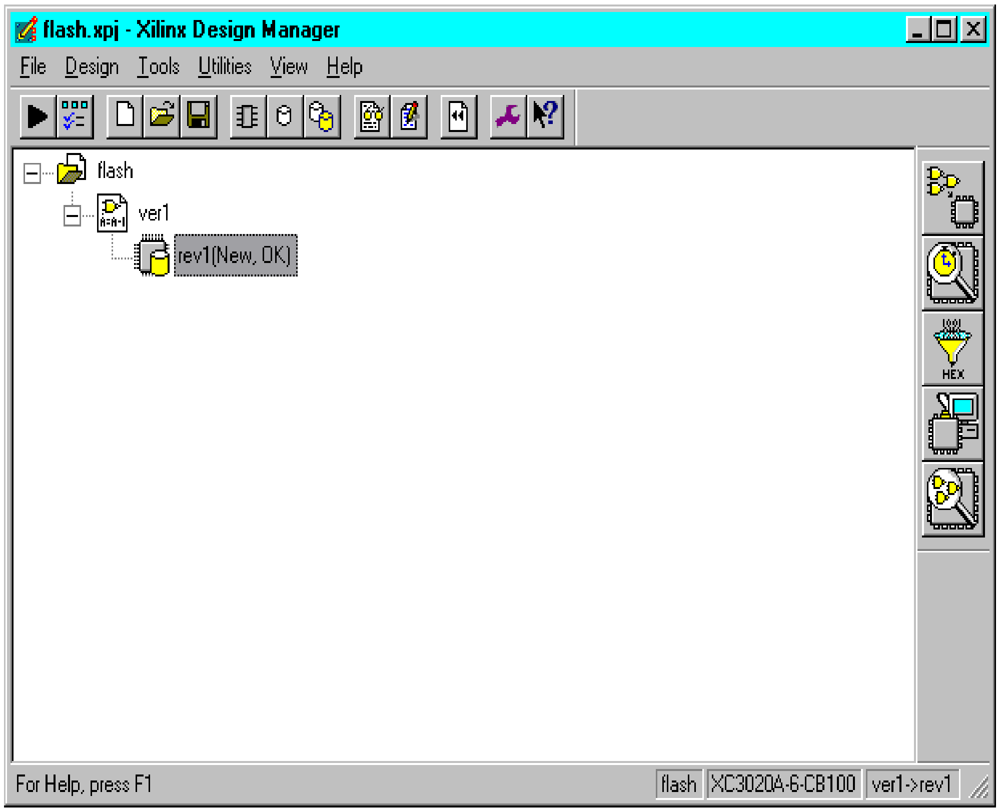
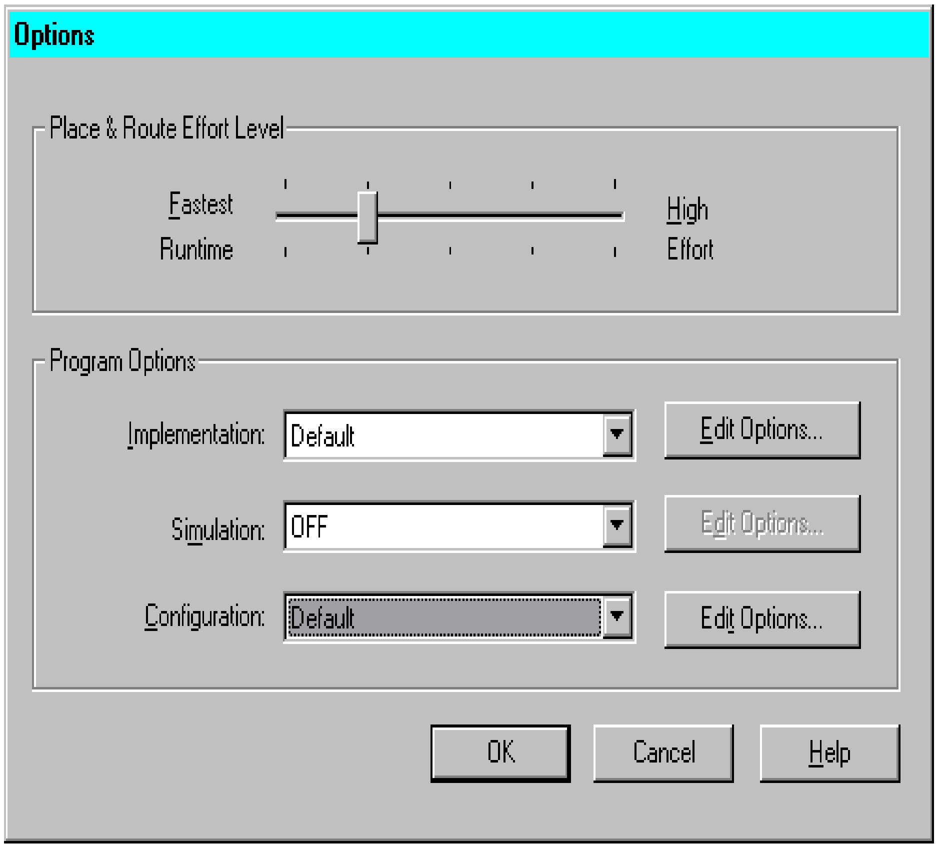


**Figure 3-4 XC3000 Implementation Revision in the Design Manager**

2. Select **`Design`** → **`Options`**.

   The Options dialog box appears, as shown in the following figure.


Options

Place & Route Effort Level

Program Options

Implementation: Default

Simulation: OFF

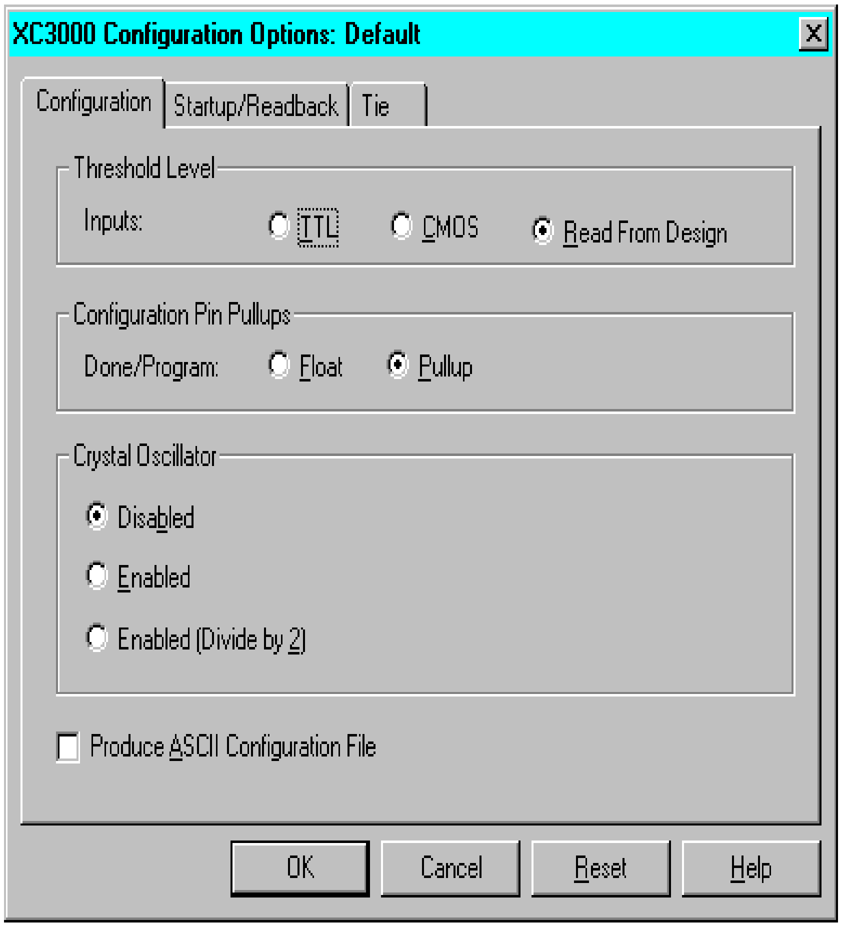
Configuration: Default

**Figure 3-5 Design Manager Options Dialog Box**

3. In the Program Options field, select **`Default`** from the Configuration drop down list.

4. Click the **`Edit Options`** button corresponding to the configuration template.

   The Configuration Options dialog box appears, as shown in the following figure.



XC3000 Configuration Options: Default

Configuration | Startup/Readback | Tie

Threshold Level

Inputs: TTL [✓ Read From Design] CMOS

Configuration Pin Pullups

Done/Program: Float [✓ Pullup]

Crystal Oscillator

[✓ Disabled]

Enabled

Enabled (Divide by 2)

☐ Produce ASCII Configuration File

**Figure 3-6 Design Manager Configuration Options Dialog Box**

5. In the Configuration tab, select **`Pullup`** next to the **`Done/ Program`** pin in the Configuration Pin Pullups group box to enable a 2 to 8 kilohm pull-up resistor on the D/P pin.

6. Click **`OK`** to return to the Options dialog box or if you want to enable the readback options, continue with the "Creating Files for Verification and Debugging (XC3000)" section.

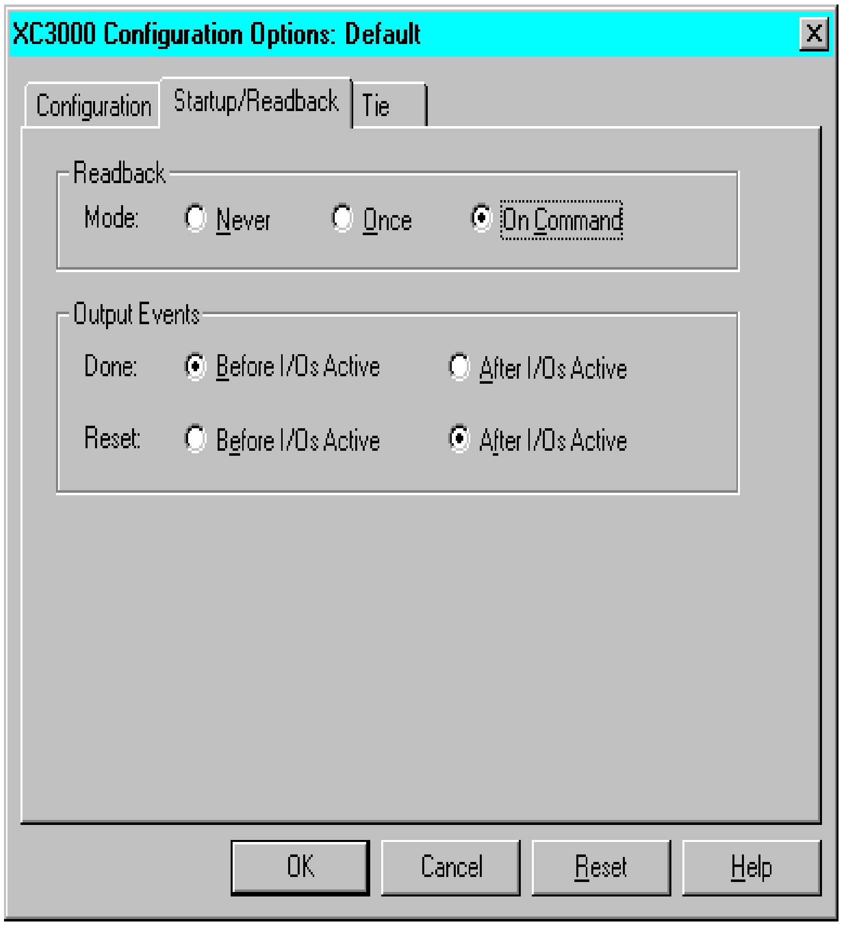
7. In the Options dialog box, click **`OK`**.

8. In the Design Manager, now implement the design, **`Design`** → **`Implement`**.

## Creating Files for Verification and Debugging (XC3000)

Follow these steps to implement XC3000 designs for verification and debugging.

1. Follow steps 1 through 8 in the preceding “Creating Downloadable Files (XC3000)” section to enable a pull-up resistor for the D/P pin for device configuration.
2. Select the Startup/Readback tab of the Configuration **`Options`** dialog box. The Startup/Readback tab appears, as shown in the following figure.


XC3000 Configuration Options: Default

Configuration | Startup/Readback | Tie

Readback

Mode: ( ) Never ( ) Once (●) On Command

Output Events

Done: (●) Before I/Os Active ( ) After I/Os Active

Reset: ( ) Before I/Os Active (●) After I/Os Active

**Figure 3-7 Configuration Options Startup/Readback Tab**

3. Enable the readback capability by choosing **On Command** for the Readback option.

   The software generates a logic allocation file (*design_name*.ll). The Hardware Debugger uses the *design_name*.ll file to identify bits in the readback bitstream that represent the values of design I/Os, latches, and flip-flops.

4. Click **OK** in the Configuration Options dialog box to return to the Options dialog box.

5. In the Options dialog box, click **`OK`**.
6. In the Design Manager, now implement the design, **`Design`** → **`Implement`**.

## Creating Files for a Single XC4000, XC5200 or Spartan Device

To configure an XC4000, XC5200 or Spartan device, you need a bitstream, which can be either a BIT file, an RBT file, or a PROM file. To verify and / or debug a design, you must use a BIT file and have a logic allocation file (*design_name*.ll) in your design directory. Use the Design Manager to generate the necessary configuration files.

### Creating Downloadable Files (XC4000/XC5200/ Spartan)

Follow these steps to prepare your XC4000, XC5200 and Spartan designs for downloading.

1. From the Design Manager, click the left mouse button on the implementation revision, as shown in the following figure.

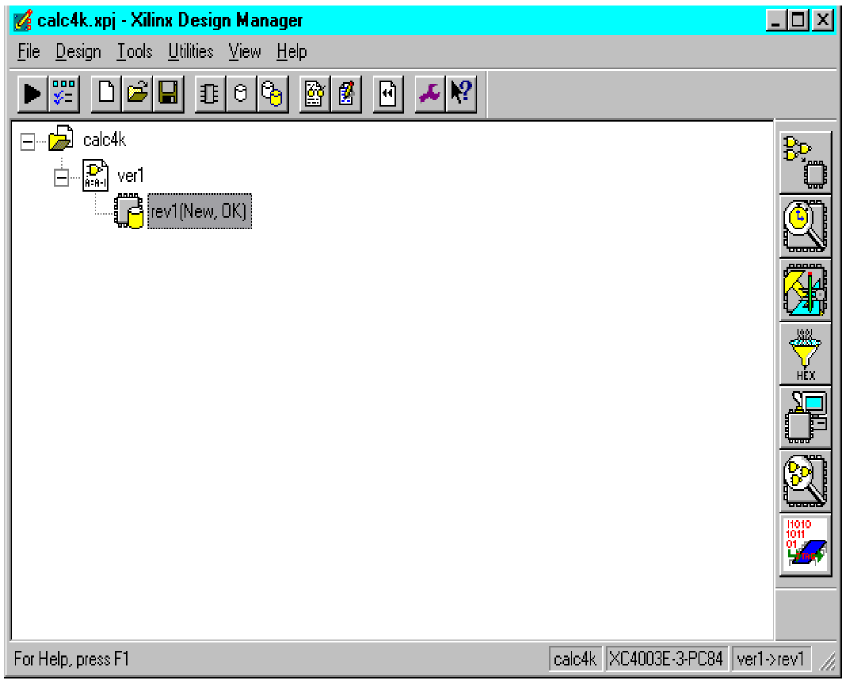


**Figure 3-8 XC4000 Implementation Revision in the Design Manager**

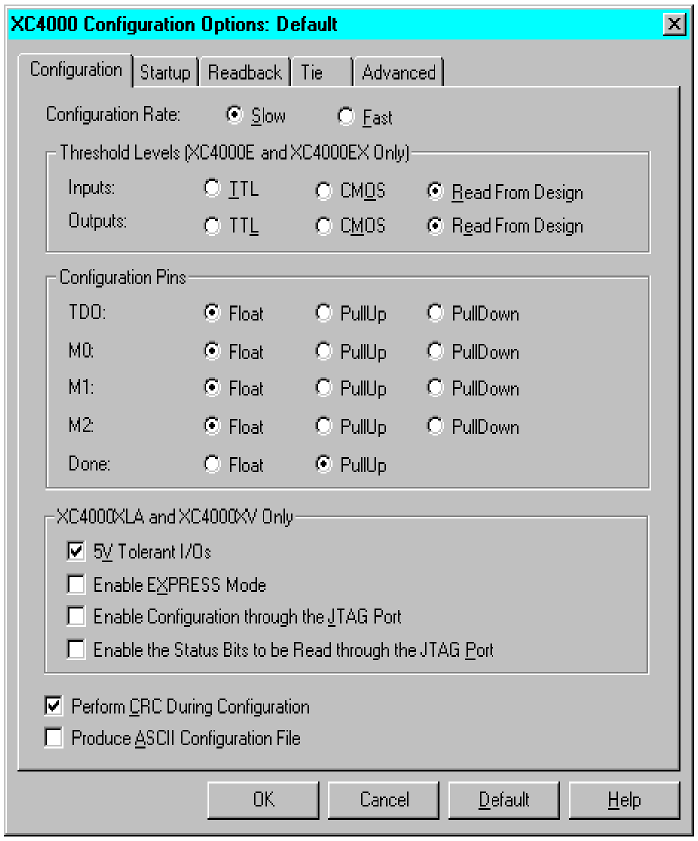
2. Select **`Design`** → **`Options`**.

   The Options dialog box appears, as shown in the “Design Manager Options Dialog Box” figure.

3. In the Program Options Field, select **`Default`** from the configuration drop down list.

4. Click the **`Edit Options`** button corresponding to the configuration template.

5. The Configuration Options dialog box appears, as shown in the following figure.



Configuration Rate: [✓ Slow] Fast

Threshold Levels (XC4000E and XC4000EX Only)

| | | | |
|---|---|---|---|
| Inputs: | TTL | CMOS | [✓ Read From Design] |
| Outputs: | TTL | CMOS | [✓ Read From Design] |

Configuration Pins

| | | | |
|---|---|---|---|
| TDO: | [✓ Float] | PullUp | PullDown |
| M0: | [✓ Float] | PullUp | PullDown |
| M1: | [✓ Float] | PullUp | PullDown |
| M2: | [✓ Float] | PullUp | PullDown |
| Done: | Float | [✓ PullUp] | |

XC4000XLA and XC4000XV Only

- [✓] 5V Tolerant I/Os
- [ ] Enable EXPRESS Mode
- [ ] Enable Configuration through the JTAG Port
- [ ] Enable the Status Bits to be Read through the JTAG Port

- [✓] Perform CRC During Configuration
- [ ] Produce ASCII Configuration File

**Figure 3-9 Design Manager Configuration Options Dialog Box**

6. Select **`PullUp`** next to the DONE pin in the Configuration Pins group box to enable a pull-up resistor for the DONE pin.

7. This step is optional. Select **`Perform CRC During Configuration`** (default) to perform a cyclic redundancy check of your bitstream during configuration.

   CRC bits are checksum bits that the FPGA uses to verify that the bitstream transmitted correctly.

8. This step is optional. Select **`Produce ASCII Configuration File`** to create a rawbits text (RBT) file, which is an ASCII representation of your configuration bitstream.

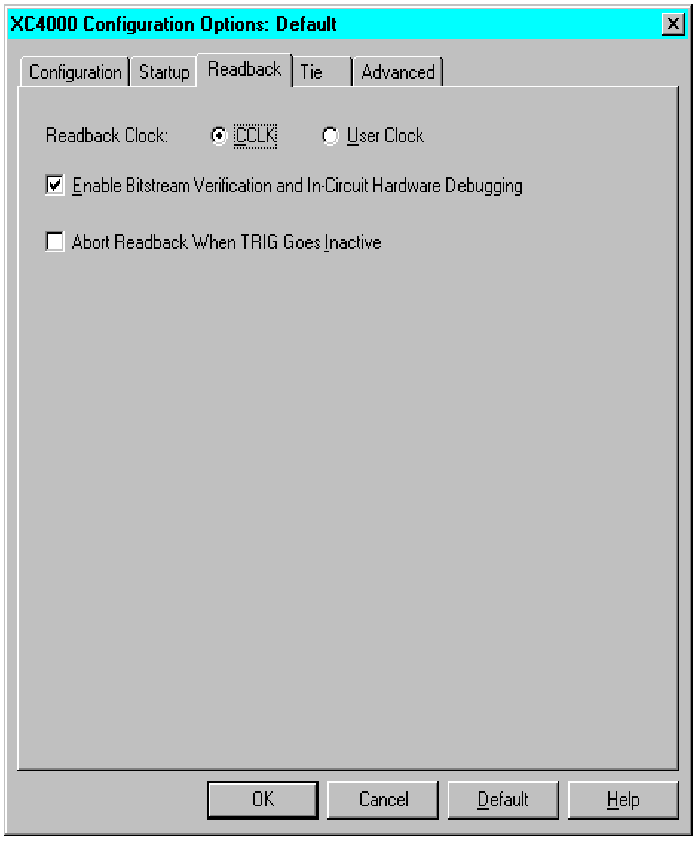
You can use the RBT file for download or to visually inspect your bitstream.

9. Click **`OK`** to return to the Options dialog box or if you want to enable the readback options, continue with the "Creating Files for Verification and Debugging (XC4000 / XC5200)" section.
10. In the Options dialog box, click **`OK`**.
11. In the Design Manager, now implement the design, **`Design`** → **`Implement`**.

## Creating Files for Verification and Debugging (XC4000/XC5200/Spartan)

Follow these steps to implement XC4000, XC5200 and Spartan designs for verification and debugging.

1. Follow steps 1 through 11 in the preceding "Creating Downloadable Files (XC4000 / XC5200 / Spartan)" section to enable a pull-up resistor for the D / P pin for device configuration.
2. In the Configuration Options dialog box, select the Readback tab. The Readback tab of the Configuration Template dialog box appears, as shown in the following figure.


XC4000 Configuration Options: Default

Readback Clock: ● CCLK ○ User Clock

[✓] Enable Bitstream Verification and In-Circuit Hardware Debugging

[ ] Abort Readback When TRIG Goes Inactive

**Figure 3-10 Configuration Options Readback Tab**

3. Select **`CCLK`** as the readback clock.

4. Select **`Enable Bitstream Verification and In-Circuit Hardware Debugging`**.

   This option generates the logic allocation file (*design_name*.ll). The Hardware Debugger uses the *design_name*.ll file to identify bits in the readback bitstream that represent the values of design I/Os, latches, and flip-flops.

**Note:** You cannot probe I/Os when working with an XC5200 device.

5. Click **`OK`** to return to the Options dialog box.

6. In the Options dialog box, click **`OK`**.
7. In the Design Manager, now implement the design, **`Design`** → **`Implement`**.

## Creating Files for Verification (Virtex)

For more information about creating files for verification using Virtex devices, read the Virtex Configuration and Readback Application Note (http://www.xilinx.com/apps/xapp.htm#xapp138).

## Creating Files for Multiple Devices (Daisy Chains)

To configure a daisy chain of devices, you need a PROM file.

1. Produce the configuration data (BIT files) for each device, referring to the appropriate single device section.
2. Concatenate the device BIT files using the PROM File Formatter. For more information, refer to the *PROM File Formatter Guide*. Ensure that the PROM file contains the bitstreams in the same order as the devices on the target board.

**Note:** You can use a PROM file to download a daisy chain; however, you cannot use it to verify or debug a daisy chain.

Chapter 4

# Connecting Your Cable

This chapter describes the various download cables and the operations you can perform with each cable. It also explains how to connect your cable to your computer and target board, and how to set the target board’s configuration mode.

Additionally, this chapter explains which connections are needed to download, verify, and debug. Information on diagnostics and troubleshooting for the MultiLINX™ Cable and the XChecker™ Cable is also included. This chapter contains the following sections.

- “Cable Support”
- “Cable Basics”
- “Cable Descriptions”
- “Cable Limitations”
- “Cable Baud Rates”
- “Connecting Cable to Host System”
- “Setting Up the Hardware”
- “Connecting Cable to Target System”
- “Setting the Cable Options”
- “Resetting the Cable”
- “MultiLINX Diagnostics”
- “XChecker Diagnostics”
- “Troubleshooting the MultiLINX Cable”
- “Troubleshooting the XChecker Cable”

## Cable Support

The Hardware Debugger supports the following cables.

**Table 4-1 Cable Support**

| Name | Function | Platform |
|---|---|---|
| MultiLINX Cable ( Model: DLC6) | Download, Read-back & Verify | PC[a] |
| XChecker Cable (Model: DLC4) | Download, Read-back & Verify[b] | PC, Workstation |
| Parallel Cable III (Model: DLC5) | Download Only | PC |
| Serial Cable | Download Only | PC, Workstation |

a. The MultiLINX Cable is only supported by the Hardware Debugger on PCs using Window 98/ NT with USB port support.
b. Limitations: You can us the XChecker with devices smaller then 256K bits. The XChecker is not recommended for use with larger devices.

## Cable Basics

The Hardware Debugger communicates with your device via a cable. The cable can be a MultiLINX, XChecker, Parallel, or Serial cable. You can use any one of the four cables to download configuration data to a device or daisy chain of devices, but you can use the MultiLINX Cable to verify and the XChecker Cable to verify and debug.

The MultiLINX Cable supports all devices and has a fast USB communication port. The MultiLINX Cable supports Slave Serial and SelectMAP modes.

Perform the following steps when connecting your cable.

1. Determine the most suitable cable to use based on the tasks you want to perform.
2. Connect the cable to your host system.

3. Configure the target board to accept the needed cable connections.
4. Set your FPGA to slave serial mode configuration using the mode pins.
5. Connect the cable to your target system.
6. Power up the target board.
7. Start the Hardware Debugger.
8. Set the cable options from the Cable menu.

## Cable Descriptions

This section describes the cables supported by the Hardware Debugger. There are four different types of cables: MultiLINX, XChecker, Parallel, and Serial.

For detailed cable information, refer to the "Cable Hardware" chapter of the *Hardware User Guide*.

### MultiLINX

You can use the MultiLINX Cable to download and verify. The MultiLINX Cable hardware communicates with the host over the Universal Serial Bus (USB) at up to 12M bits/sec, or at variable baud rates over an RS-232 interface at up to 57600 bits/sec.

For detailed MultiLINX Cable information, refer to the "MultiLINX™ Cable" chapter of the *Hardware User Guide*. You can access the following mentioned application notes with descriptions of device-specific design techniques and approaches from the support page at (http://support.xilinx.com/xapp/xapp168.pdf).

The "Getting Started with the MultiLINX Guide" application note is a quick reference to everything you need to know to use the MultiLINX Cable.

- Describes using a USB port, Mixed Voltage environments, connections for all the supported Modes.
- Describes how to setup a Prototype application for use with the MultiLINX Cable.

- Describes all the cables, their capabilities, and associated software tools.

Communication between the host system and the MultiLINX Cable is dependent on host system capability. The following table lists the valid baud rates for the supported platforms.

**Table 4-2 Valid Baud Rates**

| Baud Rates | | |
|---|---|---|
| **Cable** | **PC** | **WorkStation** |
| MultiLINX Cable (USB) | 1M-12M (Currently USB is supported only on Win98/95C.) | USB is currently not supported on the WorkStation. |
| MultiLINX Cable (RS-232) | 9600, 19200, 38400, and 57600 | 9600, 19200, and 38400 |

## XChecker Cable

You can use the XChecker Cable to download, verify, and debug. The XChecker Cable assembly houses the following internal circuitry.

| | |
|---|---|
| Xilinx FPGA | Functions as an interface between the XChecker software and the target FPGA. |
| Static RAM | Stores the configuration data for download and readback. |
| Oscillator circuit | Provides a system clock to facilitate download and readback of configuration data. |

The XChecker Cable requires a standard DB-9 or DB-25 RS-232 serial port and may require a DB9/DB25 adapter. The cable has 14 signal connections, plus VCC and GND. It comes with two header connectors and two flying lead connectors. The following figure shows top and bottom views of the cable.

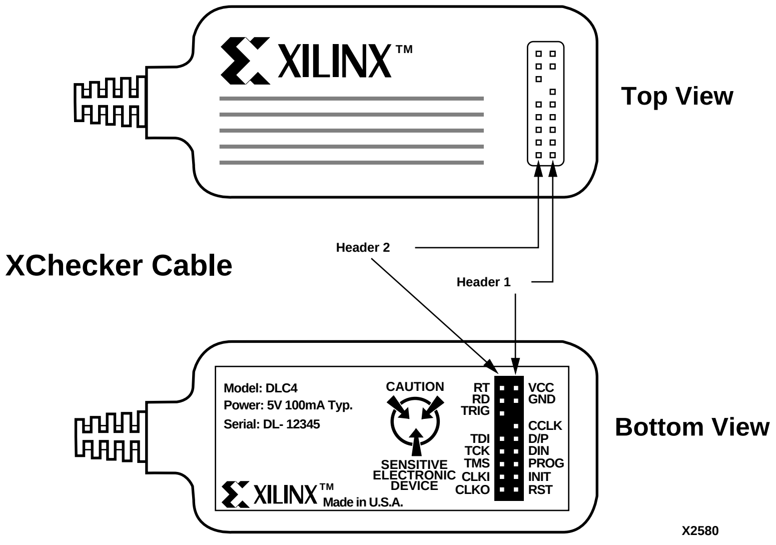


**Figure 4-1 XChecker Cable**

You can use the XChecker Cable with a single FPGA or several connected in a daisy chain to download configuration data. When used to read back data or as a logic probe, the XChecker Cable can only be used with one device at a time. The XChecker Cable transmits configuration data to all target FPGAs at 921 kHz.

Communication between the host system and the XChecker Cable is dependent on host system capability. The following table lists the valid baud rates for the supported platforms.

**Table 4-3 Valid Baud Rates**

| | Baud Rate | | | |
|---|---|---|---|---|
| **Platform** | **9600** | **19200** | **38400** | **115200** |
| Sun | X | X | X | |
| HP | X | X | X | X |
| PC | X | X | X | X |

X indicates supported baud rate.

## Additional XChecker Hardware

The XChecker hardware is the cable assembly with internal logic as described in the previous section, a test fixture, and a set of headers to connect the cable to your target system. In addition, a 3 V Adapter for use with low-voltage parts is available as optional equipment. The following figure shows the XChecker Cable hardware and accessories.

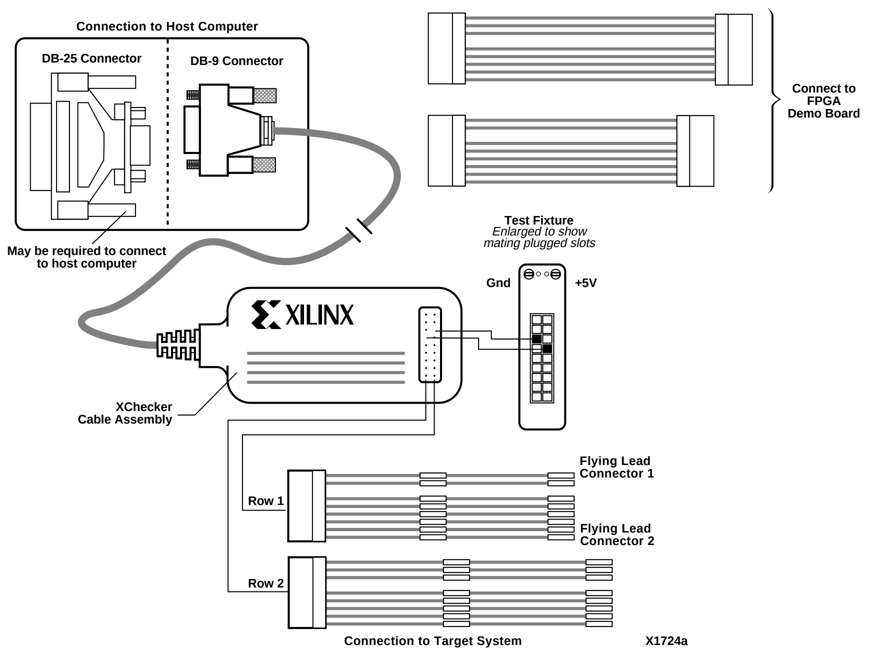

Connection to Host Computer
DB-25 Connector
DB-9 Connector
May be required to connect to host computer
Connect to FPGA Demo Board
Test Fixture
Enlarged to show mating plugged slots
Gnd
+5V
XILINX
XChecker Cable Assembly
Flying Lead Connector 1
Flying Lead Connector 2
Row 1
Row 2
Connection to Target System
X1724a

**Figure 4-2 XChecker Hardware and Accessories**

The optional 3 V adapter accepts VCC supply voltages from the target system from +2.9 V to +5.25 V. The 3 V adapter contains a voltage step-up circuit that generates the 5 V supply voltage needed by XChecker.

Because the 3 V adapter can accept input voltages up to 5.25 V, there is no need to remove the adapter when moving the XChecker Cable between low-voltage systems and higher 5 V systems. Except for the voltage conversion, the 3 V adapter is completely invisible to the XChecker hardware or the target system.

## Parallel Cable

You can use a parallel cable to download configuration data. The Xilinx parallel cable has a 6-lead flying header connector with VCC, GND, CCLK, D / P, PROG, and DIN, and also includes a 25-pin male connector.

**Note:** The parallel cable is supported on the PC only.

### Serial Cable

You can use a serial cable to download configuration data. This cable has a 6-lead header connector with VCC, GND, CCLK, D/P, PROG, and DIN, and also includes a female RS-232 serial connector.

## Cable Limitations

The MultiLINX Cable should be compatible in supporting Readback & Verify for all the FPGAs supported by the XChecker Cable. In addition to the supported devices, the MultiLINX Cable will support the devices that were not supported by the XChecker Cable since the MultiLINX Cable has no RAM size limitations. These devices include those devices in 4000E, 4000XL, and SPARTAN whose bitfile size is more than 256K bits. The MultiLINX Cable will also support Readback & Verify functions in the new Virtex family.

### XChecker Hardware Drawbacks

- Cannot support readback for devices whose bitfile size is more than 256K bits.
- Only supports RS-232.
- Has less user control (only 2 sets of 8 flying wires each).

### MultiLINX Hardware Advantages

- Fast download, readback and debug using the USB port up to 12M bits/sec.
- More configuration modes are supported.
- Supports both RS-232 ports and USB ports.
- Compatible with the currently supported devices for Readback & Verify.
- Supports new devices that are not supported by XChecker due to RAM size limitation.
- Works at low voltages (3.3V).
- Supports both Slave Serial and SelectMAP configuration modes.

## Cable Baud Rates

The supported Baud Rates for all four cables are shown in the following table.

**Table 4-4 Cable Baud Rates**

| Cable | PC | WorkStation |
|---|---|---|
| MultiLINX Cable (USB) | 1 M-12 M | USB is currently not supported on WorkStation. |
| MultiLINX Cable (RS-232) | 9600, 19200, 38400, and 57600 | 9600, 19200, and 38400 |
| XChecker Cable | 9600, 19200, 38400, and 115200 | 9600, 19200, and 38400 |
| Serial Cable | 9600, 19200, and 38400 | 9600, 19200, and 38400 |
| Parallel Cable | 9600 | Not supported on the WorkStation. |

## Connecting Cable to Host System

To install the cable, you must first connect it to the host system.

### MultiLINX Cable

If you have a MultiLINX Cable, connect it to the USB or RS-232 port. A DB-9/DB-25 adapter may be required to connect the cable to your serial port. If you have a different serial port connection, you need to use the appropriate adapter.

### Parallel Cables

If you have a parallel cable, connect it to the parallel port.

### Serial and XChecker Cables

Connect your serial or XChecker Cable to your system's RS-232 serial port. A DB-9/DB-25 adapter may be required to connect the cable to your serial port. If you have a different serial port connection, you need to use the appropriate adapter.

## Setting Up the Hardware

When using the MultiLINX Cable the default mode is slave serial. For PROM files and Virtex bitfiles, the Communication Setup dialog box will popup when download is attempted.

When using the XChecker, parallel, or serial cables you must set up the configuration mode of the devices being configured as slave serial. You must set M0, M1, and M2 to VCC. If you intend to use them as user I/Os, use 4.7 kΩ pull-ups. Refer to *Development System Reference Guide* and *The Programmable Logic Data Book* for information on how to set the mode pins.

## Connecting Cable to Target System

This section covers cable connection to the target device. You need appropriate pins on the target system for connecting the target system board to the header connectors on the cable.

**Warning:** The cable draws its power from the target system through VCC and GND. Therefore, power to the cable, as well as to the target FPGA, must be stable. Do not connect any signals before connecting VCC and GND. The input/output pins of the internal XChecker FPGA should always be at a potential that is lower or equal to their respective rail voltage to avoid internal damage.

### MultiLINX Cable Connectors

The MultiLINX Cable supports four types of flying lead connectors. For detailed information on the MultiLINX Flying Wires and cable modes refer to the “MultiLINX™ Cable” chapter of the *Hardware User Guide*.

### XChecker Cable Connectors

The XChecker Cable supports two types of connectors. You can connect to the pins of your target FPGA with a flying lead connector and to the FPGA demonstration boards with a header connector.

- Flying lead header connectors have eight standard individual female connectors on one end that connect with 0.025 inch square male pins. Each lead is labeled to identify the pin.
- Header connectors are standard 9-pin (8 signals, 1 key) header connectors that fit 0.025 inch square male pins. The pin layout is shown in the "XChecker Cable" figure. The header connectors are keyed to ensure that they are properly inserted into the cable assembly.

Each type of connector, has two different subconnectors. One is keyed for downloading signals; the second is keyed for readback signals. Header 1 is the download subconnector and fits on the outer-most subconnector socket. Header 2 is the readback subconnector and fits on the inner subconnector socket.

## XChecker Cable Pins

Refer to the tables in this section for information on how to connect the signal pins for specific applications. The "XChecker Operation Mode Connections" table shows the necessary connections for each application type, and the "Cable Connections and Definitions" table describes the pins and how to connect them.

**Note:** Not all of the signal pins are required for each function.

**Table 4-5 XChecker Operation Mode Connections**

| Cable Header | Pin Name | Download | Verification | Synchronous Logic Probe | Asynchronous Logic Probe |
|---|---|---|---|---|---|
| 1 | VCC | X | X | X | X |
| 1 | GND | X | X | X | X |
| 1 | CCLK | X | X | X | X |
| 1 | D/P | X | | | |
| 1 | DIN | X | | | |
| 1 | PROG (XC4000 only) | X | | | |
| 1 | INIT (XC3000/ XC4000 only) | X | | | |

**Table 4-5 XChecker Operation Mode Connections**

| Cable Header | Pin Name | Download | Verification | Synchronous Logic Probe | Asynchronous Logic Probe |
|---|---|---|---|---|---|
| 1 | RST | Opt | Opt | Opt | Opt |
| 2 | RT | | X | X | X |
| 2 | RD | | X | X | X |
| 2 | TRIG | | | Opt | Opt |
| 2 | TDI | | | | |
| 2 | TCK | | | | |
| 2 | TMS | | | | |
| 2 | CLKI | | | Opt | |
| 2 | CLKO | | | X | |

X = Connect as specified in the “Cable Connections and Definitions” table.

Opt = Optional

**Table 4-6 Cable Connections and Definitions**

| Signal Name | Function | XC3000 | XC4000 | XC5200 |
|---|---|---|---|---|
| VCC | *Power* — Supplies VCC to cable (5 V, 100 mA, typically) | Connect to target system. | | |
| GND | *Ground* — Supplies ground reference to cable | Connect to target system ground. | | |
| CCLK | *Configuration Clock* — Provides configuration clock to target system during configuration and readback | Connect to target system Configuration Clock. Ensure all devices are in slave serial mode if using download cable to download. | | |
| D/P | *Done/Program* — Signals the end of configuration (For XC3000 devices, a High-to-Low transition on D/P coupled with a High to Low on Reset, causes the device to reprogram.) | Connect to $D/\overline{P}$ pin with a 10-50 kΩ pull-up resistor. | Connect to target system DONE pin and rely on internal 2-8 kΩ pull-up resistors. | |

**Table 4-6 Cable Connections and Definitions**

| Signal Name | Function | XC3000 | XC4000 | XC5200 |
|---|---|---|---|---|
| DIN | *Data In* — Provides configuration data to target system during configuration and is tristated at all other times | Connect to target system's lead device DIN pin. | | |
| PROG (XC4000 Only) | *Program* — 300ns or greater Low pulse causes device to reprogram (A Low indicates the device is clearing its configuration memory.) | N / A | Connect to target system $\overline{\text{PROG}}$ with 10-50 kΩ pull-up resistor. | |
| INIT | *Initialize* — Indicates start of configuration for XC3000 / XC4000 parts.<br>A logical zero on this pin during configuration indicates a data error | Connect to target system INIT with a 10-50 kΩ pull-up resistor. | | |
| RST | *Reset* — During configuration, a Low pulse causes XC3000A devices to restart configuration<br>After configuration, this pin can drive Low to reset target FPGA internal latches and flip-flops<br>RST is the active high for XC4000 / XC5200 devices | Connect to target FPGA $\overline{\text{RESET}}$ pin with 10-50 kΩ pull-up resistor. | User-programmable connection; requires a 10-50 kΩ pull-up resistor | |
| RT | *Read Trigger* — XChecker output Hardware Debugger provides Low-to-High transition on RT to initiate readback | Connect to M0 / RTRIG with 10-50 kΩ pull-up resistor. | User-programmable connection; requires 10-50 kΩ pull-up resistor | |

**Table 4-6 Cable Connections and Definitions**

| Signal Name | Function | XC3000 | XC4000 | XC5200 |
|---|---|---|---|---|
| RD | *Read Data* — XChecker input Hardware Debugger receives the readback data through the RD pin after readback is initiated. | Connect to M1/RDATA through pull-up resistor in slave serial configuration mode; requires a 10-50 kΩ pull-up resistor if using I/O pad as input or output | User-programmable connection; requires 10-50 kΩ pull-up resistor if using I/O pad as input or output | |
| TRIG | *System Trigger* — XChecker input High on this pin signals the XChecker electronics to initiate a readback and causes the RT pin to go High | Connect to target system readback trigger and to an external pin if using an external signal to trigger readback. | | |
| TDI<br>TCK<br>TMS | Reserved<br>(These pins can be used for JTAG Programmer device configuration.) | N/A | | |

**Table 4-6 Cable Connections and Definitions**

| Signal Name | Function | XC3000 | XC4000 | XC5200 |
|---|---|---|---|---|
| CLKI | *Clock Input* — Transmits your system clock to the XChecker electronics<br>Clock must be between 120 kHz and 10 MHz<br>Connect this pin to target system clock to synchronize the readback trigger with target system clock | Connect to source of target system clock for synchronous debugging. | | |
| CLKO | *Clock Output* — Drives target system clock<br>Clock can come from either the CLKI pin, or it can be internally generated by the XChecker Cable when CLKI is unconnected | Connect to destination of target system clock for synchronous debugging. | | |

**Note:** XChecker does not drive the configuration mode pins (M0, M1, M2) during configuration. You must specify the logic levels for these pins externally.

## Serial and Parallel Cables

Connect all the pins of your serial or parallel cable for downloading using the guidelines in the "XChecker Operation Mode Connections" table and the "Cable Connections and Definitions" table.

## Connecting the 3 Volt Adapter to the XChecker Cable

If you are using a 3 V target board instead of a 5 V target board, you must connect the 3 V adapter to the XChecker Cable. The 3 V adapter supports VCC supply voltages from your target system that range from 2.9 V to 5.25 V. An internal voltage "step-up" circuit generates the 5 V voltage supply needed by the XChecker Cable.

Aside from the voltage conversion, the 3 V adapter is completely transparent to the XChecker hardware and the target system. Therefore, you do not need to remove the 3 V adapter when moving the XChecker Cable between 3 V and 5 V systems.

The 3 V adapter includes the J1 and J2 connectors.

**Note:** For information on testing the operation of the 3 V adapter, see the “Testing 3 V Adapter Operation” section.

## Connecting the 3 Volt Adapter

Follow these steps to connect the 3 V adapter to the XChecker Cable.

**Warning:** Use standard electrostatic discharge (ESD) precautions when connecting the adapter to your XChecker Cable. The adapter is static sensitive and can be damaged by ESD energy.

1. Ensure that you are adequately grounded before connecting or using the 3 V adapter.
2. Refer to the following figure to position the adapter on top of the XChecker assembly, aligning the adapter 18-pin female connector (J2) with the XChecker 18-pin male connector socket.

   Ensure that the Xilinx logo on the XChecker case and the 3 V adapter silkscreen are oriented the same way.

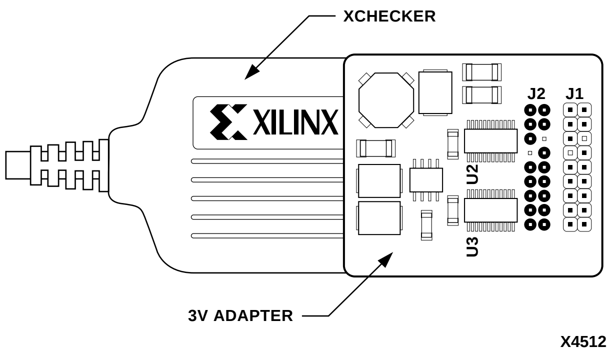


**Figure 4-3 3 V Adapter Connected to XChecker Cable**

3. Holding the adapter board by the edges, press it down until it makes a solid connection to the XChecker Cable.

### Using the 3 Volt Adapter with XC3000L and XC4000XL Parts

To use the 3 V adapter with XC3000L and XC4000XL, follow these steps.

1. Attach the 18-signal flying wire or header block to the J1 connector on the 3 V adapter.

   XChecker operation is unchanged.

2. Attach the XChecker Cable (with the 3 V adapter) to your target system.

The configuration of the XC3000L devices is the same as the XC3000, XC3000A, and XC3100 devices.

The readback clock speed of the XC3000L devices has been slowed because of lower VCC supply voltage. If the supply voltage of the target system is lower than 3 V, you might see an error message similar to the one shown below.

```
XCHECKER? verify

Design design_name has 128 probeable signals.

Readback 1847 bytes of configuration.

Verifying datafile design_name...MISMATCHED Total
of 405 bits mismatched.
```

## Connecting for Download

To connect your cable for downloading only, connect your configuration cable to your target FPGA device. Refer to the “XChecker Operation Mode Connections” table and the “Cable Connections and Definitions” table for pin assignment information. The following figure shows the XChecker Cable connected for downloading only.

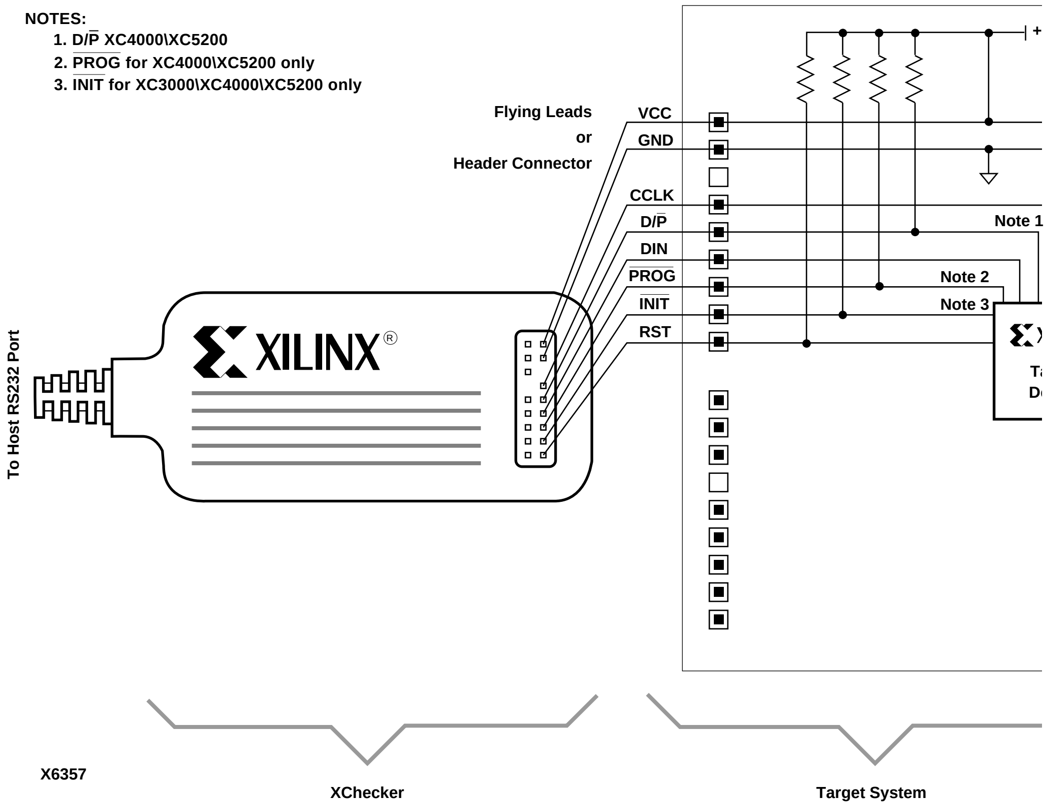


**Figure 4-4 Downloading Configuration Data**

## Connecting for Verification

To connect your cable for downloading and verification, connect your XChecker Cable to your target FPGA as shown in the following figure. Refer to the “XChecker Operation Mode Connections” table for pin assignment information.

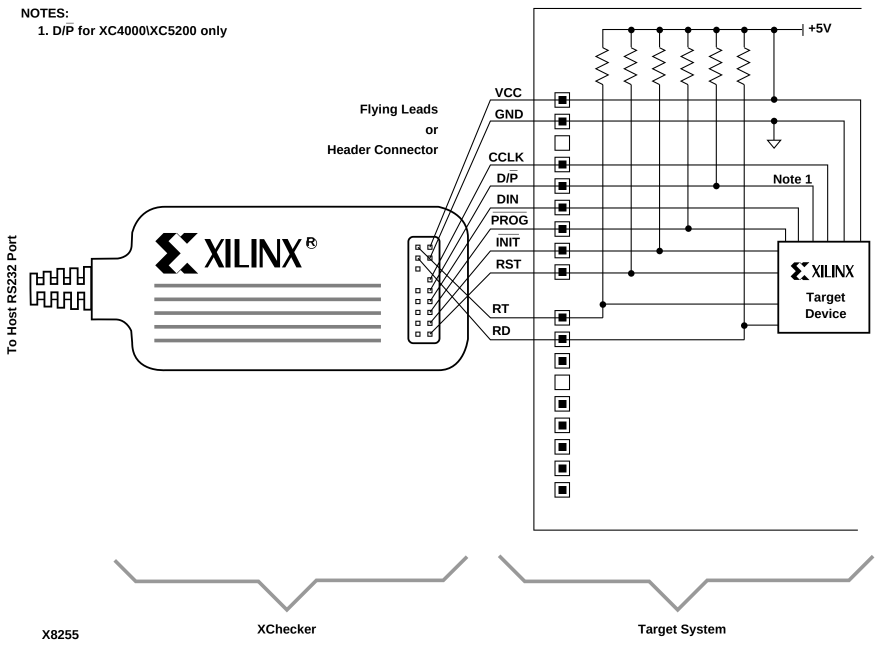


**Figure 4-5 XChecker Cable Connections for Downloading and Verification**

You can also use XChecker to verify a previously configured FPGA. To verify a previously configured FPGA, connect XChecker as shown in the following figure.

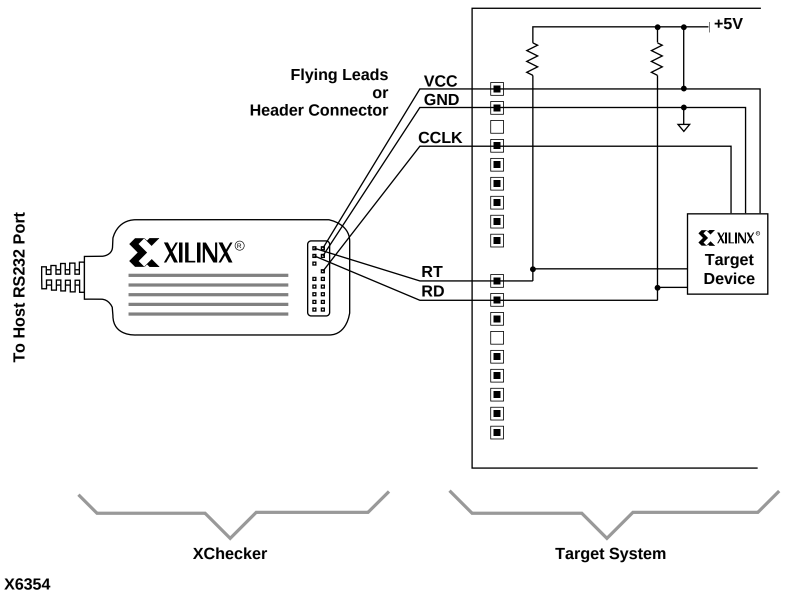


**Figure 4-6 XChecker Cable Connections for Verification Only**

## Connecting RT and RD

Connect the XChecker RT and RD pins to the FPGA RTRIG and RDATA pins, respectively. If you used the symbols MD0 and MD1, consult the device pinout tables in *The Programmable Logic Data Book* for the exact locations of M0 and M1. If you used IPAD / OPAD primitives, consult *The Programmable Logic Data Book* for IPAD and OPAD listings or the I / O Pin Assignments Report available from the Design Manager's Report Browser.

## Connecting for Synchronous Debugging

In synchronous mode debugging, you can control the target FPGA clock through the XChecker Cable. You can control the number of clock pulses applied and the frequency at which the clock cycles occur. For synchronous debugging, connect the TRIG, RT, RD, CLKO and, optionally, CLKI pins of the XChecker Cable to the target FPGA, as shown in the “XChecker Operation Mode Connections” table and the “Cable Connections and Definitions” table. Also, refer to the following figure for connection information. This XChecker Cable configuration allows you to download, verify, and debug your design in the synchronous mode.

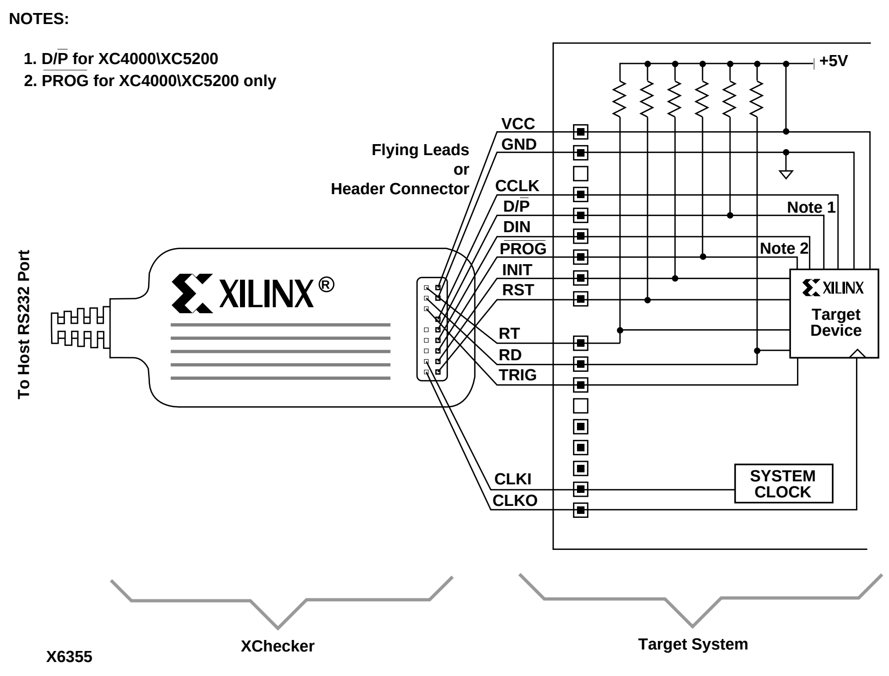


**Figure 4-7 XChecker Cable Connections for Synchronous Debugging**

## Connecting the XChecker Clock

To allow the XChecker clock to control the target FPGA system clock, connect XChecker’s CLKO pin to the pin that you assigned as the FPGA’s clock pin. All the synchronous logic should be connected to this clock source for synchronous debugging.

The source of the CLKO clock signal is from an internally generated XChecker clock oscillator or from an external clock oscillator that you provide.

### Internal Clock

To use the XChecker internal clock, connect the CLKO pin to the target FPGA. Leave the CLKI pin unconnected. Then, select the **`Debug`** → **`Settings`** → **`CLKO Clock Source`** command from the Hardware Debugger menus and select the **`Use XChecker Clock`** setting from CLKO Clock Settings dialog box.

### External Clock

To use an external clock, connect the user clock to the XChecker CLKI pin and connect the XChecker CLKO pin to the FPGA. Then, select the **`Debug`** → **`Settings`** → **`CLKO Clock Source`** command and select the **`Use CLKI`** setting from CLKO Clock Settings dialog box.

### Connecting an External Trigger

To use an external trigger, such as the terminal count of a counter or some other condition in your target board to initiate a readback, connect the external trigger signal to the XChecker TRIG pin. TRIG is active-High.

**Note:** You do not need the TRIG signal if you plan to use an internal trigger, such as the Enter key on the keyboard to initiate a readback.

### Connecting RT and RD

Connect the XChecker RT and RD pins to the FPGA RTRIG and RDATA pins, respectively. If you used the symbols MD0 and MD1, consult the device pinout tables in *The Programmable Logic Data Book* for the exact locations of M0 and M1. If you used IPAD/OPAD primitives, consult the I/O Pin Assignments Report available from the Design Manager's Report Browser.

## Connecting for Asynchronous Debugging

This configuration allows the target system to run while a readback is executed. You do not need to provide the system clock to XChecker, because readback is executed independently of the system clock.

Connect the system clock so that it controls the device flip-flops directly. Thus, the CLKI and CLKO pins on the XChecker Cable are not used for asynchronous debugging.

Connect the external trigger and the RD and RT pins as explained in the previous “Connecting for Synchronous Debugging” section. Also, refer to the following figure for connection information.

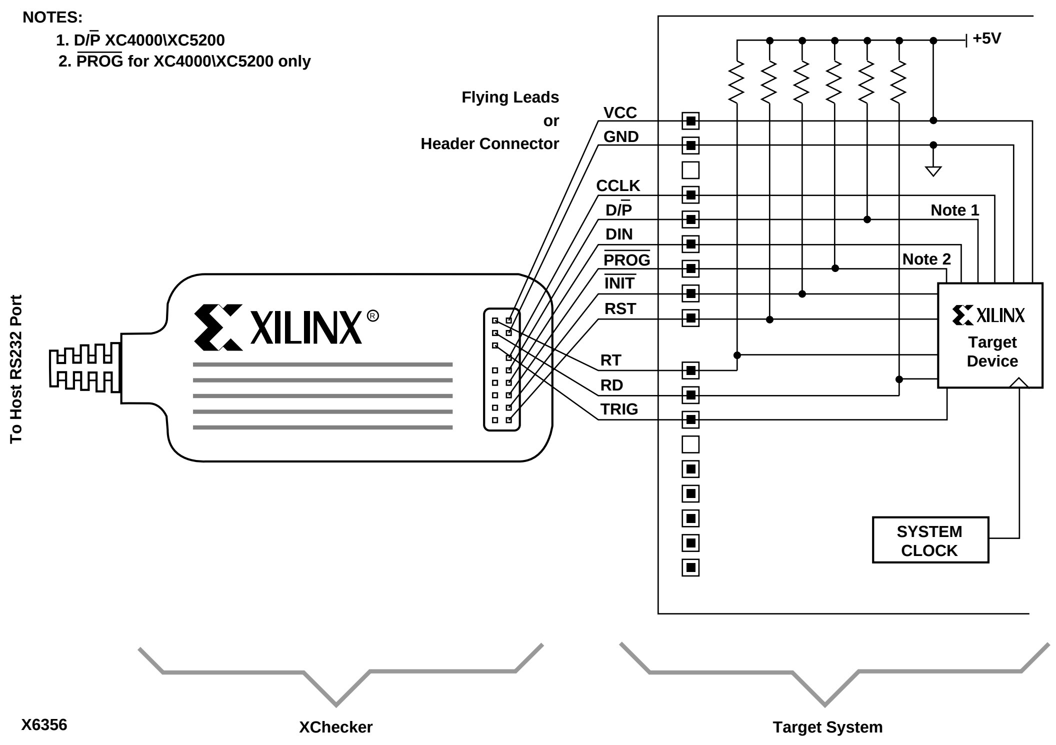
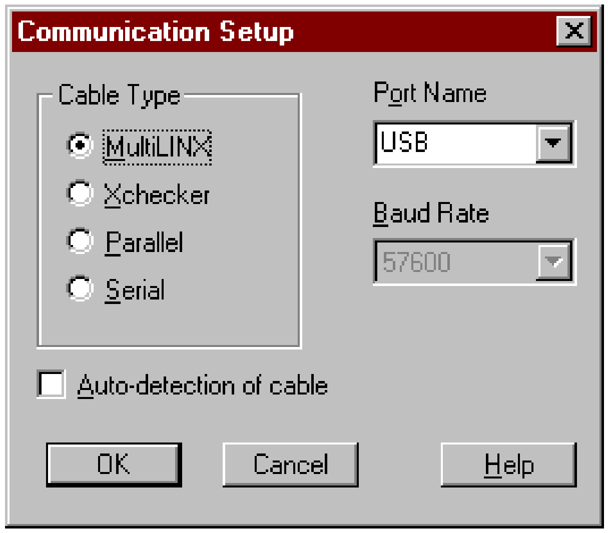


**Figure 4-8 XChecker Cable Connections for Asynchronous Debugging**

## Setting the Cable Options

After connecting the cable to download, verify, or debug, power your target board to enable the software to communicate with the cable and start the Hardware Debugger. You must then set the cable options with the following steps.

1. Select **`Cable`** → **`Communications`** to display the Communication Setup dialog box, shown in the following figure.



Cable Type: [✓ MultiLINX] Xchecker Parallel Serial

Port Name: USB

Baud Rate: 57600

☐ Auto-detection of cable

**Figure 4-9 Communication Setup Dialog Box**

2. In the Cable Type field, select the cable type that you installed for downloading. The Parallel Cable is supported for the PC only.

3. In the Port Name list box, select the port to use for downloading and readback. If the port name you want is not listed, select the blank name from the list box and type in the new port name. This list box saves up to two user-specified port names.

   The Port Name list box contains a list of valid ports for the platform, as shown in the following table. If you selected the MultiLINX Cable the USB port is displayed, If you selected a serial cable or an XChecker Cable, the serial ports are displayed. If you selected a parallel cable, the parallel ports are displayed.

**Table 4-7 USB, Serial, and Parallel Ports**

| Platform | USB Ports | Serial Ports | Parallel Ports |
|---|---|---|---|
| Sol | none | /dev/ttya, /dev/ttyb | none |
| HP 10.*x* | none | /dev/tty0p0, /dev/tty1p0 | none |
| PC | USB | com1, com2, com3, com4 | lpt1, lpt2 |

4. In the Baud Rate list box, select a communications baud rate between the cable and the host system. You cannot specify the baud rate for a parallel cable.

   Communication speed between the host system and the XChecker Cable depends on host system capability. Refer to the "Valid Baud Rates" table for a list of valid baud rates.

5. Select **`Auto-detection of cable`** if you want the software to scan for the presence of a cable and automatically establish communication. Cable information is presented in the status bar.

6. Click **`OK`** to accept the selections and close the dialog box.

## Resetting the Cable

To reset internal logic of the cable after a power glitch select **`Cable`** → **`Reset`** to reset the internal logic of the cable. The cable is reinitialized and the proper baud rate is set.

**Note:** Reconfigure your target device if you experience a power failure to the target device board. This can be done by reconnecting to the cable. The Hardware Debugger reconfigures the cable FPGA if needed, after the connection is established.

## MultiLINX Diagnostics

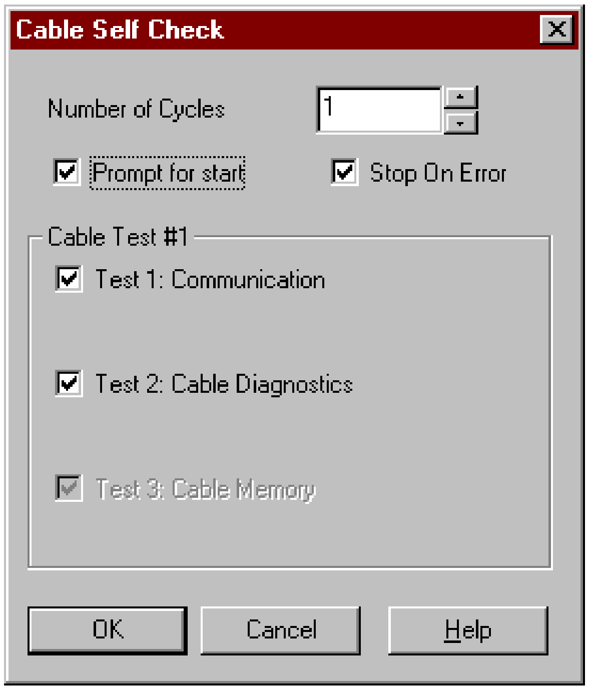
To test the MultiLINX Cable, bring up the MultiLINX Cable Self Check dialog box to perform a diagnostic of the cable.

## Testing the MultiLINX Cable

The MultiLINX Cable contains special hardware for the readback of FPGA devices. To verify that the cable is in working order, follow these steps.

1. Attach the test fixture (provided with your MultiLINX Cable) to the MultiLINX Cable pins. The test fixture is a small printed circuit card with a keyed header connector that fits onto the MultiLINX Cable pins.
2. Start the Hardware Debugger software or, if the Hardware Debugger was running when you installed the test fixture, select **`Cable`** → **`Reset`** to establish communication with the cable.
3. Select **`Cable`** → **`Self Check`** to invoke the Cable Self Check dialog box and perform a diagnostic of the MultiLINX Cable.

   The Cable Self Check dialog box, shown in the following figure is displayed.



Number of Cycles 1

[✓] Prompt for start [✓] Stop On Error

Cable Test #1

[✓] Test 1: Communication

[✓] Test 2: Cable Diagnostics

[✓] Test 3: Cable Memory

**Figure 4-10 Cable Self Check Dialog Box (MultiLINX Cable)**

4. Choose the appropriate settings in the dialog box, such as the number of cycles you want to run and types of tests you want to perform.
5. Click **OK** to start the diagnostic.

**Note:** The MultiLINX Cable does not have a separate cable self check test. The MultiLINX Cable does a BIST operation on power-up.

## XChecker Diagnostics

XChecker includes a test fixture to test the XChecker Cable and 3 V adapter to support 3 V target boards.

## Testing the XChecker Cable

The XChecker Cable contains special hardware for the readback of FPGA devices. To verify that the cable is in working order, follow these steps.

1. Attach the test fixture (provided with your XChecker Cable) to the XChecker Cable pins. The test fixture is a small printed circuit card with a keyed header connector that fits onto the XChecker Cable pins.
2. XChecker draws power from your target system, not from the host system. The test fixture has a red VCC power (+5 V) connector and a black ground connector. Plug these connectors to a 5 V DC power supply.
3. Start the Hardware Debugger software or, if the Hardware Debugger was running when you installed the test fixture, select **`Cable`** → **`Reset`** to establish communication with the cable.
4. Select **`Cable`** → **`Self Check`** to perform a diagnostic of the XChecker Cable.

   The Cable Self Check dialog box, shown in the following figure is displayed.

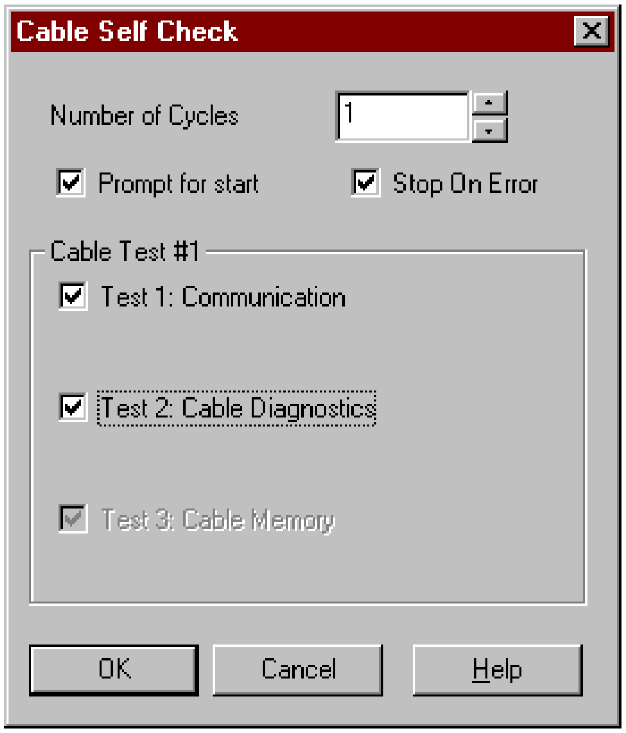


**Figure 4-11 Cable Self Check Dialog Box (XChecker Cable)**

5. Choose the appropriate settings in the dialog box, such as the number of cycles you want to run and types of tests you want to perform. For more information, see the "Cable Self Check Dialog Box" section of the "Menu Commands" chapter.

6. Click **OK** to start the diagnostic.

   A dialog box appears with the following message.

   "`Connect cable and the test header, press OK to start.`"

7. Click **OK**.

   The output is displayed under each test name in the Cable Self Check dialog box. Overall test results are shown in the Cable Self Check (Results) dialog box. For information on test failures, refer to the “Troubleshooting” section of the “Menu Commands” chapter.

## Testing 3 V Adapter Operation

To test the operation of the 3 V adapter, use the test fixture that is shipped with the XChecker Cable, and follow these steps.

1. Plug the adapter into the XChecker case, as outlined in the “Connecting the 3 Volt Adapter to the XChecker Cable” section.
2. Plug the test fixture into the male connector (J1) of the adapter.

   The test fixture is a small printed circuit card with a keyed header connector that fits onto the XChecker Cable pins.
3. XChecker draws power from your target system, not from the host system. The test fixture has two connectors: a red VCC power (+3 V) connector and a black ground connector. Plug these connectors to a 3 V DC power supply.
4. Start the Hardware Debugger software or, if the Hardware Debugger was running when you installed the test fixture, select **`Cable`** → **`Reset`** to establish communication with the cable.
5. Select **`Cable`** → **`Self Check`** from the menu to perform a diagnostic of the XChecker Cable.
6. Click **`OK`** to start the diagnostic.

   This command runs the diagnostics test. The test should produce results that match those obtained by running the same diagnostics test without the 3 V adapter connected to XChecker.

**Note:** The readback clock speed of the XC3000L devices has been slowed because of lower VCC supply voltage.

If the supply voltage of the target system is lower than 3 V, you might see an error message similar to the one shown below.

```
Design design_name has 128 probeable signals.

Readback 1847 bytes of configuration.

Verifying datafile design_name...MISMATCHED

Total of 405 bits mismatched.
```

If this occurs, correct the system voltage and download the design again.

## Troubleshooting the MultiLINX Cable

This section includes solutions to some common problems you might encounter when configuring FPGAs with the MultiLINX Cable.

**Warning:** Connecting the MultiLINX leads to the wrong signal will cause permanent damage to MultiLINX internal hardware. You must connect PWR and GND to ground.

## Troubleshooting the XChecker Cable

This section includes solutions to some common problems you might encounter when configuring FPGAs with the XChecker Cable.

### Improper Connections

**Warning:** Connecting the XChecker leads to the wrong signal will cause permanent damage to XChecker internal hardware. You must connect VCC to +5 V and GND to ground.

Always make sure that XChecker leads are connected properly for the specified mode of operation. Refer to the “Cable Connections and Definitions” table and the “XChecker Operation Mode Connections” table. Use the Cable → Logic Level of Pins command to display the logic state of the leads. This helps determine connectivity between the target system and the XChecker Cable.

For workstations (applies to Sun only), you must have read and write permissions for the port to which you connect the XChecker Cable. XChecker might issue a message stating that the cable is not connected to port tty*x*. If you see this message, follow this check list.

1. The board must have the power on because XChecker uses power from the board.
2. Check the device driver with the following command string.

   **`ls –l /dev/ttya /dev/ttyb`**

   The result should be the following.

   ```
   crw-rw-rw-  1 root12,0 month date time /dev/ttya
   ```

```
crw-rw-rw-  1 root12,1 month date time /dev/ttyb
```

3. Reconnect the XChecker Cable to another valid port.
4. Read the /etc/ttyab file. There should be two lines, as follows.

```
ttya"/usr/etc/getty std.9600"  unknown   off
local secure
```

```
ttyb"/usr/etc/getty std.9600"  unknown   off
local secure
```

   If you use a port to connect a modem for a remote login, you cannot use that port. The port must be on. Consult your System Administrator if the information the /etc/ttyab file differs from what is listed in the previous list.

## Improper or Unstable VCC

**Warning:** As with any CMOS device, the input/output pins of the internal FPGA should always be at a lower or equal potential than the rail voltage to avoid internal damage.

Never connect the control signals to XChecker before VCC and ground are connected. Xilinx recommends the following sequence.

1. Turn off power to the target system.
2. Connect VCC, ground, and then the signal leads.
3. Turn on power to the target system.

Make sure VCC rises to a stable level within 10ms. After VCC stabilizes, the level should be between 4.75 V and 5.25 V.

In the event of power glitches, reset the cable with the Cable → Reset command to reconfigure the XChecker internal FPGA device and then reconfigure the target device. For more information, see the "Resetting the Cable" section.

Chapter 5

# Programming a Device or a Daisy Chain

The Hardware Debugger enables you to program the logic of a device. This chapter explains how to use the Download menu commands to configure one or more devices. It contains the following sections.

- “Downloading Basics”
- “Preparing for Download and Verification”
- “Opening a Design File”
- “Downloading to a Target Board”
- “Verifying Design Logic”
- “Downloading and Verifying a Design”
- “Configuring Multiple Devices”

## Downloading Basics

Downloading is the process of programming a target device with the logic functions contained in your design. Verification is the process of reading back the downloaded configuration data and comparing it to the original data to ensure data was downloaded correctly.

Before beginning downloading or verification, turn on the power to your board.

**Warning:** You must connect VCC and ground before you connect the control signals to the MultiLINX Cable and the XChecker Cable. The input/output pins of their internal FPGA should always be at a potential that is lower or equal to their respective rail voltage in order to avoid internal damage.

## Preparing for Download and Verification

When you start the Hardware Debugger, the cable you installed is automatically detected. To modify the Communications settings, refer to the "Setting the Cable Options" section of the "Connecting Your Cable" chapter.

### Checking the Logic Level of Pins

The Logic Level of Pins command allows you to use the MultiLINX Cable and the XChecker Cable to display the logic levels of the cable pins and to probe digital signals on your board. It indicates the 1 or 0 value on each pin.

Steps 1 & 2 are specifically for the XChecker Cable, they are not needed for the MultiLINX Cable. For the MultiLINX Cable just connect the flying wires to the probe pins. There is no test header for the MultiLINX Cable.

**Warning:** Do not connect the XChecker leads to signals of different voltage levels than required to prevent damage to the XChecker internal hardware.

1. Connect the flying lead or header connector to the XChecker cable.
2. Attach the flying lead or header connectors to the pins or probe points.
3. Select **`Cable`** → **`Logic Level of Pins`**.

   Before configuration, the following should be true: INIT=1, PROG=1, D / P=0, CCLK=1, and DIN=1.
4. Select **`Scan Status Constantly`** if you want the software to continually scan and update the status of the cable pins.
5. If necessary, make changes to the pin configuration.

## Opening a Design File

After powering your target board, starting the Hardware Debugger, and setting the cable options, you are ready to open and download a design file. You can open BIT, RBT, or PROM files. You must open a BIT file if you want to verify and debug.

- BIT files and RBT files contain data for a single device. Use the Design Manager to create them.
- PROM files (.mcs, .tek, or .exo) contain data to program daisy chains. Use the PROM File Formatter to create them.
- Use the File → Open → Project command to open an existing project file (*design_name*.xck) that contains your design file.

**Note:** For byte wide PROMs greater than 0xFFFF (for example, 64k bytes) if the PROM start address is at an odd byte such as 0xFF, the resulting PROM cannot be used as an input design file in the Hardware Debugger for programming a FPGA device. Use either start address zero or one, starting at even bytes such as 0x10, 0x100 and so on while creating a PROM file.

1. Select **`File`** → **`New`** → **`Project`** or click the following toolbar button.

   The New Project dialog box appears.
2. In the Directories list box, select the desired directory.
3. In the List Files of Type field, specify the extension for the type of file you want to open.

   This extension is then displayed in the File Name list box. Only the files that match this extension are displayed in the File Name box.
4. In the File Name field, type the name of the design file you want to open or click on the file name in the File Name list box.
5. Click **OK**.

   If you do not have a logic allocation file (*design_name*.ll), a popup dialog box appears, informing you that debugging is not possible. The *design_name*.ll file is created during the implementation process if you have selected the appropriate readback option. Click **OK** to proceed.

After you choose a file to open, a project window appears that shows the hierarchical structure of files for each project. Each project contains bitstream, macro, and waveform data. The device name is only included in this hierarchy when you open a BIT file.

## Downloading to a Target Board

After you set up the appropriate cable and open a file to download, you can download the configuration data to a target board.

**Note:** If you plan to verify or debug a design, download a BIT file. You cannot verify or debug PROM files or RBT files. For more information, see the "Design Preparation" chapter.

1. Select **`Download`** → **`Download Design`** or click the following toolbar button.

   A download status window appears. This window includes a status bar that is updated as the data blocks are transmitted.

2. If the design is downloaded successfully, a message appears informing you that the device is configured and provides you with the transmission time.

## Verifying Design Logic

The Verify Bitstream command verifies a design that you downloaded using a BIT file. To verify or debug a device, meet the following criteria.

- Use a BIT file as input.
- Generate a logic allocation (*design_name*.ll) file in the design directory by doing one of the following.
  - Use the -l option in BitGen.
  - Enable the readback option as specified in the "Design Preparation" chapter.

    - Use the **-in** option for the Virtex family also with BitGen to generate a mask (*design_name.msk*) file.

- If you have an XC4000 or XC5200 design, do the following.
    - Set the readback clock to CClk.
    - Include the READBACK symbol in your design.
    - Include the STARTUP symbol with an inverter on the RESET pin if you plan to reset your design using the Pulse /RESET button located on the Debug Control Panel.
- Use the XChecker Cable.

  The XChecker must be configured for verification or debugging. Refer to the “XChecker Operation Mode Connections” table and the “Cable Connections and Definitions” table in the “Connecting Your Cable” chapter for information.
- Use the MultiLINX Cable.

  The MultiLINX Cable must be configured only for download and verify. Refer to the “MultiLINX Device Configuration Modes” table in the “MultiLINX™ Cable” chapter of the *Hardware User Guide.*

During verification, the XChecker cable reads the configuration data from the connected FPGA and verifies that it is the same as the downloaded configuration data. To verify your design logic follow these steps.

1. Select **`Download`** → **`Verify Bitstream`** or click the following toolbar button.

   The device’s configuration is read back and compared to the downloaded data to ensure that the device was properly configured.
2. When verification is completed, a message lets you know whether verification succeeded or failed. If verification failed, the message also informs you of the number of mismatched bits.

## Downloading and Verifying a Design

To verify a device, the design and the XChecker cable must meet the conditions described in the preceding section. To download and verify your design follow these steps.

1. Select **`File`** → **`Open`** → **`Project`** and open a project that contains a BIT file.
2. Select **`Download`** → **`Download and Verify`** or click the following toolbar button.
3. When verification is complete, a message lets you know whether verification succeeded or failed. If verification failed, the message provides the number of mismatched bits.

## Configuring Multiple Devices

You can use a PROM file to program several FPGAs at once.

### Configuring a Daisy Chain of Devices

To configure a daisy chain of devices, connect the XChecker cable to the lead device and ensure that all the devices in the daisy chain are connected correctly.

1. Connect the device pins of the master device as specified for the operation you want. Follow these steps.
2. Connect the DOUT output of each slave device to the DIN input of the next device in the chain. Refer to *The Programmable Logic Data Book* for more information on how to connect devices in a daisy chain.
3. Select **`File`** → **`Open`** → **`Project`** to open a project that contains a PROM file.
4. Select **`Download`** → **`Download Design`** or click the following toolbar button.

The Download Status window appears with a status bar that is updated as the data blocks are transmitted.

5. If the design is downloaded successfully, a message appears informing you that the device is configured.

## Verifying Individual Devices in a Daisy Chain

The Hardware Debugger and XChecker cable cannot read back or verify a daisy chain of FPGA devices. They can only read back and verify individual devices in a daisy chain.

To verify a device, the design must meet the conditions described in the “Verifying Design Logic” section. To verify an individual device, follow these steps.

1. Connect the XChecker cable directly to the readback pins of the device you want to verify.
2. Select **`File`** → **`Open`** → **`Project`** to open a project that contains a BIT file.
3. Select **`Download`** → **`Verify Bitstream`** or click the following toolbar button.

4. When verification is complete, a message lets you know whether verification succeeded or failed. If verification failed, the message provides the number of mismatched bits.

# Chapter 6

# Debugging a Device

This chapter explains how to debug a configured FPGA in either synchronous or asynchronous mode and how to save the readback data in a file. It contains the following sections.

- “Debugging Basics”
- “Debugging a Configured FPGA”
- “Synchronous Mode Debugging”
- “Asynchronous Mode Debugging”
- “Creating and Modifying a Signal Group”
- “Creating and Modifying a Signal List”
- “Saving and Loading Readback Data”

## Debugging Basics

Debugging refers to the process of reading back the internal states of a configured device using an XChecker Cable to ensure that the device is functioning as expected. After simulating a design to test it using worst-case delays, debug the device to analyze its in-circuit real-time behavior.

**Note:** The MultiLINX Cable does not currently support debug.

## Debugging a Configured FPGA

After configuring a device, you can analyze its behavior by taking snapshots of the device’s probe points. Flip-flops, RAMs, CLB outputs, and IOBs are all probe points and are read back when a snapshot of the device is taken. See the “Probe Points in FPGA Devices” table of the “Introduction” chapter for more information.

The two debugging modes for capturing the states of a device are synchronous and asynchronous.

## Synchronous

During synchronous debugging, the XChecker Cable is used to control your system clock and the state of your design. This allows you to probe the internal nodes at states that are known and stable. To control the clock, you can use the XChecker internal clock, as opposed to a system clock or a device clock. The XChecker Cable can also interface with a system clock, enabling you to start and stop the clock.

## Asynchronous

During asynchronous debugging, the Hardware Debugger does not control the clock and therefore does not control which states are captured. Use asynchronous debugging if you have an onboard microprocessor or equivalent device that allows you to control the state of your FPGA.

**Note:** If the state is not static when the internal nodes are probed, the values on some probes may reflect one state while values on other nodes may reflect a later state. This occurs because the values of the nodes can change during the time it takes for probing.

## Debugging Overview

This section summarizes the debugging process. For more details, continue with the "Synchronous Mode Debugging" section or the "Asynchronous Mode Debugging" section. To perform debugging, you must meet the criteria for verification and debugging listed in the "Verifying Design Logic" section of the "Programming a Device or a Daisy Chain" chapter.

After you download a BIT file using the Download → Download Design command, you can read back the states of the configured device using the Debug menu commands, as described in the following steps.

1. Choose a debugging mode by selecting either **`Debug`** → **`Synchronous Mode`** or **`Debug`** → **`Asynchronous Mode`**.

2. Set the appropriate options, such as the trigger type, the signals to display, and, in the case of synchronous debugging, the clock options, using the **`Debug`** → **`Settings`** submenu.

   For more information on the commands in this submenu, see the "Menu Commands" chapter.

3. Open a new waveform window by selecting **`File`** → **`New`** → **`Waveform`**.
4. Click **`Display`** to select the signals for display.
5. Click **`Read`** to read the signals that you selected for display.

   The signals are displayed in the active waveform window.

## Debugging a Previously Debugged Design

To debug a previously debugged design, specify the debug mode after opening the design. The Hardware Debugger loads the relevant settings for the specified debugging mode. The data, summarized in the following table, is saved in your project file, *design_name*.xck.

**Table 6-1 Available Synchronous and Asynchronous Settings**

| Settings | Synchronous Mode | Asynchronous Mode |
|---|---|---|
| Trigger | X | X |
| Number of clock cycles before and between snapshots | X | |
| Timeout | X | X |
| Reset before read-back | X | X |
| Clock | X | |
| Number of clocks | X | |
| Snapshots | X | |
| Groups and their radix settings | X | X |
| Displayed signals | X | X |

## Synchronous Mode Debugging

You must first download your design before you can debug it. The synchronous mode requires, the XChecker Cable, to conduct a controlled readback of your design. In this debug mode, you control the clock through the XChecker Cable.

The snapshot and clock features, which are available when performing synchronous mode debugging, enable you to define specific clock patterns to gather the device states you want. For example, if you are debugging a counter, you may want to check that the combinational values generate the state value of 10. Set the Number of Clocks Before First Snapshot to 10 and the number of snapshots to 1 to capture state 10 of the device.

In addition, you can use the following options.

- Start Clock

  This option establishes a free-running clock on the CLKO pin and advances the states of the device forward.

- Stop Clock

  This option stops the clock being applied on the CLKO pin.

- Reset FPGA

  This option uses an active-Low pulse to set the device to its initial state.

- Apply CLKO Clock(s)

  This option moves the device forward the number of states set by the Number of CLKO Clocks to Apply option.

### Pin Assignments

Before using the synchronous mode, choose whether you want to use an internal XChecker clock or an external clock connected to the XChecker CLKI pin.

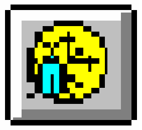
- To use the internal XChecker clock, connect the CLKO pin to the FPGA system clock input pin in place of an oscillator, and leave the CLKI pin unconnected.

- To use an external CLKI clock, connect the system clock to the XChecker CLKI pin and connect the XChecker CLKO pin to the FPGA system clock input pin.

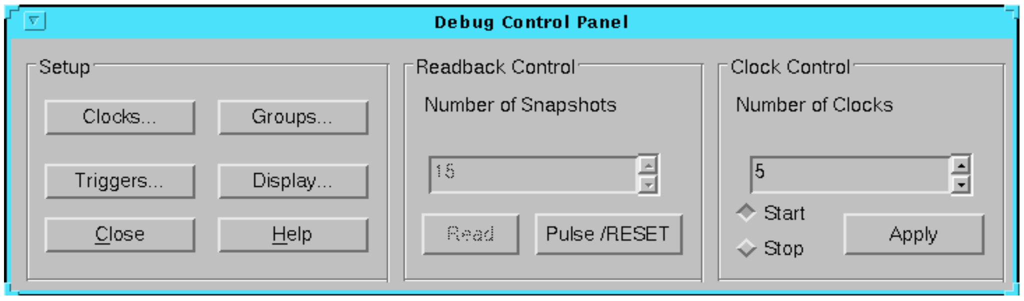
## Debugging in Synchronous Mode

To debug in synchronous mode, follow these steps.

1. Select **Debug** → **Synchronous Mode** or click the following toolbar button.

   The Debug → Settings commands and the Debug Control Panel options, shown in the following figure, are set for synchronous debugging.



Number of Snapshots: 15

Number of Clocks: 5

Start

Stop

**Figure 6-1 Debug Control Panel (Synchronous Mode)**

2. Select **Debug** → **Settings** → **CLKO Clock Source** or click the **Clocks** button in the Debug Control Panel.

   The CLKO Clock Settings dialog box appears as shown in the following figure.

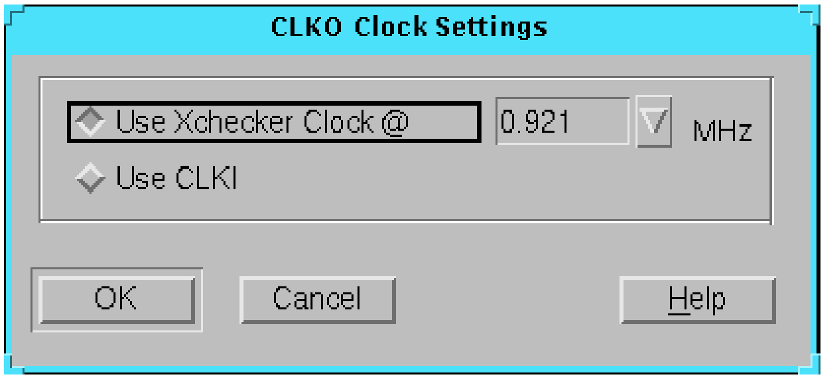


**Figure 6-2 CLKO Clock Settings Dialog Box**

3. Click the appropriate radio button for the clock you want to use: **`Use XChecker Clock`** or **`Use CLKI`**.

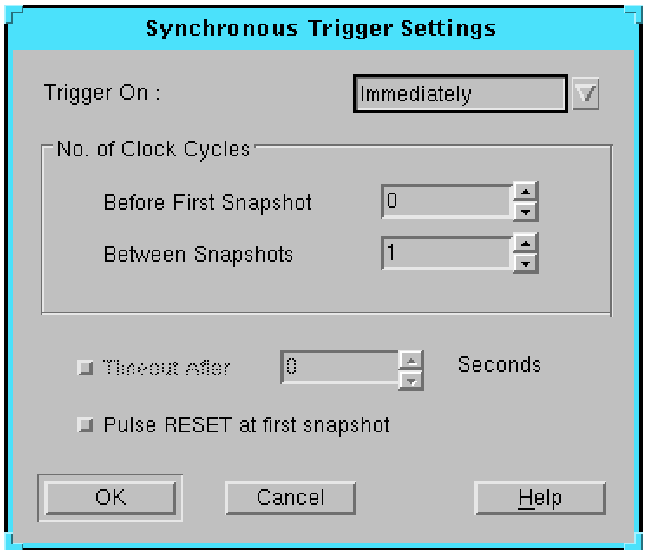
   - If you select the XChecker Clock, set the clock frequency (0.921, 2.764, 5.529, or 11.059 MHz).
   - If you select the CLKI clock, the FPGA uses the system clock connected to the XChecker CLKI pin. In this case, you cannot set the clock speed. The list box for the clock frequency is disabled.

4. After setting the desired clock settings, click **`OK`**.

   The CLKO Clock Settings dialog box closes and the status bar displays the clock settings.

5. Select **`Debug`** → **`Settings`** → **`Trigger`** or click the **`Trigger`** button in the Debug Control Panel.

   The Synchronous Trigger Settings dialog box appears, as shown in the following figure. You can select the type of trigger you want to initiate the readback.


Synchronous Trigger Settings

Trigger On : Immediately

No. of Clock Cycles

Before First Snapshot 0

Between Snapshots 1

☐ Timeout After 0 Seconds

☐ Pulse RESET at first snapshot

**Figure 6-3 Synchronous Trigger Settings Dialog Box**

6. In the Trigger On list box, select **`External`** pin (active-High on the TRIG pin of the XChecker Cable) or **`Enter Key`** to initiate a readback, or you can initiate a readback **`Immediately`** after the Read Snapshots command is executed.
7. In the Number of Clock Cycles field, specify the number of clock cycles to apply before the first snapshot and between snapshots. Use these options to cycle the device before the first snapshot and between multiple snapshots.
8. Use the **`Timeout After`** option to specify the cutoff time for trigger detection. If the trigger is not received within the specified time, the readback is canceled.
9. Select **`Pulse RESET at First Snapshot`** to reset the device each time you execute the Read Snapshots command.
10. After setting the desired trigger settings, click **`OK`**.

    The Synchronous Trigger Settings dialog box closes and the status bar displays the Trigger settings.
11. Select **`Debug`** → **`Settings`** → **`Display Signals`** or click the **`Display`** button in the Debug Control Panel to choose the signals and groups that you want to display.

For details on how to select signals for display, see the "Creating and Modifying a Signal List" section in this chapter.

12. Select **Debug** → **Settings** → **Number of Snapshots to Read** or type a number in the **Number of Snapshots** field in the Debug Control Panel.

    The Snapshots Count dialog box appears, as shown in the "Snapshots Count Dialog Box" figure, allowing you to enter the number of snapshots to read.

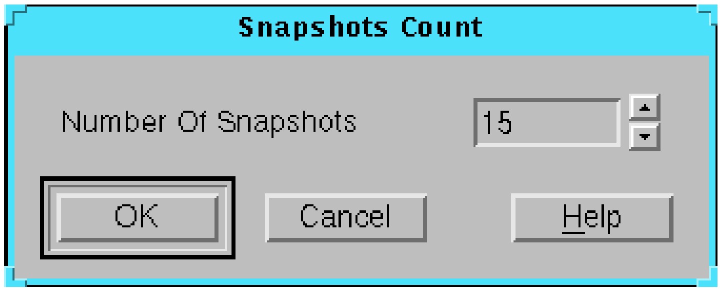


**Figure 6-4 Snapshots Count Dialog Box**

13. Enter the number of snapshots. Click **OK**.

    The Snapshots Count dialog box closes and the status bar displays the number of snapshots.

14. Select **Debug** → **Read Snapshots** or click **Read** in the Debug Control Panel to read the states of the signals that you selected for debugging.

    The device being read back returns its configuration data and the state of every probe point when a readback is triggered.

    The software then extracts the signals you selected and displays the signal values in a waveform, as shown in the following figure.

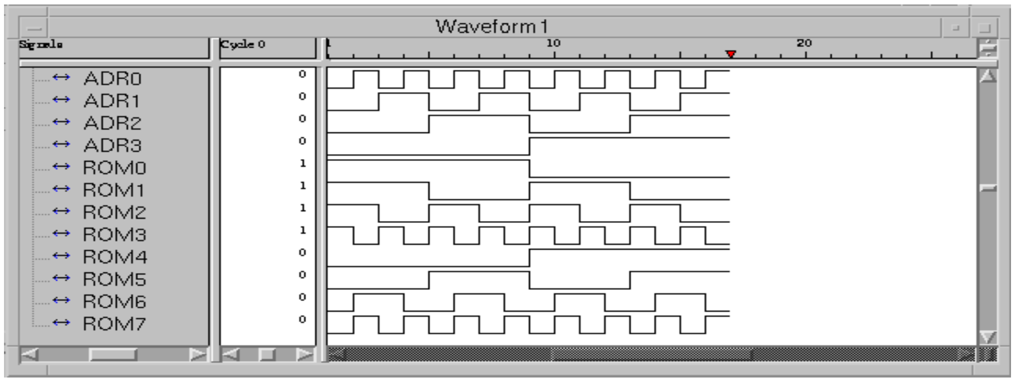


**Figure 6-5 Waveform Window**

**Note:** The readback data stream is linked to the active waveform window only. After the connection is closed, the waveform window is not updated with new information.

## Cycling the Device

You can control the clock through the XChecker Cable by applying a specific number of clocks to the device. When you apply clocks to your device, you advance the state of your device.

Follow these steps to apply Clocks to your device.

1. Select **`Debug`** → **`Settings`** → **`Number of CLKO Clocks to Apply`** and enter the number of clocks to apply or type a number in the **`Number of Clocks`** field in the Debug Control Panel.
2. Select **`Debug`** → **`Apply CLKO Clock(s)`** to cycle the device the number of clocks specified in the Number of Clocks to Apply field. Alternatively, you can click **`Apply`** in the Debug Control Panel.

**Note:** When the number of clocks to apply is set to 1, the Apply Clocks command can be used to single step the device during synchronous debugging.

### Resetting the FPGA

If you have not set the Pulse RESET at First Snapshot option in the Trigger Settings dialog box, choose **`Debug`** → **`Reset FPGA`** to issue an active-Low reset before issuing the Read Snapshots command.

### Viewing Additional Signals

Follow these steps to view new signals on your waveform, you must add these signals to the display list and read the device states again.

1. Use the **`Debug`** → **`Settings`** → **`Display Signals`** command or click **`Display`** in the Debug Control Panel to add new signals to the display list.
2. In the Display Signals dialog box, add the new signals to the display.
3. Click **`OK`**.
4. Select **`Debug`** → **`Read Snapshots`** to read the states of the new signals.

### Viewing the Waveform in Text Mode

Select **`View`** → **`Text Data`** to display the snapshots in textual form.

## Asynchronous Mode Debugging

Asynchronous mode debugging allows you to use any external clock. Because the XChecker Cable does not control the system clock, any number of clocks can occur between snapshots; and all snapshots are asynchronous to the system clock.

### Pin Assignments

In asynchronous mode, leave the XChecker CLKI and CLKO pins unconnected since you do not use the XChecker Cable to control the clock. Instead, you use an oscillator to drive the FPGA system clock directly. Connect the system clock so that it controls the target device flip-flops directly.

## Debugging in the Asynchronous Mode

In asynchronous mode debugging, you use a free-running clock to cycle the device as described in the following steps.

1. Select **`Debug`** → **`Asynchronous Mode`** or click the following toolbar button.

   The Debug → Settings commands and the Debug Control Panel options, shown in the following figure, are used for asynchronous debugging.

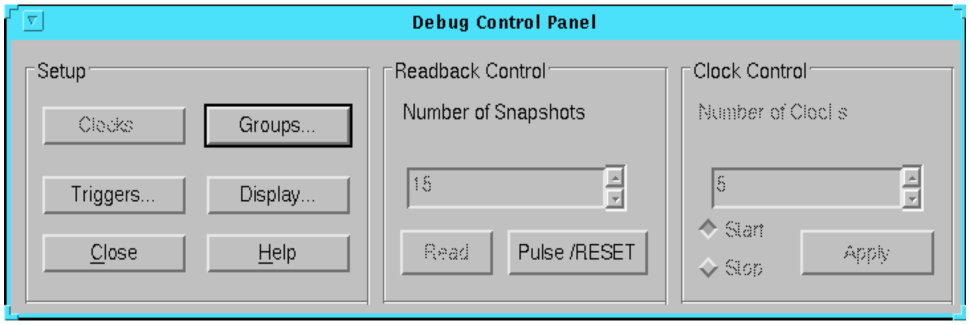


**Figure 6-6 Debug Control Panel (Asynchronous Mode)**

**Note:** Because the Hardware Debugger does not control the system clock, the clock settings are disabled.

2. Select **`Debug`** → **`Settings`** → **`Trigger`** or click the **`Triggers`** button in the Debug Control Panel.

   The Trigger command invokes the Asynchronous Trigger Settings dialog box, shown in the following figure, which enables you to select the type of trigger you want to initiate the readback.

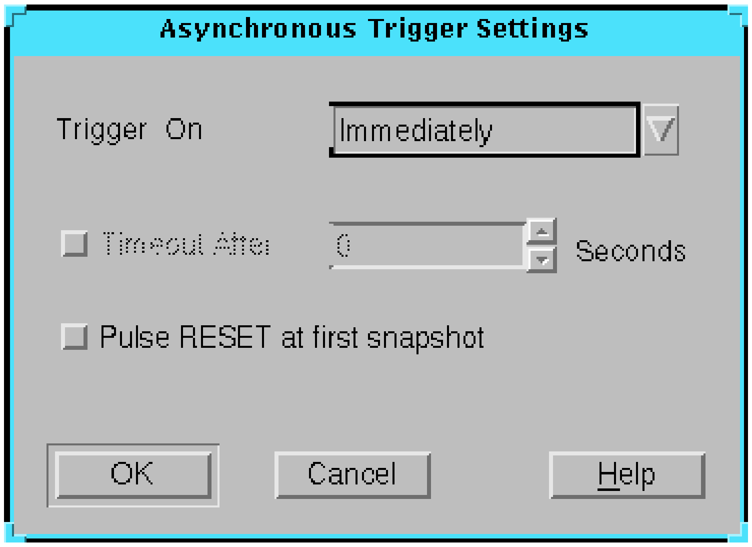


**Figure 6-7 Asynchronous Trigger Settings Dialog Box**

3. In the Trigger On list box, select **`External`** pin (active-High on the TRIG pin of the XChecker Cable) or **`Enter Key`** to initiate a readback, or you can initiate a readback **`Immediately`** after the Read Snapshots command is invoked.

4. Use the **`Timeout After`** option to specify the cutoff time for a trigger to be detected. If the trigger is not received within the specified time, the readback is canceled.

5. Select **`Pulse RESET at First Snapshot`** to reset the device each time you execute the Read Snapshots command.

6. After setting the desired trigger settings, click **`OK`**.

   The Asynchronous Trigger Settings dialog box closes and the status bar displays the Trigger settings.

7. Select **`Debug`** → **`Settings`** → **`Display Signals`** or click **`Display`** in the Debug Control Panel to choose the signals and groups that you want to display.

   For details on how to select signals for display, see the “Creating and Modifying a Signal List” section in this chapter.

8. Select **`Debug`** → **`Reset FPGA`** or click **`Pulse /RESET`** on the Debug Control Panel whenever you need to reinitialize the device.

9. Select **`Debug`** → **`Read Snapshots`** to read the states of the signals that you selected for debugging or click **`Read`** in the Debug Control Panel.

   The device being read back returns its configuration data and the state of every probe point when a readback is triggered.

**Note:** The readback data stream is linked to the active waveform window only. After the connection is closed, the waveform window is not updated with new information.

## Creating and Modifying a Signal Group

Use the Debug → Settings → Signal Groups command to create a group of signals or to add or remove signals from a previously defined group of signals. Use this option in conjunction with the Debug → Settings → Display Signals command when you are in the process of debugging a design.

### Creating a Signal Group

Follow this procedure to create a signal group.

1. Select **`Debug`** → **`Settings`** → **`Signal Groups`**.

   The Signal Groups command invokes the Signal Groups dialog box, shown in the following figure, which enables you to select the signal groups that you want to probe.

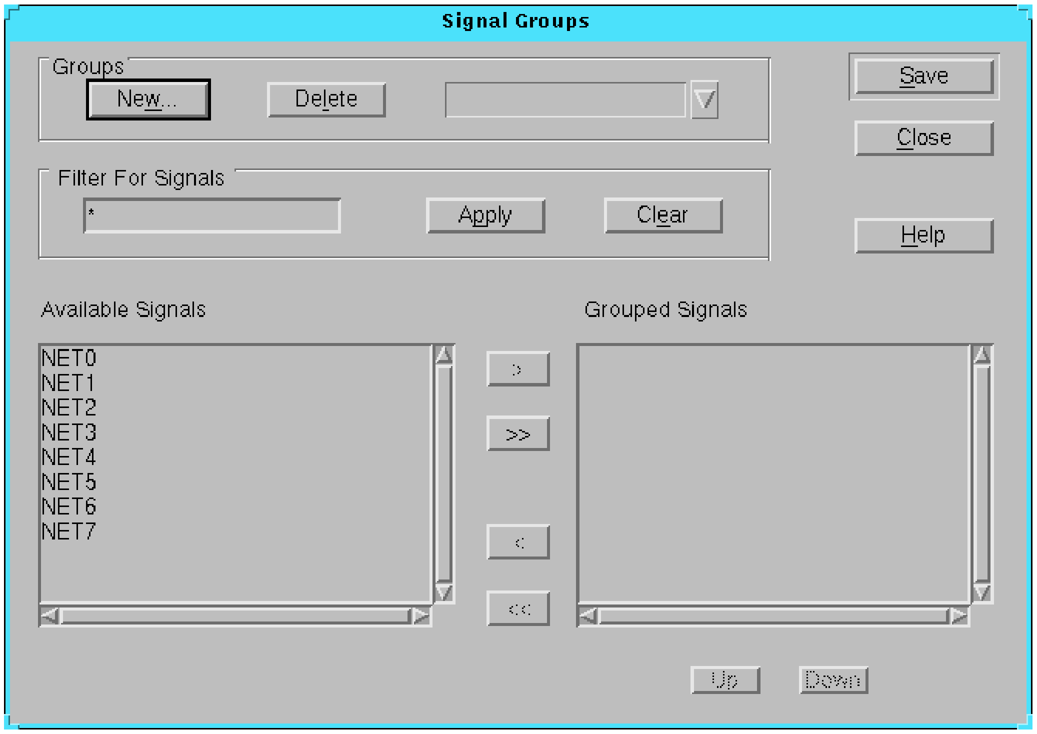


**Figure 6-8 Signal Groups Dialog Box**

2. Click **New**.

3. Type the name of the group you want to create in the Name field of the Group Name popup dialog box.

4. Use the Filter For Signals field to filter the signal names and make signal selection easier. For example, if you specify the letter A followed by a wildcard (*) all signal names that start with "A" are filtered.

   Click **Apply** to apply the filter to the signals displayed in the Available Signals list box. To redisplay the complete signal list, click **Clear**.

5. Select the signals you want to include in the group by highlighting the desired signals in the Available Signals list box and selecting the > button to move the highlighted signals to the Grouped Signals list box.

To remove signals from the Grouped Signals list box, select the signals and click the < button.

To add or remove all the signals displayed in a list box, use the << or >> button.

To rearrange the order of the signals displayed in the Grouped Signals list box, use the **`Up`** and **`Down`** buttons below the list box.

6. Click **`Save`** to save the new group.
7. Click **`Close`** to exit the Signal Groups window.
8. Refer to the “Creating and Modifying a Signal List” section for information on how to include the groups you created in your display list.

## Modifying a Signal Group

The Signal Groups dialog box contains a list of previously defined signal groups and a list of signals. Use this dialog box to add or remove signals from a group as described in the following steps.

1. Select **`Debug`** → **`Settings`** → **`Signal Groups`** or click the **`Groups`** button in the Debug Control Panel.
2. Click the down-arrow of the Groups drop-down list box and select the desired group.

   The signals currently included in the selected group are listed in the Grouped Signals list box. The signals that are not included are displayed in the Available Signals list box.
3. Add or remove the signals you want using the < or > button.
4. Use the **`Up`** or **`Down`** buttons to rearrange the order of the signals.
5. Click **`Save`** to save the new group.
6. Click **`Close`** to exit from the Signal Groups dialog box.

**Note:** To delete a group, go to the Signal Groups dialog box, select the existing group from the Groups drop-down list box, and click **`Delete`**.

## Creating and Modifying a Signal List

The signal list can consist of signals, groups, and RAM bits. Use the Display Signals dialog box to create the list of signals to be displayed, including any signal groups that you have created.

Open the Display Signals dialog box and select the signals to include in the Displayed Signals list. The Displayed Signals list is a list of signals, groups, and RAM bits you select for probing with the Hardware Debugger.

**Note:** To display a new signal group, you must first define it in the Display Groups dialog box before adding it to the Displayed Signals list. Refer to the procedure in the “Creating a Signal Group” section.

1. Select **`Debug`** → **`Settings`** → **`Display Signals`** or click the **`Display`** button in the Debug Control Panel.

   The Display Signals dialog box, shown in the following figure, appears.

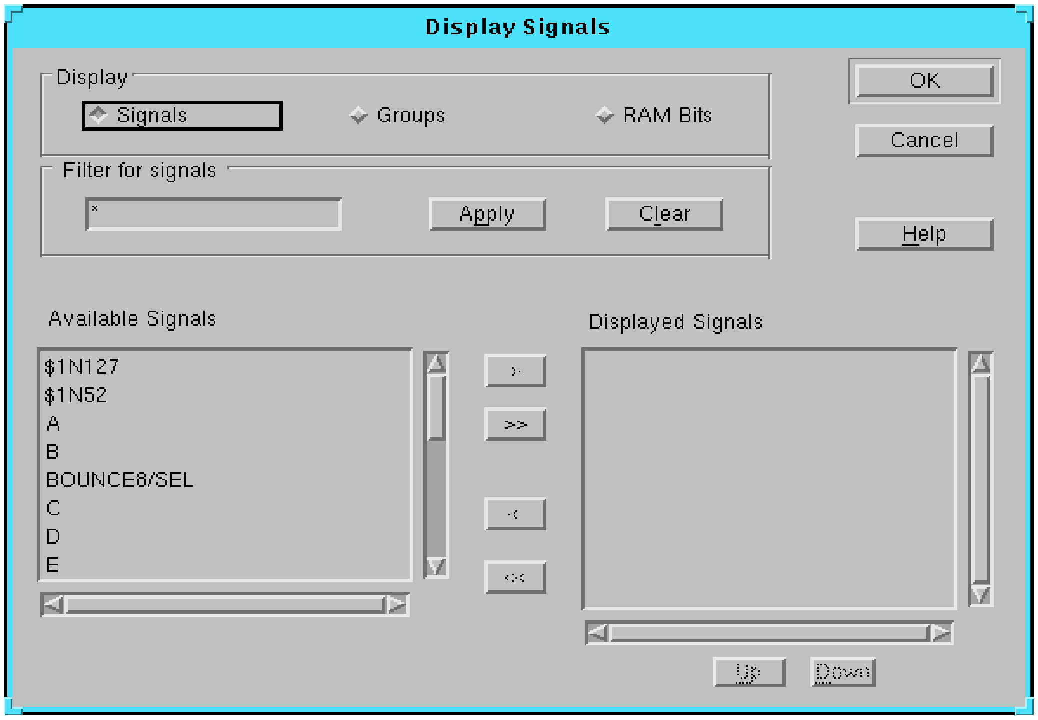


**Figure 6-9 Display Signals Dialog Box**

2. Click **`Signals`**, **`Groups`**, or **`RAM Bits`** to display the list of available signals, signal groups, or RAM bits.

3. Use the Filter for Signals field to make signal selection easier. For example, if you specify the letter “A” followed by a wildcard (*) all signal names that start with “A” are filtered

   Click **`Apply`** to apply the filter to the Signals displayed in the Available Signals list box. To redisplay the complete signal list, click **`Clear`**.

4. In the Available Signals list box, highlight one or more of the signals, groups, or RAM bits you want to probe.

5. Click the > button to move the highlighted signals to the Displayed Signals list box.

To remove signals from the Displayed Signals list box, select the signals and click the < button.

To add or remove all the signals displayed in a list box, use the << or >> button.

To rearrange the order of the signals displayed in the Displayed Signals list box, use the **`Up`** and **`Down`** buttons below the list box.

6. Click **`OK`** to exit from the Display Signals dialog box.

   The current waveform is automatically updated with the new signals, which appear as cross-hatched hexagons.

## Saving and Loading Readback Data

After generating a readback sequence of your signals, you can save the readback data you generated. Saving readback data is useful for multiple design analyses, for comparison, and design optimization.

You can view previously saved readback data textually or as waveforms. When the file is first opened, the data is displayed in a waveform. To view the data textually, use the View → Text Data command.

### Saving Readback Data

Follow this procedure to save readback data.

1. Click on the waveform that you want to save and choose **`File`** → **`Save`** or click the following toolbar button.

   The Save As dialog box appears.

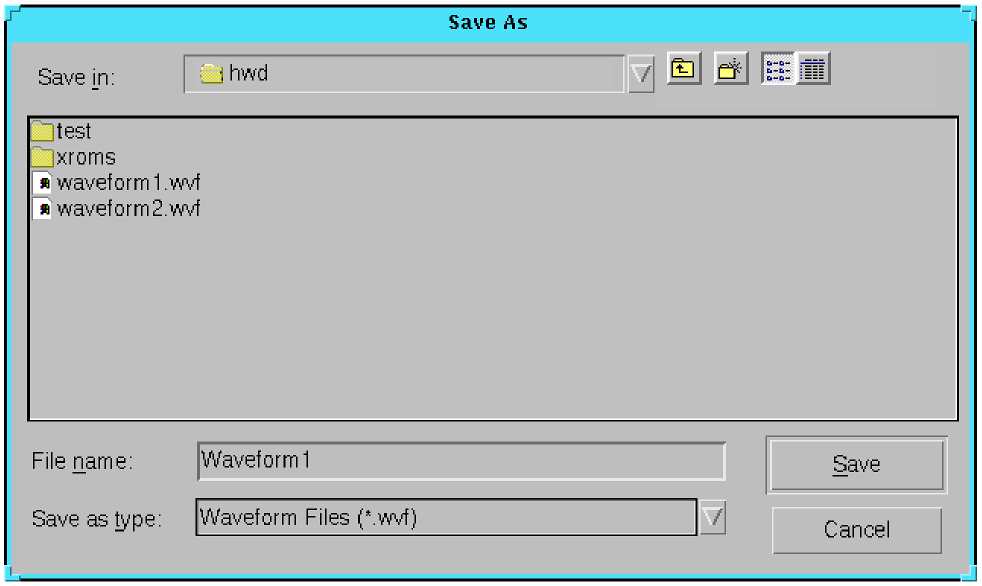


**Figure 6-10 Save As Dialog Box**

2. In the File Name field, type or select the name of the file you want to save. By default, the .wvf extension is appended to the file name.

   If you want to change the path for your readback data, select the desired directory in the Directories field.

3. Click **`OK`** to save the data.

## Viewing Previously Saved Readback Data

Follow these steps to view previously saved readback data.

1. Select **`File`** → **`Open`** → **`Waveform`** to view previously saved readback data.

2. In the Open dialog box, specify the file name and click **`OK`**.

   The data is displayed in the format in which it was saved.

3. To view the data textually, select **`View`** → **`Text Data`** or click the following toolbar button.

4. To view the data graphically, select **`View`** → **`Waveform Data`** or click the following toolbar button.

# Chapter 7

# Customizing the Interface

This chapter describes how to customize the Hardware Debugger. You can optimize its use by creating macros using predefined settings and changing the display parameters. This chapter describes these tasks in the following sections.

- “Using Macros”
- “Controlling the Waveform Display Parameters”

## Using Macros

The Hardware Debugger allows you to define macros to automate tasks. After you have completed a procedure, you can capture the steps you want to include in your macro by selecting the commands from the Console window, copying them into a macro window, and saving the macro.

Alternatively, you can type the commands directly in a macro window. For information about the available commands and syntax, refer to the “Console Commands” appendix.

### Creating a Macro

Follow these steps to create a macro.

1. Select **`File`** → **`New`** → **`Macro`**.

   A macro window appears, as shown in the following figure.

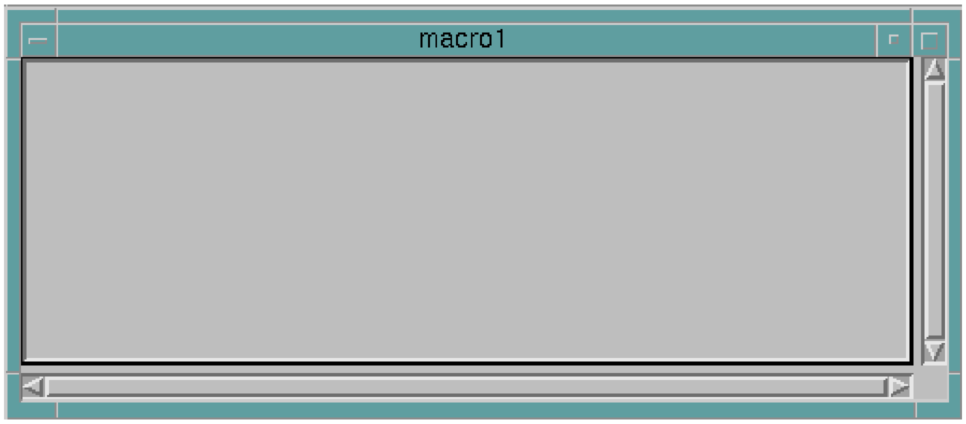
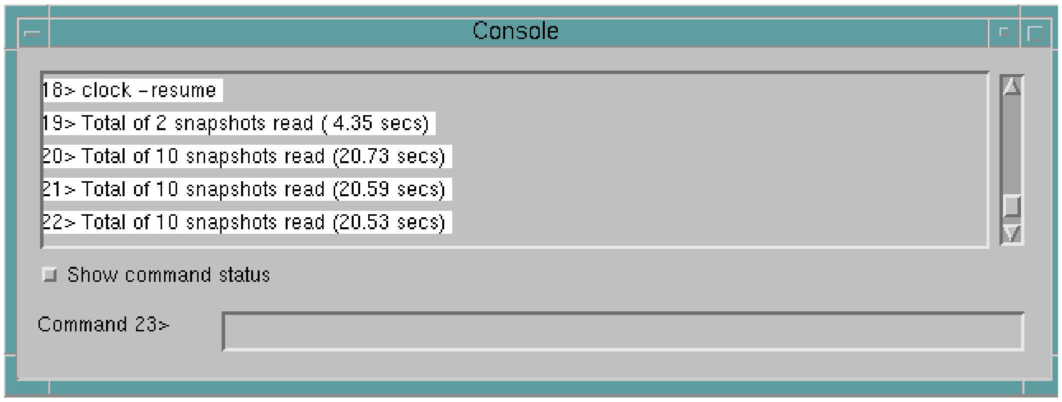


**Figure 7-1 New Macro Window**

2. Select **`View`** → **`Console`** to display the Console window and view the commands that have been executed.

   The Console window appears.



```
18> clock -resume
19> Total of 2 snapshots read ( 4.35 secs)
20> Total of 10 snapshots read (20.73 secs)
21> Total of 10 snapshots read (20.59 secs)
22> Total of 10 snapshots read (20.53 secs)
```

**Figure 7-2 Console Window**

3. Click the left mouse button at the beginning of the first command that you want to copy. Holding down the Shift key, click the left mouse button at the end of the last command you want to copy.
4. Copy the steps you want from the Console window by selecting **`Edit`** → **`Copy`**.

5. Select the macro window.
6. Paste the commands into the macro window by selecting **Edit** → **Paste**.
7. Repeat steps 3 through 6 to copy non-consecutive lines into the macro.
8. Use comments to note options and strategies, by preceding your comment text with the # character. Comments in a macro file always start with the # character on the line.
9. Save the macro using the **File** → **Save** or click the following toolbar button.

## Editing an Existing Macro

To edit an existing macro, follow these steps.

1. Select **File** → **Open** → **Macro** from the menu and open the macro you want to modify from the list of macros in the Open Macro File dialog box.
2. In the Macro window, select commands by double-clicking the left mouse button to select a word. You can also select an entire block of text by pressing the left mouse button and dragging the mouse to the end of the region you want to copy.
3. Use the **Cut**, **Copy**, and **Paste** commands from the **Edit** menu to copy the selected information and paste it at the new insertion point.
4. Select **File** → **Save** to save the change or click the following toolbar button.

### Running a Macro

To run a macro, follow these steps.

1. Open the macro by selecting **`File`** → **`Open`** → **`Macro`**.
2. Select **`File`** → **`Run Macro`** or click the following toolbar button.

### Saving the Console Log to a File

You can check the command history for a debug session by opening the Console window. You can also save the command history, or command log, to a text file.

1. Select the contents of the Console window by clicking the left mouse button at the top of the window and dragging the cursor down to the end of the window.

   To select the entire log, select the top part of the log, then click on the scroll bar to scroll down to the bottom of the page. At the bottom of the page, press the Shift key while you click the mouse button again to complete the selection.
2. Select **`Edit`** → **`Copy`** to copy the selected contents of the Console window to the clipboard.
3. Open a new file in a text editor and paste the contents of the clipboard into the file.

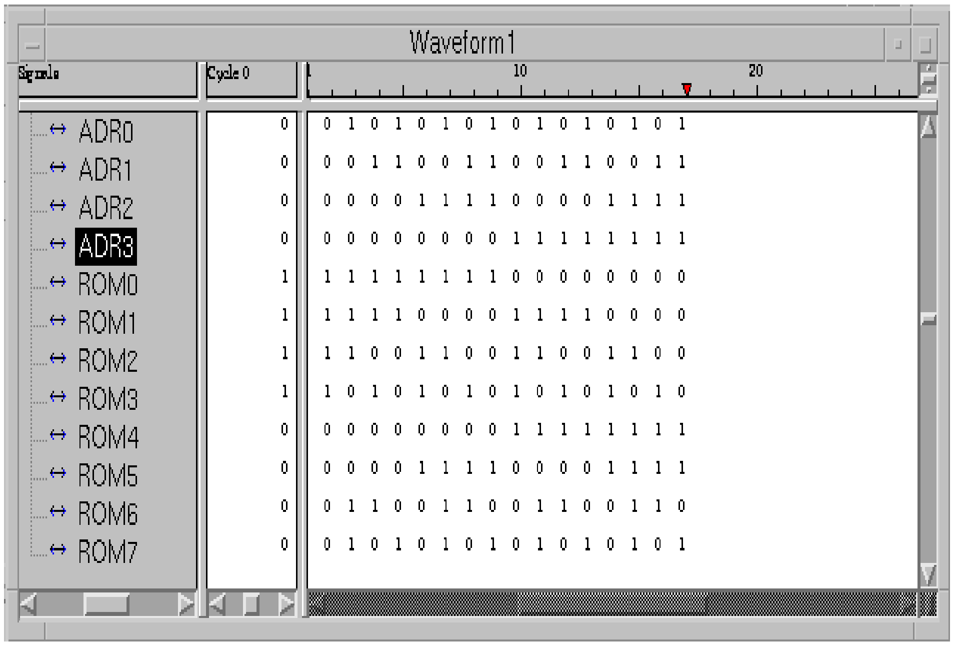
   The contents of the Console window appear inside the file.
4. Save the file using the appropriate text editor command.

## Controlling the Waveform Display Parameters

This section describes the features available for customizing your waveform window. After generating a waveform, you can modify its format by changing the display mode from graphics to text, modifying the radix of your groups, and changing the color of the waveform window background and signals. In addition, you can change the size of the waveform and the position of the windows on the screen.

## Waveform Window (Textual View)

In the text mode, the waveform window displays a textual representation of the device states you specified for viewing, as shown in the following figure. This view is a tabular display of the readback information.



```
Signals   Cycle 0
ADR0      0    0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
ADR1      0    0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
ADR2      0    0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
ADR3      0    0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
ROM0      1    1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
ROM1      1    1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
ROM2      1    1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
ROM3      1    1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
ROM4      0    0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
ROM5      0    0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
ROM6      0    0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0
ROM7      0    0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
```

**Figure 7-3 Active Waveform Window (Textual View)**

The textual waveform window contains the following areas.

### Signals

This column shows the names of the displayed signals and signal groups.

### Cycle

The data in this column reflect the position of the vertical cursor in the text data window. The header shows the number of the selected snapshot. The column shows the readback values of the selected signals and signal groups.

### Text Data Window

This column contains the following areas.

- Cycle Scale

  This area, above the text data area, shows the snapshot counts, number of applied clock cycles, and reset states.Each snapshot count appears as a red triangle. The number of clock cycles applied between consecutive snapshots appears above each snapshot mark.Each Reset state appears as a blue diamond with a vertical line.

- Text Data

  This display area shows text data for the readback values of signals and signal groups.

## Waveform Window (Graphical View)

In the graphical mode, the waveform window displays a graphical representation of the signals you specified for viewing, as shown in the following figure.

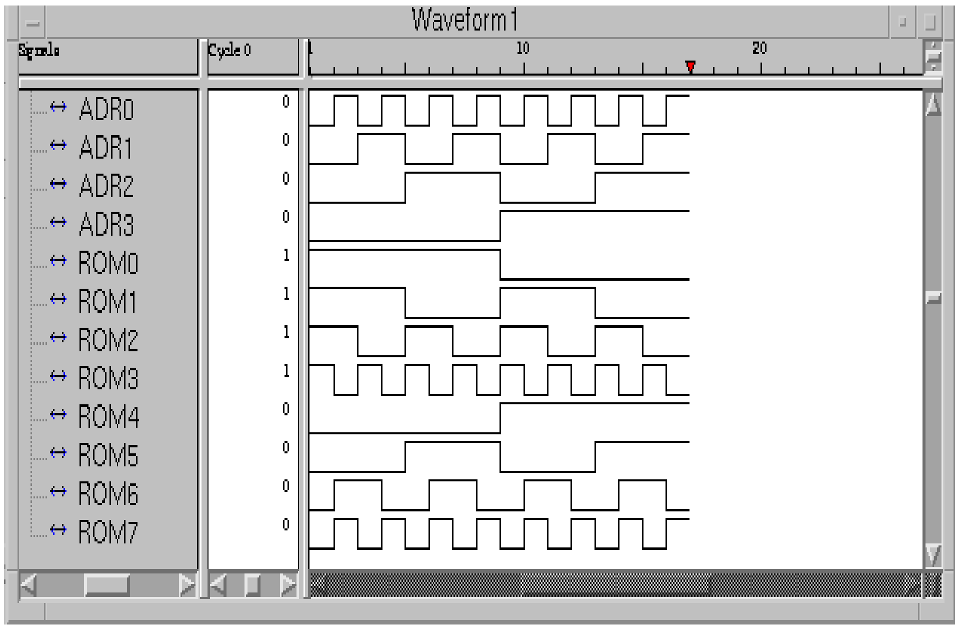


**Figure 7-4 Active Waveform Window (Graphical View)**

The Graphical Waveform window includes the following areas.

## Signals

This column shows the names of the displayed signals and signal groups.

## Cycle

The data in this column reflect the position of the vertical cursor in the waveform data window. The header shows the number of the selected snapshot. The column shows the readback value of the selected signals and signal groups.

## Waveform Data Window

This column contains the following areas.

- Cycle Scale

This area, above the waveform data area, shows the snapshot counts, number of applied clock cycles, and reset states.

Each snapshot count appears as a red triangle. The number of clock cycles applied between consecutive snapshots appears above each snapshot mark.

Each Reset state appears as a blue diamond with a vertical line.

- Waveform Data

  This display area shows waveforms for the readback values of signals and signal groups.Each signal appears as a semi-rectangular wave. Each signal group appears as a complete hexagonal wave.

**Note:** To start readback from the beginning, you can perform a waveform reset by using the Debug Reset Control. To clear, click right mouse.

## Changing the Size of the Waveform

Use the options listed under the View menu to change the size of the waveform on your screen or to update your screen.

- Select **`View`** → **`Zoom In`** or click the Zoom In toolbar button to increase the size of your waveform.

- Select **`View`** → **`Zoom Out`** or click the Zoom Out toolbar button to decrease the size of your waveform.

- Select **`View`** → **`Zoom To Fit`** or click the Zoom To Fit toolbar button to fit all of the waveform cycles into the current window.

- Select **View** → **Zoom Reset** or click the Zoom Reset toolbar button to return the display to the default state.

- Select **View** → **Refresh** to redraw the elements on your screen and update the current waveform using the latest debugging modifications.

## Controlling the Position of the Windows

To change the way the windows are displayed on your screen, follow these guidelines. The window that is in focus processes the commands you invoke.

- Select **Window** → **Arrange** to arrange the iconized windows as non-overlapping tiles at the bottom of the main window.
- Select **Window** → **Cascade** to display each window on top of the preceding window in a cascading arrangement.
- Select **Window** → **Tile Horizontally** to arrange the windows horizontally.
- Select **Window** → **Tile Vertically** to arrange the windows vertically.

## Changing the Radix of Displayed Signal Groups

Use the View → Group Radix command to display or change the radices of your signal groups. Possible radices include binary, octal, decimal, and hexadecimal. The radix is applied to all groups.

1. Select **View** → **Group Radix** from the menu.
2. Click on the desired radix.

## Changing the Color of the Waveform Window

You can change the color of the signals and background of the waveform window.

1. To modify a signal color, click on the signal in the waveform window that you want to modify.
2. Select **`View`** → **`Color Signals`**.
3. From the Color dialog box, select a color or create a custom color to select.
4. Click **`OK`** to apply the color change.
5. To modify the background color, click on the waveform window that you want to modify.
6. Select **`View`** → **`Background Color`**.
7. From the Color dialog box, select a color or create a custom color to select.
8. Click **`OK`** to apply the color change.

## Removing a Signal

Use the popup menu commands in the Waveform window to remove signal waveforms.

1. Select the signal on the waveform.
2. Select **`Delete Signals`** from the popup menu.

# Appendix A

# Glossary of Terms

This appendix contains definitions and explanations for terms commonly used in the Hardware Debugger program.

## asynchronous debugging

A debugging mode in which you capture data without controlling your system clock through the Hardware Debugger.

## BIT file

A synonym for a configuration bitstream file.

## bitstream (BIT file)

A data stream, also called BIT file, that contains location information of logic on a device, that is, the placement of CLBs, IOBs, TBUFs, pins, and routing elements. The bitstream also includes empty placeholders that are filled with the logical states sent by the device during a readback. Only the memory elements, such as flip-flops, RAMs, and CLB outputs, are mapped to these placeholders, because their contents are likely to change from one state to another. When downloaded to a device, a bitstream programs the device.

A bitstream file has a .bit extension.

## CCLK pin

The pin of the configuration cable that connects the configuration clock to the device.

## CFG_DONE pin

This pin has the same functionality as the ~PROGRAM pin on the XC4000, XC5200, and Spartan FPGAs. The CFG_DONE pin is a MultiLINX Target Interface Pin.

## CFG_RDY pin

When asserted, this pin indicates that the FPGA is ready to receive configuration data. When de-asserted, the pin indicates that either the FPGA is in the power-up mode, or a configuration error has occurred. This pin has the same functionality as the INIT pin on the Spartan, XC3000, XC4000, and XC5200 FPGAs. The CFG_RDY pin is a Multi-LINX Target Interface Pin.

## CFG_RESET pin

This pin has the same functionality as the ~PROGRAM pin on the XC4000, XC5200, and Spartan FPGAs. The CFG_RESET pin is a MultiLINX Target Interface Pin.

## CLKI pin

XChecker and MultiLINX cables clock input pin. This pin connects the system clock to the XChecker Cable. The frequency of this clock must be in the range of 120 kHz to 10 MHz. Connecting the system clock to the CLKI pin allows the Hardware Debugger to control the system clock so that snapshots are captured while the design is in a known state. If the clock source is external, the clock signal is connected to the CLKI pin; if it is internal, the clock signal is generated by the XChecker Cable electronics.

## CLKO pin

XChecker and MultiLINX cables clock output pin. This pin connects to the destination of the target system clock. Connecting the CLKO pin to the system clock allows the Hardware Debugger to control the system clock so that snapshots are captured while the design is in a known state.

## console log

A record of the commands that you executed during a Hardware Debugger session.

## control panel

A dialog box in the Hardware Debugger that consists of buttons and fields for debugging purposes. The control panel is displayed if the Hardware Debugger detects that readback is enabled for a design. The control panel commands offer an alternative to the Debug menu commands.

## CS/CS0 pin

CS on the Virtex; and CS0 on the XC4000 and XC5200 FPGAs. The CS / CS0 pin represents a chip select to the target FPGA during configuration. The CS / CS0 pin is a MultiLINX Target Interface Pin.

## CS1 pin

Chip Select to the XC4000 and XC5200 FPGAs during configuration. The CS1 pin is a MultiLINX Target Interface Pin.

## CS2 pin

Chip Select to the XC3000 FPGA while using the Peripheral configuration mode. The CS2 pin is a MultiLINX Target Interface Pin.

## debugging

The process of reading back or probing the states of a configured device to ensure that the device is behaving normally while in circuit.

## DIN pin

The Data In pin of the configuration cable connects to the DIN pin of your target device. In serial mode, the DIN pin loads the bitstream data to the target FPGA.

## DONE pin (XC4000/XC5200)

This pin connects to the DONE pin of your target FPGA. It indicates the completion of the configuration process. During configuration, this pin is Low. After configuration, this pin is High.

## downloading

Configuring or programming a device by sending bitstream data to the device.

## D/P pin (XC3000)

The dual function Done / Program pin of your configuration cable. The pin connects to the D / P pin on your target device. As an input, D / P=0 is used to initiate a device reconfiguration. As an output, D / P=1 signals the end of configuration.

## D0-D7 pins

An 8 bit data bus supporting the Express, and SelectMAP configuration modes. The D0-D7 pins are MultiLINX Target Interface Pins.

## external clock

The clock connected to the XChecker CLKI pin that the Hardware Debugger uses for synchronous mode debugging. To use an external clock, connect the system clock to the XChecker Cable using the CLKI pin and connect the XChecker CLKO pin to the system clock loads.

## GND pin

Ground (0 volt) pin of the configuration cable. This pin connects to the Ground pin of a power supply.

## group

A combination of signals that have a common output. In the case of a counter, for example, the different signals that produce the actual counter values can be grouped under the same name and share a common representation.

## INIT pin

Initialization pin on your configuration cable. This pin is connected to the INIT pin of your target device indicating when a device is ready to receive configuration data after power up. During configuration, INIT=0 indicates a configuration error.

## internal clock

The clock provided by the XChecker Cable. This clock is generated in the XChecker Cable electronics and is output on the CLKO pin. It is available in synchronous mode and is controlled from the Hardware Debugger.

## (.ll) file

The logic allocation file, which indicates the bitstream position of elements that can be probed, such as latches, flip-flops, and IOB inputs and outputs. The Hardware Debugger uses this file to locate signal values inside a readback bitstream. This file is created during the implementation process if readback is enabled.

## (.msk) file

The mask file, (.msk) file indicates which bits are configuration bits and which ones are not. This file is needed to do a verify operation on a Virtex family device using the MultiLINX Cable. This file is generated during the implementation process (BitGen) if readback is enabled in the “Configuration Template”.

## main window

The background against which other windows are displayed in the Hardware Debugger.

## menu bar

The area located at the top of the Hardware Debugger window. It includes the File, Cable, Download, Debug, View, Window, and Help menus. Refer to the “Menu Commands” chapter for information on the menu commands.

## number of clock cycles

The number of clocks that occur between snapshots. When capturing multiple snapshots during synchronous mode debugging, the number of snapshots is used as a trigger for capturing each snapshot.

## probing

The process of examining the signal states of an FPGA device.

## PROG pin

The Program pin of your configuration cable provides a reprogram pulse to XC4000 and XC5200 devices when connected to the PROG pin of the device.

## PROM file

A PROM file is the file output by the PROM File Formatter, which can be used to program one or more devices. The PROM File Formatter supports the following PROM file formats: MCS (Intel MCS-86), EXO (Motorola EXORMacs), TEKHEX (Tektronix hexadecimal), and HEX.

## RBT file

A raw BIT format file, the ASCII version of the BIT file.

## readback

The process of reading the logic downloaded to an FPGA device. There are two types of readbacks.

- A readback with a filter that extracts the configuration bits to verify that a design was downloaded correctly.
- A readback with a filter that extracts the state of design storage elements, CLB outputs, and IOB outputs to ensure that the device is behaving as expected.

## RD(TDO) pin

The readback data pin of the XChecker Cable and the MultiLINX Cable. This pin connects to the RDATA pin of the device. When connected, the pin reads data from the programmed target device.

## RDWR pin

The RDWR pin is used as an active high READ and an active low WRITE control signal to the Virtex FPGA. The RDWR pin is a Multi-LINX Target Interface Pin.

## RDY/BUSY pin

Use the BUSY pin overfertilize devices and the RDY / ~BUSY pin on the XC3000, XC4000, and XC5200 FPGAs. The RDY / BUSY pin is a MultiLINX Target Interface Pin.

## RS pin

Read Select control for the Asynchronous Peripheral configuration mode on XC4000 and XC5200 FPGAs. The RS pin is a MultiLINX Target Interface Pin.

## RST pin

The Reset pin of the XChecker Cable and the MultiLINX Cable connects to the RST pin of the target device. This pin is driven Low by the Hardware Debugger after configuration to initialize the target XC3000 FPGA internal latches and flip-flops. After configuration, this pin can be used as an active-Low reset when debugging XC4000 and XC5200 devices.

For XC4000 and XC5200 devices, the FPGA expects an active-High reset, but the Hardware Debugger produces an active-Low reset pulse. Users should invert the reset pulse before driving the GSR pin on the STARTUP symbol.

## RT pin

The readback trigger pin of the XChecker Cable and the MultiLINX Cable. This pin connects to the device RTRIG pin. When the

XChecker Cable drives this pin High, the pin initiates a readback on the target FPGA.

## snapshot

Readback data that contains the values of all storage elements, CLB outputs, and IOB inputs and outputs of a design at a point in time.

In the waveform window, each snapshot is marked by a red triangle.

## states

The values stored in the memory elements of a device (flip-flops, latches, RAMs, CLB outputs, and IOBs) that represent the state of that device for a particular readback. To each state there corresponds a specific set of logical values.

## status bar

The field located at the bottom of the Hardware Debugger window. It provides information about the commands that you are about to select or that are already being processed.

## synchronous debugging

A debugging mode in which you use the XChecker Cable to control your system clock so that snapshots can be captured while the design is in a known state.

## TCK pin

In the Hardware Debugger, an XChecker Cable and MultiLINX Cable pin reserved for future use. It can be used as a digital probe point using the Cable → Logic Level of Pins command.

## TDI pin

In the Hardware Debugger, an XChecker Cable and MultiLINX Cable pin reserved for future use. It can be used as a digital probe point using the Cable → Logic Level of Pins command.

## TMS pin

In the Hardware Debugger, an XChecker Cable and MultiLINX Cable pin reserved for future use. It can be used as a digital probe point using the Cable → Logic Level of Pins command.

## toolbar

A field located under the menu bar at the top of the main window. It contains a series of buttons that execute some of the most frequently used commands. These buttons constitute an alternative to the menu commands.

## TRIG pin

The external trigger pin of the XChecker Cable and the MultiLINX Cable. This pin is connected to an external signal used as a trigger. A Low to High transition on this pin signals the cables to initiate a readback.

## trigger

A signal that tells the Hardware Debugger to read a snapshot.

## VCC pin

Power (5 volt) pin of the XChecker Cable. This pin connects to the power pin of a 5 volt power supply.

## verification

The process of reading back the configuration data and comparing it to the original downloaded design to ensure that all of the design was received by the device.

## waveform

The graphical representation of one or more readbacks. Usually, you select a set of signals and a set of readbacks for display. Each readback represents a particular state of the memory elements of the device.

## WS pin

Write Select control for the Asynchronous Peripheral configuration mode on XC4000 and XC5200 FPGAs during configuration. The WS pin is a MultiLINX Target Interface Pin.

# Appendix B

# Console Commands

Console commands are entered from the Console window. You can use console commands as an alternative to menu commands. Console commands do not display the dialog and message boxes associated with the menu commands. To get command feedback, you can use the Force command. Additionally, you can also turn on the Show Command Status check box in the Console window. This chapter contains the following sections.

- “Conventions”
- “Alphabetical Listing of Commands”

To automate design debugging, you can copy commands from the Console window into a macro window to build macros. See the “Using Macros” section of the “Customizing the Interface” chapter for information.

**Note:** Do not use console commands until you are familiar with the graphical user interface commands because console commands must be executed in a particular order.

## Conventions

The following is a summary of the syntax used for Console commands.

- When an option occurs without parentheses “( )” next to the command name in the syntax, it is required. When it appears in square brackets “[ ],” the variable is optional.
- When two or more options occur between braces “{ },” they must be entered with the command. If the options are separated by a vertical bar “ | ,” you must choose one of the possible parameters. If one term is divided into a subset of parameters that can be

entered separately or together, each subparameter occurs between square brackets.

## Alphabetical Listing of Commands

The following summarizes the console commands discussed in this chapter.

| | |
|---|---|
| Baud | Sets the baud rate |
| Cable | Sets the cable options |
| Clock | Sets the clock options |
| Display | Selects the signals to debug |
| Download | Programs target device with the current design |
| Exit | Exits the Hardware Debugger |
| Force | Displays or hides the message boxes |
| Group | Defines groups of signals for debugging |
| Open | Loads the configuration data |
| Port | Selects the communications port |
| Readfpga | Reads the device states using the specified debugging settings (snapshot number, signals, and groups) |
| Reset | Reinitializes the target device |
| Run | Executes the macro in the current macro window |
| Setmode | Selects the debugging mode, synchronous or asynchronous |
| Trigger | Selects the source of the readback trigger |
| Verify | Verifies the design was downloaded correctly |

**Note:** The Clock, Display, Group, Readfpga, and Setmode commands are not available when using the MultiLINX Cable.

## Baud

Use this command to set the cable baud rate

### Syntax

```
baud    rate
```

### Parameters

This command has one parameter.

*rate* specifies the rate at which the data is transferred between the computer and the serial or XChecker Cable and MultiLINX Cable; this parameter can be set to {9600 | 19200 | 38400 | 15200}

**Note:** The 115200 baud rate is valid only on HP and PC platforms. For the MultiLINX Cable replace “115200” with “57600”.

### Examples

Following is an example of how to specify the Baud command.

```
baud 38400
```

### Abbreviations

You can abbreviate the Baud command as follows

| | |
|---|---|
| **baud** | **bau** |

## Cable

Use this command to set the cable options and to specify the type of cable.

### Syntax

**cable** {*option* | *name*}

### Parameters

The variables of the Cable command are further divided into the following parameters.

- *option* {–reset | –check | –pins}
  - –reset reprograms the XChecker cable.
  - –check tests the cable operation.
  - –pins checks the logic levels of the XChecker pins.
- *name* {multilinx | xchecker | serial | parallel}
  - multilinx establishes communication with the MultiLINX cable
  - xchecker establishes communication with the XChecker cable.
  - serial establishes communication with a serial cable.
  - parallel establishes communication with a parallel cable.

### Examples

Following are examples of how to specify the Cable command.

```
cable –pins
cable –xchecker
```

### Abbreviations

You can abbreviate the Cable command and its options as follows.

| | |
|---|---|
| **cable** | **cab** |
| **–reset** | **–rs** |
| **–check** | **–chk** |
| **–pins** | **–pn** |

## Clock

The Clock command is a synchronous mode debugging option. It enables you to specify the clock options, namely the clock type and speed. It also enables you to specify the number of clocks to apply to the target device during debugging.

## Syntax

The syntax of the Clock command is the following.

**`clock`** {[*source*][*speed*] | *apply_clocks* | *control_option*}

## Parameters

The variables of the Clock command are further divided into the following parameters.

- *source* {–internal | –external}
  - –internal sets the clock source to XChecker internal clock
  - –external sets the clock source to external / user clock
- *speed* –speed {1 | 3 | 5 | 11}
  - –speed sets the speed of the clock.
- *apply_clocks* –apply {*number*}
  - –apply cycles the device forward the number of clocks you specify
- *control_option* {–stop | –resume}
  - –stop interrupts the clock source
  - –resume resumes the clock source

## Examples

Following are examples of how to specify the Clock command.

```
clock –internal –speed 3
clock –apply 4
clock –stop
```

## Abbreviations

You can abbreviate the Clock command and its options as follows.

| | |
|---|---|
| **clock** | **clk** |
| **–internal** | **–int** |

| | |
|---|---|
| **–external** | **–ext** |
| **–apply** | **–ap** |
| **–stop** | **–st** |
| **–resume** | **–res** |

## Display

The Display command specifies which nets to debug.

### Syntax

The syntax of the Display command is the following.

**display** *operation list_of_signals*

### Parameters

The variables of the Display command are further divided into the following parameters.

- *operation* {–del | –add}
  - –del deletes the specified list of signals and/or group names from the display list
  - –add adds the specified signals and group names to the display list
- *list_of_signals* { [*list_of_sinals*] [*group_names*] | *}
  - *list_of_signals* is the list of signals you wish to probe
  - *group_names* is the list of defined signal groups you wish to probe
  - * is the complete list of signals and groups in your design if you are adding signals. It refers to the signals in the display list if you are removing signals.

### Examples

Following are examples of how to specify the Display command.

```
display –add A B C D E F G ALUOUT SWITCHES
display –del *
```

### Abbreviations

You can abbreviate the Display command as follows.

| | |
|---|---|
| **`display`** | **`dply`** |

## Download

Use the Download command to program a device by downloading the current design to that device.

### Syntax

The syntax of the Download command is the following.

**`download`** [–*verify*]

### Parameters

The Download command has one parameter.

-verify reads back the data that was downloaded to the target device and compares it to the original data.

### Examples

Following is an example of how to specify the Download command.

```
download –verify
```

### Abbreviations

You can abbreviate the Download command as follows.

| | |
|---|---|
| **`download`** | **`dn`** |
| **`–verify`** | **`–v`** |

## Exit

The Exit command enables you to exit the Hardware Debugger program.

### Syntax

The syntax of the Exit command is the following.

```
exit
```

### Parameters

The Exit command has no parameters.

### Examples

Following is an example of how to specify the Exit command.

```
exit
```

### Abbreviations

You can abbreviate the Exit command as follows.

|  |  |
|---|---|
| **exit** | **ex** |

## Force

The Force command enables you to display or hide information dialog boxes while a macro is running.

### Syntax

The syntax of the Force command is the following.

**`force`** *setting*

### Parameters

The parameters of the Force command are the following.

*setting* {–on | –off}

- –on displays the information dialog boxes
- –off hides the information dialog boxes

### Examples

Following is an example of how to specify the Force command.

```
force –on
```

### Abbreviations

You can abbreviate the Force command and its options as follows.

| | |
|---|---|
| **`force`** | **`for`** |
| **`–on`** | **`–o`** |
| **`–off`** | **`–f`** |

## Group

The Group command enables you to specify signal groups to debug as entities. It also allows you to delete existing groups of signals.

### Syntax

The syntax of the Group command is the following.

**`group`** {*del_group* | *new_group*}

### Parameters

The variables of the Group command can be divided into the following parameters.

- *del_group* is –del *groupname list_of_signals*
    - –del deletes the specified signals from the specified group name
    - *groupname* is the name of the group you are defining
    - *list_of_signals* is the list of signals you are adding or removing from the group
- *new_group* is *groupname list_of_signals* and adds the specified signals and group names to the display list
    - *groupname* is the name of the group you are defining
    - *list_of_signals* is the list of signals you are adding or removing from the group

### Examples

Following is an example of how to specify the Group command.

```
group aluout ALU0 ALU1 ALU2 ALU3 ALU4
group –del aluout ALU2
```

### Abbreviations

You can abbreviate the Group command as follows.

| | |
|---|---|
| **group** | **gp** |

## Open

The Open command opens a file for downloading.

### Syntax

The syntax of the Open command is the following.

**open** *file*

### Parameters

The variables of the Open command can be divided into the following parameters.

*file* {\*file_path*\*file_name*} or *file* {/*file_path*/*file_name*}

- *file_path* is the drive and directory in which your design is located
- *file_name* is the name of the design you want to open

### Examples

Following is an example of how to specify the Open command.

```
open C:\TMP\4KACALC\CALC4K.BIT
```

### Abbreviations

You can abbreviate the Open command as follows.

| | |
|---|---|
| **open** | **op** |

## Port

The Port command enables you to select the communications port you need for your cable.

### Syntax

The syntax of the Port command is the following.

**`port`** *port_name*

### Parameters

The parameters of the Port command are the following.

*port_name* is

{auto} for auto-detection of cable

{com1 | com2 | com3 | com4} for XChecker or serial cables on PC platforms

{/dev/tty00 | /dev/tty01} for XChecker or serial cables on HP platforms

{/dev/ttya | /dev/ttyb} for XChecker or serial cables on Sun platforms

{lpt1 | lpt2 | lpt3 | lpt4} for parallel cable only

### Examples

Following are examples of how to specify the Port command.

```
port com1
port lpt2
```

### Abbreviations

You can abbreviate the Port command as follows.

| | |
|---|---|
| **port** | **po** |

## Readfpga

The Readfpga command enables you to capture snapshots of the states of a device.

### Syntax

The syntax of the Readfpga command is the following.

**`readfpga`** *snapshots*

### Parameters

The parameters of the Readfpga command are *snapshots* {1–65534}

### Examples

Following is an example of how to specify the Readfpga command.

```
readfpga 4
```

### Abbreviations

You can abbreviate the Readfpga command as follows.

| | |
|---|---|
| **readfpga** | **rea** |

## Reset

The Reset command enables you to reset the FPGA. The reset signal is active-Low.

### Syntax

The syntax of the Reset command is the following.

```
reset
```

### Parameters

There are no parameters for the Reset command.

### Examples

Following is an example of how to specify the Reset command.

```
reset
```

### Abbreviations

You can abbreviate the Reset command as follows.

|  |  |
|---|---|
| **`reset`** | **`res`** |

## Run

The Run Command executes the macro defined in the current macro window.

### Syntax

The syntax of the Run command is the following.

**`run`** *file_name*

### Parameters

The Run command has one parameter.

*file_name* is the name of the design you want to run

### Examples

Following is an example of how to specify the Run command.

```
run calc4k.bit
```

### Abbreviations

You can abbreviate the Run command as follows.

|  |  |
|---|---|
| **`run`** | **`rn`** |

## Setmode

The Setmode command enables you to set the debugging mode.

### Syntax

The syntax of the Setmode command is the following.

**`setmode`** *debugmode*

### Parameters

The parameters of the Setmode command are the following.

- *debugmode* {*–sync* | *–async*}
    - –sync turns on the synchronous mode debugging
    - –async turns on the asynchronous mode debugging

### Examples

Following is an example of how to specify the Setmode command.

```
setmode –sync
```

### Abbreviations

You can abbreviate the Setmode command and its options as follows.

| | |
|---|---|
| **`setmode`** | **`md`** |
| **`–synch`** | **`–s`** |
| **`–async`** | **`–as`** |

## Trigger

The Trigger command specifies the source of the readback trigger.

### Syntax

The syntax of the Trigger command is the following.

**`trigger`** {*timing* | *clocksettings* | *option*}

### Parameters

The parameters of the Trigger command are the following.

- *timing* {–immediately | –external | –enterkey} sets the trigger type
- *clocksettings* –clock {before_first_snapshot between_snapshots} specifies the clock patterns to apply to the target device.

- *option* {–timeout *num* | –reset}
  - {–timeout *num*} specifies the cutoff time for a trigger to be received
  - –reset reinitializes the target device

### Examples

Following is an example of how to specify the Trigger command.

```
trigger –immediately –clock 2 2 –timeout 10–reset
```

### Abbreviations

You can abbreviate the Trigger command as follows.

| | |
|---|---|
| **trigger** | **tr** |

## Verify

The Verify command enables you to verify that the design was downloaded correctly.

### Syntax

The syntax of the Verify command is the following.

```
verify
```

### Parameters

The Verify command has no parameters.

### Examples

Following is an example of how to specify the Verify command.

```
verify
```

### Abbreviations

You can abbreviate the Verify command as follows.

| | |
|---|---|
| **verify** | **ver** |