RadHard/RadTolerant

Programming Guide

Microsemi®

Table of Contents

Introduction

This manual contains information about programming Microsemi RadHard and RadTolerant devices using Activator and Silicon Sculptor device programmers. The manual also contains programming failure information and post-programming guidelines. The information is meant to ensure proper programming and the highest possible programming yield of RadHard and RadTolerant devices.

Microsemi provides RadHard devices guaranteed to meet specific radiation characteristics. Through 2006, Microsemi offered the RH1020 and RH1280 devices. In addition, many of Microsemi's commercial and MIL-STD-883 devices have exhibited some capability of withstanding the effects of radiation environments. These devices include the A1020B, A1280A, A1460A, and A14100A devices and are referred to as RadTolerant devices in this guide.

System Requirements

Designer Software Version

Table 1-1 lists the required version of the Designer software to program RadHard and RadTolerant devices. Earlier versions of the software than those listed do not support and will not program RadHard devices.

***Table 1-1* • Designer Version Required to Program RadHard and RadTolerant Devices**

Microsemi SoC Products Group Part Number	DSCC Part Number	Earliest Designer Software Revision for a AFM File
RH1280-CQ172V	5962F9215603QYC	R2-1999 or later
A1020B-CQ84B	5962-9096503MTC	
A1280A-CQ172B	5962-9215601MYC	
A1460A-CQ196B	5962-9550801MYC	
A14100A-CQ256B	5962-9552101MYC	
RH1020-CQ84V	5962F9096505QTC	R2-1999 or later
RT54SX16-CQ208B	5962-9956901QYC	R1-2000 or later
RT54SX16-CQ256B	5962-9956901QXC	
RT54SX32-CQ208B	5962-9958603QYC	
RT54SX32-CQ256B	5962-9958603QXC	
RT54SX32S-CQ208B	5962-0150801QYC	R1-2001 or later
RT54SX32S-CQ256B	5962-0150801QXC	
RT54SX32S-CQ208E	5962-0150803QYC	
RT54SX32S-CQ256E	5962-0150803QXC	
RT54SX72S-CQ208B	5962-0151501QYC	
RT54SX72S-CQ256B	5962-0150801QXC	
RT54SX72S-CQ208E	5962-0151503QYC	
RT54SX72S-CQ256E	5962-0150803QXC	

PC/Workstation Requirements

Please refer to the latest version of the installation instructions at www.microsemi.com/soc for a complete listing of the minimum system requirements.

Additional Equipment

In addition to the minimum system requirements specified in the installation instructions, (available at www.microsemi.com/soc/documents/install_ug.pdf), the following equipment is also required or recommended:

Required

- ESD grounded wrist strap
- ESD workstation

Recommended

- Ionizer
- Oscilloscope

Document Organization

The *RadHard/RadTolerant Programming Guide* is divided into the following chapters:

Chapter 1 – "Programming RadHard/RadTolerant Devices" contains information and procedures to program RadHard and RadTolerant devices using APS programming software.

Chapter 2 – "Post-Programming Recommendations" discusses recommendations for post-programming burn-in and testing of RadHard and RadTolerant devices.

Chapter 3 – "Device Programming Failures" describes conditions that can cause a RadHard or RadTolerant device to fail.

Chapter 4 – "Debugging a Device with an Activator" contains information and procedures to debug a RadHard and RadTolerant devices using APS programming software and an Activator.

Appendix A – "Troubleshooting" describes some common hardware and software problems and solutions to those problems.

Appendix B – "Testing and Programming Microsemi FPGAs" discusses testing and programming methodologies for Microsemi FPGAs.

Appendix C – "Silicon Signature Decode" explains information that is contained in a Silicon Signature.

Appendix D – "AVI File Description" contains an example and description of an AVI log file.

Appendix E – "JobMaster (Silicon Sculptor only)" enables an Administrator to set up a job to precise specifications, test the results, and then protect the routine so that it cannot be modified inadvertently.

Appendix F – "Product Support" provides information about contacting Microsemi SoC Products Group for customer and technical support.

Document Assumptions

The information in this manual is based on the following assumptions:

1. You are familiar with PCs and Windows operating environments.
2. You are familiar with UNIX workstations and UNIX operating systems.
3. You are familiar with FPGA architecture and FPGA design software.

Document Conventions

The following conventions are used throughout this manual.

Information that is meant to be input by the user is formatted as follows:

`keyboard input`

Messages that are displayed on the screen appear as follows:

```
You have a fatal error.
```

The contents of a file is formatted as follows:

`file contents`

Related Manuals

This guide refers to the following manual, which provides additional information about installing hardware for use in designing and programming RadHard and RadTolerant devices:

Adaptec AVA-1505 AT-to-SCSI Host Adapter Installation Guide.

This guide contains installation information for the Adaptec AVA-1505 AT-to-SCSI Host Adapter included with the Activator hardware.

Online Documentation

The Designer software comes with online documentation. This documentation is in PDF format on the CD-ROM in the “\doc” directory. This documentation is also installed onto your system when you install the Designer software.

To view the Designer online documentation, you must have Adobe® Acrobat Reader® software installed. Microsemi provides Acrobat Reader on the CD-ROM.

Online Help

The Designer software comes with online help. To view all of the help systems in the Designer Series software, double click the online help icon in the Designer Series program group. Online help specific to each software tool is also available in Designer, ACTgen, and APS.

1 – Programming RadHard/RadTolerant Devices

This chapter discusses the recommended programming flow, and describes the procedure for programming a RadHard or RadTolerant device using an Activator connected to a PC or workstation and either APSW or APS2 software. This chapter also describes the procedure for programming devices using a Silicon Sculptor connected to a PC with the Silicon Sculptor Software installed.

RH/RT programming is supported on both the Silicon Sculptor and the Activator for all devices except the SX-S. SX-S device programming is not supported in Activator. You must use Silicon Sculptor to program SX-S devices. Also, for any SX-S failure report you don't need to submit the AVI files. Instead, you must submit the exact failure messages with the software and hardware revision number.

Programming Flow

The recommended programming flow for a RadHard or RadTolerant device has four main steps: test the Activator/Silicon Sculptor, program a commercially equivalent part, program a RadHard or RadTolerant device, and save the AVI files (where required). These steps are described in the following sections.

Test the Activator

Test your Activator before programming to ensure the Activator is working properly. Refer to the "Testing an Activator" section on page 29 for detailed information about testing your Activator.

Test the Silicon Sculptor

Test your Silicon Sculptor before programming to ensure that it is working properly. For more information on testing the Silicon Sculptor software, refer to the Microsemi SoC Products Group knowledge base at www.microsemi.com/soc/kb/search.aspx and search for the keyword s*culptor*. Choose *How to Run a Full System Self-Test on the Silicon Sculptor* (for DOS) or *How to Run Self-Test for Silicon Sculptor with SCULPTW* (for Windows) as appropriate. For more information in this manual, refer to the "Testing Silicon Sculptor" section on page 33.

Program a Commercially Equivalent Part

Before you program your device, Microsemi recommends that you program a commercially equivalent device to ensure the proper setup of the hardware and software. Table 1-1 lists commercially equivalent devices for the RadHard and RadTolerant devices. Once the you have successfully programmed a commercial device, you can program your RadHard or RadTolerant device.

Note: Due to potential timing differences between RH/RT and commercial devices, timing verification of the design for the RH/RT device may be required (even if timing verification has already been performed for the commercial device).

***Table 1-1* • Commercially Equivalent Devices for the RadHard/RadTolerant Devices**

RadHard/RadTolerant Device	DSCC Part Number	Commercially Equivalent Part
RH1020-CQ84V	5962F9096505QTC	A1020B-CQ84C
RH1280-CQ172V	5962F9215603QYC	A1280XL-CQ172C
RT1280A-CA172B	5962F9215603QYC	A1280A-CQ172C
A1020B-CQ84B	5962-9096503MTC	A1020B-CQ84C
A1280A-CQ172B	5962-9215601MYC	A1280A-CQ172C
A1460A-CQ196B	5962-9550801MYC	A1460A-CQ196C
A14100A-CQ256B	5962-9552101MYC	A14100A-CQ256C
RT54SX16-CQ208B	5962-9956901QYC	A54SX32A-CQ208
RT54SX16-CQ256B	5962-9956901QXC	A54SX32A-CQ256
RT54SX32-CQ208B	5962-9958603QYC	A54SX32A-CQ208
RT54SX32-CQ256B	5962-9958603QXC	A54SX32A-CQ256
RT54SX32S-CQ208B	5962-0150801QYC	A54SX32A-CQ208
RT54SX32S-CQ256B	5962-0150801QXC	A54SX32A-CQ256
RT54SX32S-CQ208E	5962-0150803QYC	A54SX32A-CQ208
RT54SX32S-CQ256E	5962-0150803QXC	A54SX72A-CQ256
RT54SX72S-CQ208B	5962-0151501QYC	A54SX72A-CQ208
RT54SX72S-CQ256B	5962-0150801QXC	A54SX72A-CQ256
RT54SX72S-CQ208E	5962-0151503QYC	A54SX72A-CQ208
RT54SX72S-CQ256E	5962-0150803QXC	A54SX72A-CQ256

Program a RadHard/RadTolerant Device

Use the Activator/Silicon Sculptor to program your device. Make sure you follow ESD device handling procedures when removing a RadHard or RadTolerant device from its packaging and when placing it into an Activator or programming with Silicon Sculptor. Refer to "Device Handling" and "Removing a Device from the Carrying Case" on page 19 for information about handling and unpacking RadHard and RadTolerant devices.

Save the AVI File

When you program a chip, the programming history is saved in the design_name.avi file. The AVI file contains programming data for each antifuse programmed, including the number of programming pulses applied and fuse current readings. This file is overwritten each time you select the Activate (Activator) or Program command (Silicon Sculptor). Once you have programmed your device, save your AVI file. Refer to the "AVI File Description" section on page 75 for information about the AVI file.

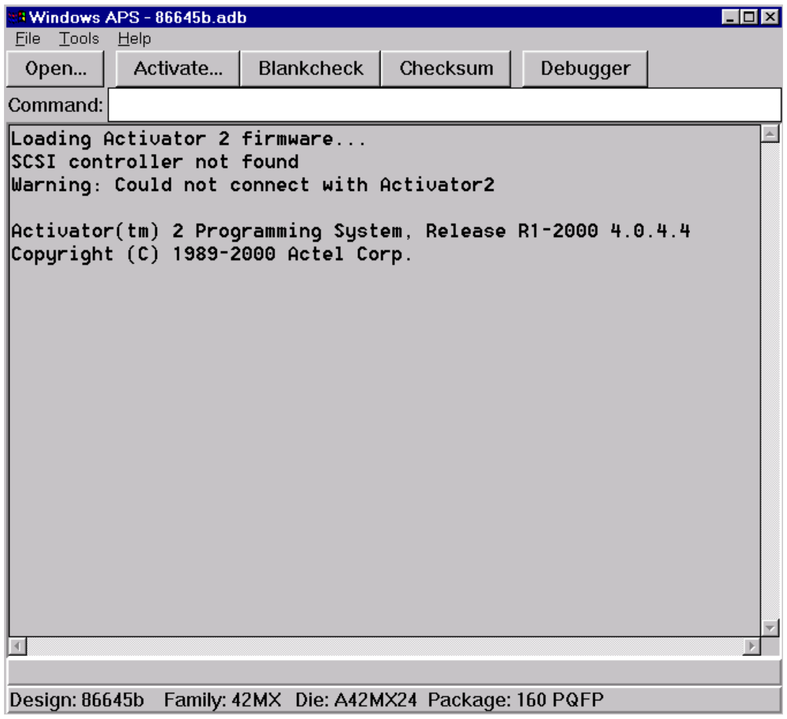
Note: The AVI file is required to program RH devices only; it is not required for RT devices. Also, note that Silicon Sculptor does not always generate an AVI file with RH devices.

If your device fails to program, refer to the "Device Programming Failures" section on page 27 for detailed information about programming failures and maximum allowed programming failure guidelines.

Note: The *.avi file is required for failure analysis of all RadHard devices, and also the RadTolerant RT1020 device. For more information, please refer to the "Returning Failed Devices" section on page 34.

Activator APSW Software Overview

This section describes the APSW interface, including information about using the interface to program RadHard and RadTolerant devices. Figure 1-1 shows the main window for APSW.



```
Loading Activator 2 firmware...
SCSI controller not found
Warning: Could not connect with Activator2

Activator(tm) 2 Programming System, Release R1-2000 4.0.4.4
Copyright (C) 1989-2000 Actel Corp.
```

Design: 86645b   Family: 42MX   Die: A42MX24   Package: 160 PQFP

***Figure 1-1* • ASPW Main Window**

Activate

The Activate button or menu command is used to program a device. Refer to "Programming a Device with Activator" on page 19 for detailed information about using the Activate command to program a RadHard or RadTolerant device.

Blankcheck

The Blankcheck button or menu command executes a test to determine if a device has already been programmed. Blankcheck displays a report for each Activator socket that has an adapter module plugged into it (the Activator 2S only has one socket). Blankcheck is performed automatically before the chip is programmed whenever you execute the Activate command.

The result of executing Blankcheck is either *blank* or *not blank* followed by the Silicon Signature, Checksum, and Security Fuse status read from the device. Only blank devices of the correct type (according to the design parameters) result in a blank status. A Security Fuse status of 0 indicates that the security fuse has not been programmed; a 1 indicates that the security fuse has been programmed.

To run Blankcheck:

Click the **Blankcheck** button, or from the Tools menu choose the **Blankcheck** command.

Checksum

The Checksum button or menu command verifies that the current programming file is the same one that was used to program the device. The Checksum command compares the checksum number, computed from the programming file (*.afm or *.adb), to the checksum number programmed into the chip. If the two numbers are identical, APSW displays PASSED. If the two numbers are not identical, the program displays FAILED, with additional comments to briefly explain why it failed.

Note: If you have already programmed the Program fuse on an RH1020 or A1020B device, you cannot read the Checksum from the device.

To run Checksum:

Click the **Checksum** button, or from the Tools menu, choose the **Checksum** command**.** The APSW window displays the checksum number, along with a Passed or Failed message. If Checksum fails, APSW displays a message that describes why.

Debugger

The Debugger button or menu command initializes the Debugger test environment. The Debugger environment enables you to test a programmed device before you place it in your system circuit. With Debugger, you can probe the chips internal nodes by applying stimulus through an Activator. Enter Debugger commands through the Command box located at the top of the APS window. Refer to the "Debugging a Device with an Activator" section on page 35 for information about using Debugger.

Note: The Debugger does not support SX, SX-A, eX, or ProASIC® devices.

To run Debugger:

Click the **Debugger** button, or from the Tools menu, choose the **Debugger** command.

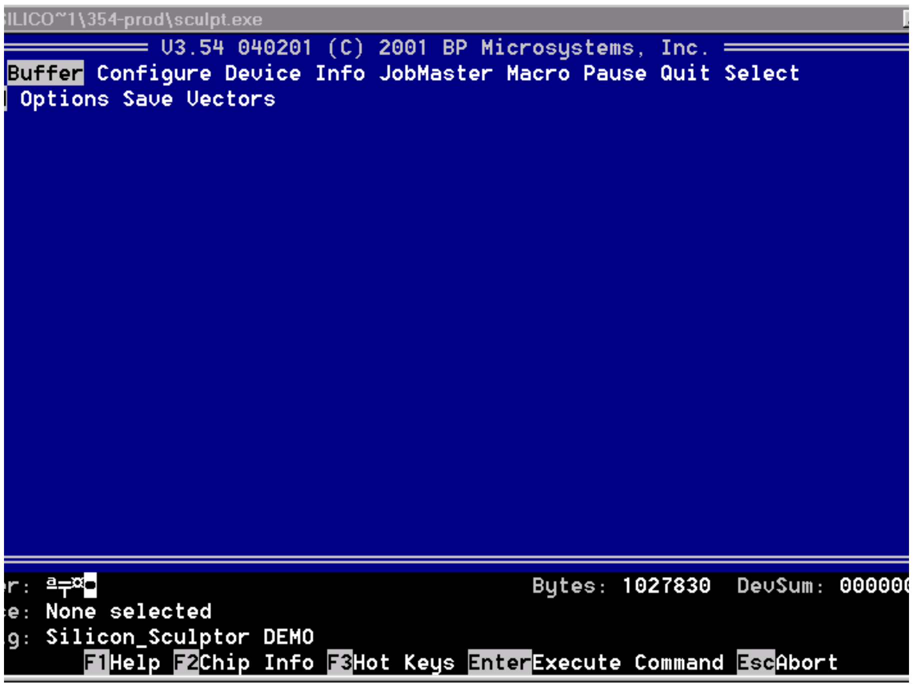
Note: If you have already initiated Action Probe, Debugger is also initialized and available for use. When you use Debugger with an Activator 2s, APSW recognizes the Adapter Module as port 4.

Silicon Sculptor Software Overview

This section describes both DOS and Windows versions of Silicon Sculptor software.

Silicon Sculptor for DOS

This section describes the Silicon Sculptor DOS Software interface including information about using the interface to program RadHard and RadTolerant devices. Figure 1-2 shows the main window for Silicon Sculptor DOS software.



```
ILICO~1\354-prod\sculpt.exe
═══════════ V3.54 040201 (C) 2001 BP Microsystems, Inc. ═══════
Buffer Configure Device Info JobMaster Macro Pause Quit Select
 Options Save Vectors

r: ª╤¤■                                  Bytes: 1027830  DevSum: 00000
e: None selected
g: Silicon_Sculptor DEMO
      F1Help F2Chip Info F3Hot Keys EnterExecute Command EscAbort
```

Figure 1-2 • **Silicon Sculptor for DOS Main Window**

AFS/Serialize – Serializes devices programmed. Use this command to select serialization pattern.

AFS/Upgrade – Enter an upgrade code if required. This command is required in order to utilize additional options (such as JobMaster) and to fix various bugs.

Buffer/Load – Loads a data file from disk into the buffer. This command loads the buffer before programming.

Buffer/Options – Sets the options that control the loading of the buffers. Selects generic buffer options such as: (1) checksum method or (2) default value. Also, sets several options pertaining to the Buffer/Load command, including: (1) changing the default directory; (2) changing the default filename; (3) disabling the listing of files in a selector box; (4) disabling automatic file type identification; and (5) enabling the loading of the filename when the software starts up.

Buffer/Save – Saves data buffer contents to disk. Use this command to save a file.

Buffer/Vectors – Not applicable for Microsemi users.

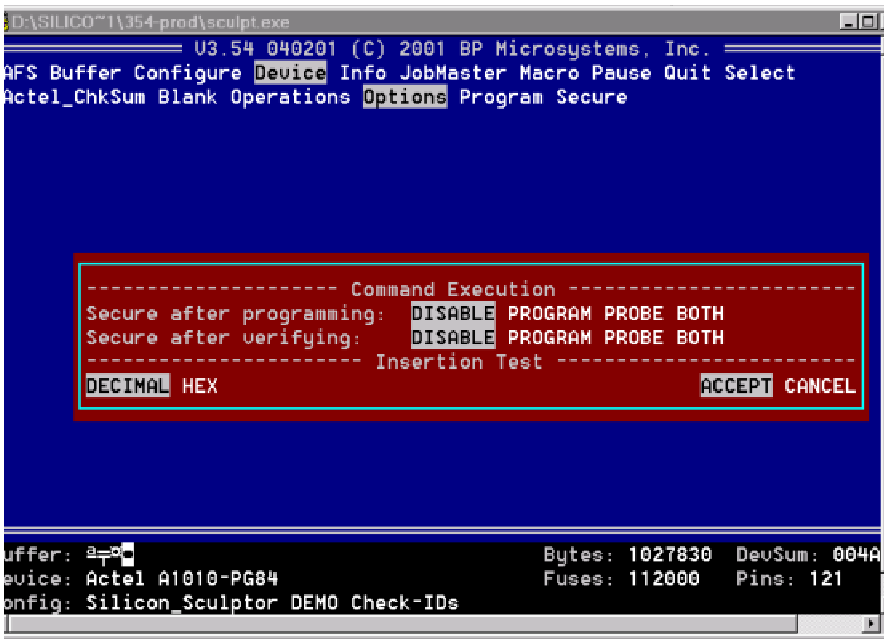
Configure – Selects the configuration options that control operation of the programmer, including: (1) whether you have a color or monochrome display; (2) which parallel port you want the software to look at first when it starts up; (3) when to save the configuration options that have been set under this Configure command and under the Buffer/Options, Device/Options, and Device/Operations commands; (4) experienced or novice mode of operation; and (5) whether the startup messages appear.

Device/Actel_ChkSum – Calculates, displays, and verifies the design checksum for single-site operation. Use this command to verify the design checksum of the program file to the checksum value programmed into the device; also displays the 16-bit checksum and the 20-bit user defined design ID.

Device/Blank – Verifies that a device is blank (all fuses unprogrammed, or open). Prior to programming, use this command to verify that a device has not been previously programmed (this operation is executed automatically prior to programming any device).

Device/Operations – Defines the number of device operations performed. Prior to using the programmer in a high-volume production setting, use this command to select how many devices you wish to operate on (concurrent programming).

Device/Options – Enables you to specify different programming, reading, and testing options to reflect your target system requirements (this should be done prior to performing a Device/Program command). Figure 1-3 shows an example of the Device/Options command in use.



```
D:\SILICO~1\354-prod\sculpt.exe
══════════════════ V3.54 040201 (C) 2001 BP Microsystems, Inc. ═════════════════
AFS Buffer Configure Device Info JobMaster Macro Pause Quit Select
Actel_ChkSum Blank Operations Options Program Secure

          -------------------- Command Execution -----------------------
          Secure after programming:  DISABLE PROGRAM PROBE BOTH
          Secure after verifying:    DISABLE PROGRAM PROBE BOTH
          ---------------------- Insertion Test ------------------------
          DECIMAL HEX                                      ACCEPT CANCEL

uffer: a=¤■                                  Bytes: 1027830  DevSum: 004A
evice: Actel A1010-PG84                      Fuses: 112000   Pins: 121
onfig: Silicon_Sculptor DEMO Check-IDs
```

Figure 1-3 **• Device/Options Command in DOS**

Device/Program – Programs a device from data in the buffer. Use this command to program your design into the device. This command also verifies that the correct device type is inserted into the socket and verifies the device is blank (unprogrammed) before you begin programming.

Device/Secure – Programs the security fuse(s). This command prevents probing of the device. You can program Security fuses on an Microsemi device that has had its Array fuses previously programmed. The ability to program Security fuses after programming Array fuses is designed to enable you to debug your device design with the Silicon Explorer diagnostic tool.

Info/Chip – Displays some characteristic information on the currently selected chip. This is a useful way to learn which package types are available for a chip and which programmers support a chip.

Info/Log – Sends information displayed in main window to a log file. You may wish to create a log of all the activity during a particular programming session; you may view this log file at any time, printed, or saved to another file name.

Info/Socket Module – Displays some characteristic information on the currently selected socket module. This is a useful way to learn which socket module is required for a particular chip.

JobMaster – Please refer to the "JobMaster (Silicon Sculptor only)" section on page 77 for more information.

Macro/Debug – Plays a sequence of commands stored in a macro file and pause after each line read from the file. This command is intended for advanced users of macro commands; it is useful when you have generated your macro files by some other means than the Macro/Record command. For example, you used an editor or wrote your own program to generate the macro files, and now you are encountering problems with the Macro/Play command.

Macro/Play – Plays a sequence of commands stored in a macro file. This command saves time when executing a series of operations for a specific chip. For convenience, you may record a macro file for each chip you program on a regular basis.

Macro/Record – Stores a sequence of commands in a macro file. Use this command to store commonly used command sequences in macro files to speed up operation and reduce operator errors. Macro files can be played back from the Macro/Play command, from the DOS command line, from a batch file, from a make file, or from another macro file.

For more information on the Macro command, please refer to the *Silicon Sculptor's User's Guide.*

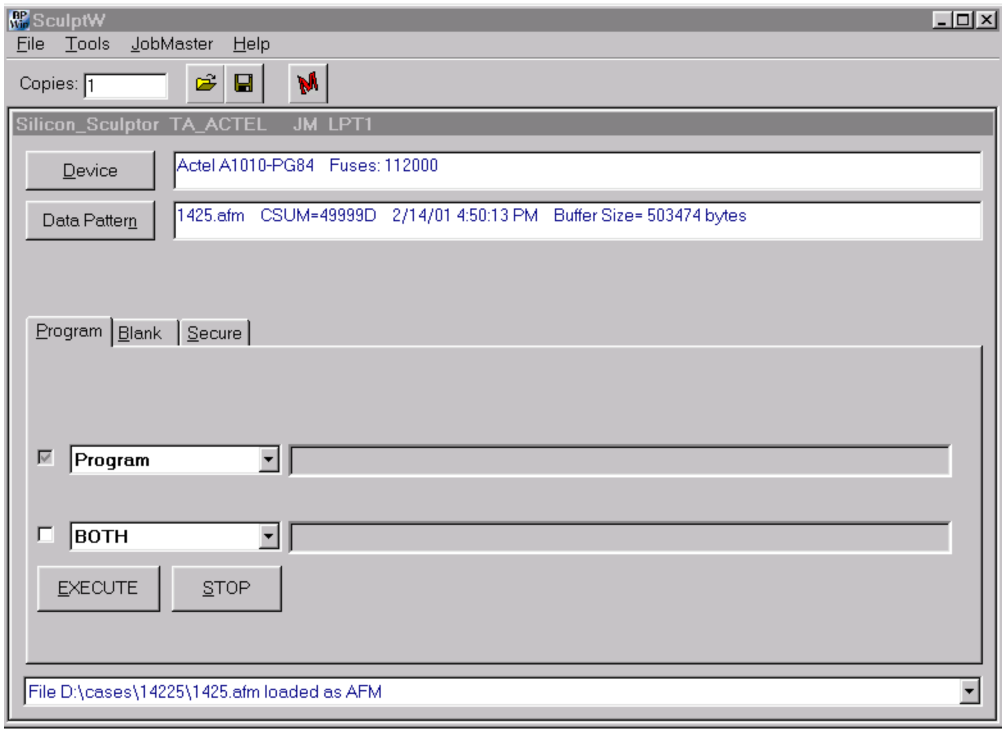
Pause – Executes a DOS shell. This command executes DOS commands, then return to the programmer in its present configuration.

Quit – Exits the program and return to DOS. You are returned immediately to DOS. Any buffer changes will be lost; however, if you have Save Configuration: set to **AUTOMATIC** in the Configure command, then all the options you have set during this session are saved and restored the next time you start the software.

Select – Specifies the Microsemi FPGA part number to select the proper programming algorithm. Selecting a device configures the programmer for the correct programming algorithm; the algorithm includes voltage, timing, and pinout requirements. You must use this command before programming or performing any other device operations.

Silicon Sculptor for Windows

This section describes the Silicon Sculptor Windows Software interface, including information about using the interface to program RadHard and RadTolerant devices. Figure 1-4 shows the main window for Silicon sculptor Windows software.


SculptW

File Tools JobMaster Help

Copies: 1

Silicon_Sculptor TA_ACTEL JM LPT1

Device: Actel A1010-PG84 Fuses: 112000

Data Pattern: 1425.afm CSUM=49999D 2/14/01 4:50:13 PM Buffer Size= 503474 bytes

Program | Blank | Secure

☑ Program

☐ BOTH

EXECUTE STOP

File D:\cases\14225\1425.afm loaded as AFM

Figure 1-4 • **Silicon Sculptor for Windows Main Window**

File

Use the commands under the File menu to control the communication with the programmer, input special code to enable advanced features, and monitor the number of copies you want to make. In addition, use the commands under the file menu to reset the program settings to their defaults and modify certain global options.

Configure

Configure re-establishes communication with the programmer. This enables you to change programmers, if necessary. It also opens a dialog box that lists several options. This dialog box enables you to switch the handler type from manual to auto-handler status and specify an experience level (novice or experienced).

Upgrade

The Upgrade dialog box allows you to input a special code, which enables Advanced Features (AFS) in the software.

Tools

Use the commands in the Tools menu to reset the configuration of Silicon Sculptor.

Programmer Diagnostic

Programmer Diagnostic runs the programmer diagnostic test. The Silicon Sculptor diagnostics ensure that the power supplies function properly and test the integrity of all the pin drivers. For more information please refer to page 68 of this manual.

Default Configure

Default Configure enables you to reset all the settings to the default configuration. When you select this option, you are asked to confirm whether or not you want to reset to the default configuration. This option is available to allow you to reset any given command back to its original settings.

Options

The Options dialog box enables you to set global options.

- Error Beep turns on or off the sounds used to indicate completion, errors, end of job, etc.
- Elapsed Time displays the duration of time of a specific operation, i.e. Read, Program, etc.

Device Marker

Not applicable for Microsemi devices.

JobMaster

For more information, please refer to "JobMaster (Silicon Sculptor only)" on page 77.

Copies

The Copies field enables you to enter the number of devices you want to program. The software automatically tracks how many you have finished, how many failed, and how many remain to be programmed. When Sculptor reaches the Copies number successfully, it displays a Job Summary report for printing.

Help

Help/Tech Support lists the phone numbers and e-mail address for reaching Technical Support. Refer to the "Product Support" section on page 81 for Technical Support contact information.

Help/About Silicon Sculptor displays the current Silicon Sculptor version number and specifics about the software.

Device Button

The Device button enables you to access to the list of devices supported by the designated device programmer via the Select Device dialog box.

- **Select Device Dialog Box –** The Select Device Dialog Box offers a list of devices supported on the programmer model you selected in Configure. Select a device to configure the programmer with the correct programming algorithm. The algorithm is specified by Microsemi and contains the voltage, timing, and pin-out requirements.

Dev Info Button

The Dev Info button lists information for the highlighted device, including certain notes and relevant precautions. The notes and precautions include algorithm settings, incorrect data pattern information, erasing procedures, etc.

Data Pattern Button

Once you make a device selection, you must choose a data pattern. Click the Data Pattern button to display the Data Pattern dialog box. Use this dialog box to load data into the buffer from a file or another device. Browse to your file and load as necessary.

Open Button

The Open button brings up the Load File into Buffer window. Several options are available before loading the file. Microsemi devices use the *.afm file.

Save Button

The Save button brings up the Save File from Buffer dialog box and saves the buffer contents to disk. You may choose any local or network drive available when you save the file.

Main Screen File Tabs

Depending on the device you select, any of the following action/function file tabs may appear in your viewing screen.

- **Actel_ChkSum -** Calculates, displays, and verifies the design checksum for single-site operation. Verifies the design checksum of the program file to the checksum value programmed into the device. Also displays the 16-bit checksum and the 20-bit user defined design ID.
- **Blank -** Verifies that a device is blank (all fuses unprogrammed, or open). Prior to programming, verifies that a device has not been previously programmed. This operation is automatically executed prior to programming any device.
- **Program -** Programs a device from data in the buffer. The program button programs your design into the device. It also verifies that the correct device type is inserted into the socket and verifies the device is blank (unprogrammed) before programming begins.
- **Secure -** Programs the security fuse(s) to prevent unwanted probing of the device. You can program Security fuses on an Microsemi device that has had its Array fuses previously programmed. The ability to program Security fuses after programming Array fuses is designed to allow you to debug your device design with the Silicon Explorer diagnostic tool. After you debug your design, you can program the security fuses to secure the device from further probing.

For more information on Software please refer to the *Silicon Sculptor's User's Guide.*

Preparation for Programming a Device

This section contains information about programming files and describes the procedure for programming a RadHard or RadTolerant device with the Activator and Silicon Sculptor Software.

Supported Device Files

To program devices with APSW, you must have a programming file. APSW can read the following programming file types:

- Database file (*.adb) for a design that has a completed layout
- Programming file (*.afm) exported from Designer using the Fuse command

Note: If Layout has been completed in Designer, you do not need to execute the Fuse command. APSW automatically extracts programming information from the *.adb file.

To program devices with Silicon Sculptor, you must have a programming file (*.afm) exported from Designer using the Fuse command.

User-Defined Silicon Signature

To specify a user-defined Silicon Signature to be programmed into a device, enter the signature from within Designer using one of the following methods:

1. Create an *.afm file containing a Silicon Signature. Select the **Fuse** command, enter the **Silicon Signature**, then export the *.afm file. Refer to the Designer Series documentation for information about how to use the Fuse command to generate an *.afm file.
2. Set the **SIG** variable. Choose **Set** from the Options menu, and enter the desired value.

Note: The Silicon Signature must be a hexadecimal number, no more than 5 digits long.

Programming Checklist

Before executing any commands from APSW, sculpt.exe (DOS), or sculptw.exe (Windows), verify the following:

1. The Activator/Sculptor hardware is powered on and the green LED is lit.
2. The appropriate Adapter Module(s) is plugged into the socket(s) on the Activator.
3. A device is inserted in the Adapter Module, with pin 1 oriented as indicated in the diagram printed on the Adapter Module.

Note: Insert a device into the Adapter Module socket after the Module is plugged in and the Activator is powered up.

Device Handling

RadHard and RadTolerant devices are CMOS devices and require proper grounding and ESD handling procedures. Although all Microsemi parts have static discharge protection built in, you should always follow ESD handling procedures when handling RadHard and RadTolerant devices. Handle devices by the corners to avoid touching the leads.

Always keep the RadHard and RadTolerant devices in their insulative carrying cases until they are used, and keep the surrounding environment clean and free of dust and debris. Periodically check the Adapter Module sockets to verify that they are free of dirt or other debris that would prevent good electrical pin connections between the device and socket.

When loading RadHard and RadTolerant devices in the Adapter Module socket, be sure to orient them so that pin 1 is oriented according to the diagram on the Adapter Module. Damage can occur if the FPGA is loaded incorrectly.

You should always wear grounded wrist straps at an ESD workstation when handling RadHard and RadTolerant devices. A calibrated ionizer should be on and functioning properly at the workstation. An ionizer air stream should be directed over the parts at all times.

Removing a Device from the Carrying Case

Use the following ESD procedure to remove a RadHard device from the insulative individual carrying case before placing it in an Adapter Module. Although RadTolerant devices are not packaged in individual carrying cases, you should follow ESD procedures when handling any Microsemi device.

IMPORTANT

You should always wear a grounded wrist strap at an ESD workstation when handling a RadHard or RadTolerant device. A calibrated ionizer should be on and functioning properly at the workstation. An ionizer air stream should be directed over the parts at all times.

1. Slowly remove the individual carrying case from the carrier box. The individual carrying case may build a charge while being removed from the foam in the carrier box.
2. Ionize the individual carrying case. Hold the individual carrying case approximately three feet from the ionizer. Expose each side of the case to the air stream for 30 to 60 seconds.
3. Remove the RadHard device from the individual carrying case. Open the case, remove the device and place the device on top of ESD sensitive foam.

You can now place the device in the Adapter Module or repackage it in a static dissipative container.

Programming a Device with Activator

Programming typically requires from 5 to 15 minutes for commercial and RadTolerant devices and 30 to 60 minutes for RadHard devices, depending on design complexity, the Microsemi device you chose, and your system environment. During programming, the Activator/Silicon Sculptor dynamically verifies that each antifuse is programmed correctly. In addition, test vectors are applied to verify that only the selected antifuses are programmed. Due to the unique, high-density architecture of Microsemi devices, you can verify the programmed state of all antifuses only during programming, not after. The following procedure describes how to program a device.

To program a device with APSW:

1. Invoke **APSW**.

   #### *UNIX Workstation*

   Type the following command at the prompt:

   ```
   apsw
   ```

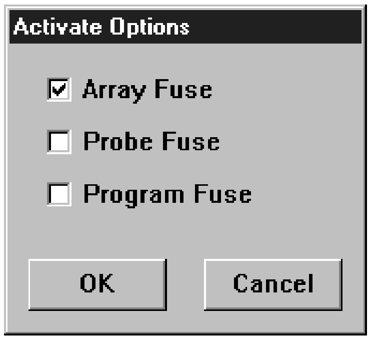
   #### *PC*

   Double-click the Windows Programming icon located in the Designer Series program group.

   This displays the APSW window (see Figure 1-1 on page 11**).**
2. Open your design. Click the **Open** button or from the File menu, choose the **Open** command. This displays the Open dialog box. Type in the design name or browse to the directory that contains the design_name.adb (or *.afm) file and select it. Click **OK.**
3. Choose fuse programming options. Click the **Activate** button or from the Tools menu, choose the Activate command. This displays the Activate Options dialog box.
4. Program your device. You can program Array fuses only, the Security fuse only, or both Array fuses and the Security fuse. For RH1020 and A1020B devices, APSW displays the dialog box shown in Figure 1-5 on page 20. For RH1280, A1280A, A1460A, and A14100A devices, APSW displays the dialog box shown in Figure 1-6 on page 20. Choose the desired fuse programming options and click **OK**. Refer to the "Programming Security Fuses" section on page 21 for a detailed discussion about programming security fuses.

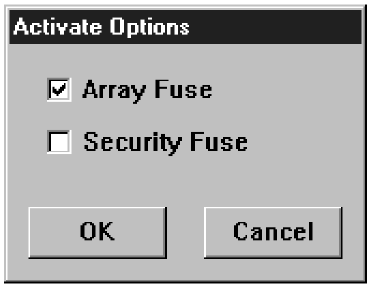
IMPORTANT

The Program or Security fuse must be the last fuse programmed.



[✓] Array Fuse
[ ] Probe Fuse
[ ] Program Fuse

Figure 1-5 • **Activate Options for RH1020 and A1020B Devices**



[✓] Array Fuse
[ ] Security Fuse

Figure 1-6 • **Activate Options for RH1280, A1280A, A1460A, and A14100A Devices**

APSW displays an Output Window on the screen and begins the programming sequence. The status bar in the Output Window displays the percentage complete of the programming sequence. When the device is 100% programmed, the finished programming status Passed or Failed is displayed on the screen.

5. Save the AVI file. Exit APSW and move or rename the AVI file for the design. The AVI file is written to the same directory as your programming file. Refer to the "AVI File Description" section on page 75 for information about AVI files.

Programming a Device with Silicon Sculptor

Use the following procedures to program a device with Silicon Sculptor for DOS or Windows, as appropriate.

To program a device with Silicon Sculptor for DOS:

1. Double-click **sculpt.exe** to invoke the software.
2. Open your design from the Buffer/Load menu. Select the file from the Directory menu and load the file with the *.afm extension for a particular device.
3. Select the corresponding device from the Select menu.
4. Run the blank check for the inserted device. Use the Device/Blank command to run the blank check.
5. Program the device. From the Device/Program menu, select the **Program** command to start programming the device.
6. Verify the checksum. When programming is complete, select the **Device/Actel_ChkSum** command to verify the checksum from the programmed device.
7. Program Array or Security fuses. You may choose to program Array fuses only, Security fuses only, or both at the same time. After programming the Array fuse you can program the Security fuse later by the Device/Secure menu.

To program a device with Silicon Sculptor for Windows:

1. To invoke the software, double-click **sculptW.exe**, or click the **Start** button, and from the Programs menu, select BP Microsystems and SculptW.
2. From the SculptW interface, click the **Device** button.
3. Select your device. Scroll down if necessary.
4. From the SculptW interface, click the **Data Pattern** button.
5. Select the desired *.afm file from the directory.
6. Run the blank check. From the SculptW interface, select the Blank tab and Press the **BLANK** button to run the blank test for the device.
7. Program the device. From the SculptW interface, select the Program tab. In the Program tab, select **Program** in the first pull-down menu. Press the **EXECUTE** button to program the device.
8. Verify the checksum. When programming is complete, again select Actel_ChkSum in the first pull-down menu. Press the **EXECUTE** button to verify the checksum from the programmed device.
9. Program Array or Security fuses. You may choose to program Array fuses only, Security fuses only, or both at the same time. After programming the Array fuse you can program the Security fuse later by the Device/Secure menu.

Programming Security Fuses

You can program Security fuses on a RadHard or RadTolerant device that has had its Array fuses previously programmed. The ability to program Security fuses after programming Array fuses is designed to allow you to debug your device design with the Debugger, ActionProbe, or Silicon Explorer diagnostic tool. After you debug your design, you can program the security fuses to secure the device from further probing.

RH1020, A1020B Security Fuse Configurations

The RH1020 and A1020B devices contain two security fuses: Probe and Program. Programming the Probe fuse disables the Probe Circuitry, which disables the use of the Debugger, ActionProbe, and Silicon Explorer diagnostic tools. Programming the Program fuse prevents further programming of the device, including programming the Probe fuse.

Table 1-2 summarizes the effects of programming the Security fuses on the PRA, PRB, SDI, and DCLK pins.

***Table 1-2* • RH1020, A1020B Security Fuse Configurations**

Mode[1]	Program	Probe	PRA, PRB	SDI, DCLK
Low	No	No	User-defined I/O	User-defined input[2]
Low	No	Yes[3]	User-defined I/O	User-defined input[2]
Low	Yes[4]	No	User-defined I/O	User-defined I/O
Low	Yes[4]	Yes[3]	User-defined I/O	User-defined I/O
High	No	No	Probe Circuit outputs[5]	Probe Circuit inputs[6]
High	No	Yes[3]	Probe Circuit disabled	Probe Circuit disabled
High	Yes[4]	No	Probe Circuit outputs[5]	Probe Circuit inputs[6]
High	Yes[4]	Yes[3]	Probe Circuit disabled	Probe Circuit disabled

Notes:

1. *The MODE pin switches the device between the normal operating mode (MODE = 0) and the Probe Circuit mode (MODE = 1).*
2. *The Program fuse must be programmed if the SDI or DCLK pins are to be used as an output or a bidirectional pin.*
3. *If the Probe fuse is programmed, the Probe Circuit is permanently disabled, which disables the Debugger, ActionProbe, and Silicon Explorer diagnostic tools.*
4. *If the Program fuse is programmed, all programming of the device is disabled, including programming the array fuses and the Probe fuse.*
5. *The PRA output and a separate I/O buffer share the use of a single device pin. The PRA output and the output function of the I/O buffer are multiplexed. The same is true for PRB. The Probe Mode that is loaded into the Mode Register will determine which output buffer is active during probing. There are three possible Probe Modes: PRA only, PRB only, and PRA and PRB.*

   When you select the PRA only mode, the PRA output becomes active and the output function of the I/O buffer associated with the PRA pin is inhibited. However, the input buffer portion of the I/O buffer associated with the PRA pin is still active. Any internal signal that appears on the PRA output is fed back through that input buffer to the internal Logic Modules. This could interfere with the expected function of the design during probing. Microsemi recommends that you use an input latch on PRA and PRB to prevent the feedback while probing. PRB will function as a normal I/O in the PRA only mode.

   The PRB only mode is functionally equivalent to the PRA only mode. PRA also functions as a normal user I/O in the PRB only mode.

   When you select the PRA and PRB mode, both the PRA and PRB outputs become active and the output function of the I/O buffers associated with both pins are inhibited. However, the input buffer of the I/O buffers associated with both pins are still active. Any internal signals that appear on the PRA and PRB outputs are fed back through the input buffers to the internal Logic Modules. This could interfere with the expected function of the design while probing. Microsemi recommends that you use an input latch on PRA and PRB to prevent the feedback during probing.
6. *The SDI input and a separate I/O buffer share the use of a single device pin. The SDI input and the input function of the I/O buffer are connected in parallel. When the Mode pin is high, both inputs are active. The same is true for DCLK. External Probe Circuit control signals sent to those pins are also sent to the internal Logic Modules. This could interfere with the expected function of the design while probing. Microsemi recommends that you use an input latch on SDI and DCLK to prevent the external Probe Circuit control signals from effecting the functionality of your design during probing.*

   If either SDI or DCLK are configured so that the output function of the I/O buffer is active, the Program fuse must be programmed. In this configuration, the signals from your design are fed back to the Shift Register and will interfere with the function of the Probe Circuitry. In addition, the I/O drivers will conflict the external SDI and DCLK drivers. Damage to both drivers could occur.

In the normal operating mode (MODE = 0), all undefined device pins in a design are automatically configured as active Low outputs.

Two exceptions are the SDI and DCLK pins. If the Program fuse is not programmed and SDI and DCLK are undefined, they are configured as inactive inputs. In this case, SDI and DCLK pins should be tied to ground. If the Program fuse is programmed and SDI and DCLK are undefined, they will become active LOW outputs.

RH1280, A1280A, A1460A, and A14100A Security Fuse Configurations

The RH1280, A1280A, A1460A, and A14100A devices contain one Security fuse. Programming the Security fuse disables the Probe Circuitry, which disables the use of the Debugger, ActionProbe, and Silicon Explorer diagnostic tools. Table 1-3 summarizes the effect or programming the security fuse on the PRA, PRB, SDI, and DCLK pins.

Table 1-3 **• RH1280, A1280A, A1460A, and A14100A Security Fuse Configurations**

Mode[1]	Security	PRA, PRB	SDI, DCLK
Low	Don't care	User-defined I/O	User-defined I/O
High	No	Probe Circuit outputs[3]	Probe Circuit inputs[4]
High	Yes[2]	Probe Circuit disabled	Probe Circuit disabled

Notes:

1. *The MODE pin switches the device between the normal operating mode (MODE=0) and the Probe Circuit mode (MODE=1).*
2. *If the Security fuse is programmed, the Probe Circuit is permanently disabled which disables the Debugger and the ActionProbe diagnostic tools.*
3. *The PRA output and a separate I/O buffer share the use of a single device pin. The PRA output and the output function of the I/O buffer are multiplexed. The same is true for PRB. The Probe Mode that is loaded into the Mode Register will determine which output buffer is active during probing. There are three possible Probe Modes: PRA only, PRB only, and PRA and PRB.*

   When the PRA only mode is selected, the PRA output becomes active and the output function of the I/O buffer associated with the PRA pin is inhibited. However, the input buffer portion of the I/O buffer associated with the PRA pin is still active. Any internal signal that appears on the PRA output is fed back through that input buffer to the internal Logic Modules. This could interfere with the expected function of the design during probing. Microsemi recommends that you use an input latch on PRA and PRB to prevent the feedback while probing. PRB will function as a normal I/O in the PRA only mode.

   The PRB only mode is functionally equivalent to the PRA only mode. PRA also functions as a normal user I/O in the PRB only mode.

   When the PRA and PRB mode is selected, both the PRA and PRB outputs become active and the output function of the I/O buffers associated with both pins are inhibited. However, the input buffer of the I/O buffers associated with both pins are still active. Any internal signals that appear on the PRA and PRB outputs are fed back through the input buffers to the internal Logic Modules. This could interfere with the expected function of the design while probing. Microsemi recommends that you use an input latch on PRA and PRB to prevent the feedback during probing. An input latch is an integral part of the I/O buffers in the RH1280 and A1280A devices.
4. *The SDI input and a separate I/O buffer share the use of a single device pin. The SDI input and the input function of the I/O buffer are connected in parallel. When the Mode pin is high, both inputs are active. The same is true for DCLK. External Probe Circuit control signals sent to those pins are also sent to the internal Logic Modules. This could interfere with the expected function of the design while probing. Microsemi recommends that you use an input latch on SDI and DCLK to prevent the external Probe Circuit control signals from effecting the functionality of your design during probing. An input latch is an integral part of the I/O buffers in the RH1280 and A1280A devices.*

   The output function of the I/O buffers associated with SDI and DCLK do not interfere with the function of the Probe Circuitry while in the Probe Mode. When the Mode pin is driven high, these outputs are inhibited. The I/O drivers do not interfere with the external drivers. However, these outputs are not observable in the Probe Mode.

In the normal operating mode (MODE = 0), all undefined device pins in a design are automatically configured as active LOW outputs. You do not need to program the Security fuse to enable SDI and DCLK as active LOW outputs.

2 – Post-Programming Recommendations

This chapter discusses recommendations for post-programming verification and testing of RadHard and RadTolerant devices. Refer to the "Testing and Programming Microsemi FPGAs" section on page 59 for additional information about the testing, programming, and reliability of Microsemi devices.

Microsemi FPGAs undergo rigorous testing and quality checks to ensure that successfully programmed devices do not require further testing. The test methodology employed for un-programmed devices includes verification of functionality for all high and low voltage transistors, a transistor junction stress test, a fuse stress test, and a functional test for all logic modules in the array. In addition, static and dynamic burn-in tests are performed for all RadHard products to further ensure device reliability. During programming, logic modules in the array are protected from high voltage signals through isolation devices on each module input and output. The isolation devices are turned off during programming, thus protecting the low voltage transistors in the module.

Static and dynamic burn-in are included in the manufacturing flow for all RadHard and RadTolerant devices. The static burn-in is a biased burn-in for 144 hours at 125ºC. The dynamic burn-in is a 240-hour, 125ºC test. The dynamic burn-in stresses the device by shifting in test commands which toggle the routing tracks and all input and output tracks. Refer to the "Burn-In of Microsemi FPGAs" section on page 66 for a detailed description of both the static and dynamic burn-in procedures. RadHard device burn-in and test is performed by Lockheed-Martin Federal Systems (LMFS) using procedures developed by Microsemi SoC Products Group and implemented at the LMFS facility in Manassas, Virginia.

A quarterly reliability report, available upon request from your local sales representative, describes the accelerated life test results for Microsemi devices. This data is for post-programmed devices and is representative of results you would obtain, assuming you use programming and handling procedures described in this guide.

Burn-in is performed on all production units, including a periodic life-test burn-in for on-going reliability as required for QML certification.

Test

Microsemi programming algorithms are designed to verify the exact programming of the specific antifuses included in the programming files. The CheckSum is programmed into the device and must match the CheckSum in the programming file. This verifies that the device is programmed as designed.

In some cases, you may wish to verify certain timing or other test conditions on programmed devices. Microsemi recommends that you perform a functional test on a programmed device before placing the device in a system. However, Microsemi does not provide test services or programs to customers for programmed devices. You must develop a test methodology or contract a third party test facility to test programmed devices.

Packaging, Sockets, Trim, and Form

All RadHard and RadTolerant devices are burned-in, tested, packaged, and shipped in Non-Conductive Tie Bar packages. Through-hole sockets are used to test and burn-in the RadHard and RadTolerant devices. These sockets are also used on the programming Adapter Modules. After programming, the user must trim and form the leads of the RadHard or RadTolerant device to correspond to the required board layout.

The sockets are produced by Enplas Corporation, Wells Electronics Inc., and Yamaichi Electronics U.S.A. Inc. They can be purchased by contacting the local Enplas, Wells, or Yamaichi sales representative. Table 2-1 shows the part number for the RadHard and RadTolerant devices and the corresponding socket part number.

***Table 2-1* • Part and Socket Number for RadHard/RadTolerant Devices**

Microsemi Part Number	DSCC Part Number	Manufacturer Part Number
RH1280-CQ172V	5962F9215603QYC	Yamaichi IC51-1964-1952
RH1020-CQ84V	5962F9096505QTC	Wells 619-1000611-001
A1020B-CQ84B	5962-9096503MTC	Wells 619-1000611-001
A1280A-CQ172B	5962-9215601MYC	Yamaichi IC51-1964-1952
A1460A-CQ196B	5962-9550801MYC	Yamaichi IC51-1964-1952
A14100A-CQ256B	5962-9552101MYC	Enplas FPQ-256-(352)-.5-01
RT54SX32-CQ208B	5962-9958603QYC	Enplas FPQ-256(352)-0.5-01
RT54SX32-CQ256B	5962-9958603QXC	Enplas FPQ-256(352)-0.5-01
RT54SX32S-CQ208B	5962-0150801QYC	Enplas FPQ-256(352)-0.5-01
RT54SX32S-CQ256B	5962-0150801QXC	Enplas FPQ-256(352)-0.5-01
RT54SX32S-CQ208E	5962-0150803QYC	Enplas FPQ-256(352)-0.5-01
RT54SX32S-CQ256E	5962-0150803QXC	Enplas FPQ-256(352)-0.5-01
RT54SX72S-CQ208B	5962-0151501QYC	Enplas FPQ-256(352)-0.5-01
RT54SX72S-CQ256B	5962-0150801QXC	Enplas FPQ-256(352)-0.5-01
RT54SX72S-CQ208E	5962-0151503QYC	Enplas FPQ-256(352)-0.5-01
RT54SX72S-CQ256E	5962-0150803QXC	Enplas FPQ-256(352)-0.5-01

3 – Device Programming Failures

This chapter contains guidelines and information about programming failures and describes programming failures that may occur when programming a RadHard or RadTolerant device.

Programming Failure Guidelines

Programming failures are a normal and expected result of antifuse-based FPGA design. Microsemi performs extensive testing to measure the characteristics of the antifuses, and programs a sample of devices from every lot to ensure high programming results. However, Microsemi cannot guarantee that all devices will program successfully, and you should expect some programming failures.

The guaranteed quality and reliability of the devices that program successfully are unrelated to the programming yield. All devices that pass the programming function are fully guaranteed to meet all electrical, timing, and radiation specifications.

Before programming a RadHard or RadTolerant device, Microsemi recommends that you program a commercially equivalent device to ensure the proper setup of the hardware and software. After you have successfully programmed a commercial device, you can program your RadHard or RadTolerant device.

RadHard Failure Rates

RadHard devices typically exhibit a 5% programming fallout. If a RadHard device fails to program, but the commercially equivalent device passes, proceed with programming additional RadHard devices. If programming failures exceed the guidelines listed in the following table, stop programming and contact your local sales representative or Application Support at 1-800-262-1060, or email us at soc_tech@microsemi.com.

Table 3-1 **• RadHard Failure Rates**

Sample Size	Maximum Failures
5	2
10	3
20	4
50	6
100	10

RadTolerant Failure Rates

RadTolerant devices typically exhibit a 1-2% programming fallout. If a RadTolerant device fails to program, but the commercially equivalent device has passed, proceed with programming additional RadTolerant devices. If programming failures exceed the guidelines listed in the following table, contact your local sales representative or Application Support at 1-800-262-1060, or email us at soc_tech@microsemi.com.

Table 3-2 **• RadTolerant Failure Rates**

Sample Size	Maximum Failures
13	3
31	5
63	8
100	10

Types of Programming Failures

This section describes failure messages that may appear when a fuse fails to program in Activator or Silicon Sculptor.

Fuse Failed to Program

When an antifuse is programmed, multiple voltage pulses are applied to the VPP pin. While the pulses are applied, IPP current is checked. If the antifuse is open (unprogrammed), there will be no IPP current. The Activator/Silicon Sculptor can tell if the antifuse has been programmed once it detects IPP current. Pulses are applied until IPP current is detected, or the maximum number of pulses is exceeded. If the antifuse does not program after the maximum number of pulses are applied, a FAILED Programming Fuse message is displayed on the screen and the failed antifuse number is shown.

Bad Fuse

The BAD FUSE failures are often caused by a poor connection of the VPP pin to the socket pin. This is especially true if the part fails programming on the first antifuse. Remove the part and check for bent pins. If the pins are not bent and the other parts continue to fail with the FAILED Programming Fuse error, then the VPP pin of the socket could be damaged. Contact Application Support at www.microsemi.com/en/design-support/application-support.

Check 6 Failure

Once the antifuse has been programmed, the Activator/Silicon Sculptor addresses the same antifuse again and checks for IPP current at a lower VPP voltage. This is to make sure that the antifuse was correctly addressed the first time and that the IPP current did not come from another source. If no current is detected with this new test, the chip fails programming and APSW/APS2 issues the Integrity test 6 failure for the antifuse. Once again, the antifuse number that failed is displayed.

Check 7 and 8 Failures

This test is only performed on the RH1020 and A1020A devices. After the Activator/Silicon Sculptor completes the CHECK 6 test, it then does two additional tests to make sure that an additional antifuse was not programmed mistakenly. The first test checks antifuses on the same column as the programmed antifuse, and the second test checks the same row. The tests are done by addressing these other antifuses, applying a voltage to VPP and making sure no IPP current is detected. If the tests fail (IPP current is detected), a FUSE INTEGRITY FAILURE failure is displayed.

Error Messages

This section describes error messages Activator/Silicon Sculptor might display during programming, and the reason the message is displayed.

Integrity Test

Incorrectly programmed fuses, like the Check 7 and 8 failures above, are reported in the following format:

```
Integrity test <test type>. <test number>
```

This message indicates that the device is a programming reject. Microsemi will replace devices that fail programming.

Fuse Current Sense Test

This test is only performed on the RH1280, A1280B, A1460A, A14100A devices. This test is done to ensure that a logic module output is not inadvertently programmed to GND or VCC. The error message's format is:

```
Fuse <fuse number>, current sense test <test number>
```

Wrong Adapter Module

If the programming adapter does not correspond to the selected Microsemi device or package, the following message is displayed:

```
Wrong adapter module
```

Old Revision Adapter Module

If the Adapter Module does not support the selected device, even though the correct package is selected, the following message is displayed:

```
Old revision adapter module
```

Not Blank

If the device has been programmed and programming is attempted, the following message is displayed:

```
Not blank
```

Testing an Activator

The following are test procedures for Activator 2/2S and programming adapters. There are no adjustable parameters on the Activator. If any test fails, the Activator or the programming adapters should be considered non-functional. Return to Microsemi for replacement or credit. See the "Returning Failed Devices" section on page 34 for information on returning a part for replacement or credit.

Note: The Activator test procedure outlined below must be executed from the APSW programming software, version R2-1999 or later.

Equipment required:

- Activator 2/2S
- Programming Adapter (Optional)
- PC with SCSI controller running Windows 95/98/NT4.0
- Digital Volt Meter (DVM)
- Oscilloscope

To set up the Activator for testing:

1. Set the vertical scale of the oscilloscope to 2V/Div; horizontal time base to 2 ms/Div.
2. Connect the positive scope probe to the Digital Volt Meter (DVM)+V and negative scope probe (ground) to DVM ground, using BNC to Banana cables.
3. Set the DVM to DC volts on the 20-volt range.

4. Connect the ground wire to pin 14 or 16 of any Activator socket and to DVM ground and scope ground (Figure 3-1).

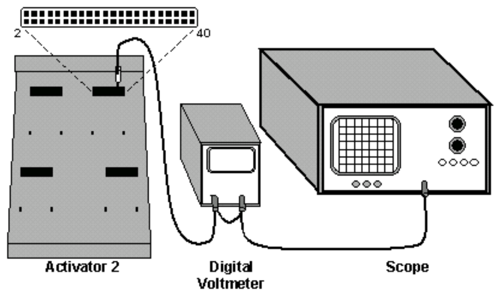
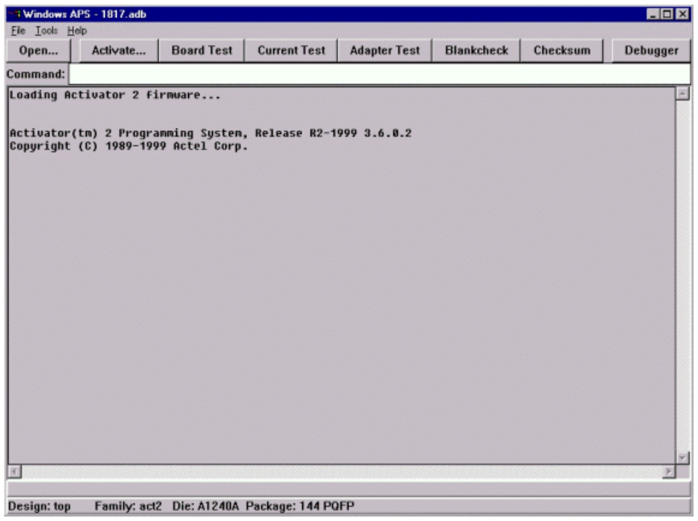


Figure 3-1 • **Activator Testing Setup Diagram**

5. To execute the test program, enter the following command from a DOS prompt:

```
apsw actst:1
```

6. Load an *.adb file to enable the test tabs. This displays the screen shown in Figure 3-2.



```
Loading Activator 2 Firmware...


Activator(tm) 2 Programming System, Release R2-1999 3.6.0.2
Copyright (C) 1989-1999 Actel Corp.
```

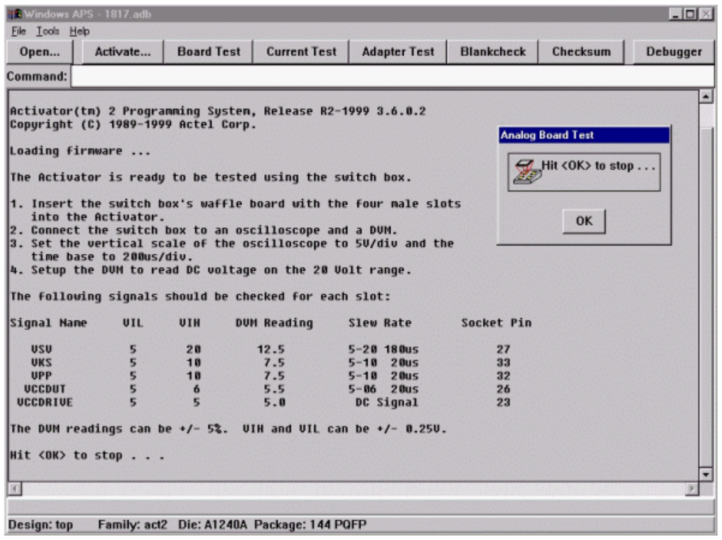
Design: top Family: act2 Die: A1240A Package: 144 PQFP

Figure 3-2 • **Loading an *.adb File**

7. To select a test, click one of the buttons: **Board Test**, **Current Test**, or **Adapter Test**.

Board Test

The Board Test checks power supply voltage level and slew rate, as shown in Figure 3-3.



```
Activator(tm) 2 Programming System, Release R2-1999 3.6.0.2
Copyright (C) 1989-1999 Actel Corp.

Loading firmware ...

The Activator is ready to be tested using the switch box.

1. Insert the switch box's waffle board with the four male slots
   into the Activator.
2. Connect the switch box to an oscilloscope and a DVM.
3. Set the vertical scale of the oscilloscope to 5V/div and the
   time base to 200us/div.
4. Setup the DVM to read DC voltage on the 20 Volt range.

The following signals should be checked for each slot:

Signal Name     VIL      VIH     DVM Reading     Slew Rate       Socket Pin

   VSV           5       20         12.5         5-20 180us          27
   VKS           5       10          7.5         5-10  20us          33
   VPP           5       10          7.5         5-10  20us          32
  VCCDUT         5        6          5.5         5-06  20us          26
 VCCDRIVE        5        5          5.0          DC Signal          23

The DVM readings can be +/- 5%.  VIH and VIL can be +/- 0.25V.

Hit <OK> to stop . . .
```

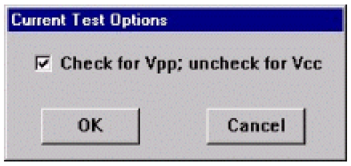
Design: top   Family: act2   Die: A1240A   Package: 144 PQFP

***Figure 3-3* • Board Test Screen**

When the test is complete, click **OK** to stop.

Current Test

This tests the Activator's ability to sense the current through a programmed antifuse. Activator displays the Current Test Options dialog box (Figure 3-4) when you select the CURRENT SENSE button.



[✓] Check for Vpp; uncheck for Vcc

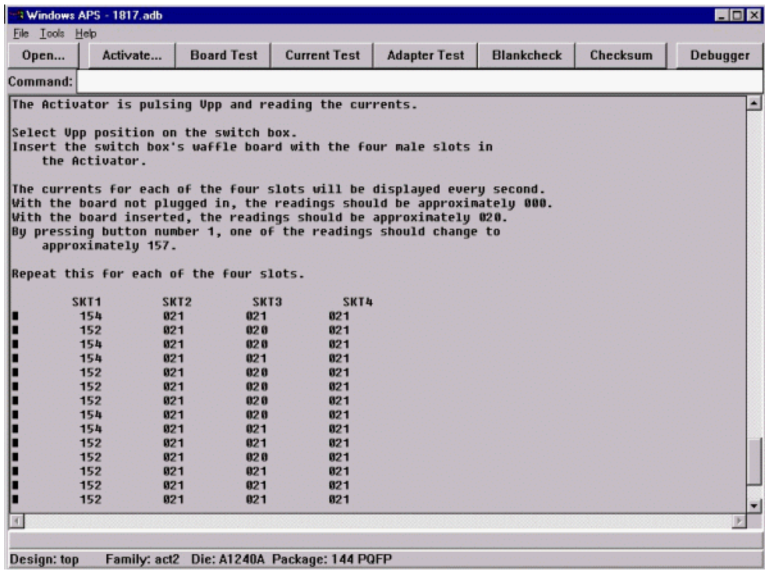
***Figure 3-4* • Current Test Options Dialog Box**

Check the box to test VPP or clear the box to test VCC. Click **OK** to continue.

VPP Testing

Insert a 637 ohm (680 ohm in parallel with a 10K ohm), 2% precision resistor between pin 14 (GND) and 32 (VPP). The currents for each of the four slots are displayed every second. With the resistors plugged in, the readings should be approximately 157. Repeat this for each of the four slots.

Figure 3-5 shows the testing result on slot 1.


Windows APS - 1817.adb
File Tools Help
Open... Activate... Board Test Current Test Adapter Test Blankcheck Checksum Debugger
Command:

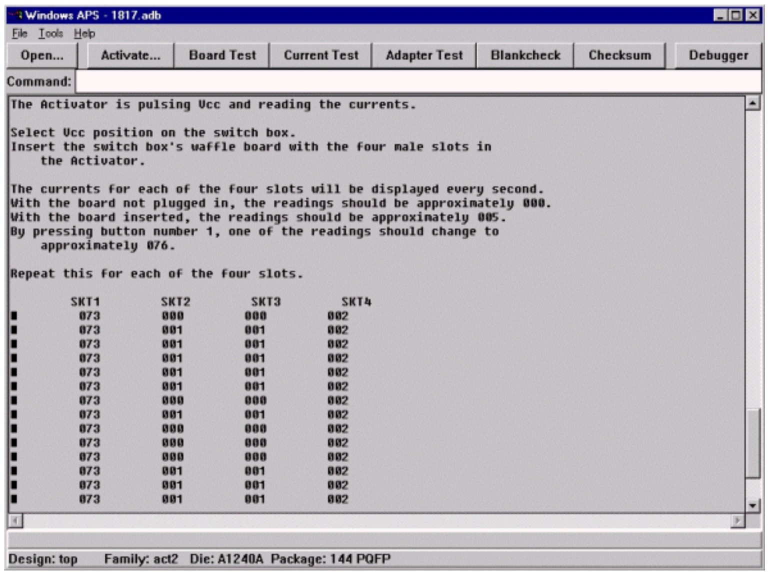
```
The Activator is pulsing Vpp and reading the currents.

Select Vpp position on the switch box.
Insert the switch box's waffle board with the four male slots in
    the Activator.

The currents for each of the four slots will be displayed every second.
With the board not plugged in, the readings should be approximately 000.
With the board inserted, the readings should be approximately 020.
By pressing button number 1, one of the readings should change to
    approximately 157.

Repeat this for each of the four slots.

        SKT1        SKT2        SKT3        SKT4
■        154        021         021         021
■        152        021         020         021
■        154        021         020         021
■        154        021         021         021
■        152        021         020         021
■        152        021         020         021
■        152        021         020         021
■        154        021         020         021
■        154        021         021         021
■        152        021         021         021
■        152        021         020         021
■        152        021         021         021
■        152        021         021         021
■        152        021         021         021

Design: top    Family: act2   Die: A1240A  Package: 144 PQFP
```

***Figure 3-5* • VPP Current Test for Slot 1 of Activator Device**

Note: The reading on the sockets that have resistor(s) should be approximately 157 ± 10.

VCC Testing

Insert a 637 ohm (680 ohm in parallel with a 10K ohm), 2% Precision resistor between pin 14 (GND) and 26 (VCC). The currents for each of the four slots are displayed every second. With the resistors plugged in, the readings should be approximately 076. Repeat this for each of the four slots. Figure 3-6 shows the testing result on slot 1.


Windows APS - 1817.adb
File Tools Help
Open... Activate... Board Test Current Test Adapter Test Blankcheck Checksum Debugger
Command:

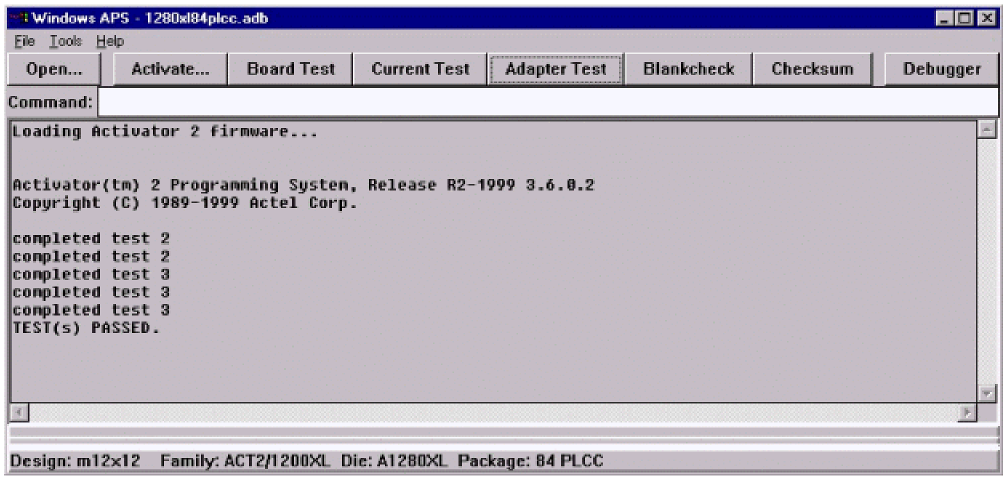
```
The Activator is pulsing Vcc and reading the currents.

Select Vcc position on the switch box.
Insert the switch box's waffle board with the four male slots in
    the Activator.

The currents for each of the four slots will be displayed every second.
With the board not plugged in, the readings should be approximately 000.
With the board inserted, the readings should be approximately 005.
By pressing button number 1, one of the readings should change to
    approximately 076.

Repeat this for each of the four slots.

        SKT1        SKT2        SKT3        SKT4
■        073        000         000         002
■        073        001         001         002
■        073        001         001         002
■        073        001         001         002
■        073        001         001         002
■        073        001         001         002
■        073        000         000         002
■        073        001         001         002
■        073        000         000         002
■        073        000         000         002
■        073        000         000         002
■        073        001         001         002
■        073        001         001         002
■        073        001         001         002

Design: top    Family: act2   Die: A1240A  Package: 144 PQFP
```

***Figure 3-6* • VCC Current Test for Slot 1 of Activator Device**

Note: The reading on the sockets that have resistor(s) should be approximately 076 ± 5.

Adapter Module Test

This test checks socket adapter modules. Select the Adapter Module button to execute the test. As the test is executed, the screen displays the test results, as shown in Figure 3-3.


Windows APS - 1280xl84plcc.adb
File Tools Help
Open... Activate... Board Test Current Test Adapter Test Blankcheck Checksum Debugger
Command:

```
Loading Activator 2 firmware...


Activator(tm) 2 Programming System, Release R2-1999 3.6.0.2
Copyright (C) 1989-1999 Actel Corp.

completed test 2
completed test 2
completed test 3
completed test 3
completed test 3
TEST(s) PASSED.
```

Design: m12x12 Family: ACT2/1200XL Die: A1280XL Package: 84 PLCC

***Table 3-3* • Adapter Module Test Screen**

The Adapter Module will fail if a device is inserted in the socket on the Adapter module. Check that the Adapter Module socket is empty before executing the Adapter Module Test.

Testing Silicon Sculptor

The DOS and Windows versions of Silicon Sculptor vary slightly. Please use the correct procedure to test your Silicon Sculptor.

Testing Silicon Sculptor for DOS

The Silicon Sculptor programmer can test its own hardware quite extensively. You are strongly advised to run a full system Self-Test on the programmer before performing any other operation. The self-test routine can detect problems in the pin-drivers, power supply, microprocessor, data cable, printer port, and several other circuits. The hardware test cannot detect problems resulting from a dirty socket. To execute the test:

1. Remove any chips from the programmer sites. Any device left in a programmer site may be damaged during testing.
2. Press Alt-D hot-key
3. Choose to test a single unit or all units
4. Watch the screen for any error messages

If you receive an error during the test, please contact Application Support at 1-800-262-1060, or email soc_tech@microsemi.com. for assistance.

Testing Silicon Sculptor for Windows

To start a self-test on your Silicon Sculptor programmer:

1. Invoke the SCULPTW software.
2. Click the **Device** button and select **Actel Diagnostics** from the list of available devices (type diagnostics). Once selected, the main window displays the Actel Diagnostics label in the Device field and a Test file tab appears.

3. Click the **Test** button to begin testing the programmer. The Test dialog box verifies the number of units you wish to test.
4. Click **OK** to begin the test. A window appears, warning you that there is no chip in the socket. After this, the test begins and runs until it is done or until you click the Stop button.

   To cancel the self-test, click on the **Stop** button at any time during the procedure. A window will appear to acknowledge the operation was aborted. To re-execute the test, simply click the **Test** button again.

   Your programmer should pass the test. Verify by checking the green PASS LED on the chassis of the programmer. If your programmer does not pass the test, the red FAIL LED activates and an error message presented on-screen:

   ```
   Error 47: Self-test failed. This unit may need service. Please call technical
   support.
   ```

   If this should happen, double-check the fidelity of the cable connections and try again. If you are still having trouble, note the exact error message and contact Application Support at 1-800-262-1060, or email soc_tech@microsemi.com.

Returning Failed Devices

Although Microsemi receives returns due to programming failures of less than 1% of parts shipped, failure rates of up to 5% – 10% may occur. In order to return failed devices, you must meet the following conditions:

1. Failure analysis (FA) is mandatory for all programming failures on RH/RT devices before Microsemi issues any replacements. A maximum of three units is necessary to perform a failure analysis. Also, request an RMA number through Microsemi's sales representatives, distributors, or customer service.
2. RMAs are authorized only for current Microsemi devices. Devices that have been discontinued will not be authorized for return.
3. All devices returned for FA and Returns should be in their original packaging. If parts are being returned for FA then you must also send the Programming File(s) and *.AVI file(s). Any parts returned to Microsemi for Failure Analysis without an RMA number and accompanying Design/.avi files will be returned immediately to the customer at the customer's expense.

Note: The *.AVI file is not mandatory for any RT devices except the RT1020.

FAILURE ANALYSIS REQUEST forms (may be collected from Customer Service) should only be used when you experience excessive programming loss has or where you have observed functional failures. The time frame of failure analysis varies greatly (it depends on the failure mechanisms). The final failure analysis report usually takes a month or longer to deliver. In addition, if during the FAILURE ANALYSIS PROCESS Microsemi is able to successfully program the units, these units will be returned to the customer against the replacement order and the units will be labeled that they are programmed.

4 – Debugging a Device with an Activator

This chapter describes how to use the Debugger tool in APSW or APS2 and an Activator to functionally debug your programmed RadHard or RadTolerant device. This includes descriptions of available debugging commands and command file usage and examples.

Note: This is not a supported feature in Silicon Sculptor. Debugging is not supported for SX, SX-A, and RTSX-S devices.

Functional Debugging With an Activator

Functional debugging of a RadHard or RadTolerant device with the Debugger tool in the APSW or APS2 software is done with a device placed in an Adapter Module and plugged into an Activator. Functional debugging is used to test a programmed RadHard or RadTolerant device by applying a stimulus to the input pins and observing the functional behavior at all internal nodes or nets and output pins.

Functional debugging is not the same as simulating. A programmed chip is required. The output results are determined from the silicon device, not from a model stored in memory. Functional debugging can be performed two ways. You can apply stimulus to the input pins by executing command-line commands or by creating command files and reading the files into APSW or APS2.

If you execute command-line commands, each command is executed before the next command can be entered. After each command is executed, output results are printed on the screen, or to a file if specified. Refer to the "Debugger Command-Line Commands" section on page 36 for a description of the command-line commands.

If you have a large number of commands to execute, you can create a command file. A command file contains a set of commands that are executed on your RadHard or RadTolerant device. A command file can also contain input test vectors. If you provide expected outputs with a test vector file, Debugger compares the chip output results with the expected outputs automatically. The Debugger saves any differences it finds for further analysis. Debugger then prints the output results on the screen, or to a file if specified in the command file. Refer to the "Using Command Files to Debug a Device" section on page 38 for information about using command files.

Running Debugger From APSW

The Debugger tool is run from within APSW. Use the following procedure to initialize Debugger and debug your device.

1. Invoke APSW.

   ### *UNIX Workstation*

   Type the following command at the prompt:

   ```
   apsw
   ```

   ### *PC*

   Double-click the Windows Programming icon in the Designer Series program group.

   The APSW window is displayed (see Figure 1-1 on page 11**).**

2. Open the design to be debugged. Click the **Open** button or from the File menu, choose the **Open** command. This displays the Open dialog box. Type in the design name or browse to the directory that contains the <design_name>.adb (or *.afm) file and select it. Click **OK.**
3. Verify that you have a programmed RadHard or RadTolerant device placed in an Adapter Module plugged into the Activator.
4. Debug your device. Click the **Debugger** button to initialize the tool. The message Debugger initialization complete is displayed when the Debugger is finished loading. Execute single command-line commands or load a command file using the loadfile command in the Command

box. Refer to the "Debugger Command-Line Commands" section on page 36 for information about command-line commands. Refer to the "Using Command Files to Debug a Device" section on page 38 for information about creating command files.

Debugger Command-Line Commands

This section lists the Debugger command-line commands and syntax, command functions, and APS2 Debugger menu command equivalents.

Command-line commands can be executed in APSW by typing the command in the Command box at the top of the APSW window and pressing Enter. Commands can be executed in APS2 Debugger by typing the command directly in the APS2 Debugger window and pressing Enter or by selecting the menu command equivalent. Table 4-1 lists the command-line syntax and function and the APS2 menu command equivalent.

Table 4-1 **• Command-Line Commands and Functions**

Command-Line Command Syntax and Function	APS2 Menu Command Equivalent
(assign <n> <vector_1> ... <vector_n>) Assigns the value <n> to each electrical node, or vector of nodes, the next time you execute the Step command. Each node must be a chip I/O or no assignment occurs. The format of <n> is as follows: 0=decimal; 0b=binary; 0h=hexadecimal; 0o=octal; 01x2=string. You can also assign a string-type constant to a vector. Valid characters are 1, 0, Z, z, X, and x. The character must be enclosed in double quotation marks. For example: assign “1Zx0” IN	**Stimulus/Assign**
(comp <n> <vector_1> ... <vector_n>) Compares <n> to each node or vector, and prints a message for each node in the list whose value is not equal to <n>.	**Compare/Comp**
(compfile <PATH\filename>) Opens the specified file to be used with the Fcomp command.	**File/CompFile**
(define (<name>) (<command>) ... (<command>)) Defines a macro. When invoked, commands specified in the macro are executed.	**Macro**
(emit “<example_text_string>”) Prints its argument to the screen and log file. The argument must be enclosed in double quotes.	No menu command
(fassign <vector_1> ... <vector_n>) Reads the value <n> from the input file defined by the Infile command. Assigns the value <n> to each electrical vector the next time you use the Step command. Each vector must be a chip I/O or no assignment occurs.	**Stimulus/Fassign**
(fcomp <vector_1> ... <vector_n>) Compares <n> to each node or vector, and prints a message for each node in the list whose value is not equal to <n>. The value checked is read from the next line in the file opened with the Compfile command.	**Compare/Fcomp**
(fprint <vector_1> ... <vector_n>) Prints the current values of all nodes in the Tablist to the file opened with the Outfile command.	**Fprint**

Table 4-1 • **Command-Line Commands and Functions (continued)**

Command-Line Command Syntax and Function	**APS2 Menu Command Equivalent**
(h <vector_1> ... <vector_n>) Assigns the value logical 1 to each electrical node or vector of nodes the next time you use the Step command. Each node must be a chip I/O or, no assignment occurs.	**Stimulus/High**
(icp <internal_node_1> <internal_node_2>) Used for In-Circuit-Probing with an ActionProbe. It brings <internal_node_1> out to the probe A pin. The second argument is optional. If given, <internal_node_2> is brought out to the probe B pin.	**ICP/Probe A**
(infile <PATH/infile_name>) Opens an input file used with the Fassign command.	**File/Infile**
(l <vector_1> ... <vector_n>) Assigns the value logical 0 to each electrical node or vector of nodes the next time you use the Step command. Each node must be a chip I/O or no assignment occurs.	**Stimulus/Low**
(loadfile <PATH/filename>) Loads the specified command file and executes all the commands in the file. The PATH consists of the full path for the command file. All commands in the command file must be enclosed in parentheses.	**File/LoadFile**
(outfile <PATH/filename>) Opens the specified file to be used with the Fprint command.	**File/OutFile**
(print <vector_1> ... <vector_n>) Prints the values of the specified vector(s). If no vector is specified, the current values of all nodes in the Tablist are printed.	**Print**
(step <n>) Debugs for <n> cycles. If <n> is not specified, a default of one cycle is used.	**Step**
(tabadd <vector_1> ... <vector_n>) Adds the named nodes or vectors to the list of nodes printed with the Print command. Names can be dropped by recreating the list using the Tablist command.	**Output/TabAdd**
(tablist <vector_1> ... <vector_n>) Initializes the Tablist to the specified nodes and vectors. If no arguments are given, prints the current contents of the Tablist.	**Output/TabList**
(vector <name> <node list>) Defines a vector <name> whose elements are the listed nodes.	**Stimulus/Vector**
(z <vector_1> ... <vector_n>) Sets all listed nodes to Z (high impedance) the next time you use the Step command. Each node must be a chip I/O or no assignment occurs.	**Stimulus/High-Z**

Table 4-1 • **Command-Line Commands and Functions (continued)**

Command-Line Command Syntax and Function	APS2 Menu Command Equivalent
(dbg-socket <n>) Chooses the socket on the Activator 2 to use during debug. The value of <n> can be 1, 2, 3, or 4. If only 1 Adapter Module is plugged in, you do not need to use this command.	Socket/dbg – socket
(repeat <n> <function>) Repeats a sequence for <n> cycles. The <function> can be a debugger command or a user-defined macro.	**Repeat**

Using Command Files to Debug a Device

This section provides example command files to illustrate the use of command files for debugging a device.

A command file contains a series of Debugger command-line commands that when loaded into Debugger are automatically executed. Use command files to run a large number of command-line commands on your device during debug.

To create a command file use a text editor or a word processor and save the file as ASCII text. Each command in the command file must be enclosed in parentheses. Use the loadfile command to load a command file into Debugger.

Two examples are shown below to illustrate how to create and use command files to debug a device. Refer to the "Debugger Command-Line Commands" section on page 36 for information about the available command-line commands.

Command File Example 1

The following example shows a command file followed by an explanation of the file, an input file, an output file, and a comparison file:

```
(vector P P0 P1 P2 P3 P4 P5 P6 P7)
(vector Q Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7)
(tabadd PE CEP CET UD P CLK Q TC)
(infile “/designs/example/example.pat”)
(outfile “/designs/example/example.out”)
(compfile “/designs/example/example.cmp”)
(define (clk10)(repeat 10(1 CLK)(step)(h CLK)(step)(fprint)
(fcomp Q))
(define (up) (1 CET CEP) (h PE UD)(step)(clk10))
(define (down) (h PE) (1 CEP CET UD)(step)(clk10))
(define (load) (1PE CLK) (fassign P)(step)(h CLK)(step))
(load)
(up)
(load)
(down)
```

The first *vector* command defines eight parallel load input bits as vector P. The second vector command defines counter outputs as vector Q.

The *tabadd* command causes the signals PE, CEP, CET, UD, P, CLK, Q, and TC to be displayed or printed when the print or fprint command is executed.

The *infile* command defines and opens a file containing input test vectors (see "Example Input File"). The *outfile* command defines and opens a file for receiving Debugger output results (see "Example Output File" section on page 40). The *compfile* command defines and opens a file containing the expected output values to be compared to the actual output values (see "Example Comparison File").

The *define* commands create the following user macros:

clk10. The clk10 macro provides 10 clock pulses to the CLK input, prints all of the nodes specified in the tabadd command to the outfile, and compares the status of the vector Q to the expected results.

up. The up macro specifies the counter to count up for 10 cycles.

down. The down macro specifies the counter to count down for 10 cycles.

load. The load macro reads a load vector P from the infile and loads the counter.

The **load, up, load, and down** commands execute the macros, as follows: *load* loads [00000000] into the counter, *up* cycles the counter up 10, *load*, loads [11111111] into the counter, and *down* cycles the counter down 10.

Example Input File

The following input file (infile) was used in this example.

```
0b00000000
0b10000000
0b01000000
0b11000000
```

Example Comparison File

The following comparison file (compfile) was used in this example.

```
00010001
10010001
01010001
01111111
10111111
00111111
11011111
01011111
10011111
00011111
11101111
01101111
10101111
```

Example Output File

The following output file (outfile) was created as a result of the command file used in this example.

***Table 4-2* • Output File Example**

Step	PE	CEP	CET	UD	P	CLK	Q	TC
000005:	1	0	0	1	00000000	1	10000001	0
000007:	1	0	0	1	00000000	1	01000001	0
000009:	1	0	0	1	00000000	1	11000001	0
000011:	1	0	0	1	00000000	1	00100001	0
000013:	1	0	0	1	00000000	1	10100001	0
000015:	1	0	0	1	00000000	1	01100001	0
000017:	1	0	0	1	00000000	1	11100001	0
000019:	1	0	0	1	00000000	1	00010001	0
000021:	1	0	0	1	00000000	1	10010001	0
000023:	1	0	0	1	00000000	1	01010001	0
000028:	1	0	0	0	11111111	1	01111111	0
000030:	1	0	0	0	11111111	1	10111111	0
000032:	1	0	0	0	11111111	1	00111111	0
000034:	1	0	0	0	11111111	1	11011111	0
000036:	1	0	0	0	11111111	1	01011111	0
000038:	1	0	0	0	11111111	1	10011111	0
000040:	1	0	0	0	11111111	1	00011111	0
000042:	1	0	0	0	11111111	1	11101111	0
000044:	1	0	0	0	11111111	1	01101111	1
000046:	1	0	0	0	11111111	1	10101111	1

Command File Example 2

The following example is a command file for an 8-bit counter circuit:

```
(define (clock) (emit “Clocking\n”) (h clock) (step) (l clock) (step))
(define (clear) (l clr) (step) (h clr) (step)
(emit "The Counter is Cleared\n"))
(vector outputs out7 out6 out5 out4 out3 out2 out1 out0)
(vector inputs in7 in6 in5 in4 in3 in2 in1 in0)
(emit "Enabling Counter B\n")
(h cen1)
(step)
(clear)
; Set input signalsRevision 1
(h in1 in3 in5 in7)
(l in0 in2 in4 in6)
(step)
(print inputs outputs)
(repeat 5 (clock) (print outputs))
```

The define commands create the clock and clear macros.

The vector command is used to define the inputs and outputs vectors.

The emit command writes text out to the screen.

The h and l commands are used to set particular signals to logic 1 (h) or logic 0 (l).

The print command prints the current state of inputs and outputs vectors.

The repeat command prints the output vector to the screen after every clock cycle.

Example Output File

The following output file (outfile) was created as a result of the command file used in this example.

```
Enabling Counter
The Counter is Cleared
inputs   = 10101010
outputs  = 00000000
Clocking
outputs  = 00000001
Clocking
outputs  = 00000010
Clocking
outputs  = 00000011
Clocking
outputs  = 00000010
Clocking
outputs  = 00000011
```

A – Troubleshooting

This appendix describes common problems you may encounter when programming with Activator or Silicon Sculptor.

Activator

This section describes some common problems you may encounter with the Activator, Adaptec 1505 SCSI card, or APSW or APS2 software and their solutions. If you are still unable to resolve your problems after reading this Appendix contact Application Support.

Driver Does Not Load under Windows

If the software driver does not load during boot-up, check for the following:

- The 1505 card is installed properly.
- The IRQ and I/O address settings match between the 1505 card and the software driver.
- The software driver is the correct one.
- There are no hardware conflicts.

Activator Hardware

This section describes problems you may encounter with an Activator and their solutions.

Green Power Light is Blinking

Problem: The green power light is blinking after the power is turned on.

Solution: A self test has failed. Contact Microsemi for a replacement Activator.

Activator Communication Link Down

Problem: The following error message appears:

```
ERROR: Activator communication link down. Exiting...
```

Solution: There is a problem with the connection between the Activator and the PC, or the Activator has lost power. Check the connections (the Activator power light should be illuminated), and try re-invoking APS.

Firmware Load Failed

Problem: The following error message appears:

```
Firmware load failed. WARNING: Could not connect with Activator.
```

Solution: No contact with the programmer is found. Check that you have turned on the Activator, that the SCSI cable is correctly installed, and the SCSI cable connection to the workstation is secure. The locking arms on the SCSI board can be misaligned easily.

Also, verify that the device driver(s) are installed correctly. Refer to the Hardware Installation chapter specific to your computer for information about configuring drivers.

Adapter Module

Problem: The following error message appears:

```
FAILED—Wrong adapter module
```

Solution: The incorrect Adapter Module is inserted in the Activator. Remove the Adapter Module and replace it with the correct one. The design may have also been configured with a different package or device type.

AFM File

AFM File Generation

Problem: Generating the *.afm file is taking a long time.

Solution: Depending on device type, device utilization, and machine speed, this process could take a few minutes. If 15 minutes have passed without completion, the hard disk may be out of memory. Exit APS and check available disk space. If there is less than 1 megabyte, free up some disk space and try generating the *.afm file again.

AFM Generation Failure

Problem: *.afm file generation failed

Solution: The software could not find a valid *.fus file. The file may not be present or it may have been created with a release prior to ALS 1.22. You must regenerate the *.fus file using Designer.

Programming

Integrity Failure

Problem: The following error message appears:

```
FAILED—fuse XXX integrity test 6, 7, or 8
```

Solution: This message often indicates that the device is bad. If you observe a programming failure rate in excess of 5%, contact Application Support at 1-800-262-1060, or email soc_tech@microsemi.com.

Programming Failure

Problem: The following error message appears:

```
FAILED—programming fuse XXXX
```

Solution: This message often indicates that the device is bad. If you observe a programming failure rate in excess of 5%, contact Application Support at 1-800-262-1060, or email soc_tech@microsemi.com.

Security Fuse Programmed

Problem: You want to verify if the Security Fuse has been programmed.

Solution: Execute the BlankCheck command.

SCSI Controller

Problem: SCSI Controller not found.

Solution: The SCSI controller board has not been installed in the PC, or there is an I/O address conflict. Also, verify that the device driver(s) are installed correctly. Refer to the Hardware Installation chapter specific to your computer for information about configuring drivers.

If you are working on a workstation, verify that no other APSW or APS2 processes are running. Only one APSW or APS2 process may run at a time.

Device Programmed

Problem: You want to determine if a part was programmed.

Solution: Execute the CheckSum or BlankCheck command.

Silicon Sculptor for DOS

The information in this chapter may help you solve or identify a problem with your programmer. If you have a problem that you cannot solve, please call us. We are dedicated to making Silicon Sculptor programmers as trouble-free as possible.

Testing the Hardware

The programmer can test its own hardware quite extensively. The self-test routine can detect problems in the pin-drivers, power supply, microprocessor, data cable, printer port, and several other circuits. The hardware test cannot detect problems resulting from a dirty socket (see below).

To execute the test:

1. Remove any chips from the programmer sites.
2. Press Alt-D hot-key
3. Choose to test a single unit or all units
4. Watch the screen for any error messages

If you receive an error during the test, please call Technical Support for assistance.

Power-on Self-Test (POST)

When power is applied to the programmer, it performs a power-on self-test (POST). This test checks RAM, ROM, CPU, analog circuits, and basic system integrity.

Note: Do not attempt any programming operations until the POST is complete.

If the POST fails, the red ERROR LED will be on. Failure codes are:

- 3 short flashes Cannot Self-Calibrate
- 2 short flashes ROM checksum error
- 1 short, 1 long flash RAM error

Error Messages

The following is a list of some of the most common error messages you may encounter when you work with Silicon Sculptor for DOS.

Error 3: Cannot reset hardware

The software cannot establish communications with the programmer. Here are some suggestions:

- Be sure the programmer has proper power and that the power LED is on.
- Make sure the cable from the programmer to the computer is properly connected to a parallel printer port. If you are using a ribbon cable, this is probably the problem (ribbon cable connectors are designed for use inside a chassis where the cable is not flexed). You should use a shielded 25 conductor cable (not an RS-232 cable).
- Your LPT port could be the culprit. If you have multiple parallel ports, you may have the ports configured incorrectly; that is, two at the same address. Some laptops have the ability to disable the port. If you have one, make sure the LPT port is enabled.
- Another program may be interfering with the port such as a print cache. When running under Windows, you increase the potential of another program trying to access the same parallel port and changing the expected status at the port.
- If you have a hardware lock key between the programmer and the port, then try removing it.

- Last but not least, the programmer may be damaged. Try another computer and/or parallel port and see if it works there. See the "Product Support" section on page 81.

Error 4: Excessive current detected. The protection circuit has shut off the power

The command was aborted to protect the programmer and hopefully not damage the chip. The device was taking too much current from the programmer. Possible causes include:

- The device may be inserted backwards and the continuity test has been turned off or did not successfully detect the device.
- The wrong algorithm could be selected and improper voltages were applied to the chip in the programmer site.
- If using a programming adapter, there could be a short.
- You may want to remove the chip and run the hardware test (Alt-D) to make sure all the pin drivers are functioning correctly. If the hardware passes the test, be sure you have the correct algorithm (device entry) selected for your device. If the error still occurs and you are sure the device is inserted correctly, then you should suspect a faulty device.

Error 5: Hardware time-out

This error message is generated when the software was waiting on a response from the programmer while executing a command and the programmer did not respond within the expected amount of time. This error may result from several causes. You may be experiencing communication errors (see Error 3: Cannot reset hardware above). There may be a bug in the software for this particular algorithm (see Error 10: Error in programming algorithm below. See also Power-on Self Test above.

Error 6: Wrong model number

See Error 3: Cannot reset hardware above for possible causes.

Error 8: LPTx: is not a functioning port

The parallel port LPTx (where x=1, 2, or 3) that is selected with the Configure command does not exist in your computer, is not functional, or has a bad cable connected to it.

Error 9: Programmer execution error

The programmer failed an internal consistency check. See Error 3: Cannot reset hardware and Error 5: Hardware time-out above for possible causes.

Error 10: Error in programming algorithm. Please call technical support

The software has detected an internal error. You should contact Microsemi to report the error. You may need to obtain a software update.

Error 11: There is no data in the buffer. You must load a file

A command tried to read data from the buffer to program or verify a chip, but nothing has been loaded into the buffer yet or the buffer was recently cleared.

Error 14: There is no chip in the programmer site

Be certain that your chip is inserted correctly. If the chip was inserted correctly, remove it and run the hardware self-test to be sure your programmer is functioning correctly (Alt-D). A defective chip may cause this error.

Error 15: The chip is not inserted in the programmer site correctly.

The continuity test determined that the chip in the programmer site does not have continuity on all the proper pins. You should examine these pins carefully. Possible causes are:

- A bent pin.
- The chip is not in the proper position in the programmer site.
- The chip has a different number of pins than the chip selected.
- The algorithm selected has a '*', indicating it requires an adapter, but you did not use the adapter, or vice-versa.
- The socket is dirty and not making a connection.

- The wrong socket module or adapter is being used for this device.

Error 16: The chip is inserted backwards

The chip has passed the continuity tests, but appears to have the GND and VCC pins improperly placed in the socket. If the PLCC, or QFP is not accidentally rotated, then the device is probably defective. Try a known good device.

Error 17: Out of base memory. You should have at least 200k free.

Your computer's configuration does not have enough RAM available to run the software. You should have 640K RAM installed with at least 200K available for program execution. Memory resident programs, such as network drivers, may reduce the RAM available to the programmer, so you may need to remove these programs from your CONFIG.SYS and AUTOEXEC.BAT files. If you are using DOS 5.00, you can specify that DOS be loaded into high memory, saving base memory for Silicon Sculptor software. See your DOS manual for details. The mem or chkdsk command will show you how much conventional memory is available.

Error 18: Temporary file error.

Our software's virtual memory manager is trying to store data that is currently not needed in RAM to the disk. The program was unable to create a temporary file or the disk is full. You should make sure you have plenty of disk space (the larger the data files, the larger the requirement for temporary disk space) and set the DOS environment variable TMP to point to the directory you wish to use for swap space. The program does take advantage of EMS memory if you have an expanded memory manager installed. This is much faster than using the disk for temporary swap space. If you want to specify that your hard disk (C:) can be used for temporary file storage, then execute the following command in your AUTOEXEC.BAT file:

```
SET TMP=C:\
```

Error 21: Cannot program

Not able to program the device in the programmer site. See Errors While Programming in this chapter.

Error 23: Invalid electronic signature in chip (device ID).

The chip may be damaged.

Error 24: Invalid electronic signature in chip (algorithm ID).

The chip may be damaged.

Error 25: Invalid electronic signature in chip (manufacturer ID).

Microsemi FPGAs have electronic identifiers that specify the manufacturer, the device code, and the proper programming parameters. The most common cause of this error is selecting one type of device in the Selection menu and inserting a different device in the socket.

Error 26: Device is not blank.

The Device/Blank command was executed or the Blank check before programming: option was enabled in the Device/Options dialog box and the device in the programmer site is determined to have programmed data. Possible causes are:

- The device was previously programmed and cannot be erased.
- The wrong algorithm was used.

Error 27: Device is not secured.

An attempt to secure a device was made, but it failed. See Errors While Programming in this chapter.

Error 31: Database file is invalid. The *.EXE file is corrupted.

The *.EXE file you are executing has been corrupted. You should get a new copy from Microsemi. See the "Product Support" section on page 81.

Error 32: Sorry, algorithm not found. Please call technical support.

The *.EXE file you are executing has been corrupted. You should get a new copy from Microsemi. See the "Product Support" section on page 81.

Error 33: You must reselect the chip you want to program.

The device was selected before establishing communications with the programmers, perhaps prior to turning on the programmer or before switching to a different programmer. Simply reselect the chip and you will be in business again.

Error 36: You must properly install the correct socket module.

On the Silicon Sculptor, the software interrogates the socket module before each operation to determine the correct mapping for the algorithm selected. You will get this error if:

- There is no socket module installed.
- The socket module installed does not support the device you have selected (e.g., you have selected a 100 pin device and you have a 208 pin PQFP socket module attached).
- The socket module installed is not supported by the version of the software you are using. Use the latest version.
- The pinout has not yet been defined for this package type. It may be an oversight on our part. If so, please call technical support and inform us of this problem.

Error 39: Device already secured.

The device cannot be legitimately programmed because it has been secured.

Error 41: Error reading file.

The Buffer/Load command was executed inside a macro file and the buffer could not be loaded. This error message is not displayed on the screen, but is returned to DOS when the software is being run via a batch file.

Error 43: Error in macro file.

A macro file was being played back and an error was detected in the syntax of the file. Possible causes are:

- The macro file is corrupted.
- The macro file was recorded with an earlier version (<V2.00) of the software.
- The macro file was generated by a user's application or text editor and does not conform to the proper macro file format.

Error 44: Internal error. Please call technical support.

The software detected an internal inconsistency. This may be caused by the computer not performing correctly.

Error 45: Hardware requires calibration. Please call technical support.

The self-test (Alt-D) has detected that the hardware is improperly calibrated. The unit must be returned for repair. See the "Product Support" section on page 81.

Error 46: AFS software required to execute this function.

This is a function that is available to users that have purchased the Advanced Feature Software only. In order to use the chosen function you must buy the AFS upgrade. See the "Product Support" section on page 81.

Error 47: Self test failed. This unit may need service. Please call technical support.

The self-test (Alt-D) has detected a hardware problem. The unit may need to be returned for repair. Note the exact error message and see the "Product Support" section on page 81.

Error 50: Device sum does not match sum specified in AFS/Options.

The sum calculated on the device does not match the sum entered in the AFS/Options Checksum Verify command. Check this option to see if a mistake was made when entering the sum value. Also check the buffer checksum to see if it matches the value entered for Checksum Verify or if any data in the buffer has changed.

Error 52: DynCall Stack Underflow.

The internal dynamic linker underflowed its reference table. If this error reoccurs, then call the Technical Support Line.

Error 53: DynCall Stack Overflow.

The internal dynamic linker overflowed its reference table. If this error reoccurs, then call the Technical Support Line.

Error 60: The demo period for this programmer has expired.

This programmer is a demo from Microsemi and the demo period has expired. Call the Microsemi SoC Products Group Sales Department for an upgrade code to extend the Demo period.

Error 61: Concurrent programmer did not initialize properly.

The Silicon Sculptor Concurrent programmer did not initialize correctly. Cycle the power on the programmer and try your operation again. If you continue to get this error message, send the programmer in for repairs.

Error 65: Concurrent Unit has the wrong socket module.

The specified programming site does not have the same socket module as the master site. The site must contain the same socket module as the master programmer in order to program devices on that site. The site has been temporarily disabled. Starting a new device operation with the correct socket module on the site will re-enable the site.

Error 66: Concurrent unit has the wrong technology adapter.

The specified programming site has the wrong technology adapter (TA). Cycle the power on the programmer. If the error persists, call Technical Support.

Error 67: Concurrent unit has the wrong BIOS.

The specified programming site has the wrong BIOS. Cycle the power on the programmer. If the error persists, call Technical Support.

Error 68: Concurrent unit has the wrong number of pin drivers.

The specified programming site has the wrong number of pin drivers. Cycle the power on the programmer. If the error persists, call Technical Support.

Error 69: Concurrent unit is not available.

The specified programming site is not responding to commands. Verify that the programmer number is correct. Cycle the power on the programmer and try again. If the error persists, call Technical Support.

Error 70: The buffer data cannot be used to program this device.

You loaded a file type that is not a valid option for the currently selected device. Re-select the device and load the buffer again. If the error persists, call Technical Support.

Warning Messages

Warning: Device is not blank

You will get this warning when using the Device/Program command with the Blank check before programming operation enabled in the Device/Options dialog box. You are given the option to Abort, Retry, or Ignore.

Warning: Device has been secured

You will get this message only on devices that have the ability to read the security bit prior to performing any other operation. You are given the option to Abort, Retry, or Ignore.

Self-diagnostics Test

Remove any chips from the programmer sites before running the diagnostics test. Select **Actel Diagnostics** from the Select menu and press <enter> on Device/Test, or use the hot-key combination

ALT-D, to invoke the self-diagnostics test. If you already know which site is failing you can run the diagnostics on just that site. If the entire system is not working, select All Sites. The software will indicate which site is bad.

Silicon Sculptor for Windows

You can get technical support from Microsemi SoC Products Group any time that you experience a problem that you cannot solve. We kindly request that you have the following information ready when you contact us:

- The model number of the programmer (title bar of secondary screen).
- Your software version number (from the top of the main screen).
- The exact error message and error number you received.
- The exact algorithm that was selected.
- The exact part number on the chip you were trying to program.
- The command you executed.
- The results of running the self-test command on your programmer (BP Microsystems Diagnostics).

It is also useful to have a print-screen of the error. You may be asked to upload your file and/or send in your devices so we can analyze the error at the factory.

If you need to return your programmer to Microsemi for any reason, you must call and get a Return Material Authorization (RMA) number before shipping; mark the RMA number clearly on the shipping container. Be sure to include a description of the problem experienced, a return address, contact person and a phone number.

Testing the Hardware

The programmer can test its hardware quite extensively. The self-test routine can detect problems in the pin-drivers, power supply, microprocessor, data cable, printer port, and several other circuits.

The hardware test cannot detect problems resulting from a dirty socket.

Remove any chips from the programmer site(s) before you proceed.

The ActelWin software enables you to test all applicable parts within the device programmer for accuracy. This test helps to ensure that the programmer is running at performance standards.

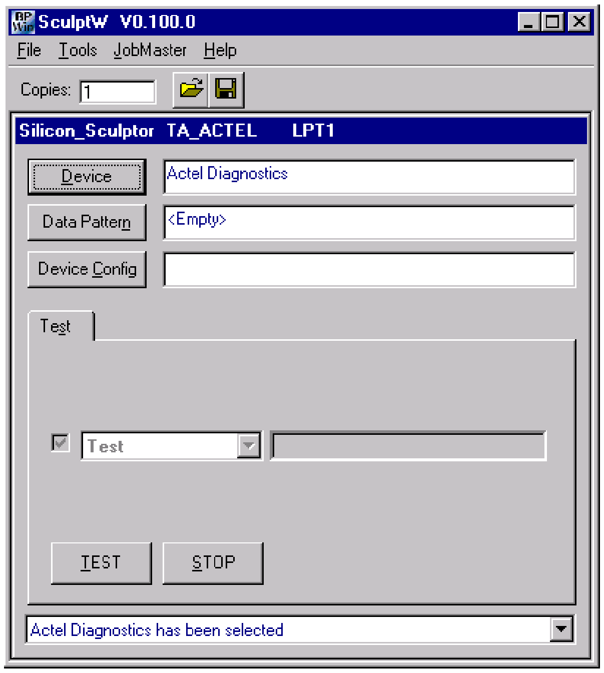
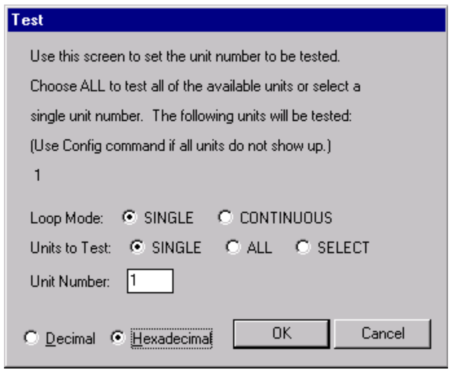


Figure A-1 • **SculptW Dialog Box with Test Information**

To begin running the self-test on your device programmer, click the Device button, and select **Actel Diagnostics** from the list of available devices (type diagnostics). Once selected, the main window displays the Actel Diagnostics label in the Device field and a Test file tab appears (Figure A-1). To begin testing the programmer, click the Test button. The Text dialog box opens, as shown in Figure A-2.


Test

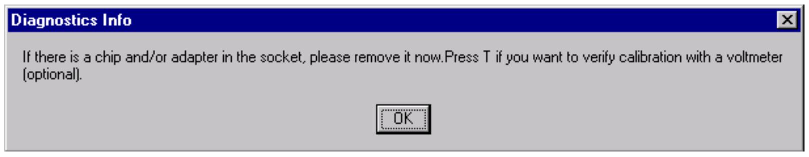
Use this screen to set the unit number to be tested.
Choose ALL to test all of the available units or select a single unit number. The following units will be tested:
(Use Config command if all units do not show up.)
1

Loop Mode: [✓ SINGLE] CONTINUOUS
Units to Test: [✓ SINGLE] ALL SELECT
Unit Number: 1

Decimal [✓ Hexadecimal]

Figure A-2 • **Test Dialog Box**

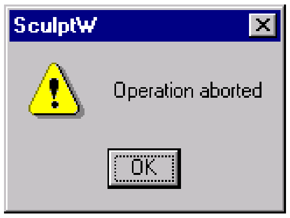
The Test dialog box verifies how many units you wish to test. After you click OK to begin the test, a window appears to prompt you that there is no chip in the socket (Figure A-3).



If there is a chip and/or adapter in the socket, please remove it now.Press T if you want to verify calibration with a voltmeter (optional).

Figure A-3 • **Diagnostics Info for Self Test Window**

The test begins and runs until it is done or until you click on the Stop button. If you decide to cancel the self-test, click on the Stop button at any time during the procedure. A window appears to acknowledge the operation was aborted (Figure A-4). To re-execute the test, simply click the Test button again.



Operation aborted

Figure A-4 • **Self-test Abort Window**

Your programmer should pass the test. Verify by checking the green PASS LED on the chassis of the programmer. If your programmer does not pass the test, the red FAIL LED activates and an error message presented on-screen:

Error 47: Self-test failed. This unit may need service. Please call technical support.

If this should happen, double-check the fidelity of the cable connections and try again. Note the exact error message and call us for technical assistance if you are still having trouble (refer to the "Product Support" section on page 81).

Software Updates

The control software for your programmer is updated on a frequent basis (typically every two months) to add features and provide you with support for new chips. Software updates can be obtained from Microsemi SoC Products Group.

Software upgrades may be obtained through our Internet address (www.microsemi.com/soc), but depending on the type of programmer you have (engineer, production or automated), upgrades and renewals may need to be made by contacting our sales department.

If you decline the software/hardware upgrade and your software support runs out, you will receive the following message when you select a part:

```
Error Code 57: Device Not Enabled
```

Your programmer is designed to be highly flexible and programmable, allowing it to program a wide variety of chips. Consequently, when a problem does arise, it can usually be fixed with a software update.

We recommend that you obtain the latest software revision before calling our support line with a software problem. The solution for many of our technical support calls is the latest version of the software.

FAQs

The questions below represent a list of the most commonly asked questions from callers. It does not represent a complete list. Please visit the Microsemi SoC Products Group website for more information.

Why does the ActelWin NT 4.0 software keep failing?

The device drivers are not installed or are not being recognized by the software. The software that runs on an NT environment is attempting to establish communication with the device programmer through the drivers. When the drivers are not found, it fails the software. To remedy this, you need to install, or re-install, the device drivers for the software. Please refer to the steps listed below the following question to obtain the device drivers and verify that the software is recognizing them.

Why isn't ActelWin running in the NT environment?

The Windows NT 4.0 system requires the installation of a systems device driver. The first step to take is to find out whether a device driver is installed or not.

1. Open the control panel. Select Control Panel from the Settings menu, under the Start menu.
2. Open Devices. Double-click the **Devices** icon.
3. Select the device BPNTDriver. Verify that the device driver status is set to started and startup is set to automatic. If these selections are not designated for the device driver, click to select the driver. To set the status to start, click the **Start** button. NT attempts to start the device driver when you click the Start button. To set the driver startup to automatic, click the **startup** button and select automatic.

If for some reason the device driver does not install or you need to uninstall the driver, go to the command prompt. Go to the NTDrvr sub directory of the default installation directory. For example, the default installation directory for Silicon Sculptor is C:\siliconwin.

- To uninstall from this directory type: drvinstl -u
- To install from this directory type: drvinstl -i

There are registry settings for the device driver and as soon as there is a full definition they will be added here.

In the DOS version, the configuration settings were saved to a file. Where did they go?

With the introduction of the Actel software, the configuration settings are no longer saved to a file. These settings are now being saved in the Windows 95/98/NT 4.0 Registry and are not deleted when they are uninstalled. This allows you to continue to use previous settings when upgrading to a new version of the software. There may be cases however, when you are required to delete these settings. To do so, execute the following steps:

1. Run **regedit**. Select **Run** from the Start menu in the Windows taskbar.
2. Select **ActelWin**. Select ActelWin from the VB and VBA Program Settings file list.
3. Delete the **ActelWin** registry settings. The ActelWin registry settings have now been deleted. The new registry settings appear in this directory upon re-installation or upgrade of the software.

Errors While Programming

If you experience problems while trying to program a chip, try to narrow down the problem. If you receive a Cannot program or Cannot erase error messages while programming:

- Make sure you have selected the proper programming algorithm.
- The device may have been previously secured; use Blank Check to verify whether or not the device is blank.
- The device may have a newer die than the one supported by the programming algorithm

Cleaning a Dirty DIP Socket

If the DIP socket becomes dirty, it sometimes fails to make contact with all the chip pins. The simple fix is to place your chip in the socket, push the lever down, and slide the chip left and right a few times. Microsemi also recommends cleaning sockets with a blast of high-pressure air on a regular basis.

If this does not resolve the problem, run the hardware self-test described above. If your hardware passes the self-test, there may be an error in the programming algorithm you are using or Microsemi may have updated the programming algorithm for your device. In either event, your can probably correct your problem with a software update.

Error Messages and Suggested Actions

Error messages are usually generated by the device programmer. When you receive an error message, refer to the appropriate device programmer manual's Troubleshooting section. The following is a list of error messages that could originate from the software.

Version Mismatch – BPEng.dll and BPAlg.db versions mismatched.

The engine and Algorithm database versions do not match. Accepting this error message prompts an exit from the software. Please contact technical support for further assistance. Refer to Appendix A for Technical Support contact information.

BPPgmr.ocx – BPPgmr.ocx internal error

ActWin generates this error when the unique coding sequence attached to both the BPPgmr.ocx and the BPEng.dll do not match. Accepting this error message prompts an exit from the software. Please contact technical support for further assistance.

Algo Database Error – Error loading BPAlg.db

This is a catch all error code. ActWin generates this error if anything not listed above causes an error while loading the Algorithm Database. Accepting this error message prompts an exit from the software. Please contact technical support for further assistance. Refer to Appendix A for Technical Support contact information.

BPNtIoDll – No parallel port is available.

ActelWin NT 4.0 generates an error if the device drivers are not installed. The software attempts to establish communication with the programmer through the driver. When the driver is not found, it fails the software. In order to remedy this, you must install the device drivers for the software. Please refer to the "FAQs" section on page 53 for procedures.

Error 3: Cannot reset hardware.

The software cannot establish communication with the programmer. Here are some suggested actions:

- Be sure the programmer has proper power and that the green PASS LED is on. Since the programmer performs an automatic Power On Self Test (POST) upon startup, it could be that the programmer failed the test and has signaled the software into the default DEMO mode. If this should happen contact Technical Support.
- Make sure the cable from the programmer to the computer is properly connected to a parallel printer port. If you are using a ribbon cable, this is probably the problem (ribbon cable connectors are designed for use inside a chassis where the cable is not flexed). You should use a shielded 25-conductor cable (not an RS-232 cable).
- Check your LPT port. If you have multiple parallel ports, you may have the ports configured incorrectly; that is, two at the same address. Some laptops have the ability to disable the port. If you have one, make sure the LPT port is enabled.
- Another program (such as a print cache) may be interfering with the port. When running under Windows, you increase the likelihood of another program trying to access the same parallel port and changing the expected status at the port.
- If you have a hardware lock key between the programmer and the port, try removing it.
- Last but not least, the programmer may be damaged. Try another computer and/or parallel port and see if it works there.

Error 5: Hardware time-out.

The software issues this error message when it is waiting on a response from the programmer while executing a command and the programmer does not respond within the expected amount of time. This error may result from several causes. You may be experiencing communication errors (see Error 3: Cannot reset hardware above). There may be a bug in the software for this particular algorithm (see Error 10: Error in programming algorithm below). Refer to the "Testing the Hardware" section on page 50 for more information.

Error 6: Wrong model number.

See Error 3: Cannot reset hardware above for possible causes.

Error 8: LPTx: is not a functioning port.

The parallel port LPTx (where x=1, 2, or 3) that is selected with the Configure command does not exist in your computer, is not functional, or has a bad cable connected to it.

Error 9: Programmer execution error.

The programmer failed an internal consistency check. See Error 3: Cannot reset hardware, and Error 5: Hardware time-out, for possible causes.

Error 10: Error in programming algorithm. Please call technical support.

The software has detected an internal error. You should contact Microsemi to report the error. You may need to upgrade your system with the most recent release of the software. Refer to Appendix A for Technical Support contact information.

Error 11: There is no data in the buffer. You must load a file or read a chip.

A command tried to read data from the buffer to program or verify a chip, but nothing has been loaded into the buffer yet or the buffer was recently cleared.

Error 14: There is no chip in the programmer site.

Be certain that you inserted your chip correctly. If the chip was inserted correctly, remove it and run the hardware self-test to be sure your programmer is functioning correctly (Actel Diagnostics). A defective chip may cause this error. When using an autohandler, the contactor may not have closed or the connection between the programmer and the contactor may be disconnected.

Error 15: The chip is not inserted in the programmer site correctly.

The continuity test determined that the chip in the programmer site does not have continuity on all the proper pins. You should examine these pins carefully. Possible causes are:

- A bent pin.
- The chip is not in the proper position in the programmer site.
- The chip has a different number of pins than the chip selected.
- The algorithm selected has a “*”, indicating it requires an adapter, but you did not use the adapter, or vice-versa.
- The socket is dirty and not making a connection. Refer to the "Cleaning a Dirty DIP Socket" section on page 54 for more information.
- The wrong socket module or adapter is being used for this part.
- The device may be a very low power device that is not properly detected by our continuity methodology. If so, please let us know.

Note: It's not easy to get continuity on an LCC device in a PLCC socket. If you are trying to do that, then you may need to add a spacer between the chip and the lid in order to apply the proper force to the device pins. The best solution is to purchase an LCC socket module that does not require any such modification. Also, LCC devices do not work at all in the autoeject sockets designed strictly for PLCC devices.

Error 16: The chip is inserted backwards.

The chip has passed the continuity tests, but appears to have the GND and VCC pins improperly placed in the socket. If LCC, PLCC, or QFP is not accidentally rotated, then the device is probably defective. Try a known good device.

Error 17: Out of base memory. You should have at least 200K free.

Your computer's configuration does not have enough RAM available to run the software. You should have 640K RAM installed with at least 200K available for program execution. Memory resident programs, such as network drivers, may reduce the RAM available to the programmer, so you may need to remove these programs from your CONFIG.SYS and AUTOEXEC.BAT files. If you are using DOS 5.00, you can specify that DOS be loaded into high memory, saving base memory for Silicon Sculptor software. See your Microsoft Windows manual for details.

Error 18: Temporary file error.

Our software's virtual memory manager is trying to store data that is currently not needed in RAM to the disk. The program was unable to create a temporary file or the disk is full. You should make sure you have plenty of disk space (the larger the data files, the larger the requirement for temporary disk space). The program does take advantage of EMS memory if you have an expanded memory manager installed. This is much faster than using the disk for temporary swap space.

Error 21: Cannot program.

Not able to program the device in the programmer site. Refer to the "Errors While Programming" section on page 53 for more information.

Error 23: Invalid electronic signature in chip (device ID).

The chip may be damaged or the programming algorithm may have changed.

Error 24: Invalid electronic signature in chip (algorithm ID).

The chip may be damaged or the programming algorithm may have changed.

Error 26: Device is not blank.

The Device/Blank command was executed or the Blank check before programming option was enabled in the Device/Options dialog box and the device in the programmer site is determined to have programmed data. You may have used the wrong algorithm or the device has been programmed previously.

Error 27: Device is not secured.

An attempt to secure a device was made, but it failed. Refer to the "Errors While Programming" section on page 53 for more information.

Error 31: Database file is invalid. The *.EXE file is corrupted.

The *.EXE file you are executing has been corrupted. You should uninstall and then reinstall the software to fix the executable. Refer to the "Product Support" section on page 81.

Error 32: Sorry, algorithm not found. Please call technical support.

The *.EXE file you are executing has been corrupted. You should get a new copy from Microsemi SoC Products Group. Refer to the "Product Support" section on page 81.

Error 33: You must reselect the chip you want to program.

The device was selected before establishing communications with the programmers, perhaps prior to turning on the programmer or before switching to a different programmer. Reselect the chip and the error should not re-occur.

Error 36: You must properly install the correct socket module.

You get this error message if:

- There is no socket module installed.
- The socket module installed does not support the device you have selected.

- The socket module installed is not supported by the version of the software you are using. Use the latest version.
- The pinout has not yet been defined for this package type. It may be an oversight on our part. If so, check the different potential situation and in case of persistent error, please call technical support and inform us of this problem.

Error 39: Device already secured.

The device cannot be legitimately programmed, read, etc., because it is secured. If it is a PLD it may still be functionally tested with the Test command under the Test file tab.

Error 44: Internal error. Please call technical support.

The software detected an internal inconsistency. This may be caused by the computer not performing correctly. Refer to the "Product Support" section on page 81.

Error 46: AFS software required to execute this function.

This is a function that is available to users that have purchased the Advanced Feature Software only. In order to use the chosen function you must buy the AFS upgrade.

Error 47: Self-test failed. This unit may need service. Please call technical support.

The self-test (Actel Diagnostics) has detected a hardware problem. The unit may need to be returned for repair. Note the exact error message and contact Technical Support. Refer to the "Product Support" section on page 81.

Error 57: You must purchase support for this device to use it.

The device that you selected is not supported in the default device set for this programmer. Call the Microsemi Soc Products Group Sales line to purchase an upgrade code for your programmer (refer to the "Product Support" section on page 81 for contact information).

Error 70: The buffer data cannot be used to program this device.

You loaded a file type that is not a valid option for the currently selected device. Re-select the device and load the buffer again. If the error persists, call Microsemi SoC Products Group Technical Support. Refer to the "Product Support" section on page 81 Technical Support contact information.

B – Testing and Programming Microsemi FPGAs

This appendix explains the testing and burn-in that Microsemi devices undergo before shipment to customers, as well as a study of the programming process. This systematic and thorough testing, burn-in, and control of the programming process ensures that customers no not need to perform additional testing or burn-in of Microsemi devices.

Testing has long been a struggle for users of masked gate arrays. To avoid board-level, system-level, or even possible field failures, the system designer must expend great effort in developing test vectors for gate array designs. Even after the vectors are developed, fault coverage for typical designs may be only about 70 percent, with about 95 percent coverage being the best possible. With a 70 percent fault coverage, typical masked gate array designs are likely to have 2 to 5 percent defective devices.[1]

In general, field programmable logic devices have allowed users to avoid the need to develop test vectors. These devices allow tests to be performed by the semiconductor vendor prior to programming. However, most one-time programmable logic devices have not yet achieved the functional quality levels of other semiconductor devices, because they don't allow the chip manufacturer to access and test all internal gates. Early one-time programmable devices had poor test coverage, and users were often disappointed to see functional failure rates of more than 10 percent on parts that had passed programming. Over time, on-chip test circuits and testing techniques have greatly improved, and now one-time programmable devices have functional defect rates in the range of 0.1 to 1 percent[2]. Although this failure rate is low for individual chips, putting 10 such chips on a single board can still mean a board failure rate of 5 to 10 percent.

Testability of Microsemi FPGAs

Although Microsemi's FPGA families use a one-time programmable technology, the device's unique architecture permits a degree of testability comparable to reprogrammable devices. Special test modes allow functional testing of unprogrammed devices at essentially 100 percent fault coverage. This testability is independent of the large number of equivalent gates. To show how this is accomplished, we will first review the architecture of the Microsemi FPGAs and describe how they are programmed.

RadHard and RadTolerant FPGA Products

RadHard products include the RH1020 and RH1280. RadTolerant products include the A1020B, A1280A, A1460A, and A14100A devices. The RH1020 an A1020B are 2,000 gate devices based on the ACT 1 architecture. the RH1280 and A1280A are 8,000 gate devices based on the 1200XL architecture. The A1460A, and A14100A are 6,000 and 10,000 gate devices based on the ACT 3 architecture.

Architecture

The basic building block of all Microsemi FPGAs is the logic module. Each logic module is programmable and capable of implementing all two-input logic functions, most three-input functions, and many other functions up to eight inputs. With an architecture similar to a channeled gate array, logic modules are organized in rows and columns across the chip (Figure B-1 on page 60). Adjacent to each row of logic modules are routing channels. Horizontal routing channels are shown in the figure, but vertical channels also run through the logic modules. These are used to configure a logic module and connect inputs and outputs of logic modules to implement a design. Surrounding the array of logic modules and routing channels are I/O buffers and test circuits.

Within the routing channels are programmable antifuse (PLICE) elements. The antifuse is normally open and is programmed to form an electrical connection between routing elements. An antifuse that connects

1. *Henshaw, "User Requirements for Fault Coverage." Wescom Proceedings, 1990, p. 179.*
2. *AMD PAL Device Data Book, 1988, pps. 3-106.*

a horizontal routing track to a vertical track is called a cross-antifuse. An example of a logic module interconnection (or a net) is shown in Figure B-2. Here the output from Module 3 is connected to a horizontal routing track by programming a cross-antifuse. Another cross-antifuse is programmed to connect an input to Module 4. In a similar manner, the output of Module 3 is connected to the input of Module 2. Notice that not all horizontal tracks are continuous across the chip. Often, tracks are broken into a series of smaller tracks called segments. Segments are useful because it is often desirable to connect logic modules that are close to each other, and a full horizontal track would waste routing resources and slow down circuit performance. Sometimes, however, it is necessary to connect two segments to form a longer segment.

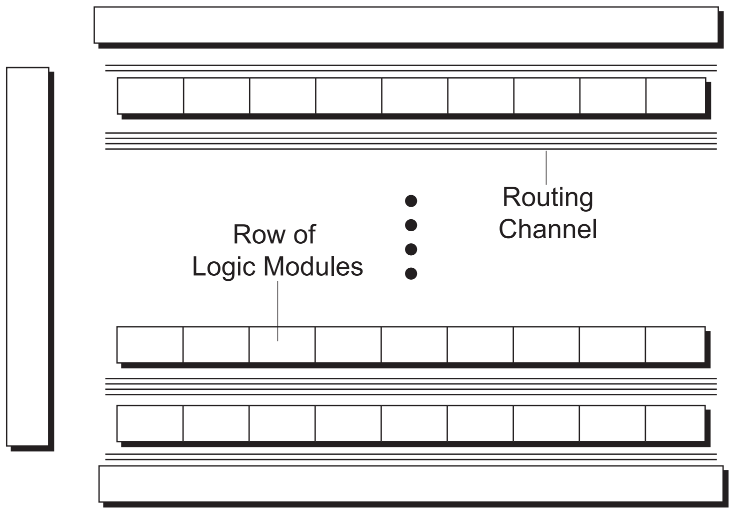


Figure B-1 • **Logic Module Architecture**

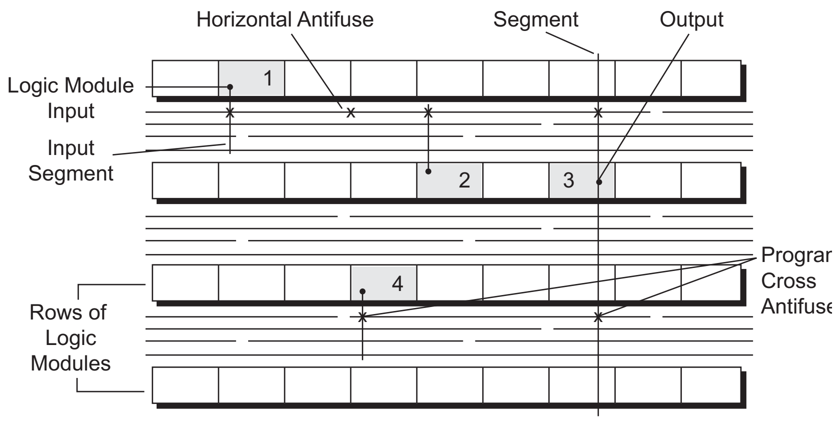


Figure B-2 • **Logic Module Interconnect**

This can be done by programming a special type of antifuse referred to as a *horizontal antifuse*. As an example, the output of Module 3 is also connected to the input of Module 1 by programming two cross-antifuses and one horizontal antifuse. Vertical antifuses are used to connect two vertical segments (not shown).

A more detailed example of the Microsemi FPGA architecture is shown in Figure B-3 on page 61. Six logic modules (two rows, three columns) are shown. Between the two rows are six horizontal tracks. Down each column are five vertical tracks. Note that the products actually have 25 to 36 (or more) horizontal and 13 to 15 (or more) vertical tracks. The circles at the intersections of vertical and horizontal tracks represent cross-antifuses. There are also circles at certain points on the horizontal tracks; these are horizontal antifuses. No vertical antifuses are shown. Notice the transistors that connect both horizontal and vertical tracks. By turning on selected transistors, various horizontal or vertical tracks can be connected even though an antifuse has not been programmed. This ability to connect tracks in unprogrammed devices is used extensively during antifuse programming and is one of the key elements responsible for the excellent testability of the Microsemi FPGAs.

Logic configuration of modules is interesting because there are no dedicated antifuses in the module to accomplish this. Instead, the inputs (and outputs) of logic modules extend into the cross-antifuse array. Each logic module has eight to ten inputs and one output. By programming appropriate antifuses, an input can be connected to a dedicated horizontal ground line, a VCC line, or a horizontal routing track. The logic module implements a particular logic function by tying appropriate unused inputs to ground or VCC.

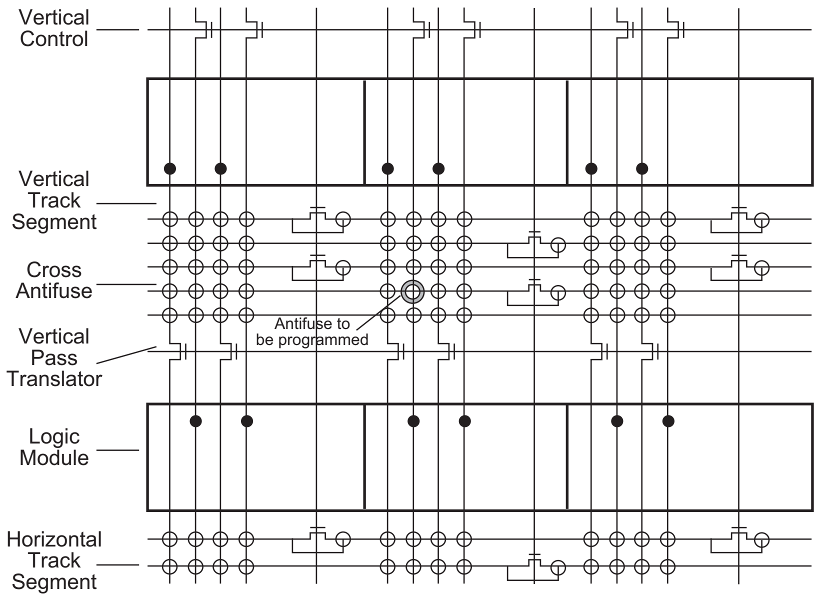


Figure B-3 • **Programmable Interconnect**

Programming

The following discussions about programming and testing modes are specific to the RH1020. However, basic concepts also apply to other RadHard and RadTolerant devices.

An antifuse is programmed by applying a sufficiently high voltage across it. This voltage is referred to as VPP. To access an antifuse deep inside the chip, it is necessary to create electrical paths from VPP and ground to the antifuse. This is done by turning on the appropriate horizontal and vertical pass transistors. (In normal chip operation, these transistors are always off.) The transistors are turned on by applying VPP to their gates. In Figure B-4 on page 62, we see an example of programming a typical cross-antifuse. VPP is applied to a vertical track at the top of the chip, and ground is applied to a horizontal track on the right side. The design of the RH1020 actually allows VPP or ground to be applied from the top, bottom, left, or right, as is most appropriate to access a particular antifuse. Notice that VPP is also applied to the gates of the horizontal and vertical pass transistors on the tracks accessing the cross-antifuse. The circled cross-antifuse now has VPP applied to it on one side and ground on the other. This voltage breaks down the antifuse’s dielectric and creates an electrical connection between the horizontal and vertical routing tracks.

There is one other important consideration when programming an antifuse. Notice that the cross-antifuses in the same vertical track as the antifuse to be programmed also have VPP applied to them on one side. This is true until the track is broken by a vertical pass transistor, below it, that is turned off.

However, the potential on the other side of the antifuses is not being driven. Should this potential be at ground, the other cross-antifuses on the vertical segment could be accidentally programmed.

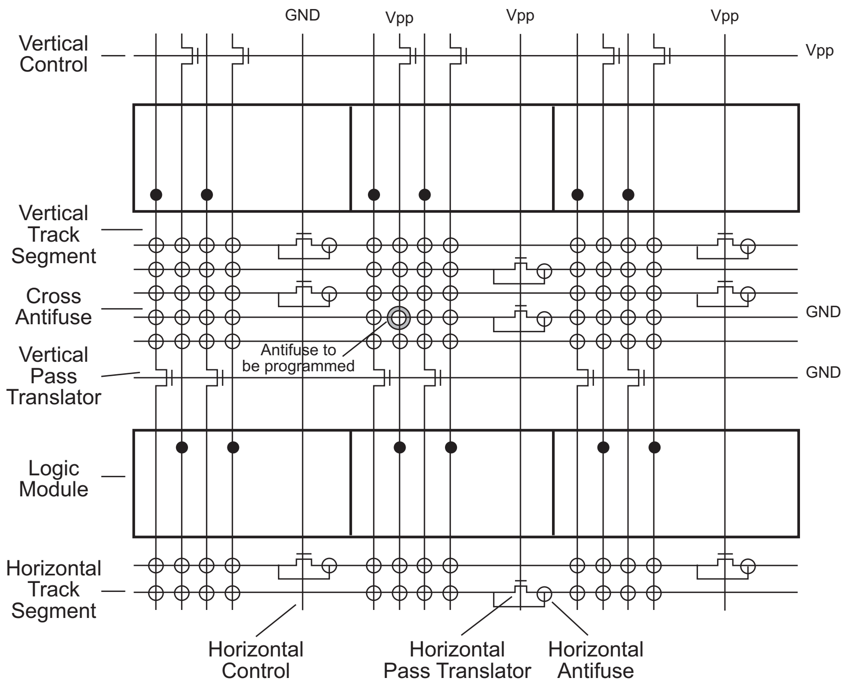


***Figure B-4* • Programmable Interconnect**

The same logic applies to other antifuses on the same horizontal track. Here, one side of the antifuse is being driven to ground, and if the other side were at VPP, extra antifuses could be programmed. This problem is solved by first applying what is referred to as a *precharge cycle*. During the precharge cycle, all horizontal and vertical tracks are charged to VPP/2.

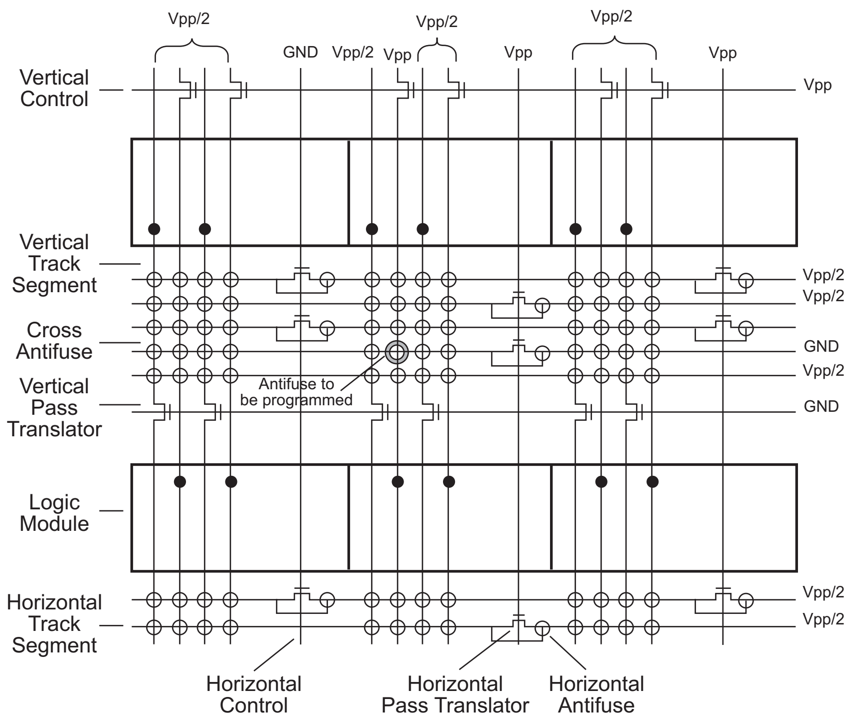


***Figure B-5* • Programmable Interconnect**

As a result, there is no voltage across the antifuses. The appropriate vertical track is then driven to VPP, and a horizontal track to ground (Figure B-5). At this point, other antifuses on the vertical track have a potential of VPP/2 across them (VPP on one side and VPP/2 on the other). This VPP/2 voltage is not sufficient to program the antifuses. Other antifuses on the same horizontal track also have VPP/2 across them (VPP/2 on one side and ground on the other). Most other antifuses in the chip still have VPP/2 on both sides and will not be programmed.

Programming Algorithm

In concept, Microsemi FPGAs are programmed in a manner very similar to many other programmable logic devices, and similar to memories such as EPROMs. The programming algorithm consists of the following steps:

1. An addressing sequence to select the antifuse to be programmed
2. A programming sequence whereby VPP is applied in pulses until the antifuse is programmed
3. A soak or “overprogram” step to ensure uniform, low antifuse resistance
4. A verify step to make sure the antifuse was properly programmed

Unlike a memory in which an antifuse is addressed by applying a parallel address, the FPGAs are addressed in a serial manner by using the special DCLK (Data Clock) and SDI (Serial Data In) pins. There is a large shift register that travels around the periphery of the chip. Bits in this shift register can be used to drive tracks to ground, VCC, VPP, or float. It is also possible to sense the level on the track (high or low) and to load this information into the shift register. By shifting in the correct address, any antifuse can be selected for programming. The shift register also plays a key role in testing the chip. This will be discussed later.

The programming sequence starts with the precharge pulse whereby VPP/2 is applied to the VPP pin. This is followed by a programming pulse that applies VPP to the pin. Following the program pulse, the voltage on the VPP pin is returned to a nominal value (about 6 V).

Refer to Table B-1 for a typical VPP waveform. The precharge/program pulse sequence is repeated until either the selected antifuse programs or a maximum number of pulses is exceeded (in which case the antifuse is considered unprogrammable and the device is rejected).

***Table B-1* • VPP Waveform**

RH1020 Programming Algorithm		**Current Parameters**
V Program	=	21 V
V Precharge	=	12.35 V
V Verify	=	6.0 V
t Program	=	150–300 µs
t Precharge	=	25 µs
I Threshold	=	–2.5 mA (to detect programmed antifuse)
I Max	=	15 mA (clamp current)
# Soak	=	30–800 pulses
Maxpulses	=	60,000

Confirmation that an antifuse has been programmed is determined by monitoring the current on the VPP pin. This current is very low (typically < 10 µa) until an antifuse is programmed. Once an antifuse is programmed, an electrical connection is made between VPP and ground, in which case currents in the range of 3 to 15 mA may be observed on VPP. Once this current is observed, the antifuse is considered programmed and enters the soak or “overprogram” cycle. Here, extra pulses are applied to the antifuse to achieve minimum antifuse resistance. Figure B-6 shows the VPP waveform for ACT 1 devices.

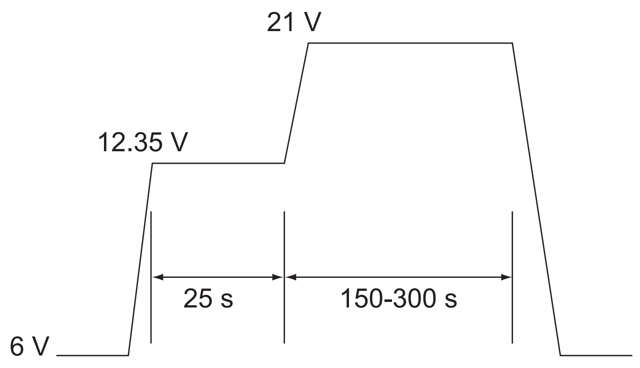


Figure B-6* • VPP *Waveform

Test Modes of Microsemi FPGAs

The unique architecture of Microsemi FPGAs allows outstanding testability of unprogrammed devices at the factory. Details of the various test modes are as follows:

- The shift register circling the periphery of the chip can be both downloaded and uploaded. This allows the use of various test patterns to ensure that the shift register is fully functional.
- All vertical and horizontal tracks can be tested for continuity and shorts. There are several ways to implement these tests. One way of doing continuity testing is to precharge the array, turn on all vertical or horizontal pass transistors on a track, drive the track low from one side of the chip, and read a low on the other side. Shorts can be detected by driving every other track low after precharge and reading back on the other side. Note that these tests also confirm that the vertical and horizontal pass transistors will turn on.
- It is important for programming to make sure that all tracks can hold the precharge level. By charging a track, floating it, and waiting a predetermined amount of time, the track can be read back and confirmed to be still high.
- Leakage of vertical and horizontal pass transistors can be tested by driving one side of a track to a voltage via the VPP pin and grounding the other side. All pass transistors except the one being tested are turned on. If excess current is detected on the VPP pin, the pass transistor is considered defective.
- There are one or two dedicated clock buffers that travel across all horizontal channels. These buffers can be tested by driving with the clock pin and reading for the proper levels at the sides of the array.
- There are two special pins referred to as Probe A and Probe B (ActionProbes). By entering a test mode, the shift register can be made to address the internal output of any logic module. This output is then directed to one of two dedicated vertical tracks, which in turn can be observed externally on the Probe A or Probe B pin. This ability to observe internal signals (even on unprogrammed parts) allows Microsemi to perform a large number of functional tests. The first such test is the input buffer test. Input buffers on all I/O pins can be tested for functionality by driving at the input pad and reading the internal I/O output node through the probe pins.
- Test modes exist to drive all output buffers low, high, or tristate. This allows testing of VOL, VOH, IOL, IOH, and leakage on all I/Os.
- One of the key tests is the ability to test functionally all internal logic modules. By turning on various vertical pass transistors and driving from the top or bottom of the chip, any of the eight to ten module inputs can be forced to a high or low. The output of the module can then be read through the ActionProbe pins. The logic module test allows 100 percent fault coverage of each module. In addition, the architecture allows modules to be tested in parallel for reduced test time.
- Microsemi FPGAs have one or two dedicated columns on the chip that are transparent to the user and used by the factory for speed selection. These columns are referred to as the Binning Circuit. Modules in the columns are connected to each other by programming antifuses. The speed of the completed test circuit can then be tested. The Binning Circuit allows the separation of units into different speed categories. It also allows the speed distribution within each category to be minimized.
- There are several tests to confirm that the programming circuitry is working. The first such test is a basic junction stress/leakage test. The program mode is enabled and VPP voltage plus a guard band is applied to the VPP pin. All vertical and horizontal tracks are driven to VPP; thus, no voltage is applied across the antifuses. The IPP current is then measured. If it exceeds its normal value, the device is rejected.
- There is a test to ensure that all antifuses are not programmed. This is referred to as the antifuse shorts test (or blank test). The array is precharged, and then the vertical tracks are driven to ground. The horizontal tracks are then read to confirm that they are still high. (A programmed or leaky antifuse would drive a horizontal track low.) The test is repeated by driving horizontal tracks low and reading vertical tracks.
- The functionality of the programming circuitry can be verified by programming various extra antifuses, on the chip, that are transparent to the user. Some of these antifuses were already described earlier when the Binning Circuit was discussed. Microsemi FPGAs also have a Silicon

Signature. In the RH1020, the Silicon Signature consists of four words of data. The first word is hardwired (no antifuses) and contains a manufacturer ID number as well as a device ID number. These numbers can be read by a programmer, and the proper programming algorithm can be automatically selected. The other words contain antifuses and are programmable. Microsemi is currently using bits in these words to store information such as the chip's run number and wafer number. Thus, each Microsemi FPGA has traceability down to the wafer level. By programming this information, the functionality of the programming circuitry is also tested. Microsemi software also allows the user to program a design ID and check sum into the Silicon Signature. By later reading this back, the user can verify that the chip is correctly programmed to a given design.

- The most important antifuse test is the stress test. When this test is enabled, a voltage applied to the VPP pin can be applied across all antifuses on the chip. (The other side is grounded.) The voltage applied is the precharge voltage plus a significant guard band. After the voltage is applied, the antifuse shorts test is again used to make sure no antifuses have been programmed. The antifuse stress test is effective at catching antifuse defects. Because the reliability of the antifuse is much more voltage dependent than it is temperature dependent, this test is also an effective antifuse infant mortality screen. Microsemi provides device reliability information in the *Reliability Report* available on the Microsemi SoC Products Group website at www.microsemi.com/soc or by contacting your local sales representative.

Burn-In of Microsemi FPGAs

As mentioned earlier, Microsemi has found that antifuse infant mortality failures can be effectively screened out during electrical testing, and it is thus unnecessary to do any kind of burn-in for standard commercial production units to screen out antifuse infant mortality failures. However, burn-in is still an effective screen for standard CMOS infant mortality failure mechanisms, and it is required for all military 883 products. MIL-883 Method 1005 allows several types of burn-in screens. These can be divided into two categories: steady state (static) and dynamic. Static burn-in applies DC voltage levels to the pins of the device under test. The device may or may not be powered up. Dynamic burn-in applies AC signals to device inputs with the unit powered up. These signals are selected so that the device receives internal and external stresses similar to those it may see in a typical application.

Static burn-in is by far the simplest to implement. By choosing appropriate biasing conditions and load resistors, it is possible to design a single burn-in circuit that can be used for both unprogrammed and programmed devices. It would not matter what pattern is programmed into the device. Static burn-in can be an effective screen for some types of failure modes, particularly those that may happen at device inputs or outputs (such as screening for mobile ionic contamination). It is not, however, very effective at stressing internal device circuits. Many internal nodes may be biased at ground without receiving any voltage or current stress. Signal lines will not toggle, and it may not be possible to screen failure modes such as metal electromigration.

A properly designed dynamic burn-in can effectively stress inputs, outputs, and internal circuits. However, dynamic burn-in of ASIC products can be very expensive because customer-specific burn-in circuits and burn-in boards must be designed and built to properly stress each design implemented in the ASIC. This results in large NRE costs and long lead times to design and build these boards. From the standpoint of burn-in, a programmed FPGA is essentially the same as a mask-programmed ASIC, and it would require similar custom burn-in circuits to do a dynamic burn-in. However, Microsemi has been able to use the testability features of its FPGA products to allow effective dynamic burn-in of unprogrammed devices. This dynamic burn-in allows users to stress circuits in a way that static burn-in would be unable to duplicate.

During burn-in of unprogrammed units, test commands are serially shifted into each device by using the SDI pin and clocked by using the DCLK pin. There are three test modes shifted into each device. The first test stresses each cross-antifuse with a voltage of VPP – 2 V. (VPP is normally set at 7.5–11 V so that each antifuse gets 5.5–9 V across it.) This voltage is applied to all vertical tracks while the horizontal tracks are grounded. Once enabled, the stress mode is held for 10 ms.

The second test mode is identical to the first except that the horizontal tracks are driven to VPP – 2 V while the vertical tracks are grounded. Note that both of these modes are similar to the antifuse stress test described earlier (although the stress voltage is lower during burn-in). Not only do these tests stress the antifuses, but they also toggle all routing tracks in the chip to VPP – 2 V and ground. All input and output tracks to the logic modules are also toggled.

The third test drives several I/O pins on the chip to a low state. Prior to this, they are at high impedance state and held at VCC through pull-up resistors. This test confirms that the burn-in is being properly implemented by looking at these I/O pins to see if they display the proper waveform. It also passes current through each I/O as it toggles low.

Although the chip is unprogrammed, these tests allow users to apply stresses to the inputs, outputs, and internal nodes that are similar to what a programmed device may see in normal operation. Once burn-in is completed, post-burn-in testing, as specified by MIL-883, is performed (including PDA) to ensure that fully compliant devices are shipped to the customers.

Conclusion

The description of the Microsemi FPGA architecture and the numerous test modes attest to the outstanding testability of these devices. All internal logic gates can be tested without programming antifuses other than the few for the Binning Circuit and Silicon Signature. Because Microsemi FPGAs are one-time programmable, the only item that is not fully tested at the factory is the programmability of all the individual antifuses. However, this is done on the programmer while the units are being programmed. Being able to test all internal gates allows Microsemi to achieve functional yields superior to other one-time programmable devices and equivalent to reprogrammable parts.

C – Silicon Signature Decode

This appendix contains information about reading and decoding information that is contained in the Silicon Signature of a device. This feature is available only in Activator. Silicon Sculptor does not support decoding of silicon signatures.

Silicon Signature Components

Information contained in the Silicon Signature enables Microsemi to provide traceability to wafer lot number in the event a Failure Analysis is required. The following information is contained in the Silicon Signature:

- Wafer lot code
- Wafer number
- Fab
- Programming voltage
- Checksum
- Silicon Signature (customer defined)
- P-fuse status
- S-fuse status
- Microsemi Device Designation

Part of the Silicon Signature is “hard-coded” into the device, meaning that the bit values are masked during wafer fab processing. Some bits are programmed into the Silicon Signature on the Activator or Sculptor during normal device programming (for example, checksum and customer’s Silicon Signature, P-fuse status, S-fuse status).

The Silicon Signature can be read by typing the command “act-read-ss” from within the APS software (either APS2 or APSW). When you execute act-read-ss, the Silicon Signature is read from the device (through the activator) and printed on the screen. The complete Silicon Signature consists of 4 to 6 (depending on the device) “words” that are printed on the screen in hexadecimal. The following sections describe how to decode the important pieces of the Silicon Signature for RH1020, A1020B, RH1280 A1280A, A1460A, and A14100A devices.

RH1020 and A1020B Silicon Signature Example

The following example shows a sample Silicon Signature for an RH1020 or A1202B device. The Silicon Signature consists of 4 words, referred to as Word 0, 1, 2, and 3. When you execute the act-read-ss command the words are printed to the screen in order (0-3) from left to right, with spaces in between each word. Figure C-1 shows the bit order and information contained in each word of the Silicon Signature.

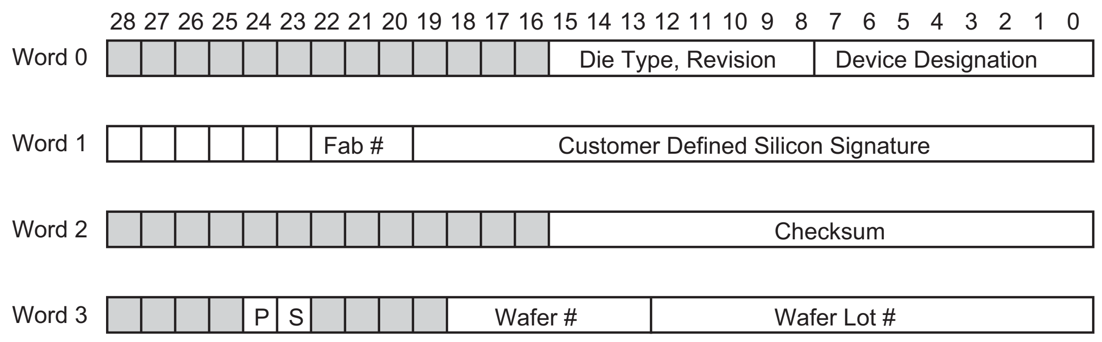


Figure C-1 **• RH020 and A1020B SIlicon Signature**

Silicon Signature Decode

The following describes the information in each word above.

Word 0: Bits 0-7 contain characters to signify the device is a Microsemi design. Bits 8-15 contain hard-coded Microsemi bits, designating die type and die revision.

Word 1: Bits 0-19 contain the customer defined Silicon Signature. Bits 20-22 contain the fab number.

Word 2: Bits 0-15 contain the checksum.

Word 3: Bit 0-12 contain the wafer lot number. Bits 13-18 contain the wafer number. Bit 23 contains the Probe fuse status (S). Bit 24 contains the Program fuse status (P).

RH1280 Silicon Signature Example

The following example shows a sample Silicon Signature for an RH1280 device. The Silicon Signature consists of 5 words, referred to as Word 0, 1, 2, 3, and 4. When you execute the act-read-ss command the words are printed to the screen in order (0-4) from left to right, with spaces in between each word. Figure C-2 shows the bit order and information contained in the Silicon Signature.

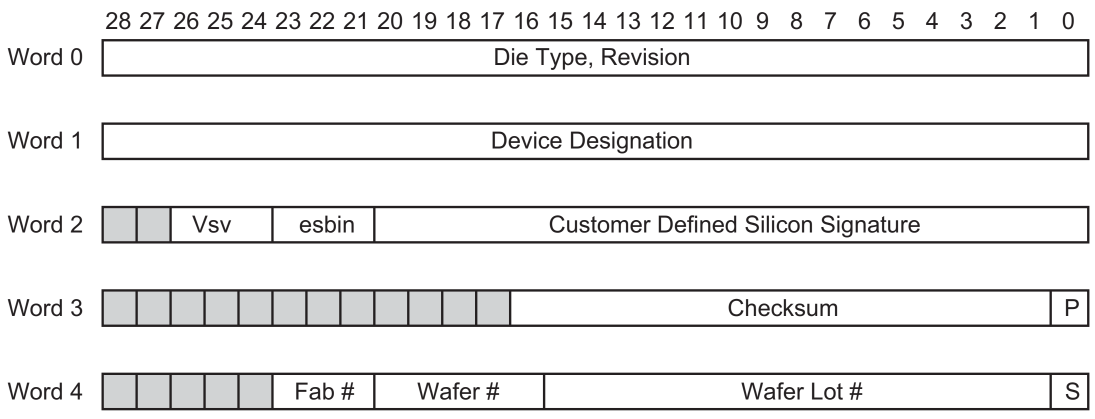


Figure C-2 **• RH1280 Silicon Signature**

Silicon Signature Decode

The following describes the information in each word above.

Word 0: Contains hard-coded Microsemi bits, designating die type and die revision.

Word 1: Contains characters to signify the device is an Microsemi design.

Word 2: Bits 1-20 contain customer defined Silicon Signature. Bits 21-23 contain programming voltage (esbin). Bits 24-26 contain the VSV voltage.

Word 3: Bit 0 contains the Program fuse status (P). Bits 1-16 contain the checksum bits.

Word 4: Bit 0 contains the Security fuse status (S). Bits 1-15 contain the wafer lot number. Bits 16-20 contain the wafer number. Bits 21-23 contain the fab number.

A1280A Silicon Signature Example

The following example shows a sample Silicon Signature for an A1280A device. The Silicon Signature consists of 4 words, referred to as Word 1, 2, 3, and 4. When you execute the act-read-ss command the words are printed to the screen in order (1-4) from left to right, with spaces in between each word. Figure C-3 shows the bit order and information contained in the Silicon Signature.

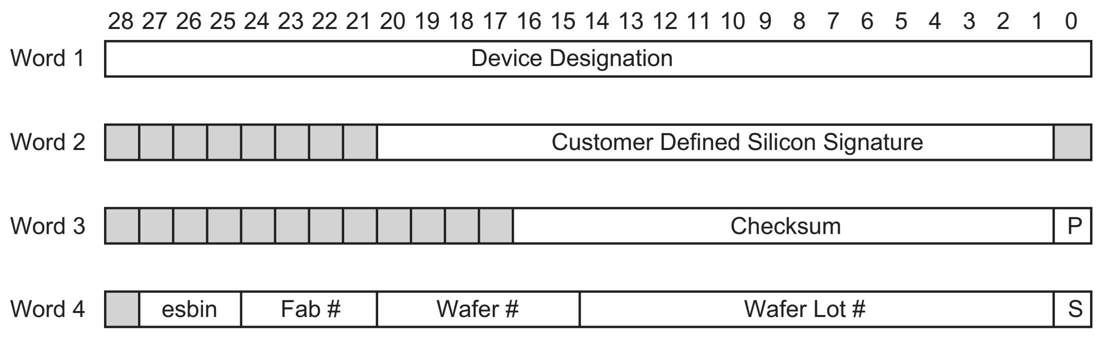


***Figure C-3* • A1280A Silicon Signature**

Silicon Signature Decode

The following describes the information in each word above.

Word 1: Contains hard-coded Microsemi bits, designating die type and die revision.

Word 2: Bits 1-20 contain customer defined Silicon Signature.

Word 3: Bit 0 contains the Program fuse status (P). Bits 1-16 contain the checksum bits.

Word 4: Bit 0 contains the Security fuse status (S). Bits 1-14 contain the wafer lot number. Bits 15-20 contain the wafer number. Bits 21-24 contain the fab number. Bits 25-27 contain programming voltage (esbin).

A1460A and A14100A Silicon Signature Example

The following example shows a sample Silicon Signature for an A1460A or A14100A device. The Silicon Signature consists of 6 words, referred to as Word 0, 1, 2, 3, 4, and 5. When you execute the act-read-ss command the words are printed to the screen in order (0-5) from left to right, with spaces in between each word. Figure C-4 shows the bit order and information contained in the Silicon Signature.

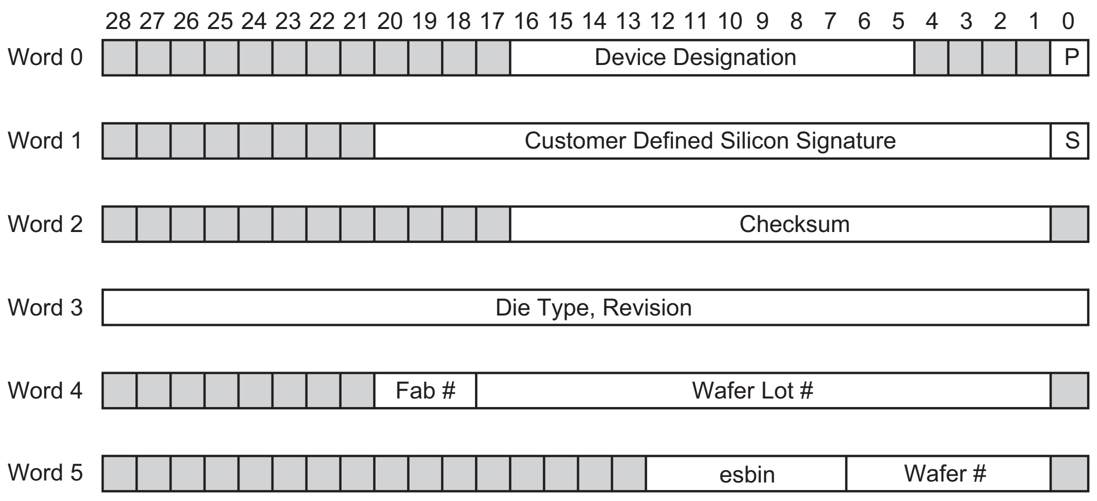


***Figure C-4* • A1460A Silicon Signature**

Silicon Signature Decode

The following describes the information in each word above.

Word 0: Bit 0 contains the Program fuse status (P). Bits 5-16 contains characters to signify the device is an Microsemi design.

Word 1: Bit 0 contains the Security fuse status (S). Bits 1-20 contain customer defined Silicon Signature.

Word 2: Bits 1-16 contain the checksum bits.

Word 3: Contains hard-coded Microsemi bits, designating die type and die revision.

Word 4: Bits 1-17 contain the wafer lot number. Bits 18-20 contain the fab number.

Word 5: Bits 1-6 contain the wafer number. Bits 7-12 contain programming voltage (esbin).

D – AVI File Description

Activator

The AVI file is a log file generated while an Microsemi FPGA is programmed. The file contains information about the number of VPP pulses applied to each fuse to program the fuse and the programming current sensed through each fuse. If a programming failure occurs, the AVI file contains information about the programming failure mode.

A new AVI file is generated each time you begin a new programming sequence. If you want to save an AVI file, you must re-name it before restarting the programming sequence. The following excerpt shows an example AVI file:

```
*************************************************************
; FILEID AVI \example.avi
; PROGRAM Activator (tm) 2 Programming System 3.1.1
; VAR DDFDIE c:\ACTEL/data/a1200/1280/G1280.ddf
; VAR DDFPACKAGE c:\ACTEL/data/a1200/1280/qfp172.ddf
; VAR FUS \designs\mod25\mod25.fus
; VAR AFM \designs\mod25\mod25.afm: Compressed
; VAR SIG
; ACT-FUSE: Fuse 1
; SILICON-SIGN 4:   5F80000 FFE 600000 FD335C
; ICC-STANDBY:22---
; IPP-STANDBY:0
; ESBIN: 3
; START-TIME Mon Mar 18 17:59:12 1996
;4321
1:14611------
2:1467------
3:14610------
4:1468------
5:1468------
6:1469------
7:14611------
8:1467------
9:1464------
10:1468------
11:14610------
12:14620------
13:14610------
14:1465------
15:14615------
16:14618------
17:1469------
18:14611------
*************************************************************
```

The first thirteen lines of the file shown above contain header information. This information is obtained from both the programming file (*.afm or *.fus) and the unit being programmed. Each line of the AVI file header is preceded by a semi-colon. The header contains the following information:

- The complete Silicon Signature (read from the device) prior to programming. Refer to the "Silicon Signature Decode" section on page 69 for additional information about the Silicon Signature.
- The ICC standby value of the device before it is programmed. In the example, the ICC standby of 22 corresponds to a standby of 2.2 mA.
- The device esbin number, which corresponds to the VPP used during programming.

The remainder of the AVI file contains information about the programming sequence. As each fuse is programmed, a new line is added to the AVI file. The first column of numbers in the AVI file is the fuse number. A typical RH1280 design requires approximately 15700 fuses to be programmed. The AVI file would contain over 15700 lines, including header information.

There are 8 additional columns in the AVI file. The 8 columns are listed in groups of 2, with each group corresponding to a socket on the Activator 2 (sockets are numbered 1-4). If the Activator 2s is used, the single socket is always recognized as socket 4.

The first column in a group of 2 contains the final programming current sensed by the Activator through each fuse as it is programmed. The programming current is listed in milliamps, and no decimal point appears in the number. For example, a programming current listed as 146 corresponds to current of 14.6 mA. It is not unusual for different fuse types to have different programming currents.

The second column in a group of 2 contains the number of VPP pulses that were required to program a specific fuse. The maximum number of programming pulses reported in the AVI file is 64464 pulses. There is an inherent variation in the number of pulses required to program a fuse depending on the fuse type. A wide distribution of VPP pulse counts can be expected.

After the array fuse programming information, test results from the end of programming tests executed by the Activator are stored. If any of these tests fail, the failing test is indicated in the AVI file.

Silicon Sculptor

An AVI file is generated automatically for RH/RT part while programming a RH/RT part in Silicon Sculptor. This file format is almost same as the *.avi file generated while programming a part in Activator. But the file extension for Silicon Sculptor is *.txt instead of *.avi. Here is a portion of an *.avi file from the Silicon Sculptor 3.56 while programming a RT part:

```
FILEID  alpha11.txt
 PROGAMMER  Silicon_Sculptor
 SOFTWARE VERSION  V3.56
 DEVICE RT1020-CQ84
 SILICON SIGN:  022DFB  060000  700000  01222F
 ESBIN  0
 AFM-CHECKSUM  9E63
 START-TIME  08/08/01 16:57:00

 Fuse No    Pulses     Current (mA)
 00001        236        14.9
 00002        473        8.8
 00003        332        10.7
 00004        211        7.0
```

E – JobMaster (Silicon Sculptor only)

JobMaster enables an Administrator to set up a job to precise specifications, test the results, and then protect the routine so that it cannot be modified inadvertently. Once a master device is programmed and approved, JobMaster is ready to begin full-scale programming immediately, without further set up.

JobMaster has two modes for the production facility:

- Administrator(s) - designated to set up jobs within the software, as well as regulate changes made to any existing jobs.
- Operator(s) - runs the jobs to program devices.

Installation

JobMaster is supplied with the BP Software; however, you must have an access code to enable it. To receive an access code, you must purchase the JobMaster software from the BP Microsystems sales department (http://www.bpmicrosystems.com). Once you have purchased the software, an access code is generated and either e-mailed or faxed to you. After you obtain access to the software, you must:

1. Configure JobMaster. Select **JobMaster/Configure**.
2. Set the **LIBPATH** environment variable to include the directory where sculpt.exe and bpjob.dll reside.
3. Set the **sculpt.exe** environment variable to your local hard drives. Do not share the sculpt.cfg files among multiple PCs.
4. Reboot your PC and run **sculpt.exe**.
5. Select the **Upgrade** option from the AFS menu and enter the upgrade code that was shipped with your diskettes or provided by the BP Microsystems Sales Department.
6. Specify your JobMaster Database directory. Go to the JobMaster/Configure menu to specify your JobMaster Database Directory. This will be the directory where all of your users can access the database. Choose whether or not you want to enable categories and tab to **Accept.**
7. Press **Enter** to set your options.

Operation

The sections below describes how to make full and best use of the JobMaster software.

Creating a New Job

This section assumes you understand the basics of programming a device using the BP software. Please refer to the BP software user guides for more information on programming with BP software. Before creating a new job within JobMaster, make sure you have completed the following steps:

1. Select the device that you want to program.
2. Load the buffer with the information to be programmed into the device.
3. Set any special configuration options. Once you have chosen at least one device and have loaded contents and set any special configuration, you are ready to create a new job.
4. Select **JobMaster/New.**

You can define the Category and Item/Part Number fields in any way you choose.

We suggest the following options in these fields:

- Reference a manufacturer in the Category field, if you intend to create a specific manufacturer database. For example, use Category: Microsemi to create a Microsemi database.

- Reference a customer in the Category field and the job number or location on the circuit board in the Item Number field.

The device you have programmed is automatically entered into the first Device line. You can add up to five additional devices, provided they have identical programming requirements. See "Adding a Device to an Existing Job" on page 78 for more information.

The Note option is another user-definable field that is displayed whenever an Operator chooses the job.

If the Verify Checksum of Data option is enabled, the system stops and prevents the job from running if data has changed since the job was created.

Note: We strongly recommend that the Verify Checksum of Data option be set to Enabled, unless you have a data file that is going change frequently.

Once you have set the options, select **Accept** and press **Enter**. A screen prompts you to verify that you want to add the new job to the database. If the information is correct, press **Enter** again to save it in the database. If you need to make changes, select **Cancel** to return to the New Job screen.

Copying an Existing Job

To copy the Category, Item Number or Device from a currently existing job press **Enter** when the cursor is on an empty field. This brings up a selection box that you can use to select currently existing entries. Highlighting a selection and pressing **Enter** inputs the information into the new job.

Adding a Device to an Existing Job

JobMaster permits the programmer to list as many as six devices under a single job description.

Note: It is essential that all the devices listed for a single job be absolutely compatible. They must accept exactly the same configuration and programming algorithms.

To add a device, select **JobMaster/Load**. Then select **JobMaster/Updates** and bring up the change screen and add the device to the next available device line by tabbing to an empty device line and pressing **Enter**. This brings up the device selector dialog screen. Once all of your changes are complete, select **Accept** and press **Enter**. The update job is saved over the existing job in the database.

Updating a Job

To update a job, select J**obMaster/Load** to load a job. Use the BP software to change the Device/Configuration, Buffer/Options, or other parameters, then select **JobMaster/Update** to accept the changes as part of the current job.

Note: JobMaster keeps track of the date, time and author of each job's creation, and of every subsequent revision. This information is displayed whenever you select the JobMaster/Load option.

Locking The Programmer In Operator Mode

To lock the programmer in Operator Mode, select **JobMaster/Options** and set the default mode to **Operator Mode**. Tab to **Accept** and press **Enter**.

When you select the Operator Mode default, you are prompted to enter a password. If you enter a password, you must use it to get out of Operator Mode. If this protection is not desired, do not enter a password, tab to **Accept** and press **Enter**.

Password Protection

The Operator Mode selection is stored both in the programmer itself and in the BP software. In either case (for PC or Software), the password is required to unlock that programmer and exit the Operator Mode.

Running A Job (Program Mode)

Assuming the Operator Mode default has been selected, the programmer will “make up” in Operator Mode. Operator Mode offers only four options: Program, Verify, Stop and Quit. To begin programming, tab to the Program option and press **Enter**.

1. Enter the Category, Item Number, and Device. JobMaster asks for a Category, Item Number and Device to program. Enter these at the prompts. You can always press Enter on an empty field to see a list of available options from the database. If more then one device has been specified for the job, you will be prompted with a list from which to choose. Simply tab to the appropriate selection and press **Enter**.

Once you have selected the device, JobMaster displays any Notes that were added when the job was created.

1. Enter the number of devices. Type this number and press **Enter**. This refers to the number of devices that should be successfully programmed, not the number of attempts, to allow for any rejects that may normally occur.
2. Evaluate summary screen, and continue. A summary screen appears, listing Category, Item Number, File, Checksum, Operation, and the Note. If everything is correct, select **Accept** and press **Enter.** This system begins programming devices.

Deleting A Job

To delete a job from the JobMaster database, you must have access to the Administrator password.

1. Go to **JobMaster/Delete**
2. Select the **Category** and **Item Number** of the job you wish to delete. A screen appears asking you to verify the delete option.
3. Press **Accept** to continue with the delete process, choose **Cancel** to cancel the delete process.

Returning To The Normal Mode

To return to the BP Software to normal mode upon startup, simply enter the Administrator password, and then go to JobMaster Configure. Tab down to the Default Mode field and select **Normal.** When the BP Software is started up again, it will be in normal access mode.

F – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 650.318.8044

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Customer Support website (www.microsemi.com/soc/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Index

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

5029106-1/8.01