User's Guide

Debug64700

Notice

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

© Copyright 1993, 1994, Hewlett-Packard Company.

This document contains proprietary information, which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company. The information contained in this document is subject to change without notice.

HP is a trademark of Hewlett-Packard Company.

Microtec is a registered trademark of Microtec Research Inc.

OSF/Motif and Motif are trademarks of the Open Software Foundation in the U.S. and other countries.

SunOS, SPARCsystem, OpenWindows, and SunView are trademarks of Sun Microsystems, Inc.

UNIX is a registered trademark of UNIX System Laboratories Inc. in the U.S.A. and other countries.

Hewlett-Packard
P.O. Box 2197
1900 Garden of the Gods Road
Colorado Springs, CO 80901-2197, U.S.A.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013. Hewlett-Packard Company, 3000 Hanover Street, Palo Alto, CA 94304 U.S.A. Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Printing History

New editions are complete revisions of the manual. The date on the title page changes only when a new edition is published.

A software code may be printed before the date; this indicates the version level of the software product at the time the manual was issued. Many product updates and fixes do not require manual changes, and manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual revisions.

Edition 1	B1471-97010, April 1993
Edition 2	B1471-97011, April 1994

In This Book

This book describes the Debug64700 interface, which connects the HP 64700 Debug Environment to the SoftBench Environment. If you are learning about SoftBench or the HP 64700 Debug Environment, you should explore each of these systems first before studying this manual, which explains how the two systems are integrated together. This book is organized into four parts whose chapters are described below.

Part 1. Quick Start Guide

Chapter 1, Getting Started, presents a step-by-step tutorial on how to perform a few basic tasks using the Debug64700 interface.
Chapter 2, Setting Up Your Project for Debug64700, describes the steps you need to take to set up your project so that it works correctly with the Debug64700 interface.

Part 2. User's Guide

Chapter 3, Controlling the Debug64700 Interface, shows you how to start up and use the Debug64700 interface.
Chapter 4, Debug64700 Interaction with Other SoftBench Tools, describes how the interaction between Debug64700 and other SoftBench tools is set up.

Part 3. Reference

Chapter 5, Debug64700 Application Resources describes the common user-modifiable X resources for the Debug64700 application.
Chapter 6, SoftBench Messages Supported, describes the messages that can be forwarded to the Debug64700 tool to generate SoftBench Messages as well as the SoftBench Messages the Debug64700 tool can receive.
Chapter 7, Common Problems and Solutions, provides information about some of the common problems you may encounter when using the Debug64700 tool.

Part 4. Installation Guide

Chapter 8, Installation, contains information related to the installation and setup of Debug64700.

For More Information ...

- For an overview of the SoftBench System, go to the *SoftBench Overview Manual*. For more specific information on the SoftBench System, go to the manuals referenced in the *SoftBench Overview Manual*.

- For information on your emulator, go to the *Emulator User's Guide Manual* for the particular emulator you are using.

Contents

Part 1 Quick Start Guide

1 Getting Started

Step 1. Start the demo 15
If you have problems starting the demo 20
Step 2. Select and start the HP 64700 system 21
Step 3. Use the Static Analyzer 23
Step 4. Use the Static Analyzer graph to set breakpoints 25
If you could not set a breakpoint using the Static Analyzer 27
Step 5. Invoke the SoftBench Editor 28
Step 6. Rebuild the target executable file 29
Step 7. Use the AxLS Branch Validator 30
Step 8. Use the Software Performance Analyzer (HP 64700 emulation system only) 32

2 Setting Up Your Project for Debug64700

Step 1. Compile your programs for Static Analysis 35
Step 2. Set X resources for the new action keys 36
Step 3. Verify that SoftBench is properly configured 37
Step 4. Start the Debug64700 system 38

Part 2 User's Guide

3 Controlling the Debug64700 Interface

To start the Debug64700 interface 45
To change the target executable (context) 46
To select an HP 64700 system 47
To select the interfaces to start 48
To specify a configuration file 49
To specify interface startup commands or command files 51
To set the load options 53
To view interface status 54
To start additional interfaces 55
To access Debug64700 online help 57

4 Debug64700 Interaction with Other SoftBench Tools

The SoftBench Editors 61
The SoftBench Builder 62
The SoftBench Static Analyzer 65
The HP Branch Validator 66

Part 3 Reference

5 Debug64700 Application Resources

Debug64700*interfacePreselect*String: Emul 70
Debug64700*systemName*String: em68000 71
Debug64700*configFile*String: c68000.EA 71
Debug64700*debugCommand*String: debuggercommand.com 71
Debug64700*emulCommand*String: emululatorcommand 71
Debug64700*perfCommand*String: performancecommand 71
Debug64700*loadOptions*String: Executable 72
Debug64700*loadNotification*String: True 72
Debug64700*autoStart*String: False 72
Debug64700*bbaOptions*String: SHOW-HISTOGRAM 72

6 SoftBench Messages Supported

SoftBench Edit Commands 75
SoftBench Build Commands 76
SoftBench BBA Command 77
SoftBench Generic Message Commands 78
SoftBench Messages Received 80

7 Common Problems and Solutions

Cannot Set Breakpoint from Static Analyzer to Debug Environment 84
Executable File Doesn't Load 85
What Does the Space in the Context Mean? 86
Can I Load Multiple Executables Into the Debug Environment? 86
Why Does the System Ask for a Configuration File? 87
Where Does the Executable File and Configuration File Come From When Running Remote? 87
What is the Advantage of Using the Supplied Compile Scripts to Generate Static Analysis Object Files? 88

Part 4 Installation Guide

8 Installation

Supported Emulator Systems and Software Versions 93
Installing the Debug64700 Software 94
Customization of the SoftBench Environment 95
Updating the MANPATH Environment Variable 97
Using Debug64700 Remotely 98
Adding Magic Strings 100
Installation Verification 101

Index

Part 1

Quick Start Guide

A one-glance overview of the product and a few task instructions to help you get comfortable.

1

Getting Started

Getting Started

This tutorial gives you step-by-step instructions on how to perform a few basic tasks using the Debug64700 interface. It uses the demo program supplied with the Debugger/Emulator, Debugger/Monitor, or Emulator/Analyzer interface software. This tutorial assumes:

- You have the HP AxLS C cross compiler for the HP emulator or ROM monitor you have purchased. If you do not have the HP AxLS cross compiler installed, go to the chapter titled "Setting Up Your Project for Debug64700".
- You have already verified operation of your emulation or ROM monitor system and SoftBench.

The following steps show you many of the advantages of using the Debug64700 system in combination with SoftBench.

This chapter shows you how to:

1 Start the demo.

2 Select and start the HP 64700 system.

3 Use the Static Analyzer.

4 Use the Static Analyzer graph to set breakpoints.

5 Invoke the SoftBench Editor.

6 Rebuild the target executable file.

7 Use the AxLS Branch Validator.

8 Use the Software Performance Analyzer (HP 64700 emulation system only).

After running this demo, go to the next chapter titled "Setting Up Your Project for Debug64700", which explains the steps you need to take to set up your system for the Debug64700-to-SoftBench connection.

Step 1. Start the demo

A demo program and its associated files are provided with the emulator's Graphical User Interface.

1 Change to the demo directory with the **cd $HP64000/demo/debug_env/hpXXXXX** command (where XXXXX is the HP model number for your emulator). For example, if you're using an HP 64742 emulator (for Motorola 68000), enter the command:

```
$ cd $HP64000/demo/debug_env/hp64742 <RETURN>
```

HP64000 is a shell variable which is set to the directory where HP 64000 software has been installed (typically /usr/hp64000).

For the Debugger/Monitor, use the equivalent emulation directory for your processor:

68000	hp64742
68302	hp64746
68020	hp64748
68030	hp64747
68040	hp64783
6833x	hp64749
68340	hp64751
68360	hp64780

2 Start the Debug64700 demo.

```
$ ../debug64700/Startdebug64700 <RETURN>
```

or, to run the demo with the Debugger/Monitor:

```
$ ../debug64700/Startdebug64700 -mon <RETURN>
```

The -mon option compiles the files for the Debugger/Monitor. Refer to the "READMEMON" file in the demo directories to determine how to set up the demo files for your ROM monitor target.

As the Startdebug64700 script runs, the following information is displayed.

```
You are currently in the directory:
/usr/hp64000/demo/debug_env/hp64742

You must run this demo in a writable directory!
Would you like to copy the demo files to a different directory?
[y/n] (y)
```

Type "y" and press <RETURN>, or press <RETURN>.

```
Please enter the directory (full path name) to which you
want the demo files to be copied (default: $HOME/demo).
>
```

Enter a full path directory name, or select the default by pressing <RETURN>.

```
Copying files to $HOME/demo/debug_env/hp64742 ...
Copy succeeded ...
Re-making to update symbol paths and create Static Analysis files ...

/usr/hp64000/bin/cc68000static -I. -LM -OG -c main.c
/usr/hp64000/bin/cc68000static -I. -LM -OG -c init_system.c
/usr/hp64000/bin/cc68000static -I. -LM -OG -c update_sys.c
/usr/hp64000/bin/cc68000static -I. -LM -OG -c proc_spec.c
/usr/hp64000/bin/ld68k -c Linkcom.k -L -o ecs.x main.o init_system.o
update_sys.o proc_spec.o >ecs.MAP
```

At this point the files have been copied to your specified directory and have been recompiled to update the symbol paths. Additionally, static analysis files (for example, main.q), to be used by the Static Analyzer tool, have been created.

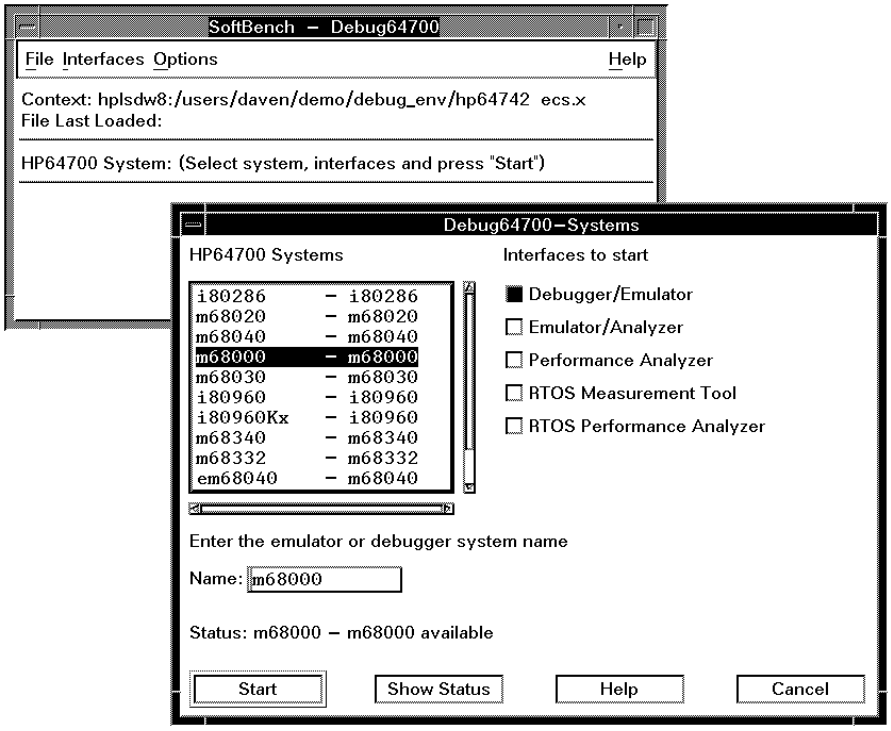
```
Making a local version of softinit in this directory.
Appending the "-demo" option to the DEBUG (debug64700) tool.
Starting SoftBench using this local softinit ...
```

A local version of softinit has been created using the default softinit file (default=/usr/softbench/config/softinit) and it has been modified specifically for this demo. This version of softinit is only to be used with this demo program. Finally, SoftBench is started using this softinit file. If SoftBench is already running, you will see the message below instead of the Starting SoftBench ... message.

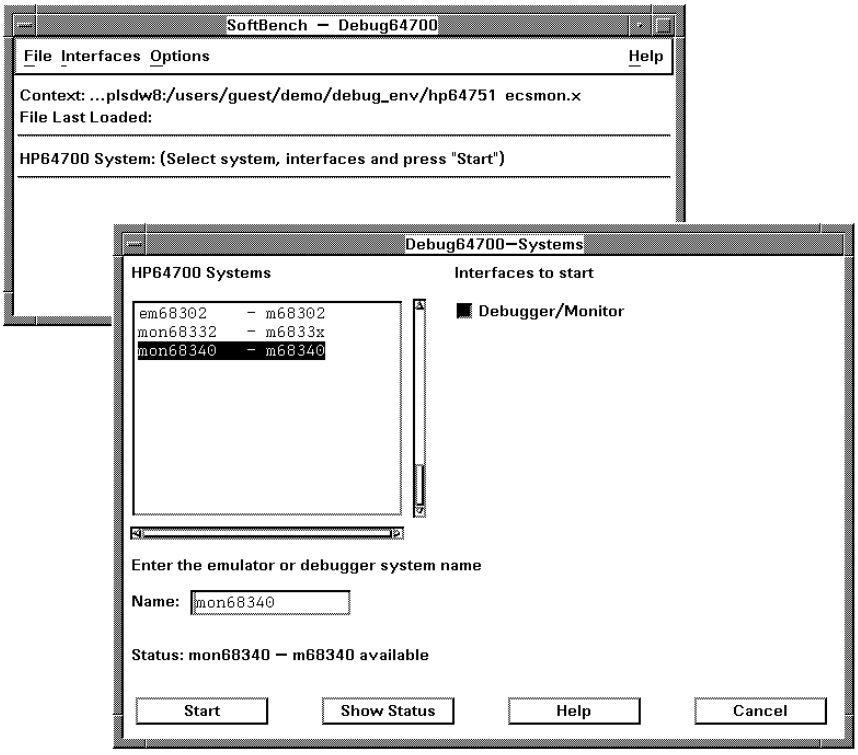
```
SoftBench is being reinitialized to use the local softinit

Starting "debug64700 -demo"
```

This informs you that the script is now starting debug64700 in a demonstration mode of operation. In addition to seeing the normal

SoftBench startup windows you will also see the SoftBench-Debug64700 interface as shown in the following figure.



If you are using the Debugger/Monitor, the SoftBench-Debug64700 interface will look like the following figure.




```
Now you can start and restart demo by either:

  1) Double clicking on "ecs.x" in the Development Manager
  2) Using Tool Manager Start by double clicking "DEBUG"
     after you have changed the context to include the file "ecs.x"
  3) Entering the command "debug64700 -demo" in a terminal window

If you quit SoftBench the demo setup will be lost.
You can restore it by changing directory to your writable demo
directory
and executing either:
        "Startdebug64700"
              or
        "softbench; softmerge -f
$HOME/demo/debug_env/hp64742/softinit"


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
NOTE: You should now issue the following command in this
      terminal window to change to the new demo directory:

      cd $HOME/demo/debug_env/hp64742
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

The above text explains how to restart this demonstration. Don't stop SoftBench until after you've completed the rest of this demonstration.

3 Change to the new demo directory (as listed in the NOTE above). For example:

```
$ cd $HOME/demo/debug_env/hp64742 <RETURN>
```

If you have problems starting the demo

☐ Does your PATH variable contain paths to the /softbench/bin and /hp64700/bin directories?

☐ If the HP 64000 products are not installed in /usr/hp64000, have you exported the shell variable HP64000 and set it to the equivalent path of /usr/hp64000?

Note: On Sun Solaris, HP 64000 products are typically installed in /opt/hp64000.

☐ Can you start SoftBench manually by entering "softbench"?

If you cannot start SoftBench, refer to the *SoftBench Installation* manual.

☐ If SoftBench is not installed in /usr/softbench, have you exported the shell variable SOFTBENCH and set it to the equivalent path of /usr/softbench?

To set shell variables such as HP64000:

If using sh(1) or ksh(1), enter:

```
$ HP64000=/mydir/usr/hp64000 <RETURN>
```

```
$ export HP64000 <RETURN>
```

If using csh(1), enter:

```
$ setenv HP64000 "/mydir/usr/hp64000" <RETURN>
```

Step 2. Select and start the HP 64700 system

After the demo is successfully started (as described in Step 1), a "Debug64700—Systems" dialog box is opened so that you can select and start the HP 64700 system you'll be using.

If you have started this demo with the "-mon" option for the Debugger/Monitor, you must select a Debugger/Monitor system. Only the Debugger/Monitor interface is available to be selected and must be selected for the demo to continue.

1 Select the HP 64700 system either by positioning the mouse pointer over the appropriate system name and clicking the *select* mouse button, or by positioning the mouse pointer in the "Name" field and typing the name of the system.

You must select an HP 64700 system that matches the directory where you started the demo script. For example, if you're using the HP 64742 emulator, select a system whose processor type is "m68000".

If you have no system names to choose from, you must edit the $HP64000/etc/64700tab.net file to add them. You can use the File→Edit HP64700 Systems command in the "SoftBench — Debug64700" window. Refer to your emulator *User's Guide* and the comments in the 64700tab.net file for information about proper contents.

2 After selecting an HP 64700 system, the status is shown in the "Status" field. You can also display the status of the HP 64700 system by clicking on the "Show Status" button.

If you typed a name into the "Name" field, you must click on the "Show Status" button to get valid information about the status of the system.

The "Status" field should show that the system is available.

If a system is not available because it is locked, you can click on the "Unlock" button to make it available. Because unlocking a HP 64700 system may cause another user to lose a valuable measurement, check with the user before you unlock their session.

If a system is unusable (you get the message: HP64700 I/O error), verify that the system has power and that it is connected through the LAN or a serial port.

3 Start the HP 64700 system by clicking on the "Start" button.

It takes a few seconds for the Debugger/Emulator interface to start and the demo configuration file (Configall.EA) executable file (ecs.x) to be loaded.

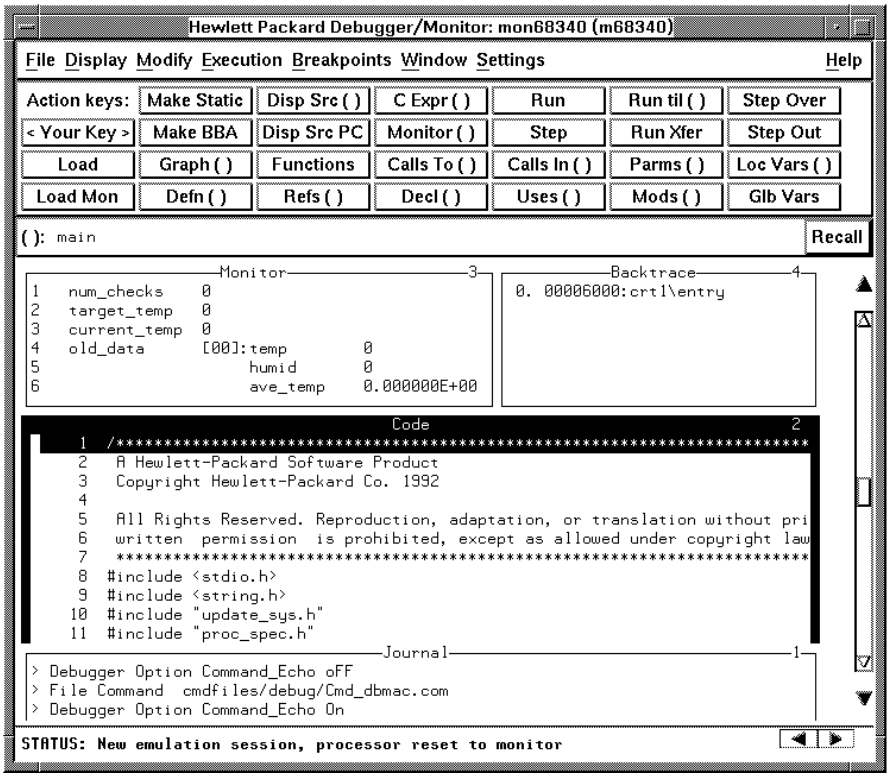
The Debugger/Emulator interface is preselected in the demo program. If you did not purchase this interface, the Debugger/Emulator selection will be half-toned and unselectable. If this is the case, select the Emulator/Analyzer interface instead.

You must select one or the other of these interfaces, or both, to continue this demonstration. It is assumed that you have the Debugger/Emulator interface selected.

To get additional information about starting the Debug64700 system click on the "Help" button.

Step 3. Use the Static Analyzer

If the demo has proceeded correctly to this point, you should see the Debugger/Emulator or Debugger/Monitor interface as shown below.



Notice the four lines of action keys. The top two rows control operations associated with the Debugger, and the bottom two rows demonstrate features of the Debug64700 connection to SoftBench.

You can customize these action keys by modifying the application's X resources. Refer to your Debugger/Emulator and Emulator/Analyzer *User's Guides* for information on customizing action keys.

Wait until the interface is fully initialized (until the half-toned action keys appear normal) before performing the following steps.

1 Click on the **Functions** action key.

This action key sends a message to the Static Analyzer to display all of the functions in the demonstration program.

Two windows appear: one called Static Analyzer Output, and the other called SoftBench - Static Analyzer. Close the Static Analyzer Output window by clicking on "Close". In the SoftBench - Static Analyzer window, you should see a list of all the functions in the demonstration program (ecs.x).

2 Make sure "main" is in the entry buffer (labeled "():" in the Debugger interface), and click on the **Calls In ( )** action key.

This action key sends a message to the Static Analyzer to display all of the function calls inside the function "main".

The list of functions shown in the SoftBench - Static Analyzer window are now just the functions called by the function "main".

3 In the same way, you can click on the action keys labeled:

Calls To ()
Decl ()
Parms ()
Uses ()
Loc Vars ()
Mods ()
Refs ()
Glb Vars

Refer to your *SoftBench Static Analyzer Users Guide* for a complete discussion of the operations associated with these action keys.

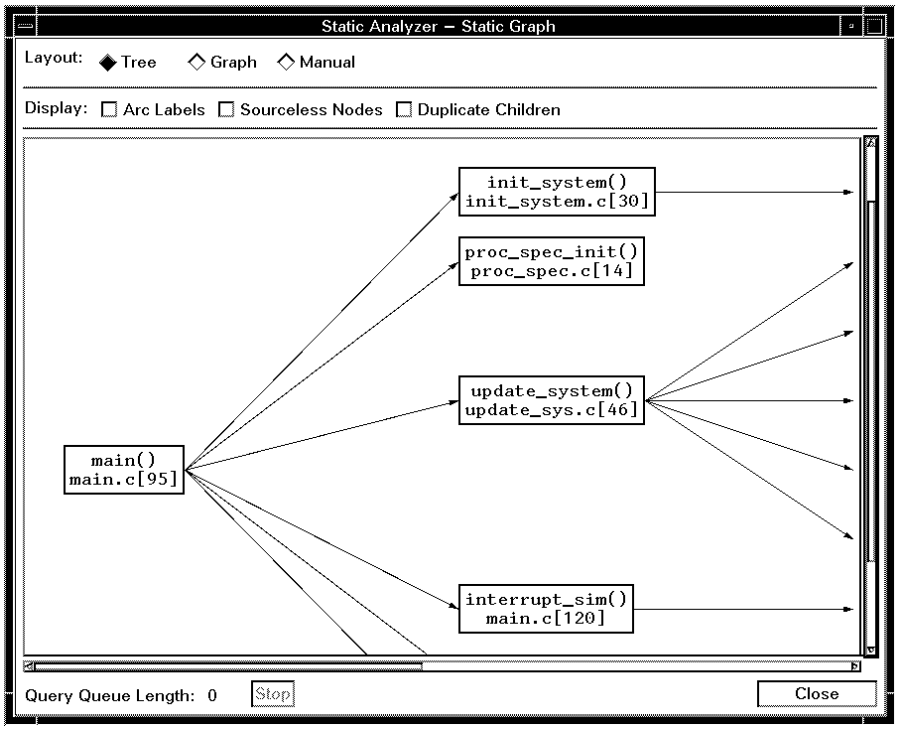
Step 4. Use the Static Analyzer graph to set breakpoints

The Static Analyzer graph is only available in SoftBench version B.00.00 or greater. If you are using an earlier version of SoftBench, skip this step.

1 Make sure "main" is in the entry buffer, and click on the **Graph ()** action key.

2 Select main in the Static Analyzer using the left mouse button; then, access the pop-up menu with the right mouse button and select "Recursive Calls Within" (also with the right mouse button).

A graphical representation of the function call graph is shown.



3 Select the arc pointing to the function update_system using the left mouse button; then, access the pop-up menu with the right mouse button and select "Set Break Point" (also with the right mouse button).

   In the Debugger, you will see the breakpoint displayed in the Breakpoint window.

4 To verify that the breakpoint is set, choose the **Execution→Run→from Transfer Address** command in the Debugger window.

   Program execution should halt on the line that calls the function update_system.

If you could not set a breakpoint using the Static Analyzer

☐ Is the Debug64700 interface still running?

If you exited the interface, the communication between SoftBench and the Debug Environment cannot occur. Choose the File→Exit→Released command from the Debugger window, and restart the demonstration.

☐ Make sure that "main" is in the entry buffer "():" before you press the "Graph ()" action key.

☐ If the breakpoint will not set, make sure the Static Analyzer knows the context of the DEBUG tool (debug64700).

Pressing any of the action keys to send a Static Analyzer command will set this context.

Step 5. Invoke the SoftBench Editor

1 With the current program counter at the address of the breakpoint (as shown by the highlighted line in the Debugger's Code window), choose the **File→Edit→At PC Location** command.

This command sends a message to the SoftBench Editor that is currently selected as the EDIT tool. This editor may not be your default editor because the demo program is using the default tool configuration file "softinit" supplied with SoftBench.

2 Choose the **File→Quit File** command in the SoftBench Editor window.

You may want to experiment with other edit commands and verify that the SoftBench editor is invoked each time. The Debugger's Breakpoints→Edit/Call Macro command may or may not invoke the SoftBench editor. Newer versions of the Debugger will invoke the SoftBench editor.

Step 6. Rebuild the target executable file

- If you have the HP AxLS compiler installed, or if you have modified the make file to work properly with your compiler (where ecs.x is the target of your makefile), press the **Make Static** the action key.

  This action key changes the dates on the ".c" files and sends a message to the SoftBench Build tool to build the target ecs.x file. (A better software development practice would be to remove the ".o" files or to invoke the "clean" target of the Makefile.)

  When the build completes, the executable file ecs.x file is reloaded into the Debugger, and you are notified of the load.

  Also, note that if the Static Analyzer is active, it rereads the static analysis files.

Step 7. Use the AxLS Branch Validator

If the AxLS Branch Validator product is installed, you can configure your executable for branch analysis. If this product is not installed, go on to the next step.

1 Select the **Make BBA** action key.

This action key changes the dates on the ".c" files and sends a message to the SoftBench Build tool to build the target ecs.bba file (which creates a branch analysis version of the ecs.x executable file). (A better software development practice would be to remove the ".o" files or to invoke the "clean" target of the Makefile.)

The new ecs.x executable file is loaded after the build is complete.

2 After the load of the executable file, choose the Debugger's **File→Store→BBA Data...** command, enter the file bbadump.data, and click on the "OK" button.

First, this command copies the branch analysis data to a file named "bbadump.data". Then, a SoftBench message is sent to the Branch Validator to display a histogram of Branch Coverage using this data file. This first histogram should all be zero's because you have not started running the executable program.

3 Run the executable by choosing the Debugger's **Execution→Run→From Transfer Address** command.

4 After a second or two, break the execution using a <CTRL>c.

On some Debugger/Monitors, you may have to press the target Abort button to break execution.

5 Again, dump the branch analysis data by choosing the Debugger's **File→Store→BBA Data...** command, entering the file name bbadump.data, and clicking on the "OK" button.

Notice that the histogram reflects the percent of branches executed in the various functions of the program.

Step 8. Use the Software Performance Analyzer (HP 64700 emulation system only)

If you have the Software Performance Analyzer installed, you should be able to start the Performance Analyzer interface and directly make performance measurements. If the Software Performance Analyzer is not installed, go to the next chapter.

1. Click on the "Start Additional..." button in the Debug64700 window. Select the "Performance Analyzer" interface, and click on the "Start" button.

2. With the Performance Analyzer interface running and the emulator running the user program, click on the **Func Duration** action key to obtain a Function Duration histogram.

   If you are using the Motorola 68040 microprocessor, you must compile your program with the marker preprocessor first. To do this, click on the Make Mark action key to invoke the marker preprocessor before you enter the Performance Analyzer.

   Refer to the *Software Performance Analyzer User's Guide* or online help in the Software Performance Analyzer for a complete description of the advantages of using markers.

2

Setting Up Your Project for Debug64700

Setting Up Your Project for Debug64700

This chapter describes the steps you need to take to set up your project so that it works correctly with the Debug64700 interface.

This chapter shows you how to:

1. Compile your programs for Static Analysis.
2. Set X resources for the new action keys.
3. Verify that SoftBench is properly configured.
4. Start the Debug64700 system.

After you have performed these steps, you can select and start an HP 64700 system and use it with SoftBench as was shown in Steps 2 through 8 of the "Getting Started" chapter.

Step 1. Compile your programs for Static Analysis

The Static Analyzer can work in two modes:

- Source File Mode. The mode uses source files and does not require any compiling. This mode is simpler to use at the expense of limited Static Analysis features. In particular, the Static Analysis features of Global Variables, Function Local Variables, Function Parameters, and Variable Modification cannot be shown.
- Object Mode. This mode allows the full features of the Static Analyzer to be used. To enter this mode you must compile your files in a way to create ".q" Static Analysis files.

If you are using the HP AxLS or the MRI C Cross compilers, you can create the ".q" Static Analysis files by invoking a special static compile script called XXXXXXstatic in place of your standard compile command (where XXXXXX is your standard compile command).

If you are not using the HP AxLS or MRI C cross compilers, you will not be able to create these Static Analysis files, and you will have to rely upon Source files. (Note that, as an independent exercise, you could create your own static analysis scripts for your compiler.)

1 Edit the Makefile and change XXXXX to XXXXXstatic.

For example, if you are using the HP AxLS compiler cc68000, replace cc68000 with cc68000static.

2 Touch all of the ".c" files ("touch *c") so they will be recompiled.

3 Finally execute the "make <target>" command to create a new executable file and generate the static analysis files at the same time.

Step 2. Set X resources for the new action keys

Action keys in the Debugger/Emulator, Emulator/Analyzer, and Performance Analyzer interfaces are defined by setting X resources. The Debug64700 demo files include a template for setting up SoftBench-related action keys in these interfaces.

1 Copy the "Xdefaults.template" file to your home directory.

```
$ cp
$HP64000/demo/debug_env/debug64700/Xdefaults.template
$HOME/. <RETURN>
```

2 Edit the "Xdefaults.template" file and replace all occurrences of the string "INSERT_EMULATOR" with the processor name of the emulator that you are using (for example, m68000 for the HP 64742/3/4 emulators).

Remember, the last action key must not have a line continuation character, "\", on the line. (Refer to your emulator manual for a complete description of action key format.)

In addition, you may need to change the Make action keys to properly make your executable. For more complete details, refer to the "The SoftBench Builder" section in the "Debug64700 Interaction with Other SoftBench Tools" chapter.

Save your changes and exit the editor.

3 If the RESOURCE_MANAGER property exists (as is the case with HP VUE — if you're not sure, you can check by entering the **xrdb -query** command), use the **xrdb** command to add the resources to the RESOURCE_MANAGER property. For example:

```
$ xrdb -merge -nocpp $HOME/Xdefaults.template <RETURN>
```

Otherwise, if the RESOURCE_MANAGER property does not exist, append the temporary file to your $HOME/.Xdefaults file. For example:

```
$ cat $HOME/Xdefaults.template >> $HOME/.Xdefaults
<RETURN>
```

Step 3. Verify that SoftBench is properly configured

SoftBench must be properly configured to understand the Debug64700 tool.

1 Verify that the Debug64700 tool is listed in your $HOME/.softinit file.

If not, copy the description of the Debug64700 tool from the system softinit file (/usr/softbench/config/softinit).

The Debug64700 tool must be the last DEBUG tool in your .softinit file (put it at the end of the .softinit file).

2 Start SoftBench in the directory where your demo is located. You can start SoftBench by entering the **softbench** command.

Step 4. Start the Debug64700 system

- Double-click on the executable file list in the DM.

  For example, if you are using the Debug Environment demo, you can double-click on ecs.x to start the Debug64700 system.

  If the type of executable file does not say "Target Executable", you must use the command line alternate method (described below) to start Debug64700.

  All files ending with ".x" or ".X" are considered to be "Target Executables" for the HP 64700 systems. Make sure you create executable files with ".x" or ".X" extensions when compiling programs for the HP 64700 systems.

- Or, enter the **debug64700 -file <executable_file>** command.

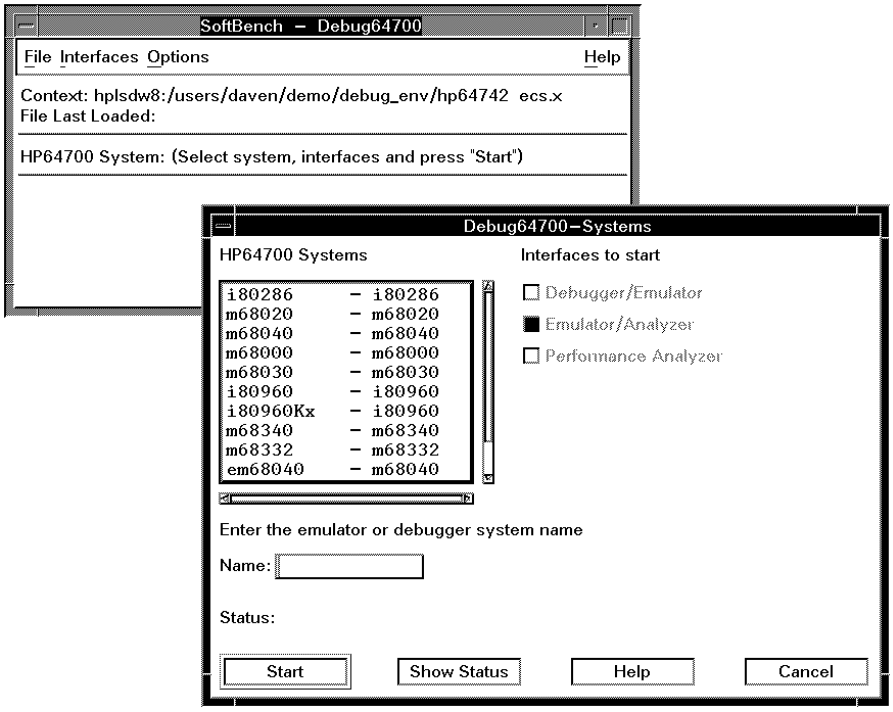
  For example, if you are using the Debug Environment demo, enter:

  ```
  $ debug64700 -file ecs.x <RETURN>
  ```

- Or, choose the **Tool→Start** command from the SoftBench Tool Manager window, and select and start the "DEBUG" tool using the Start dialog box.

  Make sure you set the context to the directory and the file to the appropriate values for your executable.

  Once the Debug64700 system is started, you should see start up windows similar to those shown in the following figure.



Refer to the man page debug64700.1 for additional details about starting the Debug64700 system.

Part 2

User's Guide

A complete set of task instructions and problem-solving guidelines, with a few basic concepts.

3

Controlling the Debug64700 Interface

Controlling the Debug64700 Interface

This chapter shows you how to:

- Start the Debug64700 interface.
- Change the target executable (context).
- Select an HP 64700 system.
- Select the interfaces to start.
- Specify a configuration file.
- Specify interface startup commands or command files.
- Set the load options.
- View the interface status.
- Start additional interfaces.
- Access Debug64700 online help.

To start the Debug64700 interface

- Enter the **debug64700** command in a terminal window.

  To start the Debug64700 interface from a terminal interface with a Target Executable file selected, enter a command like debug64700 -file ecs.x.

  For a complete list of additional options, enter the debug64700 -help command.

- Double-click on a Target Executable file in the SoftBench Development Manager.

- In addition, the interface can be started by a LOAD SoftBench message sent to the DEBUG tool, provided that the message has a file context that is a Target Executable (a file ending in .x or .X). This Target Executable file then becomes the context of the DEBUG64700 tool and, in addition, becomes the executable file that is loaded into the HP 64700 system.

To change the target executable (context)

- When the Debug64700 tool is started, but before the HP 64700 system is started (before you click on the "Start" button), choose the **File→Set Context...** command in the Debug64700 window.

  The context is the context of the DEBUG64700 tool and also reflects the name of the Target Executable file that is loaded when the HP 64700 system is first started.

  The load options commands control the actual loading of the Target Executable and may actually cause the file to never be loaded.

  If the file portion of the context shown is a number and not a Target Executable file, then a Target Executable file has not been specified and the context will be this directory and the number specified. Also, in this mode no file will be loaded.

  If you enter a file called "NONE" the context will revert to a number and the system will not load a Target Executable file.

To select an HP 64700 system

When the Debug64700 tool is started, the Debug64700—Systems dialog box is opened. You can select an HP 64700 system in the following ways.

- **Single-click on the desired system in the list of displayed systems.**

  This list is a summary of the systems described in the 64700tab.net file.

  If the list of HP 64700 systems is out of date, you can quickly edit the 64700tab.net file by choosing the File→Edit HP64700 Systems.... command. After you edit the file to make any necessary additions or deletions, you can simply close the edit window and the list of systems will be updated.

- **Enter the system name in the "System" box and press return.**

  After selecting a system, the status of the system will be updated and shown on the "Status" line.

  If the system is locked to yourself on the current host, you will be able to connect to it without unlocking it. If the system is locked to another user, you will not be able to connect to it until you unlock it. Because unlocking a HP 64700 system may cause another user to lose a valuable measurement, check with the user before you unlock their session. To unlock a HP 64700 system, click on the "Unlock" button.

  If a system is unusable (HP64700 I/O error), verify that the system has power and that it is connected through the LAN or a serial port.

  To obtain the status of a system, enter the system name in the "System" box and click on the "Status" button.

To select the interfaces to start

When the Debug64700 tool is started, the Debug64700—Systems dialog box is opened. After the HP 64700 system is selected, interfaces that are available to be started become selectable.

If you select a Debugger/Monitor target, only the Debugger/Monitor interface will be selectable.

- Click on the check boxes for the interfaces you wish to start.

  If the system is locked to yourself, you will be able to start interfaces on the locked system. If the interface is locked to someone else, you will not be able to connect to it.

  The Emulator/Analyzer interface is preselected by default. This can be changed by setting application resources (refer to the "Debug64700 Application Resources" chapter).

  If you did not purchase the Emulator/Analyzer interface, this selection will be half-toned and unselectable. If this condition exists, select the Debugger/Emulator interface.

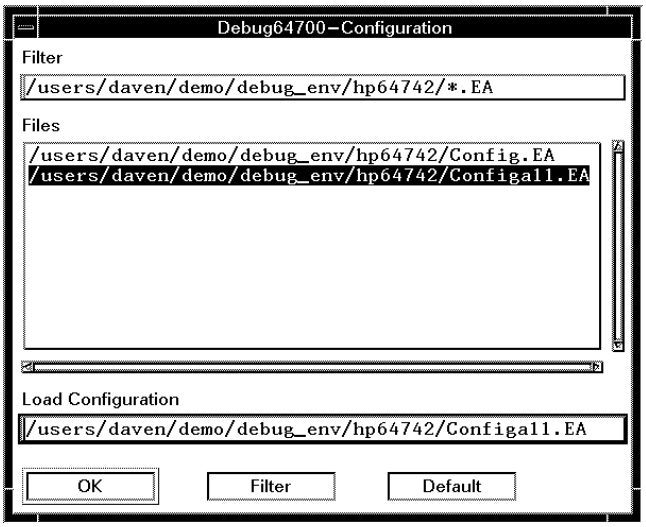
  If you have a Software Performance Analyzer, you can also select that interface.

  Additionally, if you have the RTOS measurement tool set up you will be able to select the RTOS Measurement Tool and the RTOS Performance Analyzer. You will not be able to select both the Performance Analyzer and the RTOS Performance Analyzer.

To specify a configuration file

1 After the HP 64700 system and the interfaces to be started are selected, you can specify a configuration file by choosing the **Options→Emulator Config...** command. (This menu item is only available before you start the HP 64700 system.)

If you have selected a Target Executable, you will automatically be asked to select an emulator configuration file. A dialog window will appear as shown below.



This dialog box will automatically be opened if you start an interface without specifying a configuration file.

A Debugger/Monitor will only allow you to select configuration files with .MA extensions. When using a Debugger/Monitor, you can typically click on the "Default" button to select the default configuration file. You can also specify an application resource to be DEFAULT and avoid this dialog.

2 Select the line containing your desired configuration file, or type in the name of the configuration file, and click the "OK" button.

This configuration file will be loaded into the HP 64700 system as the interface(s) are started.

If you click the "Default" button, the HP 64700 system will load the default configuration file.

To change the list of shown configuration files to another directory, you can enter the new directory in the filter window and click on the "Filter" button.

Double-clicking on a configuration file will select the configuration file and choose "OK" in one step.

To specify interface startup commands or command files

1 Choose the **Options→Start Commands...** command from the Debug64700 window.

2 In the First_Start_Commands dialog box, enter the commands or command files to be executed as the interfaces start.

Examples

For example, if you have selected the Debugger/Emulator, Emulator/Analyzer, and Performance Analyzer interfaces, you could enter the following commands.

To run the program in the Debugger/Emulator interface, enter the command:

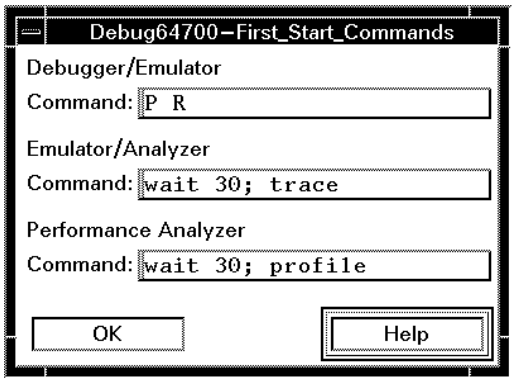
```
P R
```

To start a trace in the Emulator/Analyzer interface, enter the commands:

```
wait 30; trace
```

To start a profile in the Performance Analyzer interface, enter the commands:

```
wait 30; profile
```



Assuming you have selected a Target Executable and an emulation configuration file, these commands cause the following things to happen after clicking on the "Start" button.

- All three interfaces start.
- The Debugger/Emulator interface loads the emulator configuration and the Target Executable.
- The Debugger/Emulator then responds to its "P R" command and starts running the Target Executable.
- The Emulator/Analyzer waits 30 seconds and then executes a trace. The 30 second delay is needed to allow sufficient time for the Debugger/Emulator to initialize and complete the above loading. A delay of more than 30 seconds may be needed if you have a large Target Executable to load, or if you are running on a slower system.
- The Performance Analyzer waits 30 seconds and then executes a profile.

You can also enter the names of command files to execute when the system starts.

Remember, the commands or command files must contain commands in the syntax of the interface to which they apply. The only exception to this rule is the case of a single command file to the Debugger/Emulator. If you enter a command file for the Debugger/Emulator to execute, the Debug64700 interface will check for the existence of the command file and add the needed "F C " to it so that the command file will properly execute in the interface. If you want to run two command files at the start of the Debug64700 interface, you must enter a command like that shown below.

To run a single command file in the Debugger/Emulator:

```
comamndfile1.com
```

To run multiple command files in the Debugger/Emulator:

```
F C comamndfile1.com;; F C commandfile2.com
```

You can also specify application defaults to enter these commands. Refer to the "Debug64700 Application Resources" chapter.

To set the load options

1 Choose the **Options→Load Options...** command from the Debug64700 window.

2 In the Load_Options dialog box, select the appropriate check boxes.

This window allows you to define what happens to the Target Executable in response to system load messages. SoftBench LOAD messages can be generated in a variety of ways, such as double-clicking on a Development Manager Target Executable file, or the completion of a Build, or a command from the Build tool to debug the target. In any case, the response that Debug64700 takes from these LOAD messages is determined in this window.

If you select "Never Load" then all LOAD messages will be ignored.

If you select "Load Executable at Start Only" or "Load Symbols at Start Only", the load will only occur when the HP 64700 system is first started. All LOAD messages after the start will be ignored.

If you select "Load Executable" or "Load Symbols", every load message will generate the requested load.

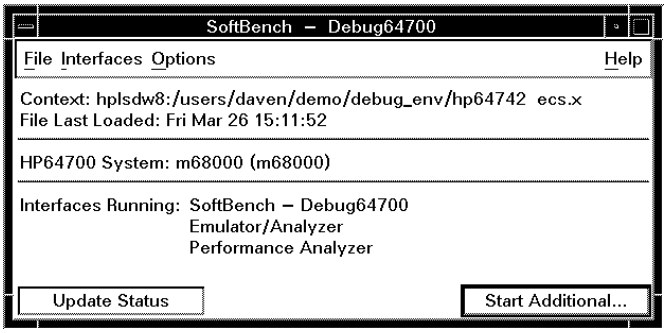
The "Load Notification" will inform you when a load has completed. Turning this option off will disable this notification.

To view interface status

If you exit an interface or start an interface of a different type than what was started initially, the interface status may become out of date. You can get a fresh update of which interfaces are running in the following ways.

- Choose the **Interfaces→Update Status** command from the Debug64700 window.
- Click on the "Update Status" button in the Debug64700 window.

After pressing the "Start" button, the user interfaces requested will take a few seconds to start. The interfaces running will be shown in the Debug64700 window, similar to the window shown below.



The interface status shown cannot distinguish between Emulation/Analyzers and RTOS Measurement Tools, nor can it distinguish between Performance Analyzers or RTOS Performance Analyzers. In addition, the status cannot display the number of Emulation Analyzer interfaces that are started.

The status reflects, instead, whether any of the interfaces are running.

You should not attempt to view the interface status while the interfaces are starting because, as they are initializing, the interfaces will not properly respond to the status request, and you will not see valid status results.

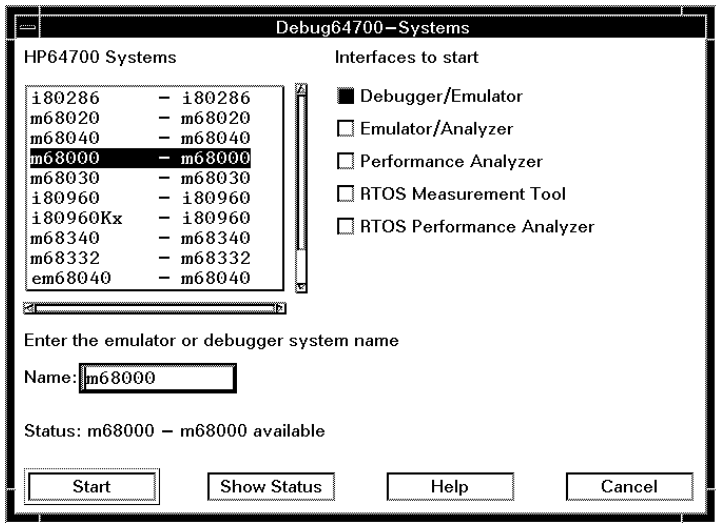
To start additional interfaces

1 Choose the **Interfaces→Start Additional...** command from the Debug64700 window.

Or:

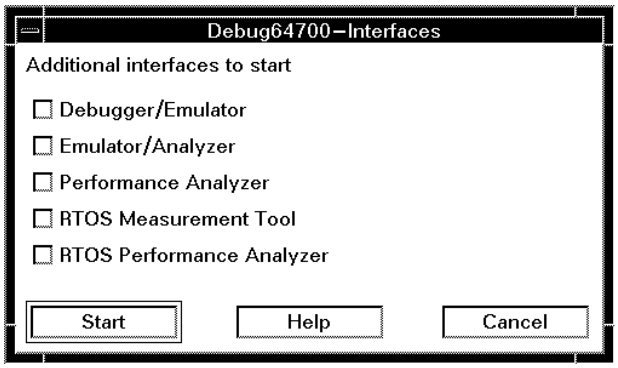
Click on the "Start Additional..." button in the Debug64700 window.

Before starting the HP 64700 system, this command will open the Systems dialog box (shown below) that is used to select an HP 64700 system, interfaces, and start the HP 64700 system.



If you do not select "Start" from this interface, the Debug64700 interface will not complete the connection to SoftBench.

After pressing the "Start" button to start the HP 64700 system, this command will now allow you to select additional interfaces to start. The Interfaces dialog box is shown in the following figure.



2 In the Systems or Interfaces dialog box, select the appropriate check boxes for the additional interfaces you want to start.

3 Simply select the interfaces to start and click on the "Start" button.

Interfaces that are half-toned out are either not available or already running. Additional interfaces of that type cannot be started.

After you click on the "Start" button, allow a few seconds for the interfaces to appear.

To access Debug64700 online help

- Choose the **Help→Application Help** command from the Debug64700 window.
- From any of the SoftBench windows, choose the **Help→List Host Topics** command, and select the "Debug64700 Overview" topic.
- Position the mouse pointer over the Debug64700 window or menu bar and press the F1 key (or "Help" on a Sun keyboard).

  For more information on using online help, refer to the "Getting Online Help" chapter in the *SoftBench Overview Manual*.

4

Debug64700 Interaction with Other SoftBench Tools

Debug64700 Interaction with Other SoftBench Tools

This chapter describes the interaction between Debug64700 and other SoftBench tools:

- The SoftBench Editors.
- The SoftBench Builder.
- The SoftBench Static Analyzer.
- The HP Branch Validator.

The SoftBench Editors

The Debug64700 tool will allow Edit commands issued from the Debug Environment to invoke a SoftBench Editor. This is an automatic operation that happens any time the Debug64700 interface is running.

For proper interaction between the Debug64700 tool and the SoftBench Editors, you must only use the TOOL based editors. The COMMAND based editors must NOT be used.

Examples

To invoke the SoftBench editor:

Load a Target Executable file into the Debugger/Emulator or Emulator/Analyzer.

Display the memory at the location of the function "main".

Use the pop-up edit command to edit the file at the line that starts the main routine. This will open the SoftBench Editor.

Now use the pop-up edit command on a different line in the same file. This should only move the cursor in your edit window.

The SoftBench Builder

The Debug64700 tool will submit Build requests to the SoftBench Build tool in response to a Build request from the Debug Environment interfaces.

To simplify the process of requesting builds, an action key that submits the Build request should be defined. An Emulation/Analyzer Build action key may look like this:

```
"Build" "forward bms 'SoftBench_Build_Target ecs.x - NONE
HP64000=$HP64000'"
```

This key requests a Build of the target file ecs.x. In addition, it passes the Compile Mode options of "NONE" and the Build Options of "HP64000=$HP64000" to the Makefile in the current working directory.

Refer to the Xdefaults.template file for additional examples of action keys that can be used to submit Build requests.

When the Build is complete, the executable file will be reloaded as specified by the load options of the Debug64700 interface.

Build Compiler and Assembler Error Messages

The SoftBench Builder has the capability of directly editing the source files where compile errors have occurred. To activate this feature simply double-click on the error in question. Then, use the "Next" and "Previous" buttons in the SoftBench Build window to navigate through your errors.

If you are using the HP AxLS C Cross Compilers or the MRI C or C++ Cross Compilers, this error navigation should work after the Debug64700 product is installed. If you are using a different compiler, you can add the error navigation by editing the builderr.fmt file. Refer to the SoftBench softbuild (1) man page for complete details.

If you are using the HP 64000 or MRI Intel family assemblers (as86, asv20, asv33, asm86) to assemble sources, you may want to improve the format of the error messages by invoking the contributed assembly scripts located in /usr/hp64000/contrib/bin. These scripts modify the assembler error messages into a format that is more suitable for the SoftBench Build tool to handle. Refer to the man page /usr/hp64000/contrib/man/man1/as86err(1) for complete details.

Using the HP AxLS Makefile Templates

Templates to help create Makefiles exist in a couple of different directories.

First, the HP AxLS C cross compilers include templates. These templates are located in the directory /usr/hp64000/lib/mf. This directory contains files CXXXXX.x.p (where XXXXX is the product number of your compiler) which can be used to help build a Makefile. These templates provide a minimal set of features for each of the AxLS C cross compilers.

A second set of templates exists in the /usr/hp64000/inst/debug64700/config directory. This set of templates, although not unique for a specific AxLS compiler, are more complete in providing a much greater number of functions. In order to use these templates with the SoftBench Builder, follow the steps below:

1 Set the following X resources:

```
Build.progTemplateFile: /usr/hp64000/inst/debug64700/config/buildt.p
Build.libTemplateFile: /usr/hp64000/inst/debug64700/config/buildt.l
Build.mkmfDefaultProgName: a.out.x
Build.mkmfDefaultLd: cc68000
```

You may want to copy the templates to a location where you can edit and customize them for your development environment; if you do this, you should modify the X resources settings so they to point to the new template locations.

2 Create a Makefile for the files in the debug environment demo.

First copy the source files, include files, and linker command file of the debug environment demo to a directory of your choosing. Make sure that you do not copy the Makefile.

```
$ cd /usr/hp64000/demo/debug_env/hpXXXXX <RETURN>
```

(Where XXXXX is your emulator product number.)

```
$ cp Linkcom.k *.c *.h /users/daven/examples <RETURN>
```

Also, if you plan on running the built Target Executable, you will want to copy the configuration file Configall.EA to this directory.

```
$ cp Configall.EA /users/daven/examples <RETURN>
```

3 Start the Build tool in the directory that you have chosen.

After the Build tool initializes, choose the Makefile→Create Program... command. In the dialog window that appears, set the appropriate values for the PROGRAM and LD shell variables.

```
PROGRAM=ecs.x
LD=cc68000
```

If you are using a different AxLS C compiler than the 68000 compiler, specify LD=ccXXXXX and in the Add'l Options field CC=ccXXXXX (where XXXXX is your AxLS C cross compiler). Now click on the "OK" button.

4 Start the Build by clicking on the "Build" button to see if you have successfully created a new Makefile.

Change the target to "help" and again click on "Build" to get a listing of the available targets that can be entered.

5 As a last test, start up your emulator and verify that the program runs.

Use the command "debug64700 -file ecs.x," or double-click on the Development Manager file ecs.x. Do not use the "-demo" option, because not all of the demo files needed by the demo have been copied into this directory.

Note: If you were previously running the the Debug64700 demo (Startdebug64700), you must exit SoftBench and restart it so the "-demo" option does not impact this test.

The SoftBench Static Analyzer

The SoftBench Static Analyzer adds a variety of additional capabilities to the Debug Environment. To completely make use of these additional capabilities, you need to compile your files specifically for Static Analysis and add action keys to the Debug Environment interfaces.

Compiling your files for the SoftBench Static Analyzer is accomplished by using a SoftBench C scanning tool called "cscan" tool. This tool creates static analysis (".q") files from your source files.

HP AxLS C and MRI C and C++ wrapper compile scripts have been generated to simplify the process of invoking the cscan tool. To invoke these scripts, simply invoke your HP AxLS cross compiler with ccXXXXXstatic instead of ccXXXXX. In the same way, you can invoke your MRI mcc68k, mcc86, mcc960 or ccc68k compiler with mccXXXstatic to generate static analysis files (.q) for use with the Static Analyzer. Refer to the man pages ccXXXXXstatic.(1), mccXXXstatic.(1) and ccc68kstatic (1) for complete details.

If you are using a compiler different than the ones listed, you can still use the Source Mode of the Static Analyzer and obtain most of the static analysis information.

The Debug64700 tool will submit Static Analysis messages to the SoftBench Static Analyzer in response to Debug Environment interface requests for static analysis. The easiest way to invoke the Static Analyzer from the Debug Environment is to create action keys that submit the Static Analyzer requests. Example static analyzer requests for the Emulator/Analyzer appear like this:

```
"Functions" "forward bms 'SoftBench_Message 5 Request STATIC
SHOW-FUNCTIONS * NOT_QUIET'"

"Calls In ()" "forward bms 'SoftBench_Message 6 Request STATIC
SHOW-CALLS-WITHIN * NOT_QUIET ()'"

"Graph ()" "forward bms 'SoftBench_Message 8 Request STATIC
GRAPH-TOKEN * () - - -'"
```

In addition to sending the Static Analyzer request, the Debug64700 tool will transmit a SET-DEBUG-CONTEXT message informing the static analyzer of the context from which the DEBUG tool is running. This allows the Static Analyzer, in return, to send set breakpoint requests to the DEBUG tool, which in this case is the HP 64700 debug environment.

The HP Branch Validator

If you have purchased the HP Branch Validator, unloading the Branch Validator data from your Debugger/Emulator or Emulator/Analyzer will automatically invoke the HP Branch Validator.

Two messages will be sent out with each unload. The first message informs the HP Branch Validator of the name of the file where the Branch Analysis information was dumped. The second message instructs the Branch Validator interface to update its display.

The second message can be controlled by an application resource that instructs the HP Branch Validator to issue a particular display command. Options include showing a histogram, a file summary, or just the branch coverage results. Refer to the "Debug64700 Application Resources" chapter for more information.

Part 3

Reference

Descriptions of the product in a dictionary or encyclopedia format.

5

Debug64700 Application Resources

Debug64700 Application Resources

You can preconfigure the Debug64700 interface and its interaction with SoftBench by setting the application's X resources.

To change these application resources, copy them from the /usr/hp64000/lib/X11/app-defaults/Debug64700 file to your home directory, modify them, and set them the same way that X resources for actions keys were set in the "Setting Up Your Project for Debug64700" chapter.

This chapter describes the most common user-modifiable X resources for the Debug64700 application:

- Debug64700*interfacePreselect*String:
- Debug64700*systemName*String:
- Debug64700*configFile*String:
- Debug64700*debugCommand*String:
- Debug64700*emulCommand*String:
- Debug64700*perfCommand*String:
- Debug64700*loadOptions*String:
- Debug64700*loadNotification*String:
- Debug64700*autoStart*String:
- Debug64700*bbaOptions*String:

Debug64700*interfacePreselect*String: Emul

This resource preselects the Emulator/Analyzer interface. By specifying one or more of: Debug, Emul, Perf, RTOSMeas, RTOSPerf, you can preselect the Debugger/Emulator, the Emulator/Analyzer, the Performance Analyzer, the RTOS Measurement Tool, and the RTOS Performance Analyzer, respectively. You can preselect two or more interfaces by specifying them together on this application default (...String: Emul Debug).

Debug64700*systemName*String: em68000

This resource defines the system name uniquely identifying an HP 64700 emulator configured in the /usr/hp64000/etc/64700tab.net file.

Debug64700*configFile*String: c68000.EA

This resource specifies the configuration file to be loaded when the HP 64700 system is started. If the value specified is "DEFAULT," the default configuration file for the HP 64700 system will be loaded. If this resource is not defined, you will be prompted to supply a configuration file.

Debug64700*debugCommand*String: debuggercommand.com

This resource specifies the command files or commands to be sent to the Debugger at the initial or first start of the HP 64700 system. The syntax of the commands or command files must conform to the syntax of the Debugger. A command file will be converted to "F C debuggercommand.com" to properly match the required syntax of the Debugger.

Debug64700*emulCommand*String: emululatorcommand

This resource specifies the command files or commands to be sent to the Emulator/Analyzer at the initial or first start of the HP 64700 system. The syntax of the commands or command files must conform to the syntax of the Emulator/Analyzer.

Debug64700*perfCommand*String: performancecommand

This resource specifies the command files or commands to be sent to the Performance Analyzer at the initial or first start of the HP 64700 system. The syntax of the commands or command files must conform to the syntax of the Performance Analyzer.

Debug64700*loadOptions*String: Executable

This resource specifies the HP 64700 load operation to be performed in response to SoftBench LOAD messages. By default the last loaded Target Executable is reloaded in response to a load message. Other options include Executable_start, Symbols, Symbols_start, or Never. The Executable_start and Symbols_start options will only load at the first start of the HP 64700 system. The Never option will cause the HP 64700 to ignore SoftBench LOAD messages.

Debug64700*loadNotification*String: True

This resource specifies that a load notification dialog should appear in response to a SoftBench LOAD message which has caused the HP 64700 system to load an executable or symbol file.

Debug64700*autoStart*String: False

The auto start resource, if set to "True," will cause the Debug64700 interface to use the predefined system and interfaces and to start the interfaces immediately. The "Start" button does not need to be clicked to make the connection. You should have the interfaces, configuration file, and system name predefined (and verify that they work correctly) before you set this option.

Debug64700*bbaOptions*String: SHOW-HISTOGRAM

This resource will specify the type of Branch Validator display to present after you execute a File→Store→BBA Data command. Available options include SHOW-HISTOGRAM, SHOW-SUMMARY, SHOW-RESULTS-ONLY. You must have the HP Branch Validator installed before this resource will become operational.

6

SoftBench Messages Supported

SoftBench Messages Supported

The Debug64700 interface can send out a variety of messages to the SoftBench environment. These messages are sent in response to forwarded commands from the Debug Environment interfaces (Debugger/Emulator, Emulator/Analyzer, and Performance Analyzer) to the Debug64700 tool. The forwarded commands are then translated into the appropriate SoftBench messages.

For example, the command 'forward bms "SoftBench_Edit_File abc.c"', when entered on the command line of a Debug Environment interface, will be translated by Debug64700 into a SoftBench Message like this: "Request EDIT WINDOW current_host current_directory abc.c".

The commands discussed in the following sections can be forwarded to the Debug64700 tool to generate SoftBench Messages.

The Debug64700 interface can also receive a variety of SoftBench messages. These messages are discussed in the section titled "SoftBench Messages Received."

SoftBench Edit Commands

SoftBench_Edit
SoftBench_Edit_File <file>
SoftBench_Edit_File_Line <file> <line number>

Each of the above Edit commands will issue a EDIT WINDOW SoftBench message. <file> is the specified file to edit, and <line number> is the line number to position the cursor. Without a file specified, a file called new.# will be opened in the current working directory. If a full path filename is entered, the directory context will be properly specified. If the full path file name starts with the string "/nfs/", a host context will also be set. If just the basename of the file is passed, the current working directory will become the host and directory context for the file.

Example: forward bms "SoftBench_Edit_File_Line /tmp/junk 35"

SoftBench Build Commands

SoftBench_Build
SoftBench_Build_Target <target>

Each of the above Build commands will issue a BUILD BUILD-TARGET SoftBench message. <target> is the specified target file to build. The current working directory will become the host and directory context where the build is to occur.

After the Build is complete, the Debug64700 tool, in response to the Build complete message, will reload the currently loaded absolute or symbol file into the Debug Environment. If the Debugger/Emulator interface is running, the load command will be passed to that interface. If the Debugger/Emulator interface is not running, the load command will be passed to the Emulator/Analyzer interface. If neither interface is running, an error message will be issued.

Example: forward bms "SoftBench_Build"

SoftBench BBA Command

SoftBench_BBA <file>

The SoftBench_BBA command is used to start up the HP Branch Validator. The SoftBench_BBA command will issue two SoftBench messages. The first message sets the Branch Validator dump file to <file>. The second message requests that a Branch Validator Histogram be displayed.

Example: forward bms "SoftBench_BBA bbadump.data"

SoftBench Generic Message Commands

SoftBench_Message 3 <type> <class> <command>
SoftBench_Message 4 <type> <class> <command><file>
SoftBench_Message 5 <type> <class> <command> <file> <data>
SoftBench_Message 6 <type> <class> <command> <file> <data> <data>
SoftBench_Message 7 <type> <class> <command> <file> <data> <data> <data>
SoftBench_Message 8 <type> <class> <command> <file> <data> <data> <data> <data>

Each of the above SoftBench_Message commands are generic message commands that can be user-defined to send almost any SoftBench message. Users can generate their favorite SoftBench Messages and place them upon Debug Environment action keys. Then, in response to pressing an action key, a SoftBench Message will be generated. The command that should be used depends upon the number of parameters that will be passed.

A SoftBench_Message 3 instructs Debug64700 to make a message consisting only of <type>, <class>, and <command>. The <type> field must be one of "Request", "Notify", or "Failure". The <class> field must be the message class of the SoftBench Tool (for example, EDIT, DM, STATIC ...). The <command> field must be the SoftBench message command for the message class. The <file> field is the specified file to act upon. If a full path filename is entered, the directory context will be properly specified. If the full path file name starts with the string "/nfs/", a host context will also be set. If just the basename of the file is passed, the current working directory will become the host and directory context for the file. The <data> fields are the data items to be passed to the associated commands.

With this generic command structure almost any SoftBench message can be generated and sent to the SoftBench Message Server. For example, this message command will invoke the Static Analyzer and have it graph the function main.

Example: forward bms "SoftBench_Message 8 Request STATIC GRAPH-TOKEN * main - - -"

The Debug Environment will issue a variety of forwarded messages (if the Debug64700 tool is running) in response to menu selections or pop-up menus. These include editing files and invoking the Branch Validator. Other forwarded commands can be added by the addition of action keys. As a

mechanism for learning the possibilities, examine the Xdefaults.template in the demo directory "/usr/hp64000/demo/debug_env/debug64700". This file shows a variety of Action keys that can be added to your debug environment to add Static Analysis and Build messages.

SoftBench Messages Received

The Debug64700 interface can receive a variety of SoftBench messages and then forward appropriate commands to the Debug Environment interfaces. In order for these messages to be received, the context of the message must match exactly (Host Directory File.x) the current context of the Debug64700 interface. Note that the context must include File.x or File.X; otherwise, the command will invoke the Host Debugger.

Request DEBUG LOAD
Notify BUILD BUILD-TARGET

These messages direct Debug64700 to reload the last loaded absolute. The BUILD-TARGET message is accepted if the host and directory match the current working directory. The file context of this message is ignored.

Request DEBUG LOAD-CONFIGURATION file

This message directs Debug64700 to load a configuration file. File must be the full path of the configuration file.

Request DEBUG BP-SET - file line
Request DEBUG BP-SET-PROC - file procedure

These messages direct Debug64700 to set breakpoints.

Request DEBUG BP-CLEAR - file line
Request DEBUG BP-CLEAR-PROC - file procedure

These messages direct Debug64700 to clear breakpoints.

Request DEBUG RUN
Request DEBUG RUN-FROM-RESET
Request DEBUG RUN-FROM-TRANSFER-ADDRESS
Request DEBUG STEP
Request DEBUG STEP-OVER
Request DEBUG STEP-OUT
Request DEBUG BREAK
Request DEBUG RESET-TO-MONITOR

These messages direct Debug64700 to run, step, break, and reset the debugger or emulator.

Request DEBUG DEBUG-COMMAND-FILE command_file
Request DEBUG EMUL-COMMAND-FILE command_file
Request DEBUG PERF-COMMAND-FILE command_file

These messages direct Debug64700 to send a command file or an individual command to the specified interface. If the specified interface is not running, the command will be ignored.

Request DEBUG EXIT-LOCKED
Request DEBUG EXIT-RELEASED

These messages direct Debug64700 to exit the debug environment session.

7

Common Problems and Solutions

Common Problems and Solutions

This chapter provides information about some of the common problems you may encounter when using the Debug64700 tool:

- Cannot set breakpoint from Static Analyzer to Debug Environment.
- Executable file doesn't load.
- What does the space in the context mean?
- Can I load multiple executables into the Debug Environment?
- Why does the system ask for a configuration file?
- Where does the executable file and configuration file come from when running remote?
- What is the advantage of using the supplied compile scripts to generate Static Analysis object files?

Cannot Set Breakpoint from Static Analyzer to Debug Environment

Problem: The Static Analyzer will not allow me to set a breakpoint to the Debug Environment, why?

Solution: The Static Analyzer needs to know the context of the DEBUG tool (Debug64700) before it can send a breakpoint to it.

The easiest solution is to press an action key in the Debugger/Emulator or the Emulator/Analyzer to send a Static Analyzer command to the Static Analyzer. Sending these commands will send the DEBUG context to the Static Analyzer.

A second method of setting the DEBUG context for the Static Analyzer is to use the command Options->Set Debug Context (in the Static Analyzer). Remember to enter the context of the Debug64700 tool exactly as it is shown in the Debug64700 or the SoftBench Tool Manager Window.

Executable File Doesn't Load

Problem: Double-clicking on a Target Executable file in the Development Manager after the Debug64700 system has started, always tries to start a new Debug64700 session. It doesn't load the executable into the existing Debug64700 as I would expect.

Solution: The Development Manager will issue a DEBUG LOAD request by double clicking a Target Executable file. This request is sent out to a DEBUG tool in the exact same context as the Development Manager (Host, Directory, File.x).

When the Debug64700 tool connects to an HP 64700 system, it starts running in a context of the executable or symbol file that is loaded into the system. If the directory to the executable or symbol file is linked to another directory, the context of the Debug64700 interface may switch to the linked directory. As an example consider the following scenario:

1. The user double-clicks on the file /usr/hp64000/demo/debug_env/hp64742/ecs.x in the Development Manager.
2. The Debug64700 interface starts, and the user starts a Debugger/Emulator.
3. When the Debugger/Emulator loads the file ecs.x, it finds that the true path to it is /disc3.11/hp64000_ln/demo/debug_env/hp64742/ecs.x. The Debug64700 interface switches to this context.
4. Another double-click on the file /usr/hp64000/demo/debug_env/hp64742/ecs.x in the Development Manager starts a new Debug64700 interface (whereas what is expected is that the ecs.x file should be loaded).
5. The solution is to change the context of the Development Manager to the directory /disc3.11/hp64000_ln/demo/debug_env/hp64742. This will force the LOAD message that is generated to match the context of the running Debug64700 system. A double-click now on the ecs.x file will generate the desired load of the ecs.x executable without the start of the new Debug64700 session.

What Does the Space in the Context Mean?

Problem: What does the "space" between the directory and file portions of the context mean? Why can't I split the context in the Debug64700 interface?

Solution: The "space" between the directory and the file indicates a separation between what is considered the directory portion of the context and what is considered the file portion of the context.

Starting the Debug64700 tool with a context of "hosta:/users/daven/demo/debug_env hp64742/ecs.x" will direct the current working directory to be "/users/daven/demo/debug_env" while the file portion of the path is considered "hp64742/ecs.x". This context will remain intact until a different executable file is loaded.

When the new executable file is loaded, the context will revert to a more standard context "host:/directory/directory file.x". Debug64700 is not designed to work in a split context mode, so entering a split context into the Debug64700 tool will not be accepted. The only exception to this is by starting in a split context mode or directing the context of the Debug64700 tool to change by using the SoftBench Tool Manager.

Can I Load Multiple Executables Into the Debug Environment?

Problem: Can I loading multiple executables into the Debug Environment by clicking on the files in the Development Manager?

Solution: This is not possible at this time. The load message generated will be for a specific context which must match exactly the context of a Debug64700 system before a load command is generated.

If you need to load multiple executables into your HP 64700 system, you should define a command file to do the load sequence. This command file could be set by an application resource to cause automatic loading of the executables at all initial starts of the HP 64700 system.

Why Does the System Ask for a Configuration File?

Problem: If continuing a session only connects to the session, and no commands are executed, why does the system ask for a configuration file?

Solution: Any time a system is continued, nothing will be loaded. No configuration files, no executables, no command files are loaded. This has been added as a safety feature to prevent users from accidentally changing their locked or running HP 64700 system.

During the start of the Debug64700 system, the complete status of a system is not fully determined until after the point where a configuration file is asked for. You can ignore this request for a configuration file (click on the Default button). If you click on "Default", the configuration file will not be used, and the current configuration will still be in use.

Where Does the Executable File and Configuration File Come From When Running Remote?

Problem: When I run the Debug64700 system on a remote machine, which machine will the Target Executable be obtained from? Where will the configuration file come from?

Solution: The context of Debug64700 running on a remote machine can become a little difficult to comprehend.

The execution host is the machine where the Debug64700 tool is running.

The data host is shown in the context of the Debug64700 interface and is the location where the Target Executable and the configuration file will be searched for.

The current working directory will be set to the data host through a change directory /nfs/data_host/... command. Therefore, most commands that either don't use a file or use a relative file name will execute relative to the data host. Commands that issue a full path name will execute based upon the full path name. If the full path name contains an /nfs/data_host, the command will work with the data_host. Otherwise, the execution host will be invoked through the command.

As an example, storing a configuration specification into a file named /tmp/junk will put the results into the /tmp directory on the execution host. On the other hand, storing the configuration specification into a file named

junk in the current working directory will store the configuration on the data host in a file named /nfs/data_host/users/daven/example/junk.EA.

The execution host, in addition to supplying the execution cycles, will also be used to find the HP 64700 systems and the application resources. All files needed to make the Debug64700 interface run (without regard to what Target Executable file will be running) must, therefore, be available on the execution host.

What is the Advantage of Using the Supplied Compile Scripts to Generate Static Analysis Object Files?

Problem: What is the advantage of using the supplied compile scripts to generate Static Analysis object files?

Solution: The supplied compile scripts will generate Static Analysis object files (files with suffix .q). These object files allow you to obtain a more complete Static Analysis of the components of a Target Executable.

In particular, using these object files allows you to get the following additional capabilities beyond the capabilities of the Source Mode Static Analysis:

- Modifications()
- Global Variables
- Parameters()
- Local Variables()
- All queries on variables are supported
- Scoping is active
- Additional C++ capabilities

Refer to your *SoftBench Program Construction Tools User's Guide* for complete details.

Part 4

Installation Guide

Instructions for installing and configuring the product.

8

Installation

Installation

This chapter contains information about:

- Supported Emulator Systems and Software Versions
- Installing the Debug64700 Software
- Customization of the SoftBench Environment
- Using Debug64700 Remotely
- Adding Magic Strings
- Updating the MANPATH Environment Variable
- Installation Verification

Supported Emulator Systems and Software Versions

The Debug64700 tool will work with the following HP emulators:

HP Product No.	Description	HP Product No.	Description
64742	68000 Emulator	64761	80960Kx Emulator
64746	68302 Emulator	64762	8086 Emulator
64747	68030 Emulator	64763	8088 Emulator
64748	68020 Emulator	64764	80C186 Emulator
64749A	68331/332 ACT Mode Emulator	64765	80C188 Emulator
64749B	68332 Emulator	64767	80C18X Emulator
64749C	68331 Emulator	64780	68360 Emulator
64751	68340 Emulator	64783	68040 Emulator
64760	80960Sx Emulator		

All emulator interfaces must be version C.05.XX or greater before Debug64700 will work. Debug64700 will NOT work with terminal mode emulator interfaces

SoftBench versions A.02.1 or B.00.XX or greater must also be installed and operational before Debug64700 can be used.

If you are using SoftBench version A.02.01, then a few features of this tool will not be available. In particular, the Static Analyzer graph and setting of breakpoints from the Static Analyzer graph will not be supported.

Installing the Debug64700 Software

Debug64700 is automatically installed when you load all of the filesets of the Emulator/Analyzer or the Debugger/Emulator products. If by chance you need to select individual partitions to install, you must select the 64700-SOFTBENCH partition to obtain the Debug64700 product.

On Sun SPARCsystems make sure you install the DEBUG64700 fileset. Refer to the *Software Installation Guide* that comes with your media for complete instructions about installing a fileset from a tape.

When installing the Debug64700 product and SoftBench at the same time, make sure you install SoftBench first. If you install SoftBench last, or install a newer version of SoftBench, you must then rerun (as root) the customize script /system/DEBUG64700/customize on HP 9000 Series 300 and HP 9000 Series 700. On Sun SPARCsystems you must run (as root) the envinstall script /usr/hp64000/bin/envinstall.

Note

Rerun the Debug64700 customize script if SoftBench is reinstalled for any reason.

The customize script expects to find SoftBench installed in /usr/softbench and /usr/local/softbench. If SoftBench is installed in a different location, you must export the shell variables SOFTBENCH and LOCALSOFTBENCH before you start the install process. On Sun SPARCsystems, set these shell variables before running envinstall.

After installing the Debug64700 product, you will need to exit SoftBench and the HP Vue environment if they are currently running. The Debug64700 product supplies additional filetypes to the Broadcast Message Server. The Broadcast Message Server will not recognize the new types until it is restarted (by exiting SoftBench and Vue).

Customization of the SoftBench Environment

The Debug64700 product will automatically do a variety of operations to the SoftBench Environment as part of the customization process. You may need to take additional action to complete the customize process for the individual users of SoftBench.

- ☐ The /usr/softbench/config/softinit file is rebuilt to include Debug64700. In the process of creating the softinit file, a copy of the original softinit file is moved to /usr/softbench/debug64700/softinit.old. If for any reason you need to restore the original softinit file, make sure that you include the definition of the Debug64700 tool. You will also need to modify individual users' .softinit files to include the Debug64700 tool if they will be using Debug64700.

Note

The definition of the Debug64700 DEBUG tool must be the last DEBUG tool in your system /usr/softbench/config/softinit and $HOME/.softinit files.

You can find the information about the Debug64700 tool that is needed in the softinit file in the file /usr/softbench/config/softinitsrc/class-defaults/debug64700init. The description of the Debug64700 tool must be on one line.

- ☐ As you customize your $HOME/.softinit file, remember that you should only use the TOOL type EDIT tools. The COMMAND type EDIT tools should not be used because in many instances Debug64700 is relying upon messages that these tools do not provide.

- ☐ The list of filetypes has been updated to add additional types for the Debug Environment. A copy of the original filetypes is moved to the file /usr/softbench/debug64700/C.softtypes.old. If HP Vue was running during the install process it must be restarted on every station that will be using Debug64700 to make it recognize the new filetypes.

- ☐ The builderr.fmt file has been updated to accept HP AxLS C Cross Compiler error messages. The original builderr.fmt has been moved to /usr/softbench/config/builderr.fmt.old.

☐ Various inconsequential files have been added to the SoftBench directories:

Debug64700 application resources have been linked into the /usr/softbench/app-defaults directory.

Action keys have been added to the Development Manager for the Emul Absolute files

Help messages have been added to the appropriate directories for Debug64700 help.

Updating the MANPATH Environment Variable

The MANPATH environment variable should include /usr/hp64000/man to get HP64000 man pages. An example MANPATH may look like this:

```
MANPATH=/usr/man:/usr/contrib/man:/usr/local/man:/usr/hp6
4000/man
```

Using Debug64700 Remotely

The Debug64700 tool can be set up to run remotely. This is where you may want the execution of the Debug64700 tool to operate on a different machine than the one where you are sitting. In setting up to run the Debug64700 tool remotely, consider these points:

- ☐ You must configure your SoftBench system to operate remotely. Can you direct a remote machine to run its editor on a file on your machine? Refer to your SoftBench Installation and make sure your system is configured to operate in this mode.

- ☐ Have you installed the Debug64700 Software on the remote machine? Have you verified that it operates properly on the remote machine?

- ☐ Change your $HOME/.softinit file on the machine that you are sitting at to use the remote machine instead of the local machine like this:

```
DEBUG TOOL FILE TARGET_EXECUTABLE remote_machine debug64700 ...
```

- ☐ You may need to make a .softenv file in your $HOME directory on your remote machine. Refer to the *Installing SoftBench Products User's Guide* for additional information.

- ☐ Make a .Xdefaults or .Xdefaults-<remote machine> file in your $HOME directory on your remote machine. In this Xdefaults file you will want to include any action keys that you want to have appear with the Debug64700 interface.

Once started in a given context, the Debug64700 system will continue to use the host and directory context for all of the files needed (configuration files, symbol files, command files, etc.). This operation will continue until the directory is changed from within the Debug64700 interfaces. When the directory is changed, the Debug64700 interfaces will operate out of the current working directory. When operating remotely, care must be taken not to change the directory away from the host from where you want to obtain your files.

Examples

As an example, if you are running remotely on a machine named johnsys, the machine that you are looking at is named yoursys, and if the absolute file that you want to work with is file.x in the directory $HOME/example of yoursys, the following operation will occur:

Your execution host will be johnsys. Your Debug64700 context will be /nfs/yoursys/$HOME/example/file.x. The Debug64700 tool will search for configuration files in /nfs/yoursys/$HOME/example. Your current working directory will be /nfs/yoursys/$HOME/example.

If you now change directory to $HOME, you have now changed back to the machine named johnsys. The /nfs/yoursys/ must be the first part of the directory that you want to change to /nfs/yoursys/$HOME.

Adding Magic Strings

If you want to improve the capabilities of the file(1) command in determining the types of files, you can add more magic strings to the /etc/magic file. To do this, append the contents of the file /usr/hp64000/inst/debug64700/etc/magic to the end of the /etc/magic file. You must do this as root.

```
# cat $HP64000/inst/debug64700/etc/magic >> /etc/magic
<RETURN>
```

Now try to get the type of the file ecs.x in the debug environment directory. Enter these commands:

```
$ cd $HP64000/demo/debug_env/hp64742 <RETURN>

$ file ecs.x <RETURN>
```

The response will be like this:

```
ecs.x: IEEE-695 object module - for Motorola 68000
```

Installation Verification

To verify installation you, should check that the SoftBench customization listed above has occurred. Once this is verified, you should be able to run the demo in the "Getting Started" chapter.

Index

! $HOME/.softinit file, 37

A action key, Calls In (), 24
action key, Calls To (), 24
action key, Decl (), 24
action key, Functions, 24
action key, Glb Vars, 24
action key, Loc Vars (), 24
action key, Make BBA, 30
action key, Make Static, 29
action key, Mods (), 24
action key, Parms (), 24
action key, Refs (), 24
action key, Uses (), 24
action keys, 23
adding magic strings, 100
application defaults, 70
application resources, Debug64700, 70
assembler error messages, 62
AxLS C cross compiler, 14
AxLS C Cross compilers, 35
AxLS Makefile Templates, using, 63

B BBA command, SoftBench, 77
branch validator, using, 30-31, 66
breakpoints, set by Static Analyzer graph, 25-26
build commands, SoftBench, 76
build compiler error messages, 62
build tool, SoftBench, 29

C C Cross compilers, AxLS, 35
C Cross compilers, MRI, 35
Calls In () action key, 24
Calls To () action key, 24
cannot set breakpoint, 84
changing the Context, 46

changing the Target Executable, 46
common problems and solutions, 84
compile programs for static analysis, 35
compile script, static, 35
configuration file, specifying, 49-50
configuration file, why does the system ask for it?, 84, 87
Context, changing, 46
cross compiler, AxLS C, 14
customization of the SoftBench environment, 95-96

D debug startup dialog, 16
debug64700 -help, 45
Debug64700 application resources, 70
Debug64700 interface and the SoftBench interaction, preconfiguring, 70
Debug64700 interface, starting, 45
Debug64700 system, starting, 38-39
Debug64700, using remotely, 98-99
Debugger/Emulator interface, preselected for demo, 22
Decl () action key, 24
defaults, application, 70
demo program, restarting, 19

E edit commands, SoftBench, 75
Emulator Config..., 49
emulator systems, supported, 93
error messages, assembler, 62
error messages, build compiler, 62
executable file doesn't load, 84-85
executable file, rebuilding, 29

F File: Edit HP64700 Systems..., 47
File: Set Context... command, 46
files, where do they come from when running remote?, 87
Filter, 50
Functions action key, 24

G generic message commands, SoftBench, 78-79
getting started, 13-32
getting started without the demo, 34
Glb Vars action key, 24

H help, debug64700, 45
histogram, 30
HP 64700 system, starting, 22
HP64700 I/O error, 22
HP64700 system, selecting, 47

I installation, 92
installation verification, 101
installing the Debug64700 software, 94
interface status, 54
Interfaces: Start Additional..., 55
Interfaces: Update Status, 54

L Load Executable, 53
Load Executable at Start Only, 53
Load Notification, 53
load options commands, 46
Load Symbols, 53
Load Symbols at Start Only, 53
Loc Vars () action key, 24

M magic strings, adding, 100
Make BBA action key, 30
Make Static action key, 29
message, HP64700 I/O error, 22
Mods () action key, 24
MRI C Cross compilers, 35
multiple command files to Debugger/Emulator, 52
multiple executables, can I load?, 84, 86

N Never Load, 53

O Options: Load Options..., 53
Options: Start Commands..., 51

P Parms () action key, 24
preconfigure the Debug64700 interface and the SoftBench interaction, 70
problems and solutions, common, 84

R rebuild the target executable file, 29
Refs () action key, 24
RTOS measurement tool, 48
RTOS performance analyzer, 48

S
selecting an HP64700 system, 47
selecting the interfaces to start, 48
setting the load options, 53
single command file to Debugger/Emulator, 52
SoftBench build commands, 76
SoftBench build tool, 29
SoftBench builder, using, 62-64
SoftBench edit commands, 75
SoftBench editor, invoking, 28
SoftBench editors, using, 61
SoftBench environment, customizing, 95-96
SoftBench generic message commands, 78-79
SoftBench messages supported, 74
SoftBench Static Analyzer, using, 65
SoftBench, starting, 37
SoftBench_BBA command, 77
software version requirements, 93
solutions to common problems, 84
space in the context, 84, 86
specifying a configuration file, 49-50
Start Additional..." button, 55
Startdebug64700 demo, running, 15-19
starting additional interfaces, 55-56
starting SoftBench, 37
starting the Debug64700 interface, 45
starting the Debug64700 system, 38-39
static analysis modes, 35
static analyzer, 29
Static Analyzer graph, use to set breakpoints, 25-26
Static Analyzer, using, 23-24
static compile script, 35
supported emulator systems, 93

T
target executable file, rebuilding, 29
Target Executable, changing, 46

U
Unlock button, 47
Update Status button, 54
Uses () action key, 24
using Debug64700 remotely, 98-99
using the HP Branch Validator, 66
using the SoftBench builder, 62-64

using the SoftBench editors, 61
using the SoftBench Static Analyzer, 65

V versions, software, 93

W what is the advantage of using supplied compile scripts?, 84, 88
where do files come from when running remote?, 84, 87

X Xdefaults.template file, 62

Certification and Warranty

Certification

Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Bureau of Standards, to the extent allowed by the Bureau's calibration facility, and to the calibration facilities of other International Standards Organization members.

Warranty

This Hewlett-Packard system product is warranted against defects in materials and workmanship for a period of 90 days from date of installation. During the warranty period, HP will, at its option, either repair or replace products which prove to be defective.

Warranty service of this product will be performed at Buyer's facility at no charge within HP service travel areas. Outside HP service travel areas, warranty service will be performed at Buyer's facility only upon HP's prior agreement and Buyer shall pay HP's round trip travel expenses. In all other cases, products must be returned to a service facility designated by HP.

For products returned to HP for warranty service, Buyer shall prepay shipping charges to HP and HP shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to HP from another country. HP warrants that its software and firmware designated by HP for use with an instrument will execute its programming instructions when properly installed on that instrument. HP does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

Limitation of Warranty

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environment specifications for the product, or improper site preparation or maintenance.

No other warranty is expressed or implied. HP specifically disclaims the implied warranties of merchantability and fitness for a particular purpose.

Exclusive Remedies

The remedies provided herein are buyer's sole and exclusive remedies. HP shall not be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory.

Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

For any assistance, contact your nearest Hewlett-Packard Sales and Service Office.