HP 82000 IC Evaluation System

Using the EDA Interface

Models D50, D100, D200 and D400

SERIAL NUMBERS

This manual affects all instruments.

HP Part No. E1299-90001
Microfiche Part No. E1299-95001
Printed in West Germany November 1993

Revision 2.0

Legal Information

Notice

The information in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MANUAL, 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 direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this manual.

Warranty

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

HILO-3 is a registered trademark of GenRad Inc.
QuickSim is a registered trademark of Mentor Graphics Corporation
Logician is a registered trademark of Daisy Systems Corporation

© Hewlett-Packard GmbH 1989

Printing History

New editions of this manual will incorporate all material updates since tha previous edition. Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a revision date at the bottom of the page. A vertical bar in the margin indicates the changes on each page. Note that pages which are rearranged due to changes on a previous page are not considered revised.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

Revision 1 May 1989.

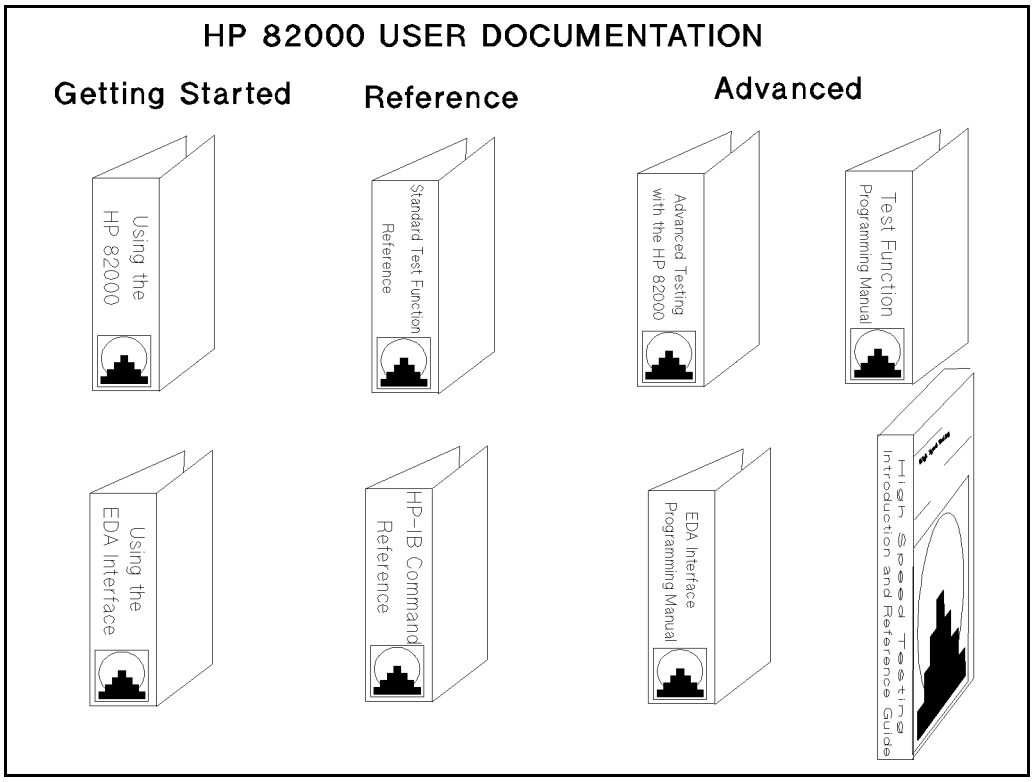
Effective Pages: ALL

Revision 2 November 1993

Effective Pages: ALL

Documentation Map

The following figures show the manuals provided for the HP 82000 IC Evaluation System. For more information on the contents of each manual and a list of tasks, refer to the Documentation Roadmap booklet.



Preface

Purpose

The purpose of this manual is to describe how to use the EDA Interface software to translate simulator files, generated on EDA stations, to functional tests that can be used to test devices on the HP 82000 IC Evaluation System.

Simulation files from the following IC simulators can be processed:

- HP74000 HILO-3 simulator
- Mentor Graphic Corporation's QuickSim simulator
- Daisy Systems Daisy Logic Simulator

Audience

This manual is intended for engineers who want to test devices using functional tests translated from simulation files.

Before using this manual, you should have attended an HP82000 training course or worked through the EDA Interface section of the Getting Started Manual. If you have not already done so, work through this section now to get an understanding of the EDA Interface functions.

For information on writing your own translators, refer to the EDA Interface Programming Manual.

Contents

This manual contains 11 chapters.

1	Introduction - explains the purpose and use of the EDA Interface software.
2	Using the EDA Interface - how to setup and translate EDA files.
3	Generating Control Files - how to set up control files to define pin and state mapping
4 to 7	Extraction Software Descriptions - how the different parts of the software work and default values
8	System Networking Tools - describes how to make connections to, and communicate with other computer systems.
9 to 11	Descriptions of the standard formats supported by the EDA Interface software.
A	Description of the pin mapping file format
B	Description of the signal mapping file format
C	Example EDA Files

Conventions used in this manual

In the descriptions in this manual the following terms are used:

pin mapping	the mapping of EDA signals to physical tester pins
pin name	name of a tester pin
signal mapping	the mapping of waveform states to tester states
signal name	name of a signal in an EDA file
simulation files	files generated by a simulator that describe the behavior of device
waveform	description of a signal in an EDA file

Contents

Index

Figures

Tables

1

What is the EDA Interface?

Introduction

The EDA Interface software allows you to automatically generate functional tests from IC simulation results. Simulation results are stored in simulation input (stimulus) files and output (log) files in various simulator-specific formats. These simulation files can be accessed or transferred from or to the HP 82000 system using the networking tools described in Chapter 8.

Translated tests can be either stored in setup files or downloaded to the tester hardware.

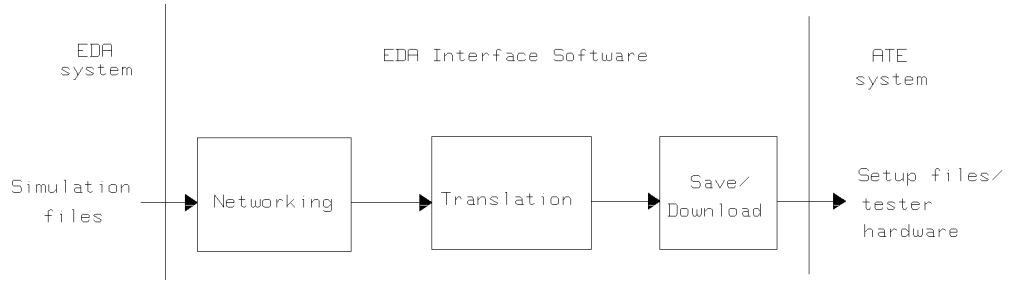


Figure 1-1. The EDA Interface Function

Supported EDA Systems

The EDA Interface software can translate simulation files produced by the following EDA simulators:

- hp74000 HILO-3™
- Mentor Graphics QuickSim™
- Daisy Systems DLS using the Logician™ system

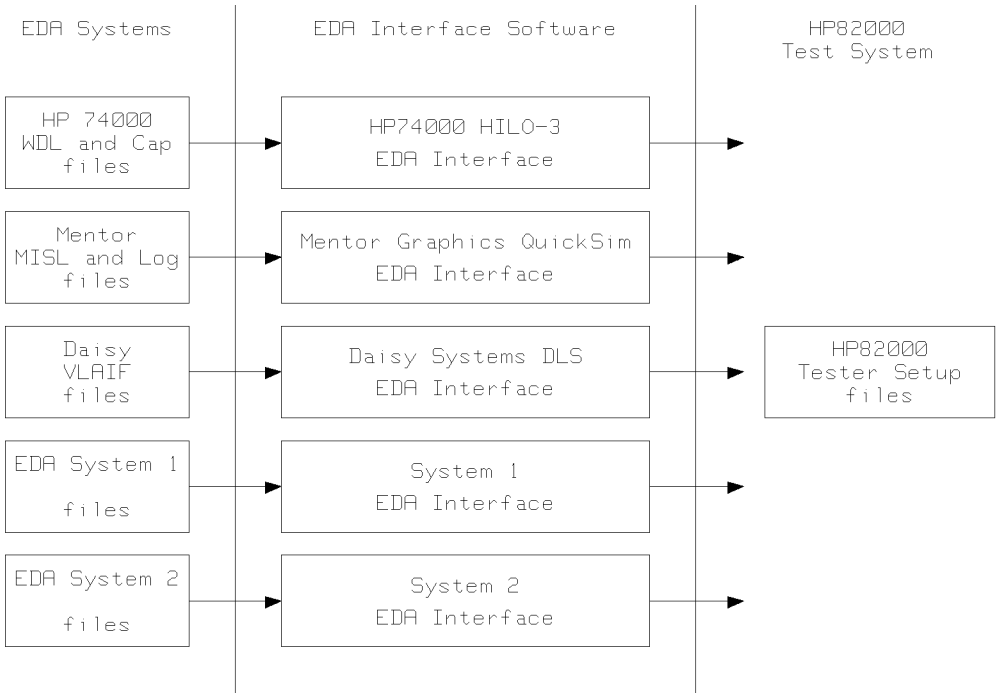


Figure 1-2. EDA Interface formats

Additionally, by using the EDA Interface Programming Toolkit, EDA Interfaces can be written for other systems and integrated into the EDA Interface software. Refer to the EDA Interface Programming Manual for more information on generating new EDA Interfaces.

EDA Interface Functionality

All of the supplied EDA Interfaces have the same functionality, which is controlled by the control parameters and functions of the EDA Interface screens (See Chapter 3). The three main functions of the software are:

- Pin Configuration Extraction
- Timing Setup Extraction
- Vector Setup Extraction

The various inputs and outputs of the EDA Interface are shown below.

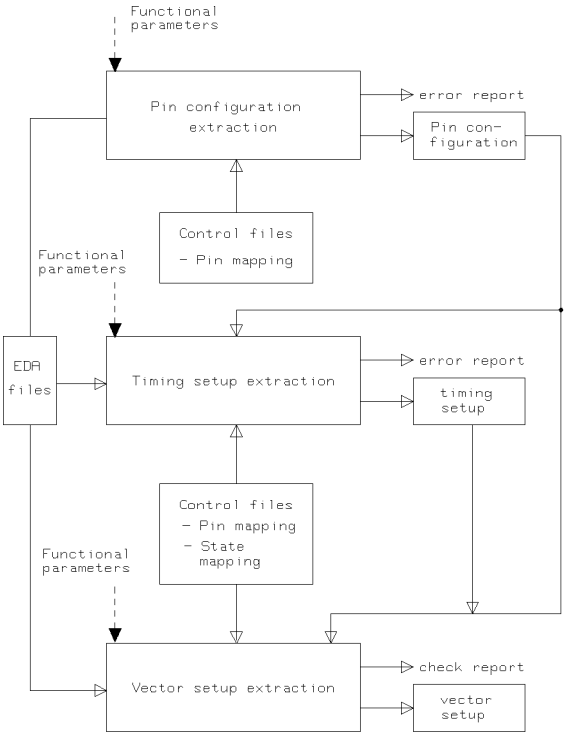


Figure 1-3. EDA Interface Functionality

- **EDA Files** These contain simulation results in EDA system-specific formats.
  - Stimulus file - simulation input that contains design pin configuration and input signal waveform information.

- Log file - simulation output that contains design pin configuration information and the logged input and output signal waveforms.

- **Control Files** These files contain information used to control how the information in the EDA files is translated.
  - Pin Mapping file - used to relate design pins and signals to tester pins
  - State Mapping file - used to relate simulator signal states to tester states.
- **Tester Setups** contain functional tests in HP 82000 system format.
  - Pin Configuration - contains the tester pins extracted from design pins and signals in the EDA files
  - Timing Setup - contains the tester pin timing extracted from the signal waveforms of the EDA log file.
  - Vector Setup - contains the vector patterns extracted from the signal waveforms of the EDA log file.
- **Function Parameters** options and parameters that control the behavior of the EDA Interface.

2

Using the EDA Interface Software

This chapter describes how to use the EDA Interface software to translate files from a CAE/EDA system to a format that can be used by the test system.

Pre-Requisites

The EDA Interface requires a certain amount of information which can either be extracted from the EDA files or explicitly specified as inputs to the software. To help ensure the best results from the translation, you should know the following information before you use the EDA Interface.

- Expected pin configuration
  - Pin names
  - Pin types
- Design pin/signal to tester pin assignments
- Timing specifications
  - Simulation period used
  - Simulation time scaling factor used
  - Expected timing information for each pin, consisting of:
    - formats
    - operation modes
    - leading and trailing edges
- Simulator state to tester state mapping

This information is required either to check the translation results, or to make additional inputs to the translation process.

Starting the PWS Software

If the PWS software is not running, you will have to start it. Depending on your system configuration, this may happen automatically when you power up the system, or by typing a command such as

```
hp82000 [-oxb]
```

at the HP-UX prompt. Use the `o` option if the tester hardware is not available. The EDA Interface software does not need hardware access to translate files.

Transferring the EDA Files

Once the PWS software has started, you can select HP-UX Shell in the AUX set of pushbuttons to access the system communication facilities. When this has been done, you have access to the EDA system and can transfer data files for translation. Refer to Chapter XX for more information on using the system networking tools.

Starting the EDA Interface Software

When you have transferred all the data files you require from the EDA system, you should click the CAE field in the Main Menu Bar to start the EDA Interface software.

The EDA Interface Main Menu appears when the program starts.

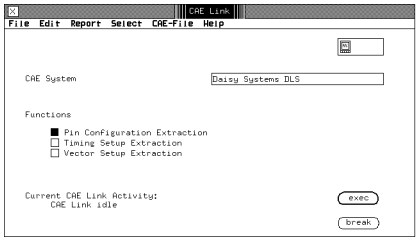


Figure 2-1. EDA Interface Main Screen

From this screen you can control the EDA Interface functions.

Selecting the EDA System

The EDA System field allows you to enter the type of simulator used to produce the simulation files. Select the EDA system corresponding to the system you used to prepare your simulation data. Options in this field are:

- hp74000 HILO-3
- Mentor Graphics Quicksim
- Daisy System DLS

Additional options may appear here if you have written your own EDA Interface. See the EDA Interface Programming Manual for more information about writing EDA Interfaces and converting files from other CAE/EDA systems.

Selecting the EDA Interface Functions

There are three functions available from the EDA Interface. These are:

- Pin Configuration Extraction - generates a pin configuration setup from the EDA data.
- Timing Setup Extraction - generates a timing setup from the EDA data and optionally modifies the mode in the pin configuration, if required.
- Vector Setup Extraction - generates a vector setup from the EDA data.

Options are available in the Function Parameters menu (under Select) or by pressing CTRL F (^F) which allow you to control how generated setups are set up and modified.

Select the functions you require by clicking the appropriate check boxes.

EDA Interface Status

A report line at the bottom of the Main Screen shows the current state of the EDA Interface software, whether it is idle or running.

Setting Up the EDA Interface

In the menu-bar, (File), (Edit), and (Report) allow you to access the standard features available in the other setup windows. (Select) is used to set up the parameters for the translation and (EDA-File) allows you to copy, print and delete the EDA files that you have transferred from your EDA workstation.

The (Select) key provides a pull-down menu containing selections allowing you to define:

- EDA System parameters
- ATE System parameters
- Control parameters
- Function parameters

Selecting the EDA files

The EDA Interface software requires at least one file from which information can be extracted. This file must be specified in the EDA file 2 text edit field.

It is possible to specify two files, in this case, the file entered in EDA file 1 text edit field will be processed by the EDA Interface processing functions.

Depending on the selected EDA system, you can specify the following file types.

Table 2-1. CAE/EDA File Types

CAE/EDA System	CAE 1 file (optional)	CAE 2 file (mandatory)
HP74000 HILO-3	WDL	Capture
Mentor Graphics QuickSim	MISL	Log
Daisy System DLS	-	VLAIF

Clicking (EDA System files) in the (Select) pull-down menu calls up the Select EDA parameters dialog box.

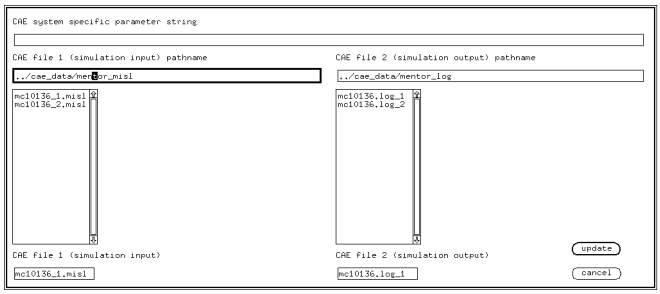


Figure 2-2. EDA System Files

Selecting the Path Name

The EDA Interface expects EDA data files to be in the `cae_data` directory of the current device. If your EDA files are not in this directory, type in the pathname to the directory containing these files.

If the files are on another system and you have not transferred them to the local system, you could enter the complete pathname to the remote system.

For example, typing in:

```
/net/CAE_Stat/users/rick/10H136/cae_data
```

would set the path to the remote computer `CAE_Stat`, and the software would search the `cae_data` directory there for simulator files.

Note

You should have performed a `netunam` command to allow access to the remote computer before starting the HP 82000 software.

Once you have selected the path, any files found in that directory will be listed in the file browser.

Selecting EDA Files

You can either select the required files with the mouse or type its name in the edit field. If you selected a file with the mouse, the filename will appear in the entry field.

`CAE file 1` is used for the optional stimulus file and `CAE file 2` for the mandatory simulation output file. If you are only entering one filename, you should enter it in theh CAE 2 file field.

Entering an EDA System-specific Parameter String

If you want to supply any additional information to the EDA Interface, you can enter this data in the field at the top of the screen. Depending on the selected EDA System, you can enter the following data. This field can also be used to enter additional information for user-written EDA Interfaces.

Table 2-2. EDA System Specific Strings

EDA System	String interpreted as:
HP74000 HILO-3	-
Mentor Graphics QuickSim	timescale (implied units nanoseconds)
Daisy Systems DLS	timescale (implied units nanoseconds)

When you have defined the files you require, pressing the enter pushbutton will read the filenames and return you to the EDA Interface Main Screen. Pressing cancel will leave the current settings of `CAE file 1` and `CAE file 2` unaltered.

Selecting ATE Setup Files

When the EDA files have been selected, you should now select the test setup files that are to be used or modified for the current translation. Clicking ATE System Params in the Select pull-down window displays the ATE System Parameters Window. The options you set in the Function Parameters window are used to decide whether the files you enter here will be modified or not.

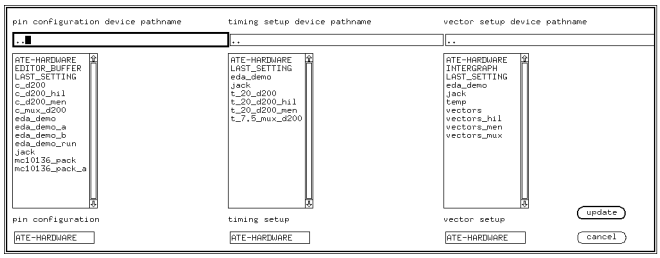


Figure 2-3. ATE System Files Window

Selecting a Pin Configuration

The default pathname is `current_device/configuration`. If you want to use pin configuration files from another directory, modify the pathname field. If the directory exists and contains files, they will be listed in the file browser.

If you want to select an existing pin configuration file, select it with the mouse. If you want to create a new pin configuration file, enter a new filename in the pin configuration entry field.

Entering `ATE-HARDWARE` will result in the translated pin configuration being loaded into the tester hardware.

Selecting a Timing Setup File

The default pathname is `current_device/timing`. If you want to use timing setup files from another directory, modify the pathname field. If the directory exists and contains files, they will be listed in the file browser.

If you want to select an existing timing setup file, select with the mouse. If you want to create a new one, enter the name of a new file in the timing setup entry field.

Entering `ATE-HARDWARE` will result in the translated timing setup being loaded into the tester hardware.

Selecting a Vector Setup File

The default pathname is `current_device/vectors`. If you want to use vector setup files from another directory, modify the pathname field. If the directory exists and contains files, they will be listed in the file browser.

If you want to select an existing vector setup file, select with the mouse. If you want to create a new one, enter the name of a new file in the vector setup entry field.

Entering `ATE-HARDWARE` will result in the translated vector setup being loaded into the tester hardware.

Leaving the ATE File Window

When you have defined the files you require, pressing the update pushbutton will read the filenames and return you to the EDA Interface Main Screen. Pressing cancel will leave the current settings of pin configuration, timing setup, and vector setup unaltered.

Selecting Control Files

Before you can start a translation, the EDA Interface software must know how the simulator file contents must be translated. The simulator files contain information about the design pins/signals and the logical states of the signals used in the simulation. If you do not select control files, the EDA Interface will use default values for mapping. Mappings defined in control files override these defaults.

There are two types of control files that can be optionally selected before a translation can take place. These are:

- pin mapping file - maps the signal names of the simulator files to tester pin names used in the pin configuration (See Pin Configuration Extraction and Using the Pin Mapping Editor). If you need to:
  - rename signals
  - combine a number of simulator signals to make up one tester pin
  - ignore signals in the EDA files

  you will have to generate a pin mapping file and specify its name here.
- state mapping file - maps the logical states used in the simulator to the logical states that can be generated by the test system.

  If your EDA files contain logical states that are not recognized by the tester, you will have to generate a state mapping file and enter its filename here. See Using the State Mapping Editor.

Both of these are selected from the Control Parameters window which is in the Select pull-down window.

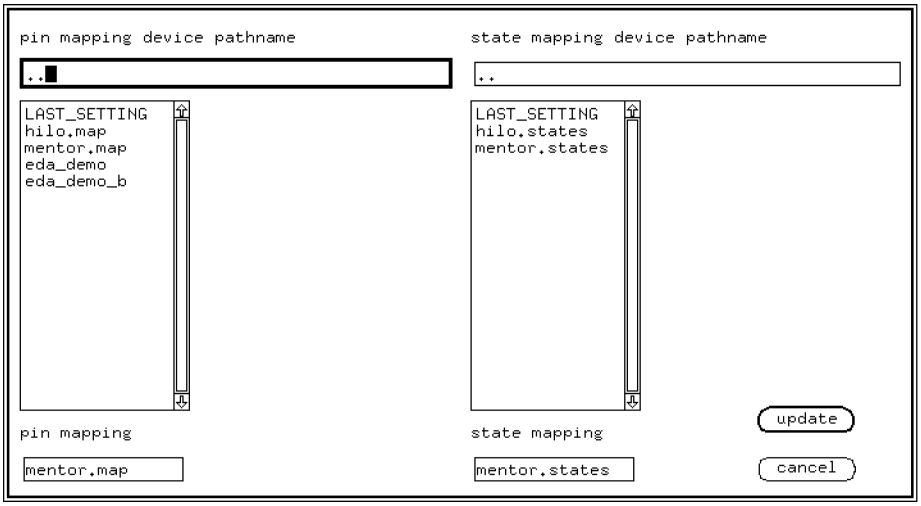


Figure 2-4. Control Files Window

Selecting the Pin Mapping Control File

Select the pin mapping file required from the browser. The default path name for these files is `current_device/cae_pmap`. If you want to use a file from another directory, enter the pathname and then select the required file.

Pin mapping control files are generated using the EDA Pin Editor which is described in Chapter 3.

Selecting the State Mapping Control File

Select the state mapping file required from the browser. The default path name for these files is `current_device/cae_control`. If you want to use a file from another directory, enter the pathname and then select the required file.

State mapping control files are generated using the EDA State Editor which is described in Chapter 3.

Leaving the Control File Selection Window

When you have defined the files you require, pressing the update pushbutton will read the filenames and return you to the EDA Interface Main Screen. Pressing cancel will leave the current settings of pin mapping and state mapping unaltered.

Setting Translation Options

Selecting Function parameters in the Select pull-down menu will display the Function Parameters window. This window allows you to set the period to be used for the translation and other extraction options.

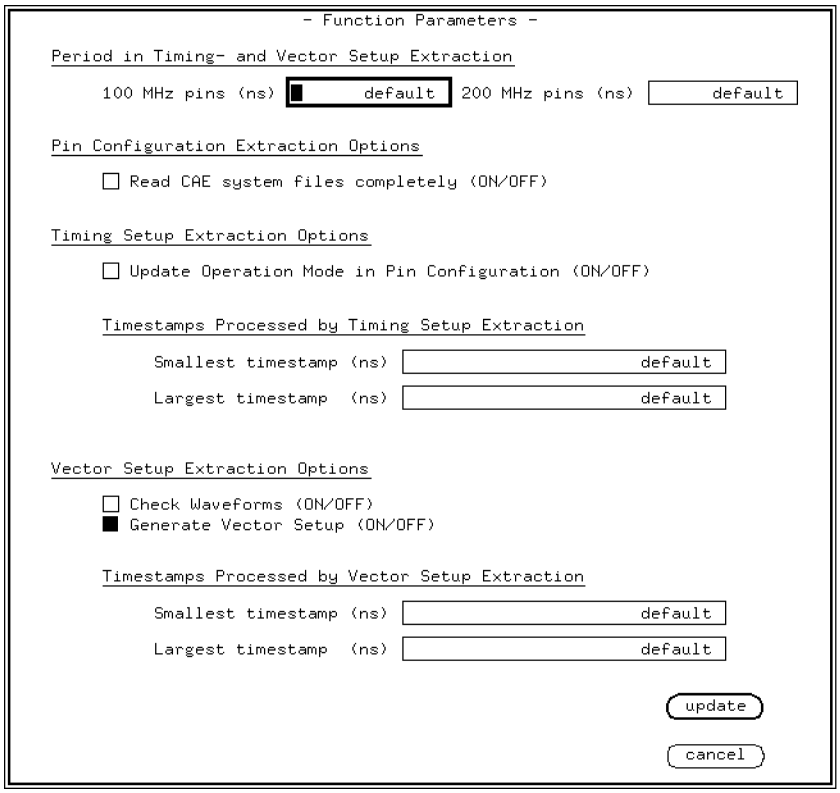


Figure 2-5. Function Parameters Window

Setting the Clock Period for the Translation

The fields `100 mode` and `200(MUX) mode` allow you to enter the system clock rate to be used for the translation. You can enter a number between XX and YY which is used to decide where the translation software breaks the simulation file into test cycles. If you enter a value of 10 ns in 100 mode, when the simulation files are read, vectors in the range 0 to 9.999 nanoseconds are placed in vector 0, vectors between 10.000 to 19.999 nanoseconds in the second, between 20.000 and 29.999 in the third and so on, if the leading and trailing edges are within the period. If you select `default`, the software will try to read the simulation file and extract the period from it.

Note The period can only be extracted from Mentor stimulus file.

Note that the two fields are linked. If you enter a value in one field, the corresponding value will be entered in the other field.

Setting Pin Configuration Options

The only option for pin configuration is reading the log file completely. If you know that all the pin definitions are at the beginning of the files, this option reads only those lines up to the first signal state transition, to make the pin configuration process faster.

If pin definitions all over the CAE files you will have to use this option to ensure that all definitions are found.

For more details, refer to Pin Configuration Extraction.

Setting Timing Setup Extraction Options

In this part of the window you can select timing setup extraction options.

This part of the software extracts and defines the format and times for the leading and trailing edges of the vector data for each tester pin as shown in the example below.

Update Pin Configuration. The Update Operation Mode in Pin Configuration check box allows you to select what happens when, after timing extraction, the pin configuration is found to be invalid. If you select this option, the operation mode of the tester pins will be modified to make them compatible with the required timing and format.

For more details, refer to Timing Setup Extraction.

Timestamps to be Processed. These two fields allow you to set the beginning and end of extraction if you do not want to process the complete file. Using these fields, you can enter the timestamps where the translation should start and stop. The defaults for these fields are the first and last timestamp (the whole file) respectively.

Vector Setup Extraction Options

There are two options that can be used for vector extraction, Check Waveforms and Generate Vector Setup.

Check Waveforms. This option allows you to check the waveforms as they are being translated. Any waveforms that cannot be produced by the tester hardware will be modified so that they can be output.

Generate Vector Setup. This option can be used in combination with the Check Waveforms option. If this option is switched off you can check the waveforms in the EDA file without creating a vector setup file.

Timestamps to be Processed. This option can be used to decide which part of the EDA file is to be read for vector data. You can specify a start and finish timestamp (in EDA file units) or use the default which is to read the complete file.

Selecting the Report Mode

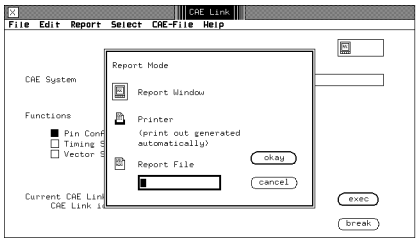


Figure 2-6. Reports Screen

You can select any combination of report media.

If you specify the Report Screen as report medium, all errors, warnings and additional messages will be written directly to the report screen.

If you specify a file as the report medium, the reports generated during the execution of the sequence of selected CAE link functions will be stored in the file.

The (Report) menu allows you to handle report files that are stored in the `../cae_report` directory.

If the printer was selected as the report medium, the report generated for each function during the execution of the sequence of selected CAE link functions will be printed when the selected sequence of functions terminates.

3

Generating Control Files

If you have special translation requirements, the EDA Interface software can optionally use two control files to enable it to translate simulation files:

- State Mapping file
- Pin Mapping file

This chapter shows you how to generate these control files using the State and Pin Mapping Editors.

State Mapping

Purpose of State Mapping

The purpose of the State Mapping file is to:

- map signal states onto tester compatible states
- specify an inaccuracy range for time computations

Many simulators use a large number of signal states to describe a signal waveform. The Timing and Vector Extraction processes derive tester waveforms from these signal waveforms. There are only five states that can be used by the tester to describe inputs and outputs. For more information on these states, refer to Processing Waveforms in Chapter 5.

The Timing and Vector Extraction processes apply a default mapping to the signal waveforms to produce tester states. If these default settings are not adequate for your purposes, you can modify the mapping of signal states to tester states using the state mapping file. The filename of the state mapping file must be specified in the Control Parameters dialog box of the `Select` menu.

Default state mappings are shown in the table below.

Table 3-1. Default State Mapping

Signal Type	Signal State	Mapped Signal State
`inp`	0	0
	1	1
	T	T
	other char	0
`out`	0	0
	1	1
	X	X (don't care)
	I	I (intermediate)
	other char	0
`dir`	0	0 (output off)
	T	T (output on)
	other char	0 (output on)

For a given timestamp, the mapping of signals with type `bid` is defined by the mapping for the `inp` and `out` states depending on which state the line is in at that time.

Starting the State Mapping Editor

Start the State Mapping Editor by clicking the CAE State Ed pushbutton in the AUX menu. When it starts, the State Mapping Editor Main Screen is displayed as shown below.

Using the State Mapping Editor

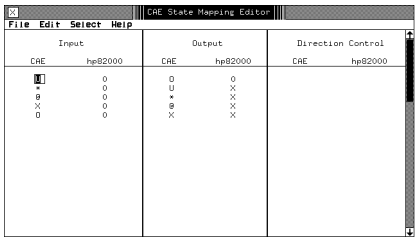


Figure 3-1. State Mapping Editor Main Screen

Mapping Simulator States

In the Main Screen, you must enter the EDA signal states and the corresponding ATE system states that will be generated for these states. For each column, input, output, and direction control you can define up to 54 states to be mapped.

Possible states for each simulator are listed in Chapters 9 to 11.

The allowed system states are:

- Input - these signals are used as DUT inputs.
  - `0` - corresponds to a logical low - a signal in this state will be mapped as a logic low input.
  - `1` - corresponds to a logical high - a signal in this state will be mapped to a logical high input.
  - `T` - corresponds to the high impedance state (tristate) - a signal in this state will result in the driver going to tristate in the current cycle.
- Output - these signals are used as DUT outputs.
  - `0` - corresponds to a logical low - the comparator will be set to expect a logical low in this cycle
  - `1` - corresponds to a logical high - the comparator will be set to expect a logical high in this cycle
  - `X` - corresponds to a don't care condition - the comparator will be masked for this cycle. The state of the expected data has no effect on the test results, it will be ignored.
  - `I` - corresponds to an intermediate state - the comparator will be set to expect the signal to be between the programmed high and low levels in the current cycle.

- Direction Control - these signals are used to set the state of bidirectional pins. Since there are two pins associated with a bidirectional pin, when one is active, the other must be in an inactive state.
  - `T` - corresponds to the output state. The driver pin will be set to tristate irrespective of the actual data being input to the DUT and the current value of the signal will be read as expected data.
  - `0` - (zero) corresponds to the input state. The expected data for the comparator will be masked and the actual data will be mapped to the drive data for the current vector. If the current state is tristate, the driver output will also be tristated.

Setting the System Inaccuracy

The system inaccuracy, which is part of the (Select) pull-down menu, allows you to define a window for timing and vector extraction. If this is set, any edges found in the EDA files that are within ±`inaccuracy` will be translated to edges at the current setting.

For example, if you set `inaccuracy` to 1000, any edges detected within ±1 nanosecond of the current edge will be mapped to the current edge. Any edges falling outside this window will be ignored.

The value entered is in picoseconds and can have a value up to 999999999.999. The value at power-on is 0 picoseconds.

Note

The inaccuracy value should be as small as possible in order to prevent combinations of state transitions that are not intended to be regarded as identical being treated wrongly.

The inaccuracy value will be automatically decreased to (period for 100 mode) ÷ 4 by the timing and vector extraction processes, if the given value is too high.

Saving the Signal Mapping File

When you have made all the entries required, save the file. The default directory for these files is `../cae_control`.

Pin Mapping File

The Pin Mapping file, generated by the Pin Mapping Editor, maps the names of signals/pins used in the IC design to tester pin names. For more information on pin mapping, refer to Chapter 4.

Starting the Pin Mapping Editor

You start the Pin Mapping Editor by pressing the CAE PinEd key in the system AUX menu.

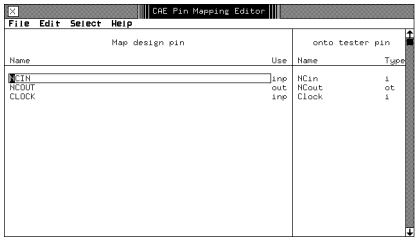


Figure 3-2. Pin Mapping Editor Main Screen

Generating a Pin Mapping file

Once the Pin Mapping Editor has started, you can edit existing files or create new Pin Mapping files.

There are two windows in the editor:

- the Main Screen - which allows you to define the mapping between signals and tester pins
- Ignore Design Pin window - where you can list the simulator pins that are to be ignored for the translation.

Entering the Simulation Signal Name

Type in the name of the signal as used in the simulation file in the Name column. This name must correspond exactly to the name in the simulation file and must not contain any spaces.

Entering the Signal Use

Under `Use` type in one of the following, depending on the purpose of the signal in the simulation file. This can be one of:

`inp`	the signal is an input signal.
`out`	the signal is an output signal.
`bid`	the signal is a bi-directional signal.
`dir`	the signal is a direction control signal.
`ana`	the signal is an analog signal, such as a power supply.
`unknown`	an empty field will set the signal to the unknown type

Entering the Tester Pin Name

Enter the name of the test system channel that is to be used for this simulator signal(s). This name consists of up to 16 characters. You can use any of the keyboard characters except blank (space), comma (,), semi-colon (;), opening quote (‘), double quotes (”), left and right round brackets ( ).

Entering the Tester Pin Type

Enter the tester pin type for each entry. This must be one of the following values:

`i`	The pin is a DUT input pin only, the channel’s comparator will not be used for measurements.
`o`	The pin is a DUT output pin only, the channel’s driver will be disabled.
`io`	The pin is a bidirectional pin, both driver and comparator resources are enabled.
`ioh`	The pin is a bidirectional pin with termination to a high level, both driver and comparator resources are enabled.
`iol`	The pin is a bidirectional pin with termination to a low level, both driver and comparator resources are enabled.
`ot`	The pin is a bidirectional pin with an active termination, comparator resources are enabled.
`dc`	The pin is connected to the DC Measurement Unit.
`unknown`	An empty field defines an unknown type.

Selecting Pins to Ignore

Once you have entered all the names of signals that are to be translated, you should use the Ignore design pin window to tell the system which signals should be ignored.

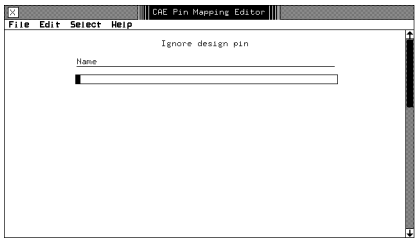


Figure 3-3. Pin Mapping Editor - Ignore Design Pins Screen

Type in the names of all the simulator signals that are to be ignored.

When the window is first opened, there is space to enter 50 signal names. If you need to enter more than 50 signal names, use Append lines in the Select pull-down menu to increase the size of the file. Each time you do this, the scroll box will get smaller to indicate how much of the file is currently visible. You can enter up to 400? names in the list.

Saving the Pin Mapping File

When you have completed the pin mapping you can save the file using the `save` or `save_as` entry in `File` pull-down menu. The default directory for pin mapping files is `../cae_pmap`.

4

Pin Configuration Extraction

This chapter describes the pin configuration extraction function of the EDA Interface software.

The pin configuration extraction process translates the design pin configuration stored in the EDA file(s) to a tester pin configuration. The process is shown in Figure 4-1.

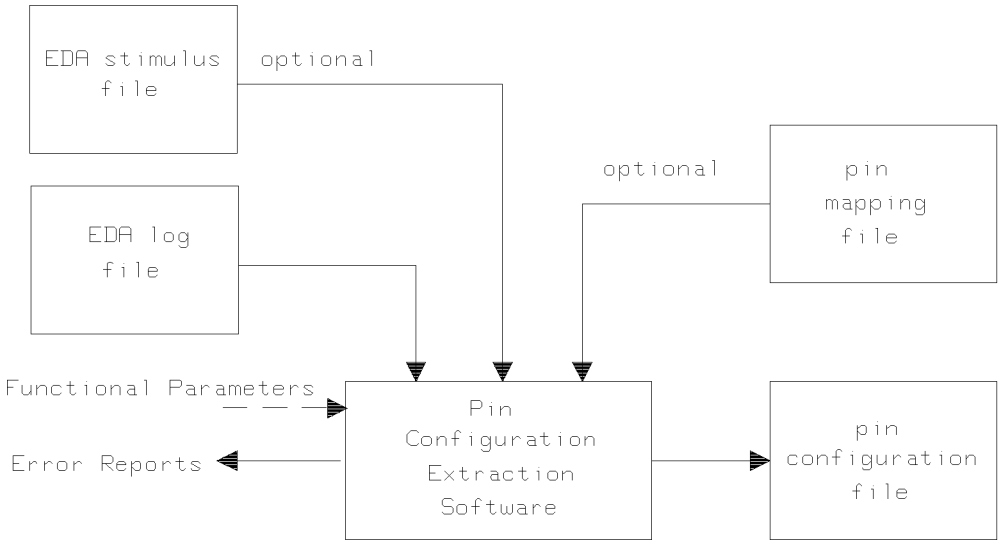


Figure 4-1. Pin Configuration Extraction

The information processed by pin configuration extraction are shown in the table below.

Table 4-1. Information Extracted by Pin Configuration

	HP 74000 HILO-3		Mentor QuickSim		Daisy DLS	
Extracted Data	WDL	Capture	MISL	Log	-	VLAIF
Signal Names	YES	YES	YES	YES	-	YES
Signal Types	YES	YES[1]	YES	NO	-	NO

[1] With some exceptions.

You can extract pin configurations using default pin mappings supplied by the system or by using a pin mapping from a pin mapping file created with the Pin Mapping Editor. These mappings will be applied instead of the default mappings.

Translation without a Pin Mapping File

The pin names and types are translated from the signal names and types in the EDA file. Any pin attributes that cannot be translated from information in the EDA files will be preset with defaults as shown below.

Table 4-2. Default Pin Configuration

EDA File Information	Extracted Pin Attributes	Defaults Assigned
Signal Name ▸	Name	None assigned
Signal Type ▸	Type	`ot` (output terminated)
	Operation Mode	`STD` (standard)
	Scan Path Mode	`parallel`
	Tester Channel	`#` (valid identifier)

Default Pin Mapping

Default pin mapping will be used while translating signal names to pin names and signal types to pin types. Each detected signal is translated to a pin.

Signal Name to Pin Name Mapping

Signal names are stripped to a length of 16 characters. Any characters not allowed by the tester are translated to underscores (_). All ASCII characters in the range 21H (33 dec) to 7EH (127 dec) are allowed except @ ( ) # , ; ” ’

Example

The signal name `@EXAMPLE:Dbus(1)` will be translated to

`_EXAMPLE:Dbus_1_`

Unique Naming

If a pin name that is extracted is the same as a previously defined pin, another unique name will be generated for that pin.

Example

The signal names `@EXAMPLE:Dbus(1)` `@EXAMPLE:Dbus(1)(3)` will be translated to

`_EXAMPLE:Dbus_1_` and `PIN_1` respectively since the second signal would map to the same name as the first.

Signal Type to Pin Type Mapping

Signal types will be translated as shown in the table below.

Table 4-3.

Signal Type		Pin Type	
Input	inp	i	input
Output	out	ot	output terminated
Direction Control	dir	ot	output terminated
Bidirectional	bid	ot	output terminated
Analog	ana	dc	DC
Unknown	-	ot	output terminated

Translation with a Pin Mapping File

If a pin mapping file was specified in the Control Parameters dialog box, the mappings in this file will have a higher priority than the default mappings. These pin mappings may be used to:

- Map signal names to pin names
- Combine a number of signals to make one pin
- Map signal types to pin types
- Ignore signals in the EDA file

Signal Name to Pin Name Mapping

A signal name may be mapped to any other pin name. With the pin mapping file entry:

`@EXAMPLE:Dbus(1) | Dbus[1]`

the signal name `@EXAMPLE:Dbus(1), out` will be translated to the pin name

`Dbus[1], ot`

The wildcard character (#) can be used to map multiple signal names to multiple pin names,

`@EXAMPLE:Dbus(#) | Dbus[#]` in the pin mapping file would result in the following translations:

`@EXAMPLE:Dbus(1),inp` will be translated to `Dbus[1],i`
`@EXAMPLE:Dbus(2),inp` will be translated to `Dbus[2],i`
`@EXAMPLE:Dbus(10),inp` will be translated to `Dbus[10],i`

The entry `#|#` in the pin mapping file translates the signals using the default mapping rules.

Combining Signals to make Pins

Up to three signals can be combined to make up a tester pin. The allowed combinations are showed in the table below.

Table 4-4. Allowed Combinations of Signals

Signal Types	Resulting Pin Types
inp	i
out	o / ot
inp + dir	i
inp + out	io / iol / ioh / nc
inp + out + dir	io / iol / ioh / nc
bid + dir	io / iol / ioh / nc
ana	dc
(unknown)	(unknown)

With the following pin mapping entry,

```
@EXAMPLE:Dbus(1),bid | Dbus[1],io
@EXAMPLE:Cpin   ,dir |
```

the signals
`@EXAMPLE:Dbus(1),bid` and `@EXAMPLE:Cpin  ,dir`

will be translated to the tester pin `Dbus[1],io`

Pin mapping file entries with **one** wildcard character in each name are allowed to combine groups of signalsi, such as bidirectional busses, to multiple tester pins.

Signal Type to Pin Type Mapping

Signals may have different usages for different pins, or may need to be modified for some reason. The following table shows how signal types can be changed due to pin mapping file entries.

Table 4-5. Signal Type Changes

	Signal Usage					
Signal Types	**inp**	**out**	**dir**	**bid**	**ana**	**(unknown)**
inp	inp	out	dir	bid	*	inp
out	inp	out	dir	bid	*	out
dir	inp	out	dir	bid	*	dir
bid	inp	out	dir	bid	*	bid
ana	*	*	*	*	ana	ana
(unknown)	inp	out	dir	bid	ana	out

Note Entries denoted by an asterisk (*) cause an error.

With the following pin mapping entry,

```
@EXAMPLE:Dbus(1),bid | Dbus[1],io
@EXAMPLE:Cpin   ,dir |
```

the signals
`@EXAMPLE:Dbus(1),(unknown)` and
`@EXAMPLE:Cpin  ,inp`

will be translated to the tester pin

`Dbus[1],io`

Ignore Signals

Signals will be ignored if their names appear in the Ignore Design Pin list of the pin mapping file. See Chapter 3 for more details of the Pin Mapping Editor.

You can also specify pin names with a single wildcard character to ignore more than one signal.

Example

If the ignore definition contains **#**, all signals except those listed in a pin mapping file will be ignored.

Order of Pin Mapping Evaluation

The pin configuration extraction process evaluates pin mapping file entries in the folowing order.

1. Pin mappings with no wildcards
2. Ignore signal definitions with no wildcards
3. Pin mappings with wildcards
4. Ignore signal definitions with wildcards

This order leads to different possibilities of pin mapping:

1. When the pin mapping file is missing or empty, the default pin mapping will be used and each signal is translated to a tester pin.
2. Subsets of signals may be selected for translation by:
   a. Specifying certain signals in the pin mapping and ignoring all others
   b. Specifying certain signals in the the ignore list and mapping all others
   c. Specifying some signals in both the pin mapping and pin ignore lists

5

Introduction to Timing and Vector Extraction

This chapter describes the common features of the timing and vector extraction processes.

Signal Waveform Processing

The timing and vector extraction processes perform their functions of extracting data from the EDA log file in two steps:

1. Each EDA state encountered in a signal waveform is mapped onto a tester state. The mapping of EDA states onto tester states is defined by either:

   a. The contents of a state mapping file - if specified
   b. A default mapping used by the extraction process.

   See Chapter 3 for more details.

2. According to the data in the pin mapping file, the resulting waveforms are combined to produce:

   a. A waveform describing the outputs of a tester pin connected to a driver - DUT input waveform
   b. A waveform describing the inputs of a tester pin connected to a receiver - DUT output waveform

The DUT input and output waveforms are analyzed by the timing and vector setup extraction processes to produce:

Table 5-1. Data Extracted by Extraction Processes

DUT Waveform	Timing Extraction	Vector Extraction
DUT Input	Driver timing values	Driver vector values
DUT Output	Defaults	Receiver vector values

Generation of DUT Waveforms from Signal Waveforms

In the following table, the Case column refers to the descriptions after the table.

Table 5-2. Waveform Generation

Signal Type	DUT Waveform derived			Case
inp	DUT input waveform	only		1
out		only	DUT output waveform	2
inp + dir	DUT input waveform	only		3
inp + out	DUT input waveform	and	DUT output waveform	4
inp + dir + out	DUT input waveform	and	DUT output waveform	5
bid + dir	DUT input waveform	only		6

Terminology

inp(s, t) Signal state of signal *s* with signal type *inp* for time $\geq t$
out(s, t) same for signal type *out* for time $\geq t$
dir(s, t) same for signal type *dir* for time $\geq t$

INP(p, t) DUT input state of tester pin *p* for time $\geq t$
OUT(p, t) DUT output state of tester pin *p* for time $\geq t$

map(i, w) Tester state derived from signal state *w* according to the rules for mapping signal states of signals with signal type *inp*
map(o, w) same for signals with signal type *out*

map(d, w) same for signals with signal type *dir*

period period for pins with operation mode *100*

Case 1 - inp

$$INP(p,t) := map(i, inp(s,t))$$

Case 2 - out

$$OUT(p,t) := map(o, out(s,t))$$

Case 3 - inp + dir

$$INP(p,t) := map(i, inp(s,t))$$

Case 4 - inp + out

$$INP(p,t) := map(i, inp(s,t))$$

$$OUT(p,t) := map(o, out(s,t))$$

Case 5 - inp + dir + out

$$INP(p,t) := \begin{cases} \text{map(i, inp(s, t))} & \text{(if map(d, dir(s, t)) = 0)} \\ \text{tristate(Z)} & \text{(if map(d, dir(s, t)) = T)} \end{cases}$$

$$OUT(p,t) := map(o, out(s,t))$$

Case 6 - bid + dir

$$INP(p,t) := \begin{cases} \text{map(i, bid(s, t))} & \text{(if map(d, dir(s, t)) = 0)} \\ \text{tristate(Z)} & \text{(if map(d, dir(s, t)) = T)} \end{cases}$$

$$OUT(p,t) := \begin{cases} \text{map(o, bid(s, t))} & \text{(if map(d, dir(s, t)) = T)} \\ \text{don't care} & \text{(if map(d, dir(s, t)) = 0)} \end{cases}$$

6

Timing Setup Extraction

This chapter describes the timing setup extraction part of the EDA Interface software.

This function generates a timing setup file from the stimulus and response waveforms found in the EDA files. The timing setup file is used as an input parameter to the vector setup extraction part of the software, which is described in the next chapter. This process is shown in Figure 6-1.

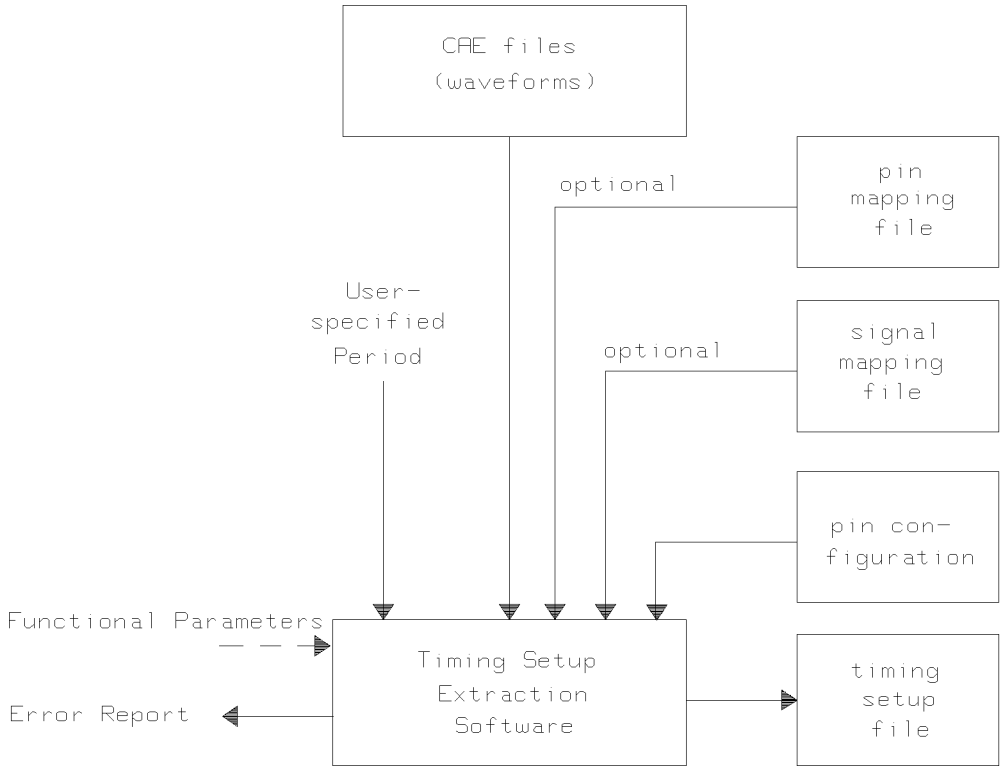


Figure 6-1. Timing Setup Extraction

Timing Setup Extraction attempts to derive a driver timing set and optional operation modes from the EDA file contents. Defaults are assigned to receiver timing settings.

Prerequisites

For a correct timing setup extraction, the following prerequisites must be fulfilled:

1. A pin configuration must exist before the software starts. This configuration file must contain at least a pin name and type for each DUT pin.
2. The information in the pin configuration file overrides all other inputs. If the software finds design pins that cannot be mapped onto pins in the pin configuration file, they will be ignored. The mapping of design pins to tester pins will be performed as described in the pin configuration extraction chapter.

Function

The function of the timing setup extraction software can be broken down into a number of steps:

1. Read the period from the EDA file or from the function parameter input.
2. Depending on the period, generate timing attributes for each pin specified in the pin configuration file. These attributes are:
   a. Format (RZ, NRZ, and so on)
   b. Edge placement (trailing and leading edges)
3. Depending on the extracted timing attributes:
   a. Generate an operation mode for all DUT pins whose timing attributes were extracted
   b. Optionally update the resource requests in the configuration file for all pins for which a mode request could be extracted.

Controlling Timing Extraction

There are options in timing extraction that allow you to:

- Specify a period to be used for the extraction - if it is not specified in the EDA files
- Update any pins specified in the pin configuration file whose operation mode are not consistent with that required for the extracted timing

There is also an option in the software which allows the number of test cycles to be chosen. Timing extraction will either extract all of the waveforms in the EDA files or you can choose timestamps at which the timing extraction should start and finish.

Special Cases

If no timing attributes can be derived for a pin, default values will be assigned to the tester pin. These defaults depend on the operation mode entry in the pin configuration file.

If no operation mode can be found for a pin, the operation mode entry in the pin configuration file will not be modified.

The default settings used for timing extraction depend on the operation mode specified for the pin.

Reading or Specifying the Period

If a period was specified in the Function Parameters menu, this value will be used to scan the signal waveforms during timing and vector extraction.

If no period was specified, the software will attempt to read a period value from the EDA files. The EDA files will be scanned from the first line to the first waveform for a period definition.

If no period value is specified or found in the EDA files, timing setup extraction will be terminated.

Updating Operation Mode

For each tester pin that requires a driver timing setup, the timing setup extraction process tries to derive an operation mode and timing parameters (format, leading and trailing edge) from the DUT input waveforms.

If no operation mode and driver timing can be extracted, a warning will be supplied and default values will be assigned for that pin.

ON	Values of operation mode for input pins that are extracted during Timing Extraction will override the operation mode values in the pin configuration.
OFF	The operation mode entry in the pin configuration will not be modified.

Timestamps Processed by Timng Setup Extraction

You can define a range of timestamps that are to be processed during the extraction process. All timestamps in the EDA log file that meet the equation:

smallest timestamp $\leq$ current timestamp $\leq$ largest timestamp

are processed.

The values for smallest and largest timestamp are determined by the entries in the respective fields in the Function Parameters dialog box.

Smallest Timestamp

The default smallest timestamp is defined as:

$$t_{sdef} = (\frac{t_s}{period} + 1) \times period$$

The smallest timestamp that can be defined is:

$$t_s = (\frac{t_{set}}{period} + 1) \times period$$

where

t_s	is the smallest timestamp in the log file
t_{set}	is the value entered in the Function Parameters dialog box for smallest timestamp

Largest Timestamp

The default largest timestamp is defined as the largest timestamp found in the file.

Timing Setup Defaults

If no timing setup file is specified, the following defaults will be used by the Timing Setup Extraction software.

Table 6-1. Defaults for Timing Setup Extraction

Pin Configuration Resource Request		Timing Setup Defaults		
Type	**Mode**	**Format**	**Leading Edge**	**Trailing Edge**
Driver	100	DNRZ	0	0
	200	DNRZ	0	-
	200MUX	RZ	0	-
Receiver	100	EDGE	90% of period	-
	200	EDGE	90% of (period/2)	-
	200MUX	EDGE	90% of (period/2)	-

7

Vector Extraction

This chapter describes the vector extraction function of the EDA Interface software.

This part of the software extract the vector information from the EDA file(s) and creates a vector setup file. This process is shown in Figure 7-1.

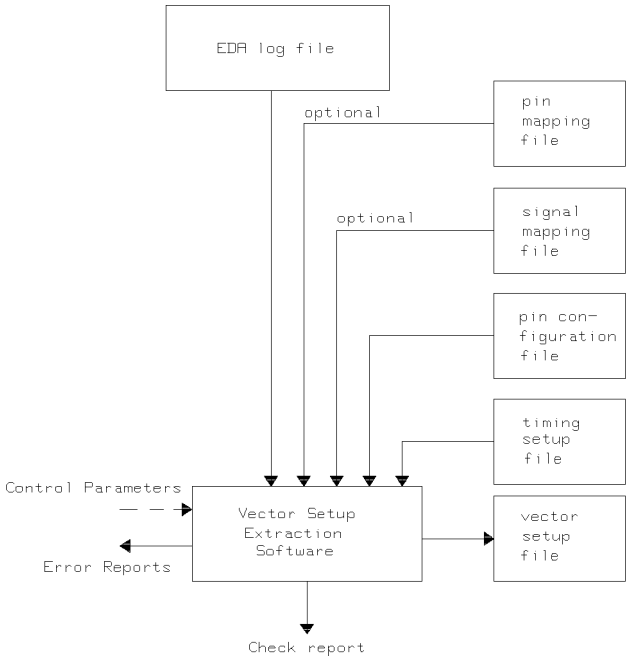


Figure 7-1. Vector Setup Extraction

Prerequisites

Before the vector extraction can take place, the folowing conditions must be met:

- A pin configuration must exist before the function is started. This file must contain, at least, the pin name, pin type, and operation mode for each DUT pin.
- The contents of the pin configuration file overrides all other inputs. Vectors will only be extracted for pins that are defined in the pin configuration file.
- A timing setup must exist before vector extraction starts. It must contain the pin names, formats, and leading and trailing edge values for each DUT pin in the pin configuration file.

Function of Vector Extraction

The vector setup extraction process can be broken down into four parts:

- Transform the signal waveforms to DUT input and/or output waveforms respectively
- Check the mapped waveforms against timing setup requirements and tester capabilities
- Sample the waveforms and generate vector values

There are two options that can be selected for the vector extraction process in the function parameter menu. These are:

- Waveform checking

  This is a waveform analysis tool to check whether the vectors in the EDA file are suited to the tester capabilities. If they are not, they will be modified to suit the tester and a report will be generated.

- Vector Setup Generation

  This option disables generation of the vector setup and tester suitability is not checked. The waveforms are sampled at a given sampling point in each period and the states that are detected at these times determine the final vector values.

Processing Signal Waveforms

Vector setup extraction processes the signal waveforms in the EDA log file in a similar manner to the timing setup extraction process in the previous chapter.

Vector setup extraction initializes the internal buffers used for DUT waveforms as shown in the table below. In order to avoid conflicts and translation errors, you should initialize the signal waveforms in the EDA file so that the initial values correspond to those listed below.

Table 7-1. Vector Setup Extraction Initial Values

Pin Configuration Operation Mode	Timing Setup Format	Initial DUT Waveform State
100	DNRZ	0
	RZ	0
	R1	1
	RC	0
	RI	0
	EDGE	X (don't care)
	WINDOW	X (don't care)
200	DNRZ	0
	EDGE	0
200MUX	DNRZ	0
	RZ	0
	R1	1
	EDGE	X (don't care)
	WINDOW	X (don't care)

Check Waveforms

The operation of this option is described below.

ON

The vector setup extraction process checks the DUT input and output waveforms against the contents of the current timing setup.

For DUT input waveforms it checks whether:

- State transitions occur at the edge placements allowed by the driver timing leading and trailing edges
- Tester states that are produced after state transitions are allowed by the current format and operation mode

For DUT output waveforms it checks whether state changes occur between the leading and trailing edges of a WINDOW format waveform.

If an error is found in the DUT waveforms, a warning will be generated.

Some defects may cause transitions to be ignored. This can result in differences in output files when this options is used. To ensure that all transitions appear in the output file, switch the Check Waveform option off.

If the DUT input waveform derived from the signal waveforms conforms exactly to a waveform that can be generated by the tester, there is no difference in the waveforms whether this option is used or not.

OFF

When the option is switched off, the DUT waveforms are produced by sampling the EDA file at sampling points dependent on the operation mode and timing setup in use. These are shown in Table 7-4, Table 7-2, and Table 7-3.

Table 7-2.
Vector Setup Extraction Sampling Points for 100/200 MHz Systems

Mode	Format	Edge Condition	Sampling Point
100	DNRZ	LE=TE	SP=LE+P_{100}/2
100	DNRZ	LE<TE	SP=(LE+TE)/2
100	DNRZ	LE>TE	SP=(LE+TE+P_{100})/2
100	R1, RZ, RC	LE<TE	SP=(LE+TE)/2
200, 200MUX	DNRZ	LE=TE	SP=LE+P_{200}/2
200MUX	RZ, R1	LE<TE	SP=(LE+TE)/2
all	EDGE	LE=TE	SP=LE
100, 200MUX	WINDOW	LE<TE	SP=(LE+TE)/2

Table 7-3.
Vector Setup Extraction Sampling Points for 400 MHz Systems

Mode	Format	Edge Condition	Sampling Point
200COM	DNRZ	LE=TE	SP=LE+P_{200}/2
200COM	RZ,R1	LE<TE	SP=(LE+TE)/2
400	DNRZ	LE=TE	SP=LE+P_{400}/2
400	RZ (ihs)	LE<TE	SP=(LE+TE)/2
200COM,400,400MUX	EDGE	LE=TE	SP=LE

Table 7-4.
Vector Setup Extraction Sampling Points for 50 MHz Systems

Mode	Format	Edge Condition	Sampling Point
25	DNRZ	LE=TE	SP=LE+P_{25}/2
25	DNRZ	LE<TE	SP=(LE+TE)/2
25	DNRZ	LE>TE	SP=(LE+TE+P_{25})/2
25	R1, RZ, RC, RI	LE<TE	SP=(LE+TE)/2
50, 50MUX	DNRZ	LE=TE	SP=LE+P_{50}/2
50MUX	RZ, R1	LE<TE	SP=(LE+TE)/2
all	EDGE	LE=TE	SP=LE
25, 50MUX	WINDOW	LE<TE	SP=(LE+TE)/2

In Table 7-4, Table 7-2, and Table 7-3:

P_{25} is the period of a 25 MHz pin

P_{50} is the period of a 50 MHz pin

P_{100} is the period of a 100 MHz pin

P_{200} is the period of a 200 MHz pin

P_{400} is the period of a 400 MHz pin

LE is the leading edge

TE is the trailing edge

SP is the sampling point

Generate Vector Setup

This option allows you to check an EDA file with the Check Waveform option without producing an output file.

ON	Vectors will be generated and stored in a file or downloaded to the tester hardware
OFF	No vectors will be generated. With Check Waveforms set to ON, this provides a check of the signal waveforms in the EDA file.

Timestamps Processed by Vector Setup Extraction

You can define a range of timestamps that are to be processed during the extraction process. All timestamps in the EDA log file that meet the equation:

smallest timestamp $\leq$ current timestamp $\leq$ largest timestamp

are processed.

The values for smallest and largest timestamp are determined by the entries in the respective fields in the Function Parameters dialog box.

Smallest Timestamp

The default smallest timestamp is defined as:

$$t_{sdef} = t_s$$

The smallest timestamp that can be defined is:

$$t_s = t_{sset}$$

where

t_s	is the smallest timestamp in the log file
t_{sset}	is the value entered in the Function Parameters dialog box

Largest Timestamp

The default largest timestamp is defined as:

$$t_{ldef} = t_l$$

The largest timestamp that can be defined is:

$$t_l = t_{lset}$$

where

t_l	is the largest timestamp in the log file
t_{lset}	is the value entered in the Function Parameters dialog box

The number of vectors that will be generated is defined by:

$$(\frac{t_{lset} - t_{sset}}{period}) + 1$$

Vector Setup Defaults

The following defaults are used by the Vector Setup Extraction software if no vector can be found or translated in a specific cycle.

Table 7-5. Input Vector Defaults

Operating Mode	Format in use	Default Vector
100, 200, 200MUX	RZ	0
	R0	0
	RI	0
	RC	0
	DNRZ	0

Table 7-6. Output Vector Defaults

Operating Mode	Strobe in use	Default Vector
100, 200MUX	EDGE or WINDOW	X
200	EDGE	0

8

System Networking Tools

This chapter describes the use of the system networking tools that are used to communicate with other computer systems.

The system software contains utilities that permit communication with other computers via the computer LAN, HPIB and RS-232 interfaces. The interface you use depends on the computer with which you want to communicate.

Communicating with an HP 9000 Computer

If the remote computer is another HP 9000 running under HP-UX, you can use the LAN interface to communicate with it. There are a number of ways of doing this, refer to the Networking Services 9000 manual for more information.

Procedure

Setting Up the Communications

1. Ensure that the NS-ARPA Network Services are installed on the remote computer.
2. Make the following changes on the remote computer:
   a. Login as root.
   b. Modify the file `/etc/hosts` to add an entry for the local computer.
   c. Make a special file in the `/net` directory for your local computer. The command to do this is:

      ```
      mknod remote_system_name n remote_system_name_complete_address
      ```
   d. Log off the remote computer.
3. Make the following changes on the local computer:
   a. Login as root.

b. Modify the file `/etc/hosts` to add an entry for the remote computer.
c. Make a special file in the `/net` directory for the remote computer. The command to do this is:

   `mknod local_system_name n local_system_name_complete_address`
d. Log off the remote computer.

4. Once all these changes have been made, reboot both computers and start the Network Services software.

Using Networking Services

If both computers are powered-up and the networking software is running, you can communicate with the remote computer.

If you just want to have access to the file system of the remote computer, you can use the `netunam` command. You should type in:

`netunam /net/name_of_remote_system user_name:`

where `name_of_remote_system` is the name you gave the system in the procedure above, and `user_name` is the name of a user on the remote computer. If there is a password associated ith the user, the system will prompt for it.

Once the remote system has been connected, you can access the required files on the remote system.

When you have finished communicating with the remote computer, you can close the connection by typing in:

`netunam /net/name_of_remote_system ""`

Example

If the remote system has the name `CAE_Stat` and there is a user on the system with the name `rick`, you would type in:

`netunam /net/CAE_Stat rick:`

To access his files, you need only insert the full pathname in front of the filename and you can than edit and copy files on the remote system. If the user has a subdirectory `sim_files`, you can copy any files in this directory by typing in:

`cp /net/CAE_Stat/users/rick/sim_files/filename .`

This would give you a copy of `filename` in the current directory. When you are working with the EDA Interface software, you do not need to physically copy

the files. If you enter the full pathname to the remote computer, the system will access the files as required.

Note

You must perform the netunam command **before** starting the hp82000 software. Entering the netunam command from the HP-UX shell in the system software will not work.

For more information on the Networking Services software, refer to the Networking Services manuals supplied with the system.

Communicating with a Mentor (Apollo) System

This section shows two methods of communicating with an Apollo workstation, using LAN and RS-232.

Communications using Local Area Network

Hardware Required

HP

HP 9000 computer with built-in LAN interface or HP 98643A LAN/300 LANIC and HP 28641 ThinMAU installed.

Mentor

Computer with AT-Bus, for example DN 4000, DN 5000, or servers DSP 3000 and DSP 4000.

or

Computer with Multibus with AT-Bus installed, for example, DN 570, DN 580, DN 590, or servers DSP 90, DSP 80 and DSP 500.

Additionally, an Ethernet controller is required for the computer.

Hardware Required

Tested Configurations

DN 4000 with HP 9000 Series 320 and 330 computers

Cables Required

1 × ThinLAN Cable, up to 125 m (410 ft). HP Part No. 92227A - 92227H
2 × BNC T-adaptor. HP Part No. 92227N
2 × Isolating Boot. HP Part No. 92227R
2 × 50 Ohm Termination. HP Part No. 92227P

Software Required

HP

HP-UX Version 6.01 or later (supplied with tester)

Mentor

AEGIS operating system Version 9.5 or later
TCP/IP software

or

UNIX running on the Apollo workstation

Hardware Connections

HP

Install the ThinLAN interface, if not already installed. Refer to the Installation Manual supplied with the LAN card.

Mentor

Install the Ethernet controller as described in the relevant Apollo manuals.

Connections

Connect the hardware as shown in the figure below.

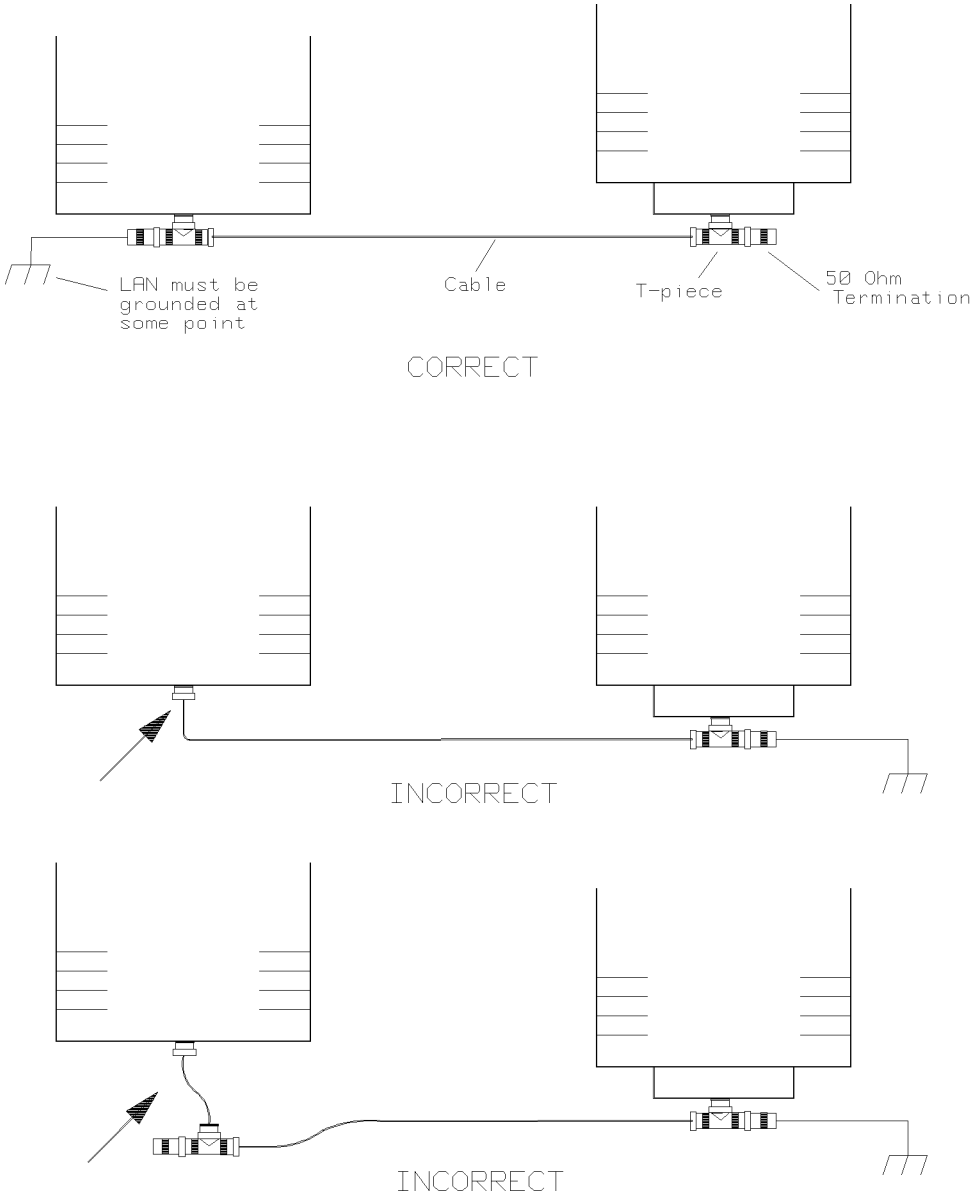


Figure 8-1. Mentor Connection Details

Note the following:

1. The T-adaptors **must** be used to ensure correct termination
2. The 50 Ohm terminators **must** be connected at each end of the cable.

Setting Up the Software

HP

The software should have been installed at system installation time. If it is not already running, refer to the NS/9000 manuals supplied with the system.

Mentor

Ensure that the TCP/IP software is installed. If not, refer to the manuals supplied with the Apollo system.

Note

To perform the following steps you need to know how to use the Apollo Editor. You should also be logged onto the system as super user.

File //<node_name/sys/tcp/hostmap/local.txt

1. Edit the file. The required syntax is shown in the file.
2. Modify the file using the following listing as a guide.

```
NET:1.0.0.0:ETHERNET:  ;Comments preceded by ";"
;
; Network address for Class 1, Network Name=ETHERNET
;
HOST:127.0.0.1:LOCALHOST
;
; To allow the system to access itself
;
HOST:1.0.1.1:cae_apol:<cpu_type>:AEGIS:TCP/Telnet,TCP/FTP:
;
; Name of the Apollo system is cae_apol. cpu_type could be
; DN 4000 etc. Operating System AEGIS, Protocols to use TCP
;
HOST:1.0.1.2:pws_statUNIX:TCP/TELNET,TCP/FTP,TCP/SMTP:
```

```
;
; Name of the HP system is pws_stat. cpu_type - ignored
; Operating System UNIX, Protocols to use TCP ...
;
```

3. Save the file.

File //<node_name>/sys/node_data/thishos

This file contains only one line containing the name of the Apollo workstation, for example, `cae_apollo`.

File //<node_name>/sys/node_data/networks

This file contains the names of the networks that are accessible from this machine. For our exemple, this should should contain the following:

```
1.0.1.1 on eth0
127.0.0.1 on lo0
```

1.0.1.1 is the address of the Apollo workstation on network 1.0 which is called eth0. The second line should be entered to allow access to the Apollo workstation.

Starting the Software

Input the following command to start the networking software

```
execute /sys/tcp/hostmap/makehost.sh
```

Starting the Networking Services

From the display manager, execute the following commands:

```
cp /sys/tcp/tcp_server
cps /sys/tcp/ftp_sever
cps /sys/tcp/telnet_server
```

These commands can also be written into the Startup file, but you should ensure that the commands do not follow directly after each other. FTP and telnet services require that the preceding services are already running before they can be started.

Transferring Files

In the following examples, the following conditions apply:

local_host = HP, hostname = PWS_Stat
remote_host = Mentor (Apollo), hostname = cae_apol

Transferring Files with the HP System

Note

This method will also work in the opposite direction when the Unix operating system is running on the Apollo computer.

To transfer a file from the Apollo system to the HP system the ftp service can be used.

1. Login on the HP 82000 controller
2. Start the ftp protocol by typing in:

   `ftp cae_apol`

   The ftp server will start and ask for a password for the login on `remote_host`. The prompt will change to `ftp>`.
3. Use the `get` command to copy the required files to `local_host`.

   `get remote_file [local_file]`

   If you do not specify a local filename the new file will have the same name as on the Apollo system.
4. Stop the ftp server and logout of the remote system.

   Type in `bye` or `quit`.

Further useful ftp commands are:

`dir`	lists the current directory on the remote system
`cd <remote_dir>`	changes the working directory on the remote system
`!<UX cmd>`	executes a command on the local host
`!pwd`	shows the current working directory on the local host.

For more information on ftp, refer to the networking manuals supplied with the test system or the Apollo system.

Transferring Files from the EDA System

To transfer a file from the Apollo system to the HP system the ftp service can be used.

1. Login on the Apollo workstation.
2. Start the ftp protocol by typing in:

   ```
   ftp PWS_Stat
   ```

   The ftp server will start and make a connection to the remote system.
3. Login into the remote system. Type in:

   ```
   log
   ```

   The system will then ask for a user name and password. Enter a valid user name for the remote system and a password if required.

   The prompt will change to `>`.
4. Define the file type to be transferred as ASCII. Type in:

   ```
   type ascii
   ```
5. Prepare the system to transfer files. Type in:

   ```
   structure file
   ```
6. Copy the files using the send command.

   ```
   send <local_file> [remote_file] If you do not specify a remote
   filename, the file will have the same name on the remote system.
   *Stop the ftp server and logout of the remote system. Type in bye.
   ```

Further useful ftp commands are:

`nlist`	lists the current directory on the remote system
`cwd <remote_dir>`	changes the working directory on the remote system
`wd <remote_dir>`	changes the working directory on the local system

For more information on ftp, refer to the networking manuals supplied with the Apollo system.

Possible Errors

The following list shows the most common problems associated with file transfer to and from the Apollo workstation.

Problem	Cause	Solution
Connection cannot be made to remote system	The software is not correctly installed	Check and correct software.
	Cables not properly connected or terminations missing	Check cables and terminations
Cursor not present or unpredictable behavior (Apollo)	Nested ftp or telnet calls	Open a separate window for each server call.
Incorrect displays or incorrectly executed commands	Control characters in text or incorrect use of Backspace	Do not use control characters in text input.
Errors in file transfer, incorrect data	Structure and type of transferred data not properly defined	Use the type and structure commands to set correct values. HP defaults are correct.

Communications using RS-232C

Hardware Required

HP

HP 9000 computer with RS232C Interface

Mentor

Apollo workstation with RS232C Interface, for example a DN 560

Cabling

Standard RS-232C Cable with pins 2 and 3 crossed (HP Part No. 12342G)

Software Required

HP

HP-UX operating system, Version 6.01 or later.

Apollo

AEGIS/DOMAIN operating system with `emt` terminal emulator installed.

Setting up the Software

HP

Note

You must be logged on as super-user (root) to perform some of the following steps. This procedure only needs to be performed once.

1. If not already installed, install the RS-232C Interface Card in the controller as described in the Installation Manual supplied with the card. Check the System Support Log to ensure that no other cards are installed with the default address 9. If there is another card with this address, change the setting of the card to an unused number. The following description assumes that address 9 was used.
2. Create a special file for a terminal connection. Type in:

```
mknod /dev/name c 1 0xYY0004
```

where `name` is the name you want to use for the remote system and `YY` is the address set in the previous step in hexadecimal. For the example this would be `09`.

3. Modify the the `/etc/inittab` file to activate the terminal connection. The entry should look like the following:

   ```
   XX:2:respawn:/etc/getty name 1200 15
   ```

   where `XX` is the next unused number and `name` is the special file created above.

   The `inittab` file accesses the file `/etc/gettydefs` to read a definition of the label `baudrate`.
4. Modify `/etc/gettype` to add the following line.

   ```
   1200# B1200 HUPCL PARENB CS7 # B1200 SANE PARENB CS7 ISTRIP IXANY
   TAB3 #login: #300
   ```

   Make sure you enter this string as one line.

   For more details on the contents of these files and eventual changes to newer HP-UX revisions, use the `man` command.
5. Modify the `.profile` file of all users who will be using this link by adding the line

   ```
   stty eof "^D"
   ```

Apollo

Note

You will have to set `raw` and `outterm` each time that you run `emt`. Depending on your configuration, some of the other settings may already be set.

Do not use the Backspace key while working with the terminal emulator.

1. Connect the two computers using the RS-232C cable.
2. Start the terminal emulator by typing in:

   ```
   emt
   ```
3. Select the interface to use by typing in: `line n`

   where `n` is 1, 2, or 3 for the interface in use.
4. Set the baudrate to 1200, even parity, seven bits per character, and two stop bits. Type in:

```
tctl -1200
tctl -parity even
tctl -bpc 7
tctl -stop 2
```

5. Set raw mode. Type in: `raw`
6. Select outterm to be `cr`. Type in: `outterm cr`
7. Use the `tctl` command to check these settings and correct that are wrong.

Note

The baud rate is set to 1200 because `etm` does not provide any handshakes. Operating at higher speeds could cause loss of data.

If you want to check a transferred file, transfer the file to the HP 9000, transfer it back with a new name and then use the `cmf` command to compare the two files.

Transferring Files

File transfers take place from the Apollo workstation.

Transferring Files to the HP 9000

1. Start the terminal emulator by typing in `emt`
2. Use function key `f1` to switch to remote and press Return to get the login prompt.
3. Login to the HP 9000 and enter a password, if required.
4. Prepare the HP 9000 to receive data by typing in:

   ```
   cat > remote_filename
   ```

   where `remote_filename` is the filename you will create on the HP 9000.
5. Switch back to local by pressing `f1`.
6. Prepare the Apollo workstation to send the data by typing in:

   `xmit local_filename` where `local_filename` is the filename on the Apollo system.
7. Press `f1` to start the transfer. When the file transfer is complete, type in:

   `^D`Return

   `^D` is the end of file marker defined in the `.profile` file.
8. When all the file transfers are complete, type in `^D` to logout of the HP 9000 and `quit` to stop the terminal emulator.

Transferring Files from the HP 9000

1. Start the terminal emulator by typing in `emt`
2. Use function key `f1` to switch to remote and press (Return) to get the login prompt.
3. Login to the HP 9000 and enter a password, if required.
4. Prepare the HP 9000 to transmit data by typing in:

   `sleep n ; cat remote_filename`

   where `n` is the delay in seconds before the start of the transfer, and `remote_filename` is the name of the file to be transferred. You should select `n` so that you can perform the following step before the transfer starts.
5. Prepare the Apollo workstation to log data. Press `f1` to go to local mode. Type in:

   `rcv local_filename`

   where `local_filename` is the name for the new file. Press `f1` followed by (Return).
6. When the file transfer is complete, press `f2` to close the new file.
7. When all the file transfers are complete, go into remote and type in `^D` to logout of the HP 9000. Return to local mode and type in `quit` to stop the terminal emulator.

Note

The HP 9000 appends a prompt to the end of the file when it is transferred. This should be removed with an editor before the file is used.

Communicating with a Daisy System

The following description shows how to connect the HP 9000 computer to a ThickLAN network connecting a number of Daisy or Sun workstations.

Hardware Required

HP

HP 9000 computer with built-in LAN interface or HP 98643A LAN/300 LANIC and HP 28641 ThinMAU installed.

Daisy/Sun

Daisy Logician™ 386
Sun Series 3 and Series 4 Workstations

Tested Configurations

The following computers were used to provide this description:

Daisy Logician™ 386 and Sun Series 3 and Series 4 Workstations connected to an HP 9000 Series 300 computer.

Cables Required

The connections were tested using the following connections to the ThickLAN:

Medium Attachment Unit (MAU) with Coaxial cable, HP Part No. 30241A
LAN Connection Module and Cable (AUI), HP Part No. 92254A to 92254H (length of cable 6 - 48 m (20 - 160 ft)
Thick LAN Cable (Yellow) HP Part No. 92253A - 92253H (length 4 - 500 m (12 - 1500 ft)

and

Interlan NT1000 Transceiver and cable (blue)

Software Required

HP

HP-UX Version 6.01 or later (supplied with tester)
NFS Service for the HP 9000 (supplied with tester)

Daisy/Sun

Daisy workstation used with DNIX operating system Sun workstations used with Sun-OS (Berkeley-UNIX 4.2)

Both workstations use the TCP/IP protocol with NFS additionally available on the Sun workstations.

Hardware Connections

HP

Install the ThinLAN interface, if not already installed. Refer to the Installation Manual supplied with the LAN card.

Connect the MAU to the Thick LAN cable using a LAN Connector, HP Part No. 30241B. Note that when the computer and MAU are removed from the LAN, the connector must remain on the LAN cable.

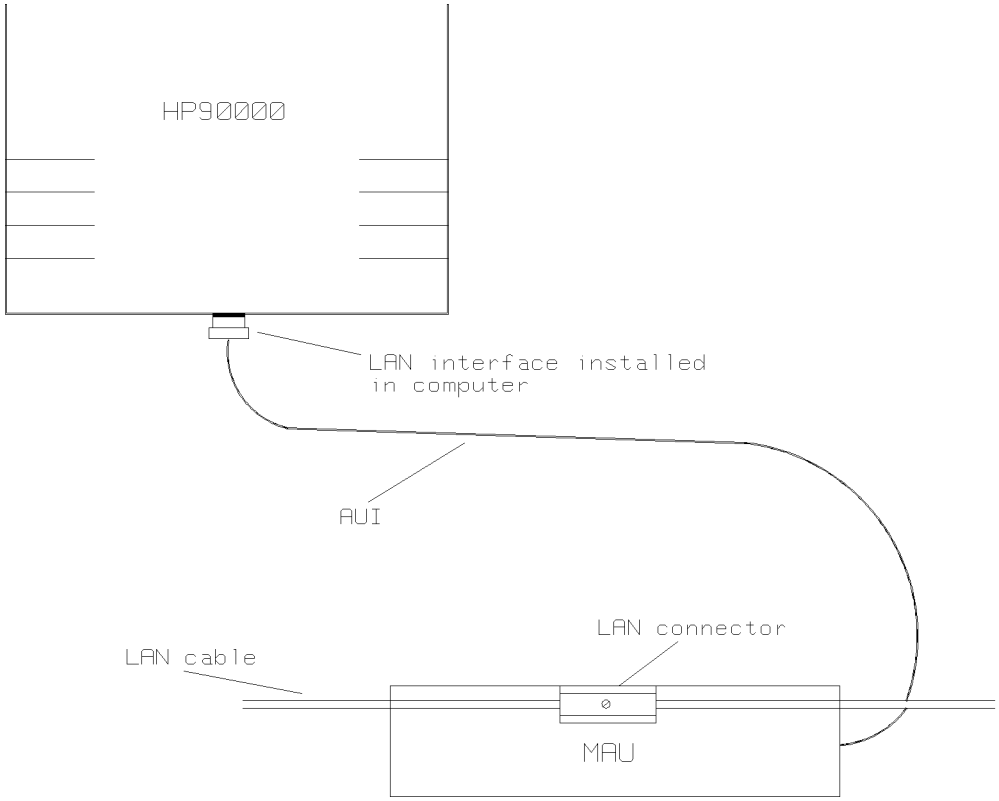


Figure 8-2. Connection Details for ThickLAN

Daisy/Sun

Connect the workstation to the LAN as described in the manuals supplied with the workstation.

Setting Up the Software

HP

The software should have been installed at system installation time. If it is not already running, refer to Appendix XXX and the NS9000 manuals supplied with the system.

Login as root and add the system name and net address to the /etc/hosts file.

Daisy/Sun

Ensure that the TCP/IP software is installed. If not, refer to the manuals supplied with the appropriate system.

Note

To perform the following steps you should be logged onto the system as super user.

File /etc/hosts

- Edit the file. The required syntax is shown in the file.
- Modify the file to include the name and net address of the HP 82000 controller.
- Save the file.

Starting the Networking Services

Refer to the appropriate manuals supplied with the system.

Transferring Files

To transfer a file from one system to the other, use the following procedure: be used.

1. Ensure that the Networking Services programs are installed and running on both systems.
2. Login on one of the systems.
3. Start the ftp protocol by typing in:

   `ftp <remote_system_name>`

   or

   `ftp <remote_system_net_address>`

   where `<remote_system_name>` is the name you entered in the etc/hosts file and `<remote_system_net_address>` is the net address.

   The ftp server will start and ask for a password for the login on `remote_host`. The prompt will change to `ftp>`.
4. Use the `get` command to copy the required files to `local_host`.

   `get remote_file [local_file]`

If you do not specify a local filename the new file will have the same name as on the Apollo system.

5. Stop the ftp server and logout of the remote system.

   Type `quit`.

Further useful ftp commands are:

`mget`	gets a group of files defined by wildcard symbols
`put` and `mput`	sends files to the remote system
`dir`	lists the current directory on the remote system
`cd <remote_dir>`	changes the working directory on the remote system
`!<Unix command>`	executes a command on the local host
`!pwd`	shows the current working directory on the local host.

For more information on ftp, refer to the networking manuals supplied with the test system or the EDA workstations.

Possible Errors

The following list shows the most common problems associated with file transfer to and from the Apollo workstation.

Problem	Cause	Solution
Connection cannot be made to remote system	The software is not correctly installed	Check and correct software.
	Cables not properly connected or terminations missing	Check cables and terminations
System cannot find remote system when `remote_system_name` is used.		Retry using system net address.
Cannot read or write files to/from remote system	Incorrect file access capabilities over network	Check and correct file access rights for required files
Incorrect displays or incorrectly executed commands	Control characters in text or incorrect use of Backspace	Do not use control characters in text input.

Other Networking Tools

The networking software included wth the HP 82000 system includes a number of networking services not mentioned in the descriptions above. These include:

telnet
netunam
nfs
rsh
rlogin

Check the manuals supplied with your EDA workstation and the Networking Services manuals supplied with the HP 82000 system to see whether any of these products can be used.

In Case of Difficulty

If you have any problems using the Networking Services software, refer to the Diagnostics section of the “LAN Node Manager’s Guide”.

9

Linking to Other EDA Systems

The purpose of this chapter is to guide you through the process of translatng files not directly supported by the EDA Interface software.

The EDA Interface Software

The figure below shows the EDA Interface used with the HP82000 IC Evaluation System. Boxes enclosed in solid lines represent it's standard features.

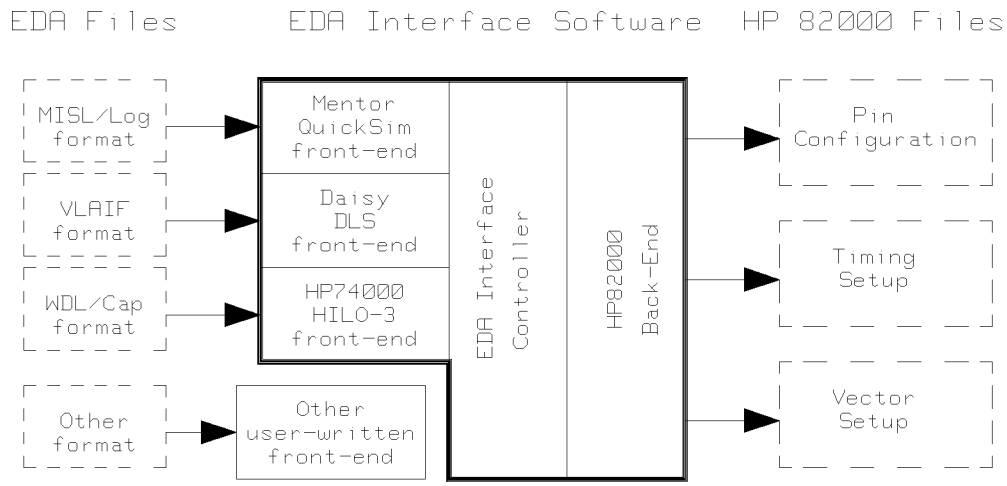


Figure 9-1. EDA Interface Structure

The standard software can directly translate simulator input/output files generated on three different systems. These system are:

- Mentor QuickSim

- Daisy DLS
- HP 74000 HILO-3

Files generated using these simulators can be directly translated into HP 82000 pin configuration, timing setup, and vector setup files. These translators can be called up directly from the EDA Interface Main Screen when they have been installed in the software.

Translating Files from Other Simulators

If you have simulator files that have been generated by another simulator, there are two methods you can use to translate the EDA files into HP 82000 pin configuration, timing setup and vector setup format. You can:

- Write a dedicated front-end for the simulator
- Convert your simulator files to a format that can be understood by the EDA Interface software.

Writing a Dedicated Front-End

You should choose this method if:

- You often need to translate files from the simulator to HP 82000 format
- Your simulator file format is different to the Mentor QuickSim Log, Daisy VLAIF, or HILO-3 formats
- You require the highest possible speed for your translations
- You have some programming experience

If you choose this method you should:

1. Order the E1295 Programmer's Toolkit which will help you write a new EDA Interface. You will have to write about 100 to 500 lines of 'C' code. It should take less than one week.
2. Write, debug, and install the new EDA Interface.
3. Translate your simulator files by invoking the new EDA Interface directly from the EDA Interface main screen.

Converting the Simulator Files to a Supported Format

You should choose this method if:

- You do not need to translate simulator files very often
- Your simulator file format is similar to the Mentor QuickSim Log, Daisy VLAIF, or HILO-3 file formats
- You have some utilities to perform the conversion quickly

If you choose this method you should:

1. Read the following chapters to decide which of the file formats is similar to your format.
2. Convert your simulator files to the selected format.
3. Translate your translated simulator files by invoking the appropriate EDA Interface.

Note

For assistance in writing new EDA Interfaces or utilities to convert simulator files to supported formats, contact your local HP representative.

10

Mentor QuickSim Description

This chapter describes the parts of the Mentor Graphics QuickSim™ simulator language that are recognized by the EDA Interface software.

QuickSim Statements

The Simulator uses two types of files:

- MISL (Mentor Interactive Stimulus Language) files which contain stimulus data
- Logfiles which contain the resulting input and output waveforms generated in the simulation

The following table shows the MISL and Logfile statements that are recognized by the EDA Interface. Statements that are not listed here are ignored by the EDA Interface.

Table 10-1. Accepted QuickSim Statements

Instruction	Format	Comments
INPUT	MISL	Input design pin definition
OUTPUT	MISL	Output design pin definition
BUS	MISL	Bidirectional design pin definition
VECTOR	MISL	Design group definition
TIMEDEF	MISL	Cycle time (clock period)
D	Logfile	Design pin/group name to surrogate assignment
T	Logfile	Time stamp
S	Logfile	State transition of pin/group
U	Logfile	Undefine of pin/group
A	Logfile	Undefine all pins and groups

The tester setups are generated from the information extracted from the following statements.

Table 10-2. Source Information for Setup Generation

Tester Setup	Simulator Statements
Pin Configuration	
- Pin Names	INPUT, OUTPUT, BUS, VECTOR, D
- Pin Types	INPUT, OUTPUT, BUS
- Pin Operating Mode	S, T
Timing Setup	
- Period	TIMEDEF
- Formats	S, T
- Edge Placements	S, T
Vector Setup	
- Vectors	S, T

Command Description

This section describes the Mentor statements and their usage.

MISL Statements

INPUT

The INPUT statement is used to define input design pins. The syntax is:

`INPUT <pin_name_1>, <pin_name_2>, ... , {pin_name_n};`

Pin names are legal DOMAIN Pascal identifiers. An identifier may be any length between 1 and 255 characters. The syntax for ‘`pin_name` is as follows:

<pin_name>	is a string of up to 255 characters commencing with a letter and containing:_, $, <, (, /, >, },), *, =, +, &, -, or letter_digit
<letter_digit>	letter or digit
<letter>	'a' .. 'z' and 'A' .. 'Z'
<digit>	'0' .. '9'

OUTPUT

The OUTPUT statement is used to define output design pins. The syntax is:

```
OUTPUT <pin_name_1>, <pin_name_2>, ... , {pin_name_n};
```

BUS

The BUS statement is used to define pins that form a bus and will have similar timing and format characteristics. The syntax is:

```
BUS pin_name_1, pin_name_2, ... , pin_name_n;
```

TIMEDEF

The TIMEDEF command is used to define the cycle time for the simulation. It can be expressed either as a frequency or a period. If more than one TIMEDEF statement is encountered, the last TIME value will be used. The DRIVE and SENSE part of the statement will be ignored. The syntax is:

```
TIMEDEF PERIOD=<time> | TIMEDEF FREQ=<frequency>
```

VECTOR

The VECTOR statement is used to define pins that form a group (see also D-statement). The syntax is:

```
VECTOR <group_name>=<pin_name_1> ... <pin_name_n>;
```

Logfile

T

The T-statement defines the current timestamp. The time unit is nanoseconds. The syntax is:

```
T<time>
```

For example, `T 4.7` indicates an event at 4.7 ns after the start of the simulation

D

The D statement maps a pin or group name to a surrogate and sets the initial states. The surrogate is an integer that is used to identify the signal for the remainder of the simulation. The syntax is:

`D<pin_name> <surrogate> <state>` or
`D<group_name> <surrogate> <state>`

For example `D /Q 23 0` assigns the name Q to the surrogate 23 which will start in state logical low. In later statements, any S statements with the surrogate 23 will indicate state changes of the signal Q.

S

The S statement sets a new state to a signal. The preceding timestamp indicates when the change of state takes place. The syntax is:

`S <surrogate> <state>`

For example, `T 20.0` followed by `S 23 1` can be read as signal Q changing to logical high 20.0 nanoseconds after the start of the simulation.

States that are allowed in Qucksim are: H, *, L, O, U, I, 1, X, 0, @, < and >.

U <surrogate>

The U statement is used to undefine a surrogate.

A

The A statement is used to undefine all surrogates.

Default Pin Name Mapping

The first slash (/) character in a pin name will be ignored. Pin or group names may be up to 255 characters in length, but only the first 16 are significant.

Note This may lead to identical design pin or group names.

Expanding Design Groups

If design groups could not be extracted from the MISL file (VECTOR statements), the EDA Interface software tries to expand design group definitions using the D statements in the logfile.

The group pin name extracted will have the syntax

`group_name'_'number`.

For example, if the statement `D ABC 123 UUUU` is found in the logfile, the design pins ABC_0, ABC_1, ABC_2, and ABC_3 will be extracted.

Note Signal names extracted from Mentor files are always converted to names in upper case (characters A .. Z). For example `Pin_1` will be converted to `PIN_1`.

Recommended State Mapping Conventions

The table below shows the recommended mapping of Mentor simulator states to HP 82000 tester states.

Table 10-3. Recommended Mentor State Mapping

Input Design Pins		Output Design Pins	
Simulator State	**Vector Value**	**Simulator State**	**Vector Value**
0, O, L, <	0	0, O, L, <	0
1, I, H, >	1	1, I, H, >	1
X, U, *, @	T	X, U, *, @	X

Prequisites and Restrictions

Bidirectional pins

Since the QuickSim simulator logs the state of a bidirectional pin at any given time, it cannot be detected whether the pin is in the input or output state. In this case, a direction control pin should always logged along with any bidirectional pins.

11

Daisy Logic Simulator Description

This chapter describes the parts of the Daisy Logic Simulator language used on the Daisy Logician™ system that are recognized by the EDA Interface software.

Daisy Logic Simulator Statements

The simulator files in VLAIF (Virtual Logic Analyzer Intermediate Format) which contain waveforms logged during a simulation

The following table shows the VLAIF statements that are recognized by the EDA Interface. Statements not listed here are ignored by the EDA Interface.

Table 11-1. Accepted Daisy Simulator Statements

Instruction	Format	Comments
$DATA_HEADER$	VLAIF	Beginning of data header section
END	VLAIF	End of data header section
$FORMAT$	VLAIF	File format
$TYPE$	VLAIF	File type
$FIELD$	VLAIF	Logged signal definitions
$TOTAL_COLUMNS$	VLAIF	Number of columns in the data section
$BASE$	VLAIF	Base used to specify time and value representation in the data section
<data_line>	VLAIF	Time and state values in the data section.

The tester setups are generated from the information generated from the following statements.

Table 11-2. Source Information for Setup Generation

Tester Setup	Simulator Statements
Pin Configuration	
- Pin Names	$FIELD$, $TOTAL_COLUMNS$
- Pin Types	-
- Pin Operating Mode	<data_line>
Timing Setup	
- Period	–
- Formats	<data_line>
- Edge Placements	<data_line>
Vector Setup	
Vectors	<data_line>

File Format Description

This section describes the Daisy statements and their usage.

VLAIF Statements

The VLAIF file consists of two sections, a header section and a data section.

The header section contains control information concerning the representation of data in the data section. The following statements are interpreted.

Header Section

$DATA_HEADER$/END

The data header section begins and ends with these statements respectively.

$TYPE$

Two file types are recognized by the EDA Interface. The statement syntax is:

```
$TYPE$
<type>
```
where <type> may be either I/O which is the DLS log file type, or VLA which is the DLS input file type for circuit analysis.

$FORMAT$

Two file formats are accepted by the EDA Interface. The syntax of this command is:

```
$FORMAT$
<format>
```

where <format> can be either TIME_VALUE or SNAP_VECTOR.

TIME VALUE

The data section consists of a simulation time and status part. The simulation time value in a data line is the current timestamp at which one or more signals have changed state. The new states are stored in the states part of the line at the position determined by the $FIELD$ statements.

SNAP_VECTOR

The data section consists of a states part only. A line of state values is given for each time unit in ascending order, starting at time 0.

$FIELD$

This statement assigns signals or groups of signals to each column of the states part of the data section. $FIELD$ statements are optional. The syntax is:

```
$FIELD$ <field>
<circuit_name> : <signal_name> <column>
```

or

```
$FIELD$ <field>
<circuit_name> : <group_name> <start_column>-<stop_column>
```

where <field> is the current $FIELD$ statement number starting with 1. <circuit_name and <signal_names> together denote the signal. States for this signal are logged in the states part at column <column>. Groups of signals can also be grouped together.

$TOTAL_COLUMNS$

This statement defines the number of columns in the simulation time and states parts of the data section. The syntax is:

```
$TOTAL_COLUMNS$
[<time_columns>]<state_columns>
```

The `<time_columns>` specifier is used with the `TIME_VALUE` file format only.

$BASE$

Describes the representation of values in the simulation time and states part of the data section. The syntax is:

```
$BASE$
[<time_base>]<state_base>
```

Time values may be represented in binary (''B'') or decimal (''D'') base defined by `<time_base>`. States may only be represented in binary format (each signal is associated to one column in the states part). The `<time_base>` specifier is used for `TIME_VALUE` format only.

Data Section

The data section is described by the statements of the header section (see above). The allowed state values of the states part are: *, 1, 0, C, S, R, F, #, H, L, c, s, r, f, Z, +, -, |, =, /, (space), U, !, @, ", <, and >.

Default Pin Name Mapping

Only the first sixteen characters of a pin name are recognized, the rest are ignored.

Note

This may lead to identical design pin or group names, since the circuit name is used as a prefix to determine a signal or group of signals. Use the pin mapping features to avoid name collisions.

Expanding Design Groups

Groups are expanded from $FIELD$ statements. The group member names have the syntax

`<group_name>_<number>`

where number starts at 0.

Recommended State Mapping Conventions

The table below shows the recommended state mappings for the Dasiy DLS simulator.

Table 11-3. Recommended Daisy State Mapping

Input Design Pins		Output Design Pins	
Simulator State	**Vector Value**	**Simulator State**	**Vector Value**
0, O, L	0	0, O, L, <	0
1, 1, H	1	1, I, H, >	1
*, #, Z, U, +, !, -, @, C, c, |, <space>, S, s, =, ", R, r, /, <, F, f, >	T	*, #, Z, U, +, !, -, @, C, c, |, <space>, S, s, =, ", R, r, /, <, F, f, >	X

Prequisites and Restrictions

Bidirectional Pins

Since the Daisy simulator logs the state of a bidirectional pin at any given time, it cannot be detected whether the pin is in the input or output state. In this case, a direction control pin should always be logged with any bidirectional pins.

Waveform Sampling

The period used to sample the logged waveforms cannot be extracted from the CAE files. It must be input via the Timing Setup Extraction options.

12

HILO-3 Description

This chapter describes the parts of the HP74200 HILO-3 simulator language that are recognized by the EDA Interface software.

HILO-3 Simulator Statements

The simulator uses two types of files:

- WDL (Waveform Description Language) files which contain pin and group definitions
- Capfile format files which contain the results of the simulation.

The following table shows the WDL and Capfile statements that are recognized by the EDA Interface. Statements not listed here are ignored by the EDA Interface.

Table 12-1. Accepted HILO-3 Simulator Statements

Instruction	Format	Comments
STIMULUS	WDL	Input design wire definition
RESPONSE	WDL	Output design wire definition
BIDIRECTIONAL	WDL	Bidirectional design wire definition
DELAYSCALE	Capfile	Simulation time unit
FINISH	Capfile	End of simulation
EOF	Capfile	End of file
= <carrier_ declaration>	Capfile	Declaration of carriers of type: EVENT, INPUT, OUTPUT, PTERM, REG, TRI, WAND, WOR, WIRE, UNID
# <time_value>	Capfile	Time value
< <circuit_name>	Capfile	Begin sub-circuit simulation level
>	Capfile	End of sub-circuit simulation level
0, 1, X, Z, E, L, H, W, T, B, P, N, R, F, U, D	Capfile	Carrier state transitions

The tester setups are generated from the information extracted from the following statements.

Table 12-2. Source Information for Setup Generation

Tester Setup	Simulator Statements
Pin Configuration	
- Pin Names	STIMULUS, RESPONSE, BIDIRECTIONAL, =
- Pin Types	STIMULUS, RESPONSE, BIDIRECTIONAL, =
- Pin Operating Mode	#, =
Timing Setup	
- Period	-
- Formats	#, =
- Edge Placements	#, =
Vector Setup	
Vectors	#, =

File Format Description

This section describes the HILO-3 statements and their usage.

WDL Statements

In the following descriptions, `<pin_name>` identifies an input or output pin or terminal. `<constant>` specifies a numeric constant with the syntax:

$$\left[\begin{array}{l}\texttt{BIN}\\ \texttt{HEX}\\ \texttt{OCT}\\ \texttt{DEC}\end{array}\right] \texttt{<number}$$

`<bus_name>` specifies a bus (group of wires).

`<i>,<j>` specify the upper and lower bounds of a bus where both `i` and `j` are positive integers (`i` < `j`).

Signals/pins in sub-circuits are identified by a name composed of:

```
<sub_circuit>:<signal_name>
```

STIMULUS

This statement is used to identify input wires or terminals and declares their initial values. It can also be used to define busses. This statement has the syntax:

```
STIMULUS <pin_name_1> = <constant> [ ... ];
```

or

```
STIMULUS <bus_name>[<i>:<j>] = <constant> [ ... ];
```

RESPONSE

This statement is used to identify output wires or terminals and declares their initial values. It can also be used to define busses. This statement has the syntax:

```
RESPONSE <pin_name_1> = <constant> [ ... ];
```

or

```
RESPONSE <bus_name>[<i>:<j>] = <constant> [ ... ];
```

BIDIRECTIONAL

This statement is used to identify pairs of identifiers (a STIMULUS and a RESPONSE) so that the simulator recognizes them as the same wire.

```
BIDIRECTIONAL <wire_identifier_1>=<wire_identifier_2> [ ... ];
```

where <wire_indentifier_1> is a dummy stimulus wire and <wire_identifier_2> is a real response wire.

Capfile Statements

The Data Capture File consists of a series of special instructions or commands. Each command begins with a special command character and ends with a space or line-feed (a terminator) unless otherwise stated. The file ends with the first occurrence of the special instruction `$EOF;`. Accepted command characters and special instructions are described below.

Command Characters

<

New level. The command starts the specification of a subcircuit simulation level. The syntax is:

`< <subcircuit_name>`

All characters up to but not including the terminator are part of the name.

>

End a level. The command ends a previous < command. A space or line feed must follow this command.

=

Carrier Declaration. The characters following this instruction define the initial value, index, type, and name of the characters. A semicolon (;) terminates this command. The syntax is:

`=<initial-value><index> <type> <identifier> <optional-data> ;`

Note the beginning (=) and ending (;) characters, and the necessary blanks after the index, type and, identifier specifiers.

`<initial-value>` is the initial state for the carrier as listed below.

`<index>` replaces the carrier name in the fourth part of the Data Capture file, and is a number to the base 94. It is represented by one or more ASCII printable characters. Printable characters are from 33 decimal (!) to 126 decimal (~).

`<type>` is a carrier type which can be one of `EVENT`, `INPUT`, `OUTPUT` for an output wire, `PTERM` for a subcircuit element, `REG`, `TRI`, `UNID` for an input wire, `WAND`, `WIRE`, or `WOR`. The initial value can be any one of the fifteen values listed below.

`<identifier>` is a wire or signal name. The following are all examples of valid identifiers: `WIREA`, `ACC[7]`, `MODA.4` `VMOD[3].2`. The identifier can be optionally preceded by < or followed by >. < indicates a subcircuit name, and > indicates the end of a subcircuit and continuation of the previous level before the last occurrence of <.

Using these characters allows the simplest possible statement of nested circuits, as in the following example.

```
<MODA
<EL[2]
=0%a unid WIREA;
<EL[3]
```

<optional-data> will always be ignored.

#

Time value. The command defines a time value using all the characters between the hash character and the next space or line terminator.

The syntax is: #<timestamp>

0, 1, X, Z, E, L, H, W, T, B, P, N, R, F, U, D

These commands change the carrier value to the specified level of the HILO-3 15-state logic set.

The syntax is, for example, 1<index>.

Special Instructions

The syntax of these commands is:

$<instruction> [<item>] <terminator>

where

<instruction> is one of the special instructions listed below.

<item> is only applicable to the instructions as listed below.

<terminator> is a space or line-feed.

The descriptions below indicate the use of the commands accepted by the EDA Interface and any definitions of <item> where applicable. Any other special instructions will be ignored.

$DELAYSCALE

This statement specifies the simulation time units for the contents of the file. The following table shows the values for <item> for the different ranges.

Value	Delayscale
12	1 ps
11	10 ps
10	100 ps
9	1 ns
8	10 ns
7	100 ns

$EOF

This statement defines the end of the Capfile.

$FINISH

This statement defines the end of a simulation.

Pin Name Mapping

The EDA Interface takes the first 16 characters of the HILO-3 pin name and maps them to system channels.

The first character must be alphabetic (A..Z or a..z). The remainder of the name can consist of any printable character except semicolon (;).

Recommended State Mapping Conventions

The recommended mappings that should be used for HILO_3 file state mapping are shown in the table below.

Table 12-3. Recommended HILO State Mapping

Input and Direction Control Design Pins		Output Design Pins	
Simulator State	**Vector Value**	**Simulator State**	**Vector Value**
0, L, N	0	0, L, N	0
1, H, P	1	1, H, P	1
Z, B, R, F, T, W, D, U, X, E	T	Z, B, R, F, T, W, D, U, X, E	X

Prequisites and Restrictions

Bidirectional Pins

Since the simulator logs the state of a bidirectional pin at a given time, there is no way of knowing whether it is in the input or output state. To extract timing and vector information, direction control signals must be associated with bidirectional pins.

Waveform Sampling Period

This cannot be directly extracted. You should enter a value for the test period in the Timing Extraction Options menu.

A

Pin Mapping File Format

The pin mapping files are in ASCII format and can be edited using a normal text editor. The information is represented by constructs that are described in the following sections.

Pin Mapping Statements

The pin mapping file consists of a statement list, which contains a number of control statements of the following types:

- Comments - these improve the readability of the pin mapping file and are ignored by the EDA Interface software
- DUT Pin Mappings - are used to map a number of design pins in the EDA input files to one DUT pin
- Design Pin Ignore List - consists of a list of design pins that will be ignored by the EDA Interface software

The first statement must the the pin mapping file header:

`hp82000,cae_pin_map,0.1` The following listing shows an example pin mapping file

```
hp82000,cae_pin_map,0.1
(comment "Pin Mapping file for device XYZ version 5.5"
)
(map a$1 to A_1
)
(ignore b
)
```

Figure A-1. Example Pin Mapping File

The example file contains one statement of all three types.

Syntax

The following pages show the individual commands used for pin mapping. Each section contains a description of the syntax and parameters, and an example of the command.

Comment

`comment`	is the statement to start a comment
`text`	any number of printable ASCII characters. The comment text must be enclosed in double quotes (”).

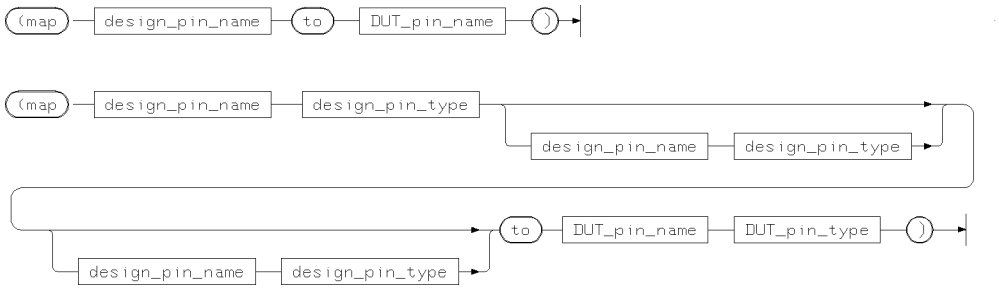
Description

The comment statement is used to improve the readability of the file.

Example

```
(comment "This is a comment line.")
```

Pin Mapping



Description

The first form of the statement is used to map design pin names to DUT pin names. The second form of the command is used to map a number of design pins to a DUT pin or to modify the DUT pin type extracted from the design pin.

`map`	the statement to start a pin mapping statement
`to`	a reserved word to indicate which design pins are being mapped to the DUT pin
`design_pin_name`	name of the design pin in the EDA files. This can be up to 255 printable characters.
`design_pin_type`	The usage of the design pin in the EDA file. This can be one of: `inp` for an input pin `out` for an output pin `bid` for a bidirectional pin `dir` for a direction control pin `ana` for an analog pin
`DUT_pin_name`	name to be used for the DUT pin in the tester setup files. This name can consist of up to 16 printable ASCII characters (hex 21 to 7E) except the following " ' ( ) ; and comma (,).
`DUT_pin_type`	The type of tester pin that is to be mapped to the design pin. This can be one of: `i` for a DUT input pin

o for a DUT output pin
io for a bidirectional pin
ioh for a bidirectional pin with a high termination
iol for a bidirectional pin with a low termination
ot for an output pin with an active termination
dc for a dc measurement pin

Restrictions

- If the optional wildcard character # is used, it must appear in each pin name of a mapping statement. The wildcard character expands to any character string between 0 and 16 characters in length.
- DUT pins can be mapped only in the following combinations:
  i with inp or inp and dir
  o with out only
  io with
  inp and out or
  inp and dir and out or
  bid with dir

Example

```
(map /NET003 to AQL
)
(map D0_+ inp
     D0_* dir
     D0_- out
 to  Data io
)
(map X#_+ inp
     X#_* dir
     X#_- out
 to  X# io
)
```

In this example, the design pin with the name /NET003 is mapped to the DUT pin AQL.

The design pins D0_+, D0_*, and D0_- are mapped to the bidirectional DUT pin Data. Drive data will be extracted from D0_+, expected data from D0_- and the direction control information will be extracted from D0_*.

Pin Mapping

In the third part of the example, the wildcard character will be expanded. In this case, if there are design pins with the names XYZ_+, XYZ_*, and XYZ_-, they will be mapped to the DUT pin XYZ.

Design Pin Ignore Statement

`ignore`	the keyword to begin an Ignore Design Pin
`design_pin`	A design pin that is to be ignored. One wildcard character (#) may be used in any name.

Description

The pin iwill be ignored by the EDA Interface software. Test vectors and pin configuration setups will not be generated for these pins.

Example

```
(ignore G1
)
(ignore F#
)
```

The design pin `G1` will be ignored as well as any pins whose names begin with `F`.

B

Signal Mapping File Format

The signal mapping files are in ASCII format and can be edited using the State Editor or a normal text editor. The information is represented by constructs that are described in the following sections.

Signal Mapping Statements

The pin mapping file consists of a statement list, which contains a number of control statements of the following types:

- Parameters - used to specify EDA system-specific parameters used in the signal mapping process
- Input Mapping - used to map the simulator states for DUT input design pins to tester DUT inputs
- Output Mapping - used to map the simulator states for DUT output design pins to tester DUT outputs
- Direction Control - used to map the simulator states for DUT direction control design pins to tester DUT direction control pins

An example state mapping file is shown below.

```
hp82000,cae_state_map,0.1
(params
inacurracy 0.005
)
(inp
map H to 1
map L to 0
map z to Z
)
(out
map h to 1
map l to 0
map x to X
map i to I
)
(crl
map i to 0
map x to 0
map z to T
map u to T
)
```

Figure B-1. Example Signal Mapping File

The example file contains one statement of all four types.

Syntax

The following pages show the commands used for pin mapping. Ecah command descriptions contains a description of the command and its syntax and an example.

Parameters

`params`	is the statement to start a parameter definition
`inaccuracy`	a reserved word to indicate an inaccuracy setting
`value`	a positive real number indicating the window size ito be used for waveform checking in nanoseconds

Description

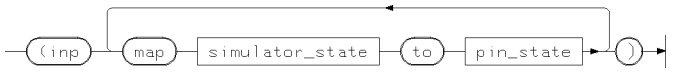
The parameter statement is used to specify the inaccuracy range used for timing setup extraction. It has the implied units picoseconds.

Example

```
(params inaccuracy 10.000)
```

sets the inaccuracy value to 10 picoseconds.

Input Mapping



`inp`	the statement to start an input state mapping statement
`map`	A reserved word to start a mapping statement
`simulator_state`	indicates the simulator input state to be mapped. It can consist of any printable ASCII character.
`to`	a reserved word to indicate which simulator state is being mapped to the DUT pin
`pin_state`	tester input state to which the the simulator state is to be mapped. This can be one of: `0` for a logical low `1` for a logical high `T` for tristate

Description

The statement maps simulator input pin states to tester input pin states.

Restrictions

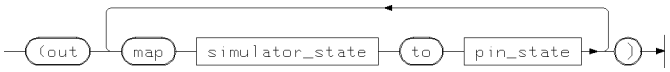
Ifthere are multiple mappings for one simulator state, the first occurence of the simulator state will be used.

Example

```
(inp
map H to 1
map L to 0
map z to T
)
```

In this example, the simulator input states H, L, and z are mapped to the tester DUT input states 1, 0, and T respectively.

Output Mapping



`out`	the statement to start an output state mapping statement
`map`	A reserved word to start a mapping statement
`simulator_state`	indicates the simulator input state to be mapped. It can consist of any printable ASCII character.
`to`	a reserved word to indicate which simulator state is being mapped to the DUT pin
`pin_state`	tester output state to which the the simulator state is to be mapped. This can be one of: `0` for a logical low `1` for a logical high `I` for intermediate `X` for don't care - the comparator result will not affect test results

Description

The statement maps simulator output states to tester output states.

Restrictions

If there are multiple mappings for one simulator state, the first occurence of the simulator state will be used.

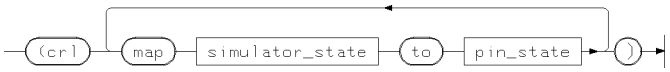
Example

```
(out
map h to 1
map l to 0
map x to X
map i to I
)
```

Output Mapping

In this example, the simulator input states h, l, x, and i are mapped to the tester DUT output states 1, 0, X, and I respectively.

Direction Control Mapping



`crl`	the statement to start a direction control pin state mapping statement
`map`	A reserved word to start a mapping statement
`simulator_state`	indicates the simulator input state to be mapped. It can consist of any printable ASCII character.
`to`	a reserved word to indicate which simulator state is being mapped to the DUT pin
`pin_state`	tester direction control state to which the the simulator state is to be mapped. This can be one of: `0` - when the direction control signal is in this state, the driver is enabled and the comparator is masked. The bidirectional pin acts as a DUT input. `T` - when the direction control signal is in this state, the driver will be tristated and the comparator will be enabled. The bidirectional pin acts as a DUT output. The state of the direction control signal overrides the state of the DUT input and output signals.

Description

The statement maps simulator direction control pin states to tester direction control pin states. When the tester 0 state is active, a bidirectional pin as as a DUT input, and when the state is T, the pin is treated as a DUT output.

Restrictions

If there are multiple mappings for one simulator state, the first occurence of the simulator state will be used.

Example

```
(crl
map i to 0
map x to 0
map z to T
map u to T
)
```

In this example, the simulator direction control signal states i and x are mapped to the tester direction control state 0 which enables the drivers. Simulator states z and u are mapped to the tester state which tristates the drivers and enables the comparators. the tester DUT input states 1, 0, and T respectively.

C

Extraction Examples

This appendix shows examples of the different EDA file formats which can be understood by the Supplied EDA Interface software.

It contains examples of pin configuration, timing, and vector extraction using the EDA Interface software for the Mentor QuickSim format.

Mentor QuickSim

Example Stimulus File

The following example shows part of a Mentor MISL file.

```
CIRCUIT EXAMPLE:

/* Stimulus file for "EXAMPLE"
 Clock : 50 MHz, synchron
*/

/********** Begin signal definitions *****************/

TIMEDEF PERIOD = 20ns;

OUTPUT Q0,Q1,Q2,Q3,NCOUT;

INPUT CLOCK,S1,S2,NCIN,CTRL;

BUS D1,D1,D2,D3;

/********** Begin signal waveform definition *********/
```

```
/*
 ...
*/

END.
```

The Mentor QuickSim input file contains the statements `INPUT`, `OUTPUT`, and `BUS` which are used to define input (`inp`), output (`out`), and bidirectional (`bid`) signals. The simulation period is 20 nanoseconds.

Example Logfile

The following example shows part of a Mentor Logfile.

```
F /user/demo/EXAMPLE | S 10 1           | S 10 1
T 0.0                | T 11.2          | T 71.2
D /CLOCK 10 0        | S 17 0          | S 17 1
D /NCIN 15 0         | S 20 0          | T 79.0
D /NCOUT 16 0        | S 19 0          | S 10 0
D /D0 11 0           | S 18 0          | T 89.0
D /D1 12 0           | T 13.5          | S 10 1
D /D2 13 0           | S 16 1          | T 91.2
D /D3 14 0           | T 19.0          | S 17 0
D /Q0 17 0           | S 10 0          | S 19 1
D /Q1 18 0           | S 22 1          | S 18 0
D /Q2 19 0           | T 29.0          | T 99.0
D /Q3 20 0           | S 10 1          | S 10 0
D /S1 21 0           | T 31.2          | T 109.0
D /S2 22 0           | S 17 1          | S 10 1
D /CTRL 23 0         | T 39.0          |
S 14 0               | S 10 0          |
S 13 0               | T 49.0          |
S 12 0               | S 10 1          |
S 11 0               | T 51.2          |
S 15 0               | S 17 0          |
S 22 0               | S 18 1          |
S 21 0               | T 59.0          |
S 10 0               | S 10 0          |
T 9.0                | T 69.0          |
```

The Mentor QuickSim output file contains the simulation results.

`D` statements define the simulation signals, for example `NCIN`, the surrogate to be used for the signal in the rest of the file (`15`) and the initial state (`0`).

`T` statements define timestamps at which state transitions occur.

`S` statements represent state transitions for signals addressed by surrogates.

Pin Configuration Example

Extracting I Tester Pins

Extracting the Pin Type from the EDA Files

Inputs MISL file
Logfile

Outputs Pin Configuration File

The `NCIN` signal with the type `inp` (input) is translated to the tester pin `NCIN` with the type `i` (input). The signal type is extracted from the `INPUT` statement of the MISL file.

Extracting the Pin Type using Pin Mapping

Inputs Logfile
Pin Mapping File

Outputs Pin Configuration File

The signal `NCIN` is extracted from the Logfile. The pin type is unknown at this point. With the pin mapping:

```
NCIN, inp | NCIN, i
```

the signal is translated to the tester pin `NCIN` with type `i`.

Extracting the Pin Type using the Pin Configuration Editor

Inputs Logfile

Outputs Pin Configuration File

Pin Configuration extraction translates the signal NCIN to the tester pin NCIN with the default pin type ot (output terminated). After the extraction is complete, you can modify the pin type to i using the Pin Configuration Editor and continues with timing and vector extraction.

Extracting O Tester Pins

Inputs MISL file (optional)
Logfile
Pin Mapping file (optional)

Outputs Pin Configuration File

Signals with no pin type specified are translated to pin type ot (output terminated) by default.

Extracting IO Tester Pins

Inputs MISL file (optional)
Logfile
Pin Mapping file

Outputs Pin Configuration File

Bidirectional signals can only be translated to io pins using a pin mapping file.

With the pin mapping:

```
NCIN, inp | NCBID, io
NCOUT, out |
```

the signals NCIN, inp and NCOUT, out are translated to the tester pin NCBID, io.

With the pin mapping:

```
DO, bid | DO, io
CTRL, inp |
```

the signals DO, bid and CTRL, inp are translated to the tester pin DO, io.

HILO-3

The following HILO-3 files contain the same information as in the previous example.

Example WDL File

```
WAVEFORM sim1
STIMULUS
 CLOCK = BIN 0;
 S1 = BIN 0;
 S2 = BIN 0;
 NCIN = BIN 0;
 CTRL = BIN 0;
 D0x = BIN 0;
 D1x = BIN 0;
 D2x = BIN 0;
 D3x = BIN 0;

RESPONSE
 Q0 = X;
 Q1 = X;
 Q2 = X;
 Q3 = X;
 NCOUT = BIN 0;
 D0 = X;
 D1 = X;
 D2 = X;
 D3 = X;

BIDIRECTIONAL
 D0x = D0;
 D1x = D1;
 D2x = D2;
 D3x = D3;

FINISH.
```

Example Capture File

```
! Simulation of "EXAMPLE" IC
$DATE 20-FEB-1989 14:25;
$CIRCUIT EXAMPLE;
$WAVEFORM SIM1;
$DELAYSCALE 9;
$MAXINDEX 14;
=0! input CTRL I1 016;
=0" input CLOCK I1 08;
=0# input S1 I1 05;
=0$ input S2 I1 03;
=0% input NCIN I1 01;
=0& wire D0 I1 01;
=0‘ wire D1 I1 01;
=0( wire D2 I1 01;
=0) wire D3 I1 01;
=X* unid Q0 I1 01;
=X+ unid Q1 I1 01;
=X, unid Q2 I1 01;
=X- unid Q3 I1 01;
=0. unid NCOUT I1 02;
#0 $STABLE;
#2 1!
#4 0* 0+ 0, 0-
#9 1"
#10 $STABLE;
#19 0" $STABLE;
#22 1$ 01
#26 1.
#29 1"
#31 1*
#36 $STABLE;
#39 0"
#40 $STABLE;
#49 1"
#51 1+ 0*
#56 $STABLE;
#59 0"
#60 $STABLE;
#69 1"
#71 1*
#76 $STABLE;
#79 0"
#80 $STABLE;
#89 1"
#91 1, 0* 0+
#96 $STABLE;
#99 0"
#100 $STABLE;
#109 1"
```

Daisy DLS

Example VLAIF TIME_VALUE Format File

```
$DATA_HEADER$                  | $FIELD$ 11
$TYPE$                         | @EXAMPLE:Q3 11
I/O                            | $FIELD$ 12
$FORMAT$                       | @EXAMPLE:S1 12
TIME_VALUE                     | $FIELD$ 30
$TOTAL_COLUMNS$                | @EXAMPLE:S2 13
4 14                           | $FIELD$ 14
$BASE$                         | @EXAMPLE:CTRL 14
D B                            | $END$
$FIELD$ 1                      | 0000    00U000LUUUU00@
@EXAMPLE:CLOCK 1               | 0090    10U000LUUUU00@
$FIELD$ 2                      | 0112    10U000LLLLL00@
@EXAMPLE:NCIN 2                | 0135    101000LLLLL00@
$FIELD$ 3                      | 0190    001000LLLLL00@
@EXAMPLE:NCOUT 3               | 0220    001000LLLLL01@
$FIELD$ 4                      | 0290    101000LLLLL01@
@EXAMPLE:D0 4                  | 0312    101000LLLLL01@
$FIELD$ 5                      | 0390    001000LLLLL01@
@EXAMPLE:D1 5                  | 0490    101000LLLLL01@
$FIELD$ 6                      | 0512    101000LO1LL01@
@EXAMPLE:D2 6                  | 0590    001000LO1LL01@
$FIELD$ 7                      | 0690    101000LO1LL01@
@EXAMPLE:D3 7                  | 0712    101000LO1LL01@
$FIELD$ 8                      | 0790    001000LO1LL01@
@EXAMPLE:Q0 8                  | 0890    101000LO1LL01@
$FIELD$ 9                      | 0912    101000LLL1L01@
@EXAMPLE:Q1 9                  | 0990    001000LLL1L01@
$FIELD$ 10                     | 1090    001000LLL1L01@
@EXAMPLE:Q2 10                 |
```

Index