VALID

Dear 7.25 Customer: 1 April 1985

The attached Changes Document describes enhancements to the 7.5 release of the optional Logic Simulator that are not described in the Reference Manual Update (900-00049) included in the documentation package for the 7.25 software release. Until another manual update is available, please refer to the 7.5 Simulator section of the Changes Document for descriptions of the new operations supported by the 7.5 Logic Simulator.

Sorry for the inconvenience;

Gary Lindstedt
Technical Publications Mgr.

2820 Orchard Parkway
San Jose, CA 95134
408/945-9400
Telex 371 9004

SOFTWARE CHANGES INFORMATION

SOFTWARE RELEASES 7.0 and 7.25

3 April 1985

TABLE OF CONTENTS

1.0 INTRODUCTION

This document describes the changes and enhancements to the SCALDsystem when the system software is updated from release 6.1 to release 7.0 and when updated from 7.0 to the current 7.25 release (systems that are updated from 6.1 are first updated to 7.0 and then updated from 7.0 to 7.25).

NOTE

> The Logic Simulator has been additionally updated from 7.25 to 7.5 and is optionally available; see "Release 7.5 of the Logic Simulator" in section 7.

The organization of the changes document follows the order of the SCALDsystem reference manual. Within each section, changes between 6.1 and 7.0 are described initially followed by the changes between 7.0 and 7.25 (if present). If your system is being upgraded from 6.1, be sure to read both sets of changes.

NOTE

> While the emphasis of the 7.25 release is bug fixes, some enhancements are incorporated.

In addition to this document, the reader should also consult the "7.25 Software Anomalies" document which describes the outstanding bugs in the 7.25 release and the recommended "workarounds" for those bugs.

2.0 DISPLAY MANAGER

The display manager includes three programs; the display manager itself, the window manager, and the help facility.

RELEASE 7.0 OF THE DISPLAY MANAGER

The 7.0 display manager's setup mode has been changed significantly. The setup mode is now entered by typing "setup" is response to the shell prompt and is no longer selected from the screen menu (the program "vgsetup" is no longer supported).

The number of screen lines is limited to 62 (not 64); the maximum number of characters per line is limited to 100.

The documentation on the display manager has been updated and moved to Chapter 2 of the reference manual.

RELEASE 7.25 OF THE DISPLAY MANAGER

The number of screen lines increased to 64 with 102 characters per line (with full-screen windows); tiled windows still are limited to 62 lines by 100 characters.

The system information log for the window manager (in log file "/tmp/wm.log") has been reduced in size. Only information regarding errors is now written and results in much smaller "wm.log" files.

RELEASE 7.0 OF THE WINDOW MANAGER

First introduction of multiple window facility (see Chapter 2 of reference manual).

RELEASE 7.25 OF THE WINDOW MANAGER

Added full-screen windows and user-defined window names (see Chapter 2 of reference manual).

If a screen cannot be saved or restored, the Display Manager displays either the message "Cannot save screen on disk" or "Cannot restore screen from disk later," as appropriate, and continues in the old window.

Screens are now saved in /u0/wmtmp and not /u0/tmp (/u0/wmtmp is removed on startup).

RELEASE 7.0 OF THE HELP FACILITY

First introduction of on-line help facility for SCALD topics (see chapter 2 of reference manual).

RELEASE 7.25 OF THE HELP FACILITY

Added documentation to allow user-defined help topics.

3.0 GRAPHICS EDITOR

RELEASE 7.0 OF THE GRAPHICS EDITOR

This 7.0 release of the Graphics Editor (GED) included two new features (text and body rotations and a new connectivity file format) as well as bug fixes. All drawings from the previous (6.1) release are compatible with the 7.0 release (i.e., the SCALD Compiler continues to read the old connectivity file format). The two new features and command enhancements are described below.

CAUTION

WARNING: NEW VIRTUAL MEMORY UNIX MUST BE INSTALLED BEFORE INSTALLING 7.0 GED!

While drawings from all previous versions of GED can be read by the 7.0 GED, pre-7.0 versions of GED **cannot** be run under the new UNIX. Conversely, 7.0 GED will **not** run under pre-7.0 UNIX systems.

For release 7.0, the following must be installed:

- /u0/lib/master.lib (which is used by the SCALD Compiler); GED no longer uses /u0/lib/ged/master.dir and /u0/scald/master.lib.
- /u0/editor/doc (contains drawing for the HELP command).

Additionally, 7.0 UNIX, new Video Graphics Board code, and Display Manager and SCALD Compiler programs must be installed.

NEW FEATURES

The following features have been added to the Graphics Editor:

- **Connectivity File Format** -- the new connectivity file format described in Chapter 3 of the reference manual is supported. The SCALD Compiler continues to read the old connectivity format. However, any new Compiler features are not supported in the old format.
- **Rotations of Bodies and Text and Text Justification** -- rotations are in increments of 90 degrees (0, 90, 180 and 270). When a body is rotated, all notes and properties also are rotated. Properties can be rotated or justified independently. Since 180 degree rotations of devices could, in some cases, reverse the order of pins (e.g., mergers) and cause subtle bugs in a user's design, a 180-degree rotation of a device is, in reality, a mirror of a 0 degree rotation (about the Y axis). Similarly, a 270 degree rotation of a device is a mirror of a 90 degree rotation (about the X axis). In a 90 degree rotation, body notes are rotated 90 degrees and left in their original justification. For mirrors, only the justification of the text is changed (left --> right, right --> left), and no further rotation is done. Text

rotations (properties and drawing notes) are actually rotated, not mirrored. To get the other four rotations, see the MIRROR command, below.

To rotate a body, use the command

```
ROTATE    |    point ....
```

Each press of the puck button rotates the device 90 degrees. To get the other four rotations (0 and 90 degrees are mirrors and 180 and 270 degrees are rotations), another version of the device is needed. This version should be a mirror about the Y-axis of the first version. To make it easier to create the mirrored version, the MIRROR command has been added to the Graphics Editor's command set. This command mirrors, about the Y axis, all lines, arcs, and text in a body drawing.

```
MIRROR    |
```

For instance, two versions of a body might be:

The MIRROR command should be used with caution, especially with bodies that have unmarked pins such as the Valid-supplied merge bodies. Reversal of the bit order can cause subtle, hard-to-find bugs within a design.

- **Justification** -- Text justification can be either left (current default) or right. To change the justification of text:

```
DISPLAY   |   { LEFT  }   point....
          |   { RIGHT }
```

The origin of the text string remains the same. For instance, DISPLAY RIGHT to a left-justified string does the following:

```
                    THIS IS A NOTE
THIS IS A NOTE
```

When a right justified string is moved, the cursor is attached to the end ("E" in the above example). As with other DISPLAY options, justification works on groups as well as single objects.

The default justification of text is set using:

```
SET     |  { LEFT_JUSTIFIED  }
        |  { RIGHT_JUSTIFIED }
```

NEW OR ENHANCED COMMANDS

- **Backannotate** -- Enhancements

  With text justification and rotation, the placement of backannotated properties has been improved for better readability. For instance, when two vertical pins are on top of a part, the annotated pin properties overwrite each other. Additionally, the size of the pin properties that are annotated is smaller: 0.6 rather than 0.8 to for improved visual separation.

  However, if an already backannotated drawing is re-backannotated, GED uses the previous positions. In order to take advantage of the new placement algorithms, do the following in GED:

```
EDIT DRAWING_NAME   {edit already annotated drawing}
FIND $*             {finds all annotation properties
                     and puts them in Group A}
DELETE A            {deletes annotation properties}
WRITE
BACKANNOTATE        {uses new placement algorithm
                     on the drawing}
```

- **Display** -- Enhancements

```
DISPLAY  |  { LEFT  }  point...
         |  { RIGHT }
```

  The LEFT and RIGHT options change the justification of text strings as previously described.

- **Group** -- Enhancements

  When a group is defined, the number of notes, wires, arcs, and bodies is listed. The count also is given when the "SHOW GROUP group_name" command is invoked.

- **Hardcopy** -- Enhancement

  The HPFILTER program that outputs plots to the HP Pen Plotter now outputs patterned lines correctly.

- **Mirror** -- New command

  See the section on rotations in "New Features" (above).

- **Pinswap** -- New command

```
PINSWAP    |   ( { pin_number } point )...
           |   ( { point      }       )...
```

The PINSWAP command only can be used after section assignment has occurred for the part. Also, pin swapping only can occur between pins that have been defined in the library as swappable (i.e., the two input pins of a NAND gate can be swapped; the input and output pins cannot be swapped).

To swap pins, either point to the two pins to be swapped or type in the new pin number for the selected pin (the selected pin is swapped with the specified pin number).

Note that properties attached by the PINSWAP command cannot be changed, only deleted and moved, and that properties are not written into the connectivity file. Once pins on a part have been swapped, the part cannot be resectioned with the SECTION command.

Only devices in libraries with chips files can be sectioned.

- **Replace** -- New Command

```
REPLACE | [<directory>]body_name[.[.[version]]]  point
```

This command replaces one device for another. The device to be replaced is selected with a puck point, and the new device is given by name. Any properties attached to the selected device are reattached to the new device. Pin properties are reattached if a pin name on the new device is the same as a pin name on the initial device. If no pin name match exists, the pin property becomes a body property. All properties with the exception of properties generated by the BACKANNOTATE, SECTION, and PINSWAP commands are maintained. Any wire connections to the original device are maintained only if the pins are in the same place.

- **Rotate** -- New Command

See description in "New Features" (above).

- **Section** -- Enhancements

```
SECtion          |     [pin_number] point ....
```

Previously, you had to step through every section explicitly before reaching the one you wanted. Now, to assign a specific section directly, first type in a pin number that uniquely defines the section before pointing at the part.

- **Set** -- Enhancements

```
SET  |  { LEFT_JUSTIFIED                }
     |  { RIGHT_JUSTIFIED               }
     |  { USER_SIM simulator_file_name }
```

LEFT_JUSTIFIED and RIGHT_JUSTIFIED change the default justification of text strings (both properties and notes).

The USER_SIM option allows you to give the UNIX path name of the Simulator to run with the SIMULATE command in GED. The default is /u0/scald/simulator/sim.

- **Show** -- Enhancements

The SHOW GROUP group_name command now prints the number of devices, wires, notes, and arcs in the group.

- **Write** -- Enhancements

If a drawing name other than the one listed in the status line is given and that drawing name is already in a SCALD directory, a warning message is given. You must select WRITE again in order to actually write the drawing. Selecting any other command aborts the write.

CHANGES TO ASCII FILE FORMAT

The ASCII file format has been changed to handle rotations; see Chapter 3 of the reference manual.

UNNAMED SIGNALS

Unnamed signals in releases prior to and including 6.1 were created dynamically when a drawing was written and caused problems for the Packager state files when a design was only written and not changed. In release 7.0, unnamed signals are now attached to the drawing (more specifically to pins; attaching the name to the pin of a net is necessary because wires are more often deleted when "prettying up" a drawing). By attaching unnamed signals to pins, the design database is larger (unnamed signals are now included in binary and ASCII files rather than just in the connectivity files), but many problems are solved.

The goal of unnamed signals is:

- o consistent between writes
- o interpretable
- o local
- o predictable

When connectivity is determined (WRITE and SHOW NET), GED produces a signal name in the form:

UN$PAGE$BODY_NAME$PATH$PIN_NAME

The drawing is ALWAYS auto path'ed before the signal names are created. GED selects the BODY_NAME alphabetically from all of the bodies on the net; in the event of identical BODY_NAMES, the lowest-numbered PATH property is used. The signal name is then attached to the lowest-numbered PIN_NAME.

There are several consequences of the new algorithm: if you just read and write a drawing, the unnamed signals are guaranteed not to change. However, if ANY changes are made to the drawing, the state is different and therefore the unnamed signal names might be different. In particular:

- o Any changes to unnamed signal names are local. Logic changes only affect the net that was changed and no other net.

- o If two unnamed signals are connected, only one of the unnamed signals is kept. Therefore, connecting two unnamed signals and then unconnecting them may change the name of one of the nets.

- o If a new device is added to an unnamed net, the name of the net doesn't change.

- o If a body is deleted from the net, the net name may change (if the pin that determined the net name is attached to that body). Re-adding the wire or body may not give the same unnamed signal name.

RELEASE 7.25 OF THE GRAPHICS EDITOR

The 7.25 release of GED consists of two new features, body rotation and color support. Also, there have been major changes to the vector file format and to the operation (but not the user interface) of the HARDCOPY command. In addition, several commands have been enhanced (note especially SET and the recovery feature).

For release 7.25, the following must be installed:

- o 7.25 UNIX
- o Video Graphics Board code
- o Display Manager
- o lpr
- o section
- o SCALD Compiler.

NEW FEATURES

The following new features have been added to the 7.25 release:

- o **Hardcopy**

  Electrostatic Plotters

  SET HPR

  SET VGB

  * There are two methods for doing HARDCOPY; using HPR and using VGB. The method supported in release 6 was VGB (using the integral graphics board for rasterizing the plot). This method had many drawbacks; it required a work station (2310) and, while plotting, you could not use the work station as it was being used for rasterization, only one work station could plot at a time as the plots were not queued, and if the graphics board or PIB crashed while rasterizing, the work station was unusable. The HPR method corrects all of these disadvantages.

  * HPR performs the rasterization using the 68010 in the S32 and also queues the plots to allow more than one work station to plot at the same time.

  * The default HARDCOPY method is HPR. New features (i.e., new plotters and HARDCOPY on the color work station) are only supported with HPR and not VGB.

  * The VGB method is included only for compatibility. While the PIB code is much more robust, be warned that this method still has the same drawbacks and problems as outlined above.

  * Compared to the VGB method, HPR will appear both faster and slower. Faster in that you will have the use of your work station much quicker. Slower in that HPR takes longer to rasterize the plot and send it to the plotter.

* The Benson metric 24" (model 9424) plotter is now supported (with HPR method ONLY). Use the command:

  SET B9424

* On a color design station (2310C), only the HPR method is supported (the VGB method does not work).

HP Plotters

* To select the hp plotter option, use the command "SET MONO_HPPLOT" to cause ONLY pen 1 to be used in the plotter (in the next maintenance release, there will be a "SET COLOR_HPPLOT" command to enable the use of more than one pen). **Note that "SET HP_PLOT" NO LONGER WORKS.**

* HP plotter now does filled dots, rotated text, rotated bodies, justified text, and patterned lines.

* More than one drawing can be plotted at a time (formerly, "HA B FOO*" did not work). After each drawing, GED asks you to type <RETURN> when ready for the next plot to allow paper to be loaded.

o **Vector File Format**

The vector file format has been modified (see Chapter 3 of the reference manual).

o **Spin Command**

In addition to Rotation (a 7.0 enhancement), bodies can be rotated in increments of 90 degrees without the mirroring effect at 180 and 270 degrees (i.e., true rotation) with the SPIN command.

o **Color Support**

Objects in GED can have an associated color attribute and, when displayed on a 2310C color monitor, are presented in the corresponding color. Note that colors can be assigned from either a color or a monochrome monitor. The following commands/functions are supported:

PAInt	color point...
	color "group_name"...
SHOw	color point...

SET	COLOR_Wire	color
	COLOR_Body	color
	COLOR_Arc	color
	COLOR_Dot	color
	COLOR_Note	color
	COLOR_Prop	color

The PAINT command allows you to color an object or group of objects on either a color or monochrome work station. When the PAINT command is selected, the command menu is replaced by a color palette menu (the color names are displayed on a monochrome monitor). You can either select the color from the palette menu or you can type in the label.

The SHOW command displays the color label (i.e., name of the color) attached to the selected object. This command is used primarily on monochrome work stations to "see" the color of an object.

The SET COLOR command sets the color for the six types of GED objects (default color for all objects is "mono") for all subsequent graphics drawing.

The available colors (color labels) are:

RED	GREEN	BLUE
YELLOW	ORANGE	SALMON
VIOLET	SKYBLUE	PINK
AQUA	PEACH	BROWN
WHITE	PURPLE	GRAY

- **Recovery From Crashes**

If GED or UNIX crashes, you can recover the drawings you were editing with GED. This feature has been slightly altered and simplified.

When GED is restarted following a crash, you will be asked if the drawings are to be restored from the undo log. If you elect to recover your drawings, they will be placed in the SCALD directory "restore.wrk." The recovered drawings are called RESTORED1, RESTORED2, ... If restore.wrk exists, it will be over written. A warning message is printed stating this, and you can elect not to recover. You must type USE RESTORE.WRK to access the recovered drawings. Whether you elect to recover or not, GED will startup.

NEW AND ENHANCED COMMANDS

- LED -- new

When running GED on a SCALDstar system, the LED command (in conjunction with the GED command in the Layout Editor) allows a network to be identified on one system and highlighted on the other. To use the LED

command, type LED in the GED window, go to the LED window (using window manager), and type GED to cause the two programs to be connected; using the SHOW NET command in either the GED or LED window highlights the net in both windows.

- SET -- Enhancements

```
SET  |  { HPr / VGb / MOno_hpplot      }
     |  { B9424                        }
     |  { COLOR_Wire color            }
     |  { COLOR_Prop color            }
     |  { COLOR_Dot  color            }
     |  { COLOR_Arc  color            }
     |  { COLOR_Note color            }
     |  { COLOR_Body color            }
```

The HPR, VGB, MONO_HPPLOT and B9424 options are discussed in the "New Features" section (above).

The COLOR options also are discussed above in the "New Features" section.

- SHOW -- Enhancements

```
SHOw  |  { GRoup  group_name          }
      |  { COLor  point                }
```

The SHOW GROUP group_name command now prints the number of devices, wires, notes, and arcs in the group.

The SHOW COLOR command prints the color of the indicated object.

MODIFICATIONS TO FEATURES

The following operations are modified in the 7.25 release:

- IGNORE

The IGNORE command now requires conformation that the SCALD directory is really to be ignored. This feature was added for two reasons: to make GED's IGNORE command more like LED's IGNORE command and to prevent time consuming errors when a directory is inadvertently ignored. Note that no message is displayed in the text area; if the current directory is deleted, the name of the previously-selected directory is displayed on the status line.

- HARDCOPY (on electrostatic plotters)

  Instead of using the VG board for rasterizing the drawings to the Versatec, this release of GED uses the utility hpr to spool drawings and uses memory (rather than the VG board) to do its rasterization. The user interface is identical to previous releases with the exception that the screen is now usable while the hardcopy is being printed.

- HARDCOPY (on hp plotters)

  To allow users to plot multiple drawings on the HP plotter, GED now prompts when it is ready to plot; you must type <RETURN> before the drawing is plotted. This interval allows time to load and/or change paper in the plotter before continuing. Filled dots and rotated and justified text are now supported.

- CHANGE

  In the CHANGE command, you must type ^I, not ^H, for the command summary.

4.0 COMPILER

RELEASE 7.0 OF THE COMPILER

The 7.0 release of the Compiler contained several major new features, some changes in the SCALD language, revisions in the execution environment, and several bug fixes. Each of these is described below.

MAJOR NEW FEATURES

The 7.0 release contains major changes in the way the Compiler is run. They are described below.

Command Line Arguments

The Compiler supports command line arguments to permit the user to specify both the name of the drawing to be compiled and the compile type without having to edit the Compiler's directive file. The form for the compile command is

```
compile <drawing name> <compile type>
```

where <drawing name> is the name of the drawing to be compiled. If the drawing name contains special characters, the name must be placed in quotes. The <compile type> specifies the argument for the COMPILE directive. Here are a few examples:

```
compile foo
      - compile the drawing FOO for LOGIC (the default).

compile foo time
      - compile the drawing FOO for TIME.

compile "drawing with strange name!" SIM
      - compile this strange drawing for SIM.
```

If a ROOT_DRAWING or COMPILE directive appears in the Compiler directives file when command line arguments are being used, the Compiler uses the command line arguments and ignores those in the directives file.

The IBM version of the Compiler has the restriction that the command line arguments must be legal CMS parameters. That is, they must be no longer than eight characters and cannot contain "special" characters such as spaces, quotes, etc. This restriction is only pertinent to the drawing name which is often likely to be "special." For those drawings, the command line arguments cannot be used; the ROOT_DRAWING directive in the directives must be used.

The VAX version of the Compiler processes command line arguments by editing the compiler.cmd file to delete ROOT_DRAWING and COMPILE directives and by placing new ones into the file based on the values of the command line arguments. It should be noted that in order for this to work correctly, these directives (ROOT_DRAWING and COMPILE), if they appear, must be the only directive on the line. For example, the following line is ILLEGAL:

```
root_drawing 'FOO';  compile LOGIC;  print_width 80;
```

Separate Compilation

The 7.0 Compiler supports separate compilation to allow a large design to be compiled in several smaller pieces that can then be linked together to form the complete design. A new program, the Linker, is supported to perform the linking function. Separate compilation is an optional mode of operation for the Compiler. The Compiler continues to support compilation of the design in one piece. Separate compilation has the following features:

- The notion of a design is introduced. The design name refers to the entire design. Whenever a drawing is compiled, the design of which it is a part is also specified.

- Compile nothing smaller than a drawing. If a design consists of one multi-page drawing, separate compilation provides no advantage over full compilation in one piece.

- Drawings that are parameterized cannot be separately compiled. An example parameter is SIZE. Any drawing that uses SIZE or any user-defined parameters cannot be compiled separately.

- When a drawing is compiled, all parameterized drawings within it are compiled out. For example, if a drawing uses an LS00 and the drawing is being compiled for TIME, the timing model for the LS00 is compiled as well since it is parameterized (uses SIZE in this case). All plumbing drawings (MERGE, NOT, etc.) are compiled out as well.

- When all of the drawings in the design have been separately compiled, they can be linked together with the Linker to form expansion and synonym files that are the same as the files that would be produced by the Compiler if the design was compiled as a single piece.

The Linker keeps track of which files have been compiled and compiles those that have not been compiled. If the drawings have been changed since the last compilation, the Linker causes the drawings to be recompiled before linking. See the section on separate compilation in Chapter 5 for details. Also see the description for the SEPARATE_COMPILE directive.

Master Library Changes

A MASTER_LIBRARY directive has been added to permit the user to add additional master library files without having to modify the system MASTER.LIB file. Any number of master library files can be specified. See the MASTER_LIBRARY directive in Chapter 5 for additional details.

The format of the MASTER.LIB file has been changed. The LIBRARY directive accepts a library name in quotes or as an identifier (if it doesn't start with a digit).

Multiple Property and Text Macro Files

Multiple files can be specified with the PROPERTY_FILE and TEXT_MACRO_FILE directives. Multiple directives can be specified in the same directives file. If any text macros are duplicated, an error is generated.

Memory Reports

The heap space estimation was altered to include overhead for the Pascal compiler's heap management. On the VAX, for example, this amounts to eight bytes per object on the heap. The heap report in the log file shows both the uncorrected (first total) and corrected (second total) values. The heap reported in the list file is the corrected value.

Memory Usage

Some optimization of memory has been done to reduce memory space needed by the Compiler (memory optimization also reduces the execution time). Reductions in memory space of 25% have been noted.

Connectivity File Syntax

The syntax of connectivity files produced by the Graphics Editor and read by the Compiler has been changed to reduce their size, to make them easier to process, and to encode new information for 7.0. See the Graphics Editor chapter (Chapter 3) for details of the new syntax. The Compiler will continue to read drawings in the old format and will accept drawings whose pages have different formats (some old, some new). If a drawing is in the new format and has multiple versions, all of the versions must be in the new format (at least, page one of each version of the drawing must be in the new format).

Change in Selection Expression Processing

The 6.1 Compiler produced an oversight if the EXPR property did not appear on the DRAWING body of each version of a multi-version drawing in preparation for the 7.0 release. In 7.0, the Compiler determines the selection expression for each drawing version by looking at the first page of the drawing (the EXPR property appears in the third line of the connectivity file). The 7.0 Compiler does NOT use MENU bodies. An error is generated if a MENU body is found. If the drawings are in the old format, that is, not written with the 7.0 Graphics Editor, the Compiler continues to process MENU bodies.

When converting multi-versioned drawings (i.e., drawings with a MENU body on page 1 of version 1) created under release 6.1:

1. Delete the first page of version 1 (the drawing containing the menu body).

2. Attach the appropriate EXPR property to the DRAWING body on the first page of each subsequent version and write the drawing to disk. Note that if an EXPR property already is attached, the drawing still must be read and written.

The MENU body was error-prone and confusing. The new mechanism is easier to use, easier to understand, and more flexible.

Change in NC Signals

All NC signals appear in the CMPEXP file as NC. The Compiler does not place a unique numeric identifier on the signal (such as NC89) nor does it output a path name (as in (FOO .BAR2P)NC. This change should not affect any users since all of the SCALD programs process NC signals in the same manner as before.

SIZE, X_FIRST, and X_STEP Processing

The 7.0 Compiler is much more forgiving when it comes to the use of the SIZE parameter. The following changes have been made:

- If the SIZE parameter is not used on any of the pins on a body, the Compiler does not generate a "default SIZE used" warning.

- If SIZE is attached to a body that does not have any pins that use the SIZE parameter, a warning is generated.

- A drawing is never SIZE replicated unless explicitly indicated. The Compiler assumes that X_STEP=SIZE without informing the user.

- If X_STEP is specified, the Compiler uses the default value for X_STEP unless otherwise indicated. No warning is generated.

The result of the changes described above is that the DEFINE body need not be placed on a drawing unless the user wishes to define some text macros; the Compiler will not generate warning messages.

DOCUMENTATION CHANGES

Chapter 5 of the reference manual has been updated to include documentation on separate compilation. The error documentation also has been updated, and the directives documentation has been changed to reflect the additional directives.

RELEASE 7.25 OF THE COMPILER

The 7.25 release of the compiler consists primarily of bug fixes. There are several changes in compiler operation that should be noted.

1. LANGUAGE CHANGE -- .PART and .PRIM extensions are now equivalent and either may be used to specify a leaf component for any type of expansion. The compiler ignores either extension except where it occurs within a drawing specified by a SCALD directory of the proper type (i.e., a leaf component for a LOGIC expansion is a .PRIM or .PART extension occurring in a drawing specified in a LOGIC_DIR SCALD directory). When compiling for TIME, only .PRIMs and .PARTs occurring in drawings specified by TIME_DIR are considered. A given drawing can not have both a .PART and a .PRIM extension for a given compile type.

2. PROPERTY CHANGE -- the MAX_DELAY property has been assigned the attributes of inherit(), permit(body), and parameter(integer); in the previous (7.0) release, the MAX_DELAY property was not assigned attributes (and therefore had the usual default attributes).

3. AUTOMATIC RECOMPILATION -- the automatic recompilation feature of separate compilation now includes "fatal errors occurred last time" (which is, for now, internally, "a file containing expansion data is missing from the database") as sufficient to cause recompilation.

RELEASE NOTE

The package contains the program "lnkcomp" which must be compiled with the VAX-11 PASCAL version 2. Its .OBJ file must be linked using the version 2 libraries when producing the executable.

5.0 TIMING VERIFIER

RELEASE 7.0 OF THE TIMING VERIFIER

The 7.0 release of the Timing Verifier includes several enhancements as well as bug fixes.

ENHANCEMENTS

The following enhancements have been added to the 7.0 release:

- **Reconvergent Fanout** (correlated skews).

  The Timing Verifier has been enhanced to understand reconvergent fanout in order to eliminate false error messages that are currently generated for circuits that count on correlated signal skews to work.

  Reconvergent fanout is where a signal fans out from a common point in a circuit through different paths, which then reconverge at some other point. When these signals come together at a primitive like a setup-hold checker, the skew which is common to them needs to be subtracted out before the check is done, or else false error messages may be generated.

  One of the most common cases of this is a shift register. Skew on the clock to the register is common to all of the bits of the register, and needs to be subtracted out before checking the setup and hold times of the shift bits.

  Other common terms for reconvergent fanout are correlated skews and common ambiguity.

  This feature is controlled by a new directive RECONV_FANOUT, whose default value is OFF.

- **Clock Period Specification**

  An enhancement has been made to allow the clock period to be specified in the assertion syntax, which can be used either in case file specifications or signal names used in drawings.

  A clock period may be specified by giving a number between the exclamation mark (!) and the assertion letter (C, P, D, or S). This clock period has to be a sub-multiple of the system clock period.

For example, with the CLOCK_PERIOD and CLOCK_INTERVALS set to 100, the following assertions can be given:

Signal Assertion	Equivalent To
'SIG !50C0-25'	'SIG !C0-25, 50-75'
'SIG !25C5-10'	'SIG !C5-10, 30-35, 55-60, 80-85'

- **Case File Extended**

  The case file capability has been extended to allow assertions to be put on either subranges or individual bits of buses. In the past, all of the bits of a bus had to have the same assertions on them.

Chapter 6 of the reference manual has been updated to reflect the above changes.

RELEASE 7.25 TIMING VERIFIER

The 7.25 release of the Timing Verifier is a maintenance release for bug fixes only.

DOCUMENTATION UPDATES

The Timing Verifier error message descriptions have been added to Chapter 6 of the reference manual.

6.0 RELEASE 7.0 OF THE PLOTTIME PROGRAM

MAJOR CHANGES

The 7.0 release of the PLOTTIME program includes the following major changes:

- PLOTTIME no longer produces a script file for GED (PLOTTIME inserts the timing diagram directly into the specified SCALD directory).
- Timing diagrams can be given a user-specified name (the default name is TIMING.TIMING.1.1).
- PLOTTIME produces a binary GED file rather than an ASCII file (performance improvement).

CAUTION

Your PLOTTIME directives file must be changed in order for the program to work correctly. See "New Directives" section in the next section and Chapter 6 of the reference manual.

The procedure to look at the resulting timing diagrams is different; see "New Way To Look At Your Timing Diagram" (below) and Chapter 6.

NEW DIRECTIVES

There are four new directives for the PLOTTIME command file (td.cmd). The first three are associated with the input and output file names.

```
DIRECTORY 'your_scald_directory';
INPUT  'your_input_file';
OUTPUT 'your_output_drawing_name';
```

The only mandatory directive is the DIRECTORY directive (the other two are optional). If the INPUT directive is missing, PLOTTIME uses the file 'plotsig.dat' for input. If the OUTPUT directive is missing, the SCALD drawing created is TIMING.TIMING.

The fourth directive is called plot_size. This directive allows PLOTTIME to stretch the timing diagram over the entire page or screen. For example, specifying a plot size = 15.0 (inches) causes the program to calculate the ns_per_inch as (end_time_start_time)/plot_size. The format of this directive is:

```
PLOT_SIZE <real number>
```

NEW WAY TO LOOK AT YOUR TIMING DIAGRAMS

In the previous release, PLOTTIME produced an ASCII file that was read by the Graphics Editor's SCRIPT command. To produce timing diagrams faster, PLOTTIME now produces binary files and associates them with the drawing name you choose. You can either use the new directive OUTPUT or the default name, 'TIMING.TIMING'.

To look at your new timing diagram, invoke GED and use either the EDIT or GET command with the drawing name and the new suffix '.TIMING'. For example, if you gave the directive:

```
OUTPUT 'my drawing'
```

you should either

```
EDIT MY DRAWING.TIMING
```

or

```
GET MY DRAWING.TIMING
```

to cause GED to display the first page of the diagram. To get the second drawing, type either:

```
EDIT ..2
```

or

```
GET ..2
```

Note that if you do not use the OUTPUT directive, type the default name:

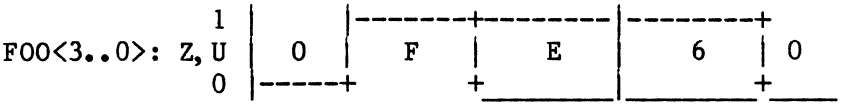
```
EDIT TIMING.TIMING.
```

or

```
GET TIMING.TIMING.
```

BUS SIGNALS

Previous releases of the PLOTTIME program drew the timing diagram for individual bits only. The 7.0 release of PLOTTIME is capable of plotting bus signals such as:



The values displayed are not necessarily hexadecimal (values are treated as strings and can be in any radix).

RELEASE 7.25 OF THE PLOTTIME PROGRAM

The 7.25 release of the PLOTTIME program is primarily a maintenance release.

ENHANCEMENTS

The precision of the clock interval variable has been changed to allow greater range in the time scale.

7.0 RELEASE 7.0 OF THE LOGIC SIMULATOR

The 7.0 release of the Logic Simulator includes a number of new features and improvements. Numerous bug fixes also have been made. Highlights of this release include the following (see the paragraphs below for details on each of these changes):

- The addition of Valid's Realchip (tm) system.
- A new radix to indicate signal strengths.
- Improvements to the signal display area in WAVEFORMS mode, including the ability to display up to 200 signals.
- Tracing of signals and tabular I/O.
- New primitives, including a J-K flip flop, bidirectional resistor and transistor, and a counter/shift register, as well as the addition of BCD arithmetic to the ALU primitive.
- Modification of the OPEN command syntax when operating under GED and utilizing the puck.
- Expanded breakpoint definition capabilities.
- Correction of a number of reported bugs (including addition of the capability to restart the Simulator under GED and correction of two conditions which would previously crash the program).

Existing circuits, Simulator models, command files, etc. do not need to be changed for compatibility with the 7.0 release.

REALCHIP SUPPORT

The 7.0 release of the Logic Simulator supports Valid's Realchip(tm) system that allows modeling of complex LSI and VLSI chips using the actual devices as hardware subroutines. The use of Realchip is described in the separate publication "Realchip User's Manual."

The Logic Simulator now accepts the directive

```
REALCHIP_LIBRARY = '<UNIX file name>'
```

which specifies the UNIX file containing the full set of device definition blocks for primitives that are to be modeled by Realchip reference elements. The format and definition of these device definition blocks is described in the "Realchip User's Manual."

This directive must be present if any Realchip models are to be used by the Simulator. Otherwise, this directive can be omitted.

Every reference element that is plugged into the Realchip system must have an associated device definition block in the Realchip library file. However, it is not an error to include extra device definition blocks in this file. At the end of the simulation session, the Simulator outputs the name and "jig ID" of each detected reference element into the "SIMLST" file.

BIDIRECTIONAL MOS SUPPORT

The 7.0 release of the Logic Simulator supports bidirectional MOS through the addition of the RES and PASS TRANSISTOR primitives and the INDETERMINATE signal strength.

The RES primitive is fully bidirectional and acts like a wire except that HARD strength signals are converted to SOFT strength when they pass through; RES primitives always have 0 delay. The RES primitive is SIZE wide, and the pins may not be bubbled.

The PASS TRANSISTOR primitive is fully bidirectional and acts like a switch. The G pin of the PASS TRANSISTOR controls whether the A and B pins are connected together. An active G pin (0 if the pin is bubbled, otherwise 1) causes the PASS TRANSISTOR to act like a wire to connect the A and B nets. An inactive G pin causes the PASS TRANSISTOR to act as if it is not in the circuit. The delay from A-to-B or B-to-A is always 0. The G pin has an input delay that assumes the value of the DELAY property on the PASS TRANSISTOR. The A and B pins of the PASS TRANSISTOR are SIZE wide and may not be bubbled. The G pin is always one bit wide and may be bubbled.

The INDETERMINATE signal strength represents a signal for which the logical level is known, but the drive strength is not. The set of signal strengths now supported is:

Signal Strength	Meaning
HARD	Driven to level without resistance
SOFT	Driven to level through resistance
MEMORY	Was driven to level, now holding due to charge storage
INDETERMINATE	Could be HARD, SOFT or MEMORY

MEMORY strength signals decay to (MEMORY strength, UNDEFINED value) after a user-settable DECAY time. This time is measured from the time the signal was last driven to the specified value. All signals in a design have the same DECAY time which is set with the DECAY_TIME <value in nanoseconds> directive.

DEPOSITing into bidirectional signals is not recommended as the deposited value will not persist very long due to the bidirectional net evaluation scheme used by the Simulator. Unidirectional drivers should be connected to those bidirectional nets that the user wishes to force to certain levels.

NEW RADIX OPTION

A new RADIX option has been added to allow the viewing and setting of signal strengths. The new RADIX is called STRENGTH and is selected with the RADIX S command. States are displayed and input for each bit of the signal using the following abbreviations:

STATE NAME	ABBREVIATION
HARD_STATE_0	h0
SOFT_STATE_0	s0
MEMORY_STATE_0	m0
INDETERMINATE_STATE_0	i0
HARD_STATE_1	h1
SOFT_STATE_1	s1
MEMORY_STATE_1	m1
INDETERMINATE_STATE_1	i1
HARD_STATE_U	hU
SOFT_STATE_U	sU
MEMORY_STATE_U	mU
STATE_Z	Z

WAVEFORMS DISPLAY IMPROVEMENTS

In WAVEFORMS mode, the display has been modified to use all available space on the screen. That is, instead of being limited to the former display size of 12 signals, the number of signals displayed in WAVEFORMS mode is now a function of the terminal used and how many rows will fit on the screen (as it already is in BUS mode). For the various terminal types, the size of the WAVEFORMS display is as follows:

Terminal Type	Max. Signals on Display
Cluster	48
Cluster running GED	12
Ann Arbor	34
VT 100	12
IBM 3270	14

In addition, signal names are now right-justified when printed in WAVEFORMS mode so that signals with long path names can be distinguished by their unique parts.

200 SIGNALS IN WAVEFORMS MODE

WAVEFORMS mode has been modified to store and display up to 200 signals (previous releases of the Simulator were limited to 12). As described above, the number of signals that can be displayed on the screen simultaneously is dependent on the type of terminal, but the signals on the display are controlled by the specification of the top row number using the new command, ROW. The following are examples of valid ROW commands:

```
ROW +168
ROW -10
ROW 33
```

The first two commands specify relative offsets to the present top row number, while the final command provides an absolute value for the top row to be displayed on the screen. The top row which is presently being displayed on the screen is indicated in WAVEFORMS mode at the upper right-hand corner of the display as "Top Row:".

OPENing a new signal in WAVEFORMS mode without specifying a row number causes the signal to be opened at the first available position. If no positions are available on the current screen, the display is scrolled to display the signal on some later row.

TRACING AND TABULAR I/O

The Simulator can now output a trace showing signal transitions. This trace can be in one of two formats (refer to Chapter 7 of the reference manual).

New Simulator Directives

The following directives have been added:

- TABULAR_TRACE { ON | OFF } ;

  Specifies the trace format. TABULAR_TRACE OFF, the default, specifies standard trace format, while TABULAR_TRACE ON specifies tabular trace format.

- BINARY_TRACE { ON | OFF } ;

  This directive is ignored for Tabular tracing. Specifying BINARY_TRACE ON causes the Value File to be output in binary. The default, BINARY_TRACE OFF, causes the Value File to be an ASCII file.

New Simulator Commands

The following commands have been added:

- TRace <signal name>

  Tells the Simulator to trace the output or outputs corresponding to the given signal or signal subrange.

  Examples: * trace foo
  * trace bar<66..33>
  * trace bar<4>

  See the LIST TRACES command later in this section.

- TRACE_All

  Trace all outputs of all signals that can be opened (this does not include NC signals). If using standard tracing, also trace the contents of all memories.

- TRACE_Close

  Close all trace output files. This usually means that all tracing for the current simulation is complete.

- TRACE_Interval <number>

  For Tabular I/O format, causes a trace record to be output every <number> nanoseconds during the simulation. <number> must not be less than 0. If <number> is 0, the default, a trace record is written every time there is at least one transition. This command is ignored if the standard trace format is being used.

- TRACE_Mem

  Trace the contents of the memory currently specified by the MEM_PATH command. This command is valid only for standard tracing.

- TRACE_Open

  Open the trace output file(s). If the simulation is using the standard trace format, the signal mapping file is output when this command is given.

- TRACE_Read <file name>

  Read in a Tabular I/O trace file from a previous run (or a hand-generated file), and use it to stimulate the circuit. The signals to be traced are first read in, followed by the list of times and signal values. As each time is reached in the simulation, the values for that time are deposited into the proper signals. To see the values being deposited as the simulation advances, use the UPDATE_INTERVAL command.

- TRACE_Start

  Begin tracing the outputs that have been specified by either the TRACE_ALL command or some TRACE commands. The TRACE_START command can be given any number of times during a simulation run. See the TRACE_STOP command.

- TRACE_STOp

  Discontinue tracing until another TRACE_START command is entered.

- List Traces

  List all signals, subranges, and memories that are currently being traced.

PRIMITIVES

The following primitives have been added to the Simulator in the 7.0 release:

- JK Primitive

  The JK primitive models the J-K Flip Flop. The primitive has input pins for J and K data inputs, set and reset functions, and a clock. Outputs consist of Q and Q-BAR data outputs. Asserting both the set and reset pins cause both outputs to go high.

- LATCH RS COMP Primitive

  The LATCH RS COMP primitive is very similar to the LATCH RS primitive except that complementary outputs are provided. The latch has one data input, and inputs for set, reset, and enable. Outputs consist of Q and Q-BAR data outputs. Asserting both the set and reset pins cause both outputs to go high.

- REG RS COMP Primitive

  The REG RS COMP primitive is very similar to the REG RS primitive except that complementary outputs are provided. The primitive has input pins for data, set and reset functions, and a clock. Outputs consist of Q and Q-BAR data outputs. Asserting both the set and reset pins cause both outputs to go high.

- RES Primitive

  The RES primitive models a bidirectional resistor. The function is described above under BIDIRECTIONAL MOS SUPPORT.

- PASS TRANSISTOR Primitive

  The PASS TRANSISTOR primitive models a bidirectional MOS transistor. The function is described above under BIDIRECTIONAL MOS SUPPORT.

- COUNTER SHIFT REGISTER Primitive

  There is an up-down counter primitive with right and left shifting capabilities. This primitive has seven inputs: MR - Master reset, CK - clock, CEP - count enable parallel input (active low), CET - count enable trickle input (active low), S - select inputs (3 bits), DI - parallel data in, and MSBIN - serial data input; it produces two outputs: DO - data out and TC - terminal count (active low). This primitive is patterned after the 100136 ECL part. Note that part delays on any part now can be specified using real numbers (for compatibility with other programs). In the Simulator, these numbers are rounded-off before simulation.

NEW and EXTENDED SIMULATOR COMMANDS

The following commands have been added or enhanced:

- OPEN Command

  The syntax of the OPEN command has been extended: now, when running under GED, several signals can be opened with one OPEN command by successively pointing at a signal in the GED window and then pointing at a location in the Simulator window. That is,

  ```
  OPEN <signal pt> <location pt> <signal pt> <location pt>
       ... <signal pt> <location pt> ;
  ```

  Note that, with this change, either a semicolon or a carriage return MUST be entered to terminate the OPEN command.

- BREAKPOINT Commands

  The expression syntax for breakpoint commands has been extended. In this release, the Simulator accepts breakpoints on multiple-bit signals and all relational operators in the syntax shown below. It is possible to set breakpoints with very complex expressions that evaluate to a single bit value (logic 0 or 1). In evaluating the expression, all relational expressions take precedence over all boolean operators except the NOT operator.

```
<relational expression> -> <term> <rel OP> <term>
                        -> <term>

<rel OP> -> <'='>          { equal }
         -> <'<>'>         { not equal }
         -> <'>='>         { greater than or equal }
         -> <'<='>         { less than or equal }
         -> <'<<'>         { less than }
         -> <'>>'>         { greater than }

<term>   -> <factor>

<factor> -> <signal>
         -> ( <expression> )
         -> NOT <factor>   { boolean NOT }
         -> 0              { constant 0 }
         -> 1              { constant 1 }
         -> & <constant>   { any constant, given in current radix }
```

  When using <rel OP> symbols, they must be separated from <term>s by spaces to prevent confusion in parsing. Note that signals and constants may be any width, but the <expression> used in a breakpoint or enable definition must evaluate to a single bit. Note also that any constants used in an expression other than 0 and 1 must be preceded by the "&" symbol.

- PLOT Command

  The PLOT command is used to build a timing diagrams file for plotting via the PLOTTIME program and GED. The format of the command is:

```
PLOT [<starting_time> <ending_time>] [<filename>]
```

  The PLOT command plots all signals that are open in WAVEFORMS mode, for the specified time range. The default values for the optional parameters are the waveform starting time and ending time, and a file named 'plotsig.dat', respectively. The file produced is an input file to the PLOTTIME program to obtain a timing diagram.

- UPDATE_INTERVAL Command

  The UPDATE_INTERVAL command directs the Simulator to update the screen at specified intervals while simulating for a longer time. If zero (0) is specified, any previously set interval is cleared and updating is disabled. The format of this command is:

  UPDATE_INTERVAL <constant>

- ERASE Command

  The ERASE command has been revised to erase the signals only in the mode in which one is operating. That is, executing the ERASE command in BUS mode will erase the current screen, but not the signals displayed in WAVEFORMS mode. ERASE also resets the top row number to 1 in WAVEFORMS mode.

- SIMULATE Command

  The SIMULATE command has been changed so that all primitive evaluations are made through the current time rather than to just before this time (note that this was an undocumented change in Release 6.0). This means that 0 is now a valid simulation interval and allows the user to deposit values into zero-delay parts and to cause their evaluation using "SIMULATE 0" without advancing simulation time.

  An additional parameter may be specified after the existing SIMULATE command to control the WAVEFORMS mode display when simulating past the end time on the display. This optional parameter specifies the percentage of the screen width that is to be occupied by waveforms after the time scale is shifted for the simulation. Formerly, when simulating past the end time, the display was redrawn to fill 50 percent of the width of the screen; this amount of fill remains the default if no parameter is specified.

  To illustrate the usage of this new parameter, assume that the WAVEFORMS display is arranged for simulations from 2100 to 3100 nanoseconds. If, after 3000 ns of simulation, the next simulation period is to be 200 ns (i.e., to 3200 ns), the final display time will be exceeded. If no percentage is specified, the display will be redrawn to show from 2700 to 3700 ns (i.e., 3200 will be in the middle of the screen). If, instead, the user wishes the new display to extend from 3000 to 4000 ns, the new SIMULATE parameter can be used to specify that 3200 ns should be displayed at a position 20 percent across the screen. Thus, the command entered by the user would be:

  SIMULATE 200 20

Since the new parameter is a percent value, inputs must be in the range 0 (to display none of the previous simulation period) to 100 (filling the screen), inclusive. Values must be whole numbers. This parameter has no effect if the specified simulation time does not exceed the final display time.

RELEASE 7.25 OF THE LOGIC SIMULATOR

NEW SIMULATOR COMMAND

- o TRACE_RESet

  Turn off circuit stimulation from input files. The TRACE_RESET command can be used at any time to disable a TRACE_READ command (for example, after a LOGIC_INIT command); additional TRACE_READ commands can then be entered.

DOCUMENTATION CHANGES

A list of the Logic Simulator error messages has been added to Chapter 7 of the reference manual.

RELEASE 7.5 OF THE LOGIC SIMULATOR

GENERAL DESCRIPTION

The optional 7.5 release of the Simulator contains several new features and added capabilities over the 7.25 Simulator. Highlights of this release are listed below and are described in more detail within this document:

- o Support for different radices in tabular I/O.
- o Separate rise/fall delays for all Simulator primitives.
- o Wire delay feedback.
- o User-specified time resolution.
- o Simple coverage analysis to indicate which signals have made a transition.

Existing circuits, Simulator models, command files, etc. do not need to be changed. Incompatibilities among versions 7.0, 7.25, and 7.5 should be reported as bugs unless otherwise described in this document.

DIFFERENT RADICES IN TABULAR I/O

In releases 7.0 and 7.25, users were limited to using binary for tabular I/O values. The 7.5 Simulator has been modified to accept values in a radix other than binary. Now, these values may also be specified in octal, decimal, or hexadecimal using a command of the following format:

TRACE <signal name>, <radix>

The radix may be specified using numerals (2, 8, 10, or 16) or characters (b, o, d, or h). The LIST TRACES command has been changed to display the trace radix following the signal name.

If the TRACE command is given without a radix, the default trace radix is used. This default is initially 2, but some other value may be specified using the TRACE_RADIX directive. The format for this directive is:

TRACE_RADIX { 2 | 8 | 10 | 16 };

The default radix may also be changed at any time using the new TRACE_RADIX command:

TRACE_RADIX [2 | 8 | 10 | 16 | b | o | d | h]

If no argument is given, the current default trace radix is displayed. Note that tabular I/O files from 7.0 and 7.25 cannot be used without appending ", 2" (to denote binary radix) after each signal name at the beginning of the file (e.g., "foo" becomes "foo, 2").

SEPARATE RISE/FALL DELAYS

Delays associated with Simulator primitives have been modified so that different times may be specified corresponding to a rise delay and a fall delay. Specification of these delays is made through the modified DELAY property or through the new properties, RISE and FALL.

The DELAY property has been modified to accept two values, a rise delay followed by a fall delay (separated by a comma). If only one value is specified, this value is used as both the rise and fall delay. Accordingly, delay can be specified in either of the following formats:

DELAY <delay time>
DELAY <rise delay>, <fall delay>

In addition, rise and fall delays can be specified using the RISE and FALL properties. Usage of these properties is as follows:

RISE <rise delay>
FALL <fall delay>

Note that both the DELAY property and the RISE and FALL properties cannot be specified on the same body or an error will result.

A new directive has been added to the Simulator to control the use of separate RISE/FALL delays. The format of this new directive is:

```
RISE_FALL { OFF | ON }
```

If the ON state is specified, simulations are performed using both the rise and fall delays specified for parts. The default state of this directive is OFF which causes all primitives to change states after the specified delay time (if only one value is given) or after the greater of the rise and fall delays.

When separate rise/fall delays are specified, the delay used for the various transitions is as follows (where X indicates any value):

old value output	new value	delay to use
X	0	fall
X	1	rise
X	U	min(rise, fall)
0	Z	rise
1	Z	fall
U	Z	max(rise, fall)

As a result of this new feature, changes were made to the set of functions provided for UCP's. The function GET_DELAY now returns the greater of the rise and fall delays. In addition, two new functions, GET_RISE and GET_FALL have been added to return the rise and fall delays of the primitive, respectively.

WIRE DELAY FEEDBACK

Wire delay feedback has been added to the Simulator. The wire delays can be fed back in either of two ways:

1. By using a directive of the form

   ```
   WIRE_DELAYS 'filename';
   ```

2. By using a command of the form

   ```
   WIRE_DELAYS filename [;]
   ```

The file must be in the format described below. Basically, each element consists of a signal name (in quotes), a bit subscript (if any), and a delay element or a list of path names of components that the signal drives with a delay for each bit. These delays are added in with any other specified delay values to determine when Simulator events are scheduled for those bits.

```
<delay file>          ::= END. |
                          <delay list> ; END.

<delay list>          ::= <signal delay list>; |
                          <signal delay list> ; <delay list>

<signal delay list>   ::= <signal name> := <stop delay list>

<stop delay list>     ::= <stop delay>; |
                          <stop delay>; <stop delay list>

<stop delay>          ::= = <quoted rise/fall range> |
                          <quoted path name> =
                              <quoted rise/fall range>

<signal name>         ::= <quoted signal name> |
                          <quoted signal name> < <bit range> >

<bit range>           ::= <bit number>  |
                          <bit number> .. <bit number>

<bit number>          ::= <integer>

<quoted rise/fall range>
                      ::= '<delay>' |
                          '<delay range>' |
                          '<rise delay range>,
                              <fall delay range>'

<rise delay range>    ::= <min delay> - <max delay>

<fall delay range>    ::= <min delay> - <max delay>

<min delay>           ::= <fixed point number>

<max delay>           ::= <fixed point number>

<delay range>         ::= <delay>, <delay> |
                          <delay> - <delay>

<delay>               ::= <fixed point number>
```

At present, the Simulator does not support the following:

```
<stop delay>        ::= <quoted path name> =
                            <quoted rise/fall range>

<min delay>         ::= <fixed point number>

<max delay>         ::= <fixed point number>

<delay range>       ::= <delay> - <delay>
```

In other words, the delay specified for a signal currently is applied to all of its inputs, and ranges for the delay values have not as yet been implemented. The following is an example of a wire delay file:

```
'FOO' <5..0>: = '2.3, 3.4';

'FOO BAR' : = '5.1';

END.
```

PERFORMANCE ENHANCEMENTS IN MOS

The performance of MOS simulation has been improved by making the decay time feature associated with MOS primitives default to "no decay". That is, unless the user explicitly specifies a decay time (using the DECAY_TIME directive or DECAY_TIME command), MOS signal strengths do not decay over time.

USER-SPECIFIED TIME RESOLUTION

A user can now specify the time resolution to be used by the Simulator through a new directive, RESOLUTION. This feature allows a user to specify a finer resolution when such capabilities are needed, or to increase the speed of simulations when a more coarse resolution is being used. The format of this directive is:

```
RESOLUTION <time>
```

The time resolution should be specified as a real number of nanoseconds and need not be a power of ten (e.g., 50 picoseconds is expressed as 0.05 and 2 microseconds is expressed as 2000). The default value is 1 nanosecond, the resolution previously used by the Simulator.

The current resolution used by the Simulator is indicated in the display area as a fixed point value labeled "Scale:". In BUS mode, this indication appears on the second line of the screen. In WAVEFORMS mode, the resolution appears near the bottom of the display area below the tick marks and time scale.

The addition of this feature affects the user interface in several areas. The most obvious of these is that the time scale indicated on the screen in WAVEFORMS mode no longer represents nanoseconds, but must be scaled by the indicated scale factor. For example, by specifying "RESOLUTION 20", each tick (formerly 1 ns) now represents 20 ns.

This feature also impacts time values entered into the Simulator or used by the Simulator. Some values are scaled based on the time resolution specified in this directive; these include the clock period, signal histories, signal delays, and decay times. These values are typically specified in nanoseconds, and this remains true; however, since the display may not be in nanoseconds, the values must be appropriately scaled by the Simulator. The following examples will help clarify time resolution. With the same scale factor of 20 used above:

1. A clock period of 500 ns divided into 10 intervals is displayed as "Clock: 25 / 10".

2. The default for signal history remains the same (1000 ns), but since each tick on the screen now represents 20 ns, history is only retained for 50 ticks.

3. Time values specified in input files (e.g., delay), do not have to be scaled by the user. The input units remain the same, but the values are scaled by the Simulator for display on the screen. For example, a 10000 ns decay time still is specified by "DECAY_TIME 10000", but signals change value after 500 ticks.

On the other hand, screen-oriented times maintain their relation to ticks on the screen (although the "real" times associated with those ticks have changed). For example, while "WAVEFORM 0 100" displays a time scale from 0 to 100 ticks, "SIM 100" advances simulated time by 100 ticks, and "CURSOR 25" sets the cursor to a location corresponding to 25 ticks. Still using the example with a scale factor of 20, the 25 and 100 ticks correspond to "real" times of 500 and 2000 ns, respectively.

The user should exercise caution when manipulating the time resolution. Too fine a resolution decreases the execution speed of the Simulator (simulating for hundreds of ticks even when no events are being scheduled) or could result in the generation of massive amounts of signal histories. On the other hand, before decreasing the resolution, one must ensure that the specification of other time values is correspondingly coarse (i.e., it probably does not make sense to specify "RESOLUTION 50" with a 20 ns clock interval).

NEW SIMULATOR DIRECTIVES

CLOCK_ON_DRIVEN Directive

In previous releases, if the clock property was specified for a signal, the Simulator built a clock generator for that signal even if it was driven by some other signal. The new directive:

```
CLOCK_ON_DRIVEN { OFF | ON };
```

has been added for building clock generators on driven signals. The default for the directive is OFF (which only permits timing assertions to be specified on undriven signals). Thus, building a clock generator on a driven signal is no longer allowed unless this directive is specified as ON.

Other Directives

Several new directives have been described previously in this document. Below is a summary of these directives (see above for a more complete description):

TRACE_RADIX { 2 | 8 | 10 | 16 };	defines default trace radix (default: 2)
RISE_FALL { OFF | ON };	enables separate primitive rise/fall delays (default: OFF)
WIRE_DELAYS 'filename';	specifies file for wire delays
RESOLUTION <time>;	specifies Simulator time resolution (default: 1 ns)

NEW/MODIFIED COMMANDS IN THE SIMULATOR

TRACE Command

The TRACE command has been modified to take advantage of the puck when running the Simulator under GED. Signals to be traced may now be specified by pointing to them with the puck using the following command format:

```
TRACE <point> <point> ... ;
```

Thus, signals can easily be specified for tracing in the default radix without typing their signal name.

ASSERTIONS Command

The ASSERTIONS command is a new command that allows timing assertions to be specified while running the Simulator. This command allows the user to specify assertions interactively rather than with the signal name given when creating the drawing in GED. Addition of this feature provides the user with an extra degree of flexibility when performing simulations since signal timing assertions are no longer fixed with the signal name and need not be compiled with the drawing. Usage of this command is as follows:

ASSERTIONS < signal name >, < timing data >

The < timing data > parameter is specified using the standard SCALD syntax for timing assertion data (e.g., 0-4). The assertion type should not be specified - the Simulator automatically adds the "!C" property to the timing data.

This command can be invoked on existing clock signals as well as any other signals in the drawing. Thus, any signal can be assigned timing assertions while in the Simulator, and assertions of existing clock signals can be re-defined. After assigning clock properties, the signal can be OPENed using either its previous or its new (with assertions) name.

COVERAGE Command

Simple coverage analysis has been added to the Simulator to enable the user to obtain a list of the signals that have made a transition during a period of simulation. This list can then be used to ensure that all signals in the circuit have been exercised.

Coverage analysis is controlled by the COVERAGE command. The format of this command is:

COVERAGE [ON | OFF]

If no parameter is given, the current status of the coverage analysis is reported. If coverage analysis is off, the Simulator will not track the number of transitions.

At any time (whether coverage analysis is enabled or not), the user can output the list of signals that have made a transition and the number of transitions that they have made by using the WRITE_COVERAGE command. This command outputs the list to a file with the specified name. The format of this command is:

WRITE_COVERAGE < filename > [, { 0 | 1 | 2 | 3 }]

If no parameters are given, the user is prompted for a file name. If the optional parameter (0 - 3) is given, the signals are processed based on the number of times that they have made a transition. The signals are sorted by the number of transitions, and the file only contains those signal names in specific groups; for example, specifying "0" indicates that only signals making 0 transitions (i.e., those that have not changed) should be output, and "1" indicates that only those signals making 0 or 1 transitions are output.

To clear the list of signals that have made a transition, the user must invoke the INIT_COVERAGE command. This command, which has no parameters, enables the user to invoke coverage analysis for different periods of simulation. Note that turning coverage analysis OFF does not clear this list - this command must be invoked each time a new list of signals is to be started (except the first, when the list is empty), regardless of the use of the COVERAGE command.

RECORD_SIGNALS Command

The RECORD_SIGNALS command causes the signal histories of all signals in the circuit to be recorded. Previously, a signal had to be OPENed in WAVEFORMS mode in order to start a recording of its history. Thus, after a period of simulation, if a signal was not OPENed, there would be no method to determine what the value of a signal was at a previous time. By invoking this command, which takes no parameters, the history of all signals is available thereafter.

Note that this command does not affect the duration of history which is maintained for all signals - history only is preserved for the interval specified using the HISTORY command (or the default). Also note that since certain storage requirements are involved in creating and maintaining history, this command should not be invoked on large circuits.

RECORD_ALL Command

The RECORD_ALL command causes the signal histories of all signals and all memories in a circuit to be recorded. This command is identical to the RECORD_SIGNALS command described above except that the history of all locations of all memories also is recorded. This command requires no parameters and has no effect on the duration of history maintained for all signals.

Note that considerable storage requirements could be involved in creating and maintaining a history of all signals and memories. Thus, this command should not be invoked on circuits with a large number of elements and/or large memories.

SCROLL Command

The SCROLL command allows the user to control the automatic scrolling feature of the Simulator. The format of this command is:

```
SCROLL [ ON | OFF ]
```

The default is ON, which causes the Simulator display to scroll in WAVEFORMS mode when a signal not currently on the screen is OPENed. Using this command to turn the feature OFF allows the user to OPEN and DEPOSIT into signals that are not on the display.

DOCUMENTATION

The 7.5 Simulator enhancements previously described have not been incorporated in the 7.25 Reference Manual Update Package (900-00049); a separate update package to update the Simulator chapter currently is being prepared.

8.0 PACKAGER

RELEASE 7.0 OF THE PACKAGER

The 7.0 release of the Packager has several major changes, new features, and bug fixes. The changes include a new name, physical part tables, and manual pin assignments.

NEW NAME

The name of the program has changed from the Post Processor to the Packager. This change has been made to more fully express the new philosophy of the Packager in the SCALDsystem. Its main task will now be to package a logical design into a physical design. The Packager is no longer concerned with the creation of files for several different physical design systems, but rather to generate a standard description of the physical design.

This change in philosophy means that interface programs are required to convert the physical designs produced by the Packager to forms that are acceptable to specific physical design systems. Valid has a variety of interface programs such as SCICARDS, HDL, SPICE, TEGAS V, and LOGCAP. Users can also write their own interface programs using the capabilities provided by DIAL.

As part of this simplification, the Packager no longer generates the Concise Net List, Concise Part List, Body Ordered Net List, Power and Ground List, and the Stuff List; these files are now generated by separate DIAL programs provided by Valid.

To run the Packager, you now type the command:

```
package
```

The Packager directives are now read from the file:

```
packager.cmd
```

Because physical DIAL programs require the expanded part and net lists from the Packager, make sure to generate these files with the directive:

```
OUTPUT EXPANDEDPARTLIST, EXPANDEDNETLIST;
```

PHYSICAL PART TABLES

Physical part tables give the users the ability to assign a property to a part that causes the Packager to create a new part type from the basic part type. For example, a resistor may have a VALUE property attached that causes a different PART_NUMBER to be used, but otherwise the part definition would be the same as for an unselected resistor. There is only one library definition for the part, and therefore only one copy of the model. The

Packager uses the properties attached to the part to differentiate it from other instances of the same part.

Another use of physical part tables is to attach new body properties to a part type without having to recreate or modify the library files containing the part types definitions. An important use of this capability is the addition of new properties to the libraries for certain interfaces such as SCICARDS. These properties describe the type and shape of each component to the interface.

By using several physical part tables, users can change the way part types are handled without changing the library files. This fact is useful when a design is processed by several different interfaces. The properties for each new interface do not have to be added to the libraries, but instead are concentrated together into their own special physical part table. A user only needs to specify which physical part table the Packager is to use.

To tell the Packager the names of the files that contain physical part tables, the user must use the PART_TABLE_FILE directive. Any number of tables can be specified with this directive. The names can be placed in a list separated by commas or listed individually with separate PART_TABLE_FILE directives. For example, the directive

```
PART_TABLE_FILE 'res.tab', 'cap.tab';
```

specifies two physical part table files, res.tab and cap.tab, and is equivalent to the directives:

```
PART_TABLE_FILE 'res.tab';
PART_TABLE_FILE 'cap.tab';
```

For more information on physical part tables, please refer to the section "Using the Packager Physical Part Tables" in Chapter 8 of the reference manual.

MANUAL PIN ASSIGNMENTS

The user can now manually assign the pins of logical parts in the drawings and have the Packager perform the specified assignments. Pins are assigned through the new Graphics Editor command PINSWAP.

In addition, only pins in the same swap group are permitted to be swapped. A swappable group of pins are those pins that are logically equivalent and belong to the same section (i.e., if two nets are swapped between two pins that are in a swappable group, the logical function of the circuit is not altered).

A common example of this occurs for the inputs of an NAND gate like a 74LS00. The two input pins are physically equivalent in terms of loading and propagation delay from input to output; if the nets to the input pins are swapped, the behavior of the circuit remains unchanged.

To define a swappable group, the library files must have the PIN_GROUP property defined. Any set of swappable pins must have the PIN_GROUP attached to them with the same value. Any pin without the PIN_GROUP property cannot be swapped with any other pins. The value of the PIN_GROUP property is not important, only that all pins of a swappable group have an identical value.

Currently, the only parts that can have pin assignments are those that already have been assigned to a section with the Graphics Editor's SECTION command. It is an error to try to PINSWAP pins of a part that has not been SECTIONed.

Once pin swaps have been performed on a part, further section assignments are no longer allowed for the part (i.e., if the user wishes to assign a part to a different section after performing pin swaps, the part must first be de-assigned by using the REPLACE command; the user can then assign the new part to the desired section).

MANUAL SECTION ASSIGNMENT ENHANCEMENT

The SECTION command has been enhanced so that the user can now directly specify the section to be assigned instead of having to step through each of the available sections. This capability is useful when the number of available sections is large as is the case with connectors.

To perform a direct section assignment, the user must first type in the pin number of a pin that uniquely identifies a section before pointing to the body or pin of the part to be assigned (a pin that uniquely identifies a section is one that is not common to more that one section of the part).

GLOBAL PROPERTIES

The packager has been changed so that it is possible to find all of the global design properties attached to a global variable. This global variable is the variable GLOBAL_PROPERTY_LIST. The properties that are in the list on this variable are all of the properties that are attached to the drawing body of the root drawing.

EXPANDED PARTS LIST CHANGES

The new 7.0 version of the SCALD Compiler outputs global properties of a design into the expansion file. To pass these properties on to the interfaces and DIAL programs, the Packager outputs these properties in the directives block of the Expanded Part List.

These global properties are printed one per line after the root drawing name, compilation time, and packaging time. As an example, we can have the following Expanded Part List

```
FILE_TYPE=EXPANDEDPARTLIST;
DIRECTIVES
 ROOT_DRAWING='TTL Example';
 COMPILE_TIME=' COMPILATION ON 13-OCT-1982 AT 11:24:37.66';
 POST_TIME='23-JAN-1984 23:50:39.70';
 REVISION='2a';
 DESIGNER='A. E. Steinmetz';
END_DIRECTIVES;
      .
      .
      .
END.
```

where REVISION and DESIGNER are global design properties.

NEW AND ENHANCED PACKAGER DIRECTIVES

The following is a list of new and changed directives for the Packager.

- HEADER_FILE

  The HEADER_FILE directive has been removed from the Packager since all the part and net lists that can have a header are no longer generated by the Packager.

- INCLUDE_IO_LIST

  The INCLUDE_IO_LIST directive now only determines if the interface pins of the design are to be output to the Expanded Net List as the IO_NET net property since the Concise Net List and Body Ordered Net List are no longer generated.

- OUTPUT

  The OUTPUT directive has been modified to remove the following output files:

  ```
  CONCISENETLIST
  CONCISEPARTLIST
  CBODYORDEREDLIST
  POWERANDGNDLIST
  STUFFLIST
  ```

  To generate these lists, use the specific DIAL interface programs after running the Packager.

- OUTPUT_FORMAT

  The OUTPUT_FORMAT directive has been removed from the Packager. To generate HDL output, use the HDL DIAL interface program after running the Packager.

- PART_TABLE_FILE

  This new directive is used to specify the names of the files containing physical part tables. For more details, see above.

- SINGLE_NODE_NETS

  This directive has been removed from the Packager since the Packager no longer generates any net lists that can have single-node nets suppressed.

RELEASE 7.25 OF THE PACKAGER

The 7.25 release of the Packager consists primarily of bug fixes.

NEW FEATURE

The 7.0 Packager outputs its list and log data in the files PCKLST.DAT and PCKLOG.DAT. For compatibility with the 6.0 release of of the Post Processor (replaced by the Packager in release 7.0), the 7.25 Packager now outputs its list and log data to the files PSTLST.DAT and PSTLOG.DAT.

DOCUMENTATION CHANGES

A (partial) list of the Packager error messages have been added to Chapter 8 of the reference manual.

9.0 RELEASE 7.0 OF DIAL

The 7.0 release of DIAL consists of many changes. Most of these changes are routines that have been added to make DIAL easier to use; some of the changes are fixes to existing DIAL routines.

DOCUMENTATION UPDATES

The section on DIAL in the reference manual has been updated to reflect the changes and moved to Chapter 9.

NEW DIAL ROUTINES

The following routines have been added to DIAL to make it easier for the user to write DIAL programs.

- o IS_CHAR_IN_STRING: Checks to see if the character passed to the routine is in the given string. If present, returns TRUE.
- o PRINT_BEFORE_CHAR: Prints the given string until an occurrence of the character is found which is then passed to the routine.
- o PRINT_AFTER_CHAR: Prints out all of the characters in the string following the first occurrence of the given character.
- o GET_INPUT_ONLY_PINS: Changes the given list of nodes to only include those nodes that describe input pins to a part. This list contains nodes that are strictly input.
- o GET_OUTPUT_ONLY_PINS: Changes the given list of nodes to only include those nodes that describe output pins to a part. This list contains nodes that are strictly output.
- o GET_BIDIRECTIONAL_PINS: Changes the given list of nodes to only include those nodes that describe bidirectional pins. This list contains nodes that are strictly bidirectional.
- o GET_INPUT_PINS: Changes the given list of nodes to include those nodes that describe all input pins, whether the pin is strictly input or bidirectional.
- o GET_OUTPUT_PINS: Changes the given list of nodes to include those nodes that describe all output pins, whether the pin is strictly output or bidirectional.

GLOBAL PROPERTIES

DIAL has been changed so it is possible to find all of the global design properties attached to a global variable. This global variable is the variable GLOBAL_PROPERTY_LIST. The properties that are in the list on this variable are all of the properties that are attached to the drawing body of the root drawing.

CHANGES TO CURRENT DIAL ROUTINES

Many changes have been made to DIAL that fix bugs and add new capabilities to old DIAL routines. Some of these changes may affect the way an interface will run.

- IMPORTANT CHANGE - This change must be made for all interfaces. Three new file variables have been added to the 7.0 version of DIAL. These variables are:

  ```
  DIALStat
  DIALBack
  inprog
  ```

  These variables must be added to the program header and the list of file variables in the user's program (see template.pas for the correct order of the variables). If this change is not made, a program will not run correctly.

- There was a problem found in some of the routines that took the next element from a list. This problem caused the routine to not return a NIL pointer when the list was empty. This problem has been fixed in the following routines:

  ```
  GET_PHYSICAL_NET
  GET_LOGICAL_NET
  GET_PART_TYPE
  GET_INSTANCE_DRAWING_NAME
  GET_GENERIC_DRAWING_NAME
  ```

- The manner in which a physical part is named has been fixed so that the name assigned to the part is no longer than the value of the variable physical_part_name. This variable is used to set the maximum number of characters in a physical part name.

- The user routine CREATE_NET_ABBREVIATION was fixed to make certain that every net name starts with an alphabetic character.

- It is now possible for DIAL to read in multiple library entries for a part. It is possible for vectored pins to be described in multiple library entries. The only limitation is that a specific element of a pin cannot be described in more than one library entry.

- Added the OUTPUT directive to DIAL. It is now possible to specify which cross reference files are to be produced by the run of a program. The possible output files are:

```
LOCALPARTXREF     - local part cross references
GLOBALSIGNALXREF  - cross reference of global signals
GLOBALPARTXREF    - global part cross references
CROSSREFERENCES   - all cross references
```

10.0 INTERFACES

Interfaces are released independently of the SCALDsystem software.

RELEASE STATUS

The following interfaces are in final release:

BOM*	McLDL	TEGAS V
CADDS	PARAGON	TELESIS
HDL	SCALD*	WIRE DELAY*
LIB*	SCICARDS	WIRE WRAP
LOGCAP/MOTOROLA	SPICE	

* Standard interfaces

Note that the individual interfaces are described in separate publications (contact your local Valid sales engineer).

CHANGES TO THE SPICE INTERFACE

The following changes have been made to the SPICE interface:

- ALTER Statement -- there is no limit on the number of .ALTER cards that can be included in the spice deck (the circuit is reanalyzed as many times as the number of .ALTER cards). Subsequent ALTER operations employ parameters of the previous change, and no topological change of the circuit is allowed.

- SOURCE-STEPPING Method -- the source-stepping method can enhance DC convergence, but it is slower than the direct use of the Newton-Raphson method and is best used as an alternative to achieve convergence of DC operating point when the circuit fails to converge when using the Newton-Raphson method. The source-stepping method is used by SPICE when the variable ITL6 in the .OPTIONS card is set to the iterations limit at each step of the source(s). For example, .OPTIONS ITL6=30 causes SPICE to use the source-stepping method with an iteration limit of 30 at each step. By default, ITL6 is 0 which means to use the Newton-Raphson method directly.

11.0 LIBRARIES

Libraries are released independently of the SCALDsystem software.

LIBRARIES AVAILABLE WITH THE 7.0 RELEASE

The following additional libraries were available with the 7.0 release:

HCA6348 (HCA6300)
ASTTL
ALSTTL
HCMOS
CMOS
MM74C

LIBRARIES SUPPORTED WITH THE 7.25 RELEASE

The following additional libraries are supported with the 7.25 release:

AMCCQ700*
CDC6000*
FCG*
HCMOS54
TISC*
SCX3MU*

* Supplied directly by the manufacturer

12.0 UTILITIES

The following utilities described in Chapter 11 of the reference manual have been changed with releases 7.0 and 7.25 of the SCALDsystem software.

FILECOPY (RELEASE 7.0)

The FILECOPY utility described in Chapter 11 of the reference manual requires the 7.0 virtual-memory UNIX and cannot be used on pre-7.0 systems (pre 7.0 versions of FILECOPY cannot be run under 7.0 UNIX).

COM232/CU232 FILE TRANSFER PROGRAM

The com232/cu232 package transfers text files between two SCALDsystem S32s or between an S32 and a DEC VAX-11 machine over a medium-speed serial line using RS-232C electrical protocol with EIA levels and a simple full duplex ASCII echo-controlled transmission protocol. Among the alternatives supplied by Valid, the com232/cu232 package provides the simplest communication using the least costly resources. The DR-11 link or Ethernet is faster and more flavorful.

The cu232 program runs on the SCALDsystem S32 and is supplied as an executable file to be run under the UNIX operating system. An executable version of the com232 program is provided in the release for file transfers between two SCALDsystem S32s. A VAX version of com232 is provided in pascal source code form for compilation on a DEC VAX-11 (the VAX version source file is called com232.vax and is located in the /u0/src directory on the S32). Knowledgeable customers may wish to use the VAX code as a model from which code can be developed to transfer binary files or to run com232 on a machine other than a DEC VAX-11.

The operating system must provide full duplex transmission using ASCII, and must allow the program to turn echoing on and off. Characters with ASCII codes 32 through 124 (octal 040-0174, hex 0x20-0x7c) must be transmitted and received as sent, unmodified in any way by the operating system. SCALDsystem UNIX, DEC VAX-11 VMS, and perhaps other operating systems meet these requirements.

Com232 is a "slave" program while Cu232 is the "master" program. In general, each end suspends echoing for the duration of a transfer. The programs use a software protocol to provide checked transmission of named text files. (By convention on UNIX systems, text files contain lines of characters terminated by a linefeed character. On VMS and some other systems, each line is a separate record and a text file is a sequence of records. Com232 uses Pascal I/O library routines to handle the conversion.) The protocol uses packet transmission with checksums and packet sequence numbers and retransmission of bad packets to implement reliable transmission. The packet format provides for transmission of file names, read/write operation codes, data, ack/nak, and end-of-file. Each packet is positively acknowledged via ack or nak. Ack/nak packets are not acknowledged. A packet is (re)transmitted until it is ack'ed, and the protocol never gives up. If the other side doesn't send

anything at all, or repeatedly sends nak or bad packets, then the protocol will appear to hang. The protocol is designed to recover from occasional transmission errors or overruns, and assumes that the line and operating systems normally provide fairly reliable communication. Each file transfer requires a separate invocation of com232; a successful end-of-file packet terminates com232. The source code for com232 contains more documentation on packet formats and protocol.

There are no higher-level constructs understood by com232. It knows nothing about SCALD directories, UNIX directories, VAX-11 VMS directories, date/time, or anything other than its packets.

Installation and Checkout - SCALDsystem to VAX

Resources required:

- o DZ-11 port on VAX-11, enabled for login at 1200 or 300 baud
- o UTB port on SCALDsystem UNIX
- o ASCII terminal located near SCALDsystem (needed only for test)
- o Four cables with 25-pin D connectors (customer-supplied)
  1 male-to-male straight-through
  1 male-to-female straight-through
  1 female-to-female straight-through
  1 female-to-female null modem (swaps pins 2 and 3)
- o /u0/src/com232.vax
  (Valid-supplied source file for VAX-11 VMS)
- o /bin/cu232
  (Valid-supplied executable file on SCALDsystem UNIX)

Procedure:

1. Move the source file /u0/src/com232.vax from the S32 to the VAX via tape or existing com232 connection.
2. On the VAX system, move the source file com232.vax to com232.pas and compile. On the SCALDsystem UNIX, change the mode of the device file that is connected to the UTB port so that the device is readable and writeable by everyone. For purposes of illustration, assume that '/dev/ttya' is chosen:

```
chmod 666 /dev/ttya
```

3. Disable logins on the UTB port by changing the first character of the corresponding line in /etc/ttys to a '0':

```
02ttya
```

4. Kill any 'getty' or 'login' process running on the line; reboot the system if necessary.

5. Connect the ASCII terminal to the DZ-11 port on the VAX-11, and login to VMS to ensure that the DZ-11 port and a minimal subset of the cables are working.

6. Logout from VMS. Disconnect the cable from the ASCII terminal and connect it to the UTB port on the SCALDsystem, but change the number of null modems by 1. If the ASCII terminal works without using the null modem, insert the null modem when connecting to the UTB port. If the ASCII terminal requires the null modem, remove the null modem before connecting the cable to the SCALDsystem.

7. Login to SCALDsystem UNIX and run cu232:

```
cu232 /dev/ttya -s 1200 -t -c comfile
```

   where the "-s 1200" selects line speed of 1200 baud; use "-s 300" for 300 baud. The file "comfile" should contain the following line

```
run com232
```

   which starts the com232 program running on the VAX.

   The response should be "Connected" and the SCALDsystem terminal should act just like a VMS terminal; login to VMS.

8. To transfer a file, type

```
~%put sfile vfile
```

   to send "sfile" from the SCALDsystem S32 to VAX-11 VMS file "vfile," or type

```
~%take vfile sfile
```

   to retrieve VAX-11 VMS file "vfile" onto SCALDsystem S32 file "vfile." If the file name is the same on both systems, the shorter forms

```
~%put sfile
~%take vfile
```

   can be used.

During a file transfer, cu232 prints a dot "." each time it transfers 512 bytes to or from the SCALDsystem disk; this serves as an indication of progress. Each time cu232 sends or receives a NAK packet or an ill-formed packet, cu232 prints a sharp "#". Cu232 signals the end of the file transfer by printing a linefeed. When not transferring a file, cu232 sends all typed characters down the line and prints all characters received from the line. The only exceptions are typed lines beginning with a tilde "~", which are interpreted as possible command lines for file transfer. To send a line beginning with a tilde, type two tildes "~~".

To exit from cu232, type the two-character line:

```
~.   (tilde -  dot -  RETURN)
```

During a file transfer, both systems have echo turned off. The transfer can be aborted by typing the "quit" character (normally control-C) on the SCALDsystem S32, followed by the interrupt character (also normally control-C) on VMS. This leaves VMS echo off. To restore VMS echo:

```
set term/echo
```

In case cu232 dies or is killed from another SCALDsystem terminal, the UNIX terminal modes need to be reset. Type

```
stty echo -raw -nl
```

and terminate the command with a LINEFEED rather than a RETURN.

Installation and Checkout - SCALDsystem to SCALDsystem

Resources required:

- UTB port on each SCALDsystem
- One male-to-male straight-through RS232C cable with 25-pin D connectors.
- One female-to-female null modem (swaps pins 2 and 3)
- /bin/cu232 and /bin/com232
  (Valid-supplied executable files on SCALDsystem UNIX)

Procedure:

1. Connect male-to-male cable to null modem connector.

2. Connect the cable to an open port on each S32. Designate one machine the "host" (i.e., the machine that establishes contact).

3. Edit the /etc/ttys file on the host machine. Change the entry for the port used to make the RS232 connection read

   ```
   02ttyX
   ```

   where ttyX is the outgoing port.

4. Edit the /etc/ttys file on the other (remote) machine and change the entry for the port used to read

   ```
   13ttyQ
   ```

   where ttyQ is the incoming port.

5. On both S32s, enter the command

   ```
   kill -1 1
   ```

   to cause the /etc/ttys file to be reread.

6. From the host machine, enter the command:

   ```
   cu232 /dev/ttyX -t -s 1200
   ```

7. To transfer a file, type

   ```
   ~%put hfile rfile
   ```

   to send "hfile" from the host S32 to the remote S32, or type

   ```
   ~%take rfile hfile
   ```

   to retrieve "rfile" from the remote S32 to the host S32 file "hfile."

Note that other operations are the same as the VAX version previously described.

RELEASE 7.0 ETHERNET

The 7.0 release includes upgrade of the UNIX operating system from System III to 4.1cBSD. The largest changes are in the areas of file system and Ethernet.

There are basically two Ethernet facilities in the Release 7.0. One is the Extended File System (EFS) which is the upgraded version of the old EFS in Release 6.0. The other one is the new network commands which are the implementation of some higher layer protocols on top of the TCP/IP and Ethernet.

CHANGES IN THE EXTENDED FILE SYSTEM

The SCALDSystem UNIX Extended File System (EFS) has been totally redesigned by VALID in Release 7.0. It is quite similar to the 6.0 Release from the user standpoint. The main differences are administrative rather than functional. The new connection manager is used to manipulate a kernel-level host table. The user should read the appropriate pages in the "SCALDsystem UNIX Programmer's Manual" for more information. Conn(8V) is run from rc(8) to initialize the connection of the local node to the site network.

The old hostable, "/etc/net/hostable" is no longer used and should be purged; the "/net" special node should remain the same.

The 7.0 Release supports open, close, read, and write system calls on remote files and directories (i.e., these system calls can be used to access file systems on a remote node) or the user can issue local commands to access remote files and directories over the network (e.g., ls, cat, page, more, vi, ex, head, tail, etc.). Note that commands that involve changing directories cannot be executed over the network (e.g., find(1) cannot be used since it executes a "chdir"). The commands rm(1) and mv(1) also are not supported on remote files, and remote device access is not permitted.

NEW NETWORK COMMANDS FROM BERKELEY UNIX BSD 4.1c

Except for the EFS in release 7.0, there are several new network commands from BSD 4.1c. These commands need to be invoked by the user explicitly. Rsh is one of these commands. Since it is differently implemented, there is a slight change from the user's point of view.

The new commands are:

Commands	Daemon Process
rsh(1)	rshd(8)
ruptime(1)	rwhod(8)
rwho(1)	rwhod(8)
rlogin(1)	rlogind(8)

For more details about these commands, see the "SCALDsystem UNIX Programmer's Manual" (900-00038).

- rsh -- the rsh(1) command is functionally the same as the one in the Release 6.0 except two options are added : -l and -n. You can remotely execute commands in pipelines quite naturally. Beware of shell metacharacters such as ">" and "*" in the remote command: they will be evaluated locally unless you protect them by quoting them with the " character.

- ruptime -- the ruptime(1) command prints a status line for each node on the local network. It shows host name, the number of users logged in at each host, the load average and the amount of time the node has been up (or down).

- rwho -- the rwho(1) command shows the users who are currently logged in at each node in the network within an hour. The option -a prints all users regardless of how long they have been logged in on the system.

- rlogin -- the rlogin(1) command is used to log into any other node on the network. If you do not have an account on the remote node, you will be prompted for a login name.

NETWORK MANAGEMENT SYSTEM

To ensure minimal security for the extended file system, the system administrator must ensure that the /etc/passwd and /etc/group files at each node in the site network are consistent (mkusr(8V) may create duplicate user id numbers). If, as a result of setting up the new passwd and group files, a previous user's file tree no longer shows ownership of his or her files (i.e., the user id numbers are different), chuid(8V) must be run on each node that the user has files.

/etc/chuid

Chuid takes the old and new password files and creates a map between the two. It then traverses the entire directory tree to make the necessary change for each single file. For more details, see chuid(8V).

After installation, new users are added to the system only by /etc/mkusr.

/etc/mkusr

/etc/mkusr adds a new user to a node and creates the login directory and setup files for the new user in the node. Note that mkusr must be run on each node on which the new user is to access. See mkusr(8V) for more details.

/bin/conn

Conn sends the ioctl commands to the connection manager. The ioctl commands include enable, disable, shutdown, listen, nolisten, and show. The show command displays the status of all nodes in the network from the incore host table. See conn(8V) for more details.

BSD 4.2 NETWORK MANAGEMENT

The bsd 4.2 network has the /etc/hosts* and /.rhosts database maintained by the system dynamically.

/etc/chkhosts

/etc/chkhosts initializes and updates the /etc/hosts* and /.rhosts files and is run from /etc/rc and the cron daemon. See chkhosts(8V) for more details.

NETWORK RELATED FILE DATABASE

The following files are new and only apply to the BSD 4.2 Network Facilities. Note that these files are maintained by the system and should not be edited or changed by the user.

```
/etc/hosts
/etc/networks
/etc/hosts.local
/etc/hosts.equiv
/etc/protocols
/etc/services
```

The /etc/net/hostable, /tmp/netlog, and /tmp/rshlog files in the old database are obsolete.

/etc/hosts

The hosts file appears as:

```
Syntax:
    network-number.ethernet-board-address   node-name       alias
Example:
    0.0x7974        mini            myname
    0.0x7729        puff            puff
    0.0x4734        robert          bob
    0.0x4714        darth           darth
```

The first field is the inter-net address and the second field is the formal node name; subsequent fields are aliases. For the local node, an alias is "myname"; for the remote nodes, aliases can be any name.

The /etc/hosts file is automatically maintained by /etc/chkhosts(8V). At installation and every boot time, this file is initialized according to the incore host table and also is updated from the cron(8) daemon every thirty minutes while the system is running.

/etc/hosts.equiv

The /etc/hosts.equiv file contains a list of "equivalent" hosts "rsh" and "rlogin" (see rshd(8) for details).

/etc/hosts.local

The /etc/hosts.local file contains the following line:

```
Syntax:
    network-number.ethernet-board-address   node-name       localnet
Example:
    0.0x7974        mini            localnet
```

The first two fields are the same as in the /etc/hosts file for the local node; the third field is the string "localnet." This file is automatically maintained by /etc/chkhosts.

/etc/networks

The /etc/networks file contains one line and can be found from release tape. There is no need to change this file.

```
Format:
    localnet        0       localnet
```

/etc/protocols and /etc/services

These files are just the protocol and service name data base and are provided in the release tape. For details, refer to protocols(5) and services(5) in the "SCALDsystem UNIX Programmer's Manual."

Pty Device Files

Rlogin needs a device-pair called a pseudo terminal (a pseudo terminal is a pair of character devices, a master device and a slave device, that emulate a terminal).

Adding Pseudo Terminals in /dev Directory

You should have several pairs of the following files in the /dev directory.

```
Format:
    /dev/pty[p-r][0-9a-f]    master pseudo terminals
    /dev/tty[p-r][0-9a-f]    slave pseudo terminals
```

Updating the /etc/ttys File

Adding the following pair of entries into the /dev/ttys file for each pseudo terminal:

```
Format:
    02ptyp0
    02ttyp0
```

Updating the /etc/ttytype File

Adding the following pair of entries into the /etc/ttytype file for each pseudo terminal.

```
Format:
    network ptyp0
    network ttyp0
```

There are ten pseudo terminals currently assigned in the release tape. If more pseudo terminals are needed to allow more users to remotely login to the same node simultaneously, add additional pseudo terminals as described above.

/etc/rc File

For the network facilities, the following commands are executed by /etc/rc:

```
/bin/conn listen --- turn on connection manager listen mode.
/bin/conn enable hostName --- check the naming conflict and
                              enable the EFS network connection.
/etc/chkhosts -clean --- initialize /etc/hosts host table.
/etc/chkhosts  --- maintain /etc/hosts host table.
/etc/rlogind   --- start remote login daemon.
/etc/rwhod     --- start remote who daemon.
/etc/rshd      --- start remote shell daemon.
```

RELEASE 7.25 ETHERNET

NEW COMMANDS

The following new network commands have been added:

- **shownet**(1V) -- displays the currently reachable (and unreachable) Valid nodes on the local Ethernet (abbreviated version of the "conn show" command).

- **ether**(8V) -- allows user to control and observe 3Com Ethernet driver and controller.

Man pages for the above commands have been included in the "SCALDsystem UNIX Programmer's Manual" (900-00038).

RESTOR UTILITY

With release 7.25, a new "-z" option has been added to the restor(8) utility to allow users to specify the file system on which restor builds its temporary work file for directory extraction (restor normally builds its work file on the "/tmp" file system). If the /tmp file system does not have sufficient space for the restor operation, the -z option can be used to specify an alternate file system. For example, the following command causes restor to build its work file on the "/u0" file system:

```
restor -xz u0
```

If "-z" is omitted or if a file system name is not specified in the command line, the /tmp file system is used by default. Note that the description of the "-z" option is **not** described on the restor(8) man page in the "SCALDsystem UNIX Programmer's Manual."

13.0 OPERATING SYSTEM

The 7.0 release of the operating system is a new implementation for SCALDsystem that is based on UC Berkeley's 4.1c release of UNIX (the 6.0 release of the operating system was based on a mixture of Bell System III and Version 7 UNIX with specific Berkeley enhancements). The 7.0 release of the operating system implements the Berkeley file system and demand-paged virtual memory.

The virtual memory support uses 4096-byte pages that are swapped out to disk to allow processes that require more physical memory than is resident on the system to be run. The maximum process size is limited by the combination of the processor's address range, the hardware and operating system memory management implementation, and the available swap space. The Motorola 68010 processor has a virtual address range of 16 megabytes. Valid reserves the upper two megabytes of virtual memory which limits the virtual process size to 14 megabytes. The standard swap space of 12 megabytes further limits the maximum virtual process size. In practice, a process can grow to approximately 9 megabytes before the available swap space is consumed.

The 7.0 release of the operating system also realizes improved performance and reliability due to the implementation of the Berkeley file system. This new file system increases the disk access speed by using larger blocks. Note that the new file system can cause a problem when converting from 6.0 to 7.0 on systems with nearly full file systems since the average file can require more disk space due to the increased block size.

The complete set of the 4.1c BSD utilities is not implemented. A number of Valid-specific utilities are provided as well as support for the graphic system and peripheral interface board.

RELEASE 7.25 ENHANCEMENTS

The following enhancements have been included in the 7.25 release:

- Larger amount of physical memory recognized (12 megabytes)
- Various virtual memory paging problems corrected
- Support for Fujitsu Eagle SMD disk drives
- Improved tape reliability

14.0 HARDWARE CHANGES

In order to run release 7.0 of the SCALDsystem software, the CPU board must be upgraded by:

1) replacing the 68000 with a 68010
2) replacing the 1Kx4 map RAMs with 4Kx4 RAMs

These hardware changes allow the UNIX operating system to implement demand-page virtual memory (see section 13.0, "Operating System").

VIDEO GRAPHICS BOARD

RELEASE 7.0 FIRMWARE CHANGES

Modifications to the Video Graphics Board (VGB) firmware with release 7.0 have changed the patterned line display. Specifically, the "draw line" call (for Display Manager) was modified so that if the display mode is not valid, it is interpreted as a line style to be drawn in replace mode. The 7.0 Display Manager understands a fifth parameter that defines a line style (see the draw vector escape sequence in the "Valid Escape Sequence Extensions" section in Chapter 2 of the reference manual).

RELEASE 7.25 FIRMWARE CHANGES

The 7.25 release of the Video Graphics Board firmware (vgin2) has added several features and enhancements; the new firmware requires the 7.25 kernel, loader, and display manager.

- Full Screen Windows -- the new vgin2 firmware supports full screen windows.

- Two-Segment Display Manager -- the vgin2 firmware supports the new 2-segment display manager (dmgr2.ld).

- Color Workstations -- Vgin2 runs on both the VGB and the new CGB (color graphics board).

The new vgin2 firmware increases the speed of the reverse video display by accepting an attribute flag instead of requiring display manager to make a second call to reverse the character. The new vgin2 also supports the underline attribute.

PERIPHERAL INTERFACE BOARD

The 7.25 release of the Peripheral Interface Board (PIB) firmware or "pib code" has been rewritten to provide an effective single-board interface to both the VAX and IBM hosts. Problems related to hardcopy requests that caused the system to hang when the Versatec showed an "offline" status (e.g., power off, cable disconnected, out of paper, paused, etc.) and the dropping of the RSCS link while performing hardcopy were corrected. In addition, printing operations that dropped characters on the Versatec now print correctly.

15.0 PASCAL COMPILER

RELEASE 7.0 OF THE PASCAL COMPILER

This section describes the optional Pascal compiler available with the 7.0 release of the SCALDsystem software. The programs are installed in /u0/pascal.

Documentation Note

Valid obtains its Pascal compiler from Silicon Valley Software, Inc. (SVS) and accordingly, the Pascal documentation also is developed by SCS. Currently, an SVS Pascal manual and an SVS Debugger manual are supplied from Valid with systems that include the Pascal compiler; if this documentation is not included, contact Valid for copies.

Shared Code

The object files required for compilation and linking of a SCALD applications program developed by a customer (e.g., to make a SCALD Simulator linked with user-coded primitives) are in /u0/pascal/shared. See the chapters on the Logic Simulator and DIAL in the SCALDsystem Reference Manual to understand how to compile and link with these files.

Pascal External References

The Pascal compiler now generates code that can be linked with C routines without an assembly-coded "wrapper." You must declare these procedures as "cexternal." The parameter list should be reversed from what it is in the C code, and the name you give in Pascal for the routine must be exactly what it is in the C code. These names, unlike most SVS Pascal procedure names, are case-sensitive and must be unique in the first eight characters.

You should not try to link code generated from the old compiler with code generated with this compiler -- the external references will not match. You must recompile everything with the new compiler.

New Debugger

There is now a symbolic debugger for SVS Pascal. For details, see the SVS Debugger Manual.

Rangechk

There is a program in /u0/pascal/shared called "rangechk" that simplifies the range checking in code generated by the SVS compiler and may speed up your program by as much as 30% (the program does NOT remove the range checking; it leaves a semantically equivalent program). To use the program, you must link /u0/pascal/shared/patch.o with your program using "cc" or "ld," and then apply "rangechk" to it -- so that the final stage of compilation and linking looks like this:

```
cc -o foo foo.o /u0/pascal/shared/wraplib.o /u0/pascal/shared/patch.o
/u0/pascal/shared/rangechk foo
```

if your program is called "foo.pas."

Revision Dates

The SVS Pascal programs should all be listed as version 2.1. Their dates are as follows:

- pascal -- 2-Feb-84
- code -- 2-Feb-84
- ulinker (Pascal obj file linker) -- 2-Feb-84
- dbg (debugger) -- 2-Feb-84

ADDENDUM

to the

SOFTWARE CHANGES INFORMATION

SOFTWARE RELEASE 7.25.8

24 June 1985

TAPE CONTENTS

The executable files on the 7.25.8 tape are:

- **/u0/editor/lib/ged.ex**
- **/usr/lib/hpf**
- **/u0/scald/packager/packager**
- **/lib/vgin2**
- **/u0/scald/dial/dial.crf**
- **/u0/scald/dial/dial.obj**
- **/u0/scald/dial/dialint.obj**
- **/u0/scald/dial/userunit.obj**
- **/usr/lib/rscs/bisync**
- **/u0/editor/lib/hpfilter.pas**
- **/u0/editor/lib/hpfilter**
- **/usr/bin/simulate**

GRAPHICS EDITOR

The GED program contains the following changes:

- Plotting with **hpr** has been fixed to correctly plot bodies containing arcs.
- The **sec**tion command has been changed to ensure that all pins are annotated, not just the first thirty.
- Arguments for a UNIX command issued from within GED no longer are passed on as lower-case letters when the input is in upper-case (e.g. **ls -R**).
- Typing **set le** designates left-justified text.
- Type-ahead works even when a softkey is used for the main portion of a command.
- Using **set** command without any arguments clears the screen; it can also be used in startup.ged.

- It is possible to include move and copy commands in scripts.

- In instances when a user attempts to **add** a part that does not exist, the system checks to see if the part name begins with a keyword which is then executed. For instance, typing **add LS00** <pt> and **add LS04** <pt> adds two bodies correctly.

- Color information is now written into body drawings.

- Rotated text in a body that has been rotated 270 degrees no longer causes a crash.

- The **led** and **sim** commands can no longer be called from startup.ged or from a script.

- Path properties now appear closer to the referenced gate after the **auto path** command is used.

HARDCOPY FOR THE GRAPHICS EDITOR

Hardcopy procedures utilizing **vgb** and **hpr** have been improved.

- The **vgb** mode can now be used for 22", 36" and 42" Versatec plotters as well as 11" plotters.

- The **ha**rdcopy command can be included within a script.

- It is possible to plot a drawing that is not currently being edited.

- Drawing names and wild cards can now be used for **ha**rdcopy. Drawings not in the current directory must be specified according to their SCALD directory.

- If a user requests a copy of a drawing whose UNIX directory is empty or non-existent, that request is ignored.

- Property names now appear on plots of old drawings using the **hpr** mode.

- Rotated bodies having bubbles are printed correctly in **hpr** mode.

- The local plot mode for **vgb** now spools plots through **Vpr** (**vpl**) instead of using PIB to send plots directly to the Versatec.

- Once a hardcopy is made, the **hpf** header files are deleted.

- It is now possible to use **ha** ***scale_factor*** to plot the current drawing.

- The **vgb** mode has improved algorithms for computing scaling and rotation and for making efficient use of paper.

- The presence of random junk files in a drawing directory does not cause **vgb** to crash while plotting.

- Plots of simulator waveforms have text and lines scaled correctly when **hpr** is used.

- Text is displayed correctly in hardcopies made with **vgb** and in screen displays following **win**dow **f**it instructions.

- An **hpr** plotting fault which caused a memory fault error message has been fixed.

- The hpfilter.pas file ensures the correct placement of right justified text by eliminating a space character at the end of each string. This file also ensures that patterned lines are plotted correctly.

PACKAGER

The Packager now has fixes for the following problems:

- The assignment of pins-to-nets is now correct in a design having vectored pins and using LEFT_TO_RIGHT bit_ordering.

- A design containing REPLICATE bodies can no longer cause difficulties with Packager operations.

- The system generates an error message whenever a user adds a HAS_FIXED_SIZE property that has a value different from the SIZE property value for the body. Likewise, an error message is generated whenever a property value for HAS_FIXED_SIZE is greater than the number of sections available on a physical package.

- A problem that can cause an infinite loop in the packager has been fixed.

- An erroneous message has been deleted. The message formerly read "ASSERT 37: Wrong Node Count during Net Evaluation."

The Packager has two new directives:

- The directive DOCUMENT_ERRORS prints revised documentation for error messages at the end of the listing file PSTLST. This directive is on by default. The error messages can be suppressed by entering **DOCUMENT_ERRORS OFF;** in the directives file.

- The directive USE_PIN_GROUP is useful in instances when a library has no PIN_GROUP property in the pin description and the user wants to do a pinswap. Formerly, the Packager only swapped pins with the same PIN_GROUP property value. This directive enables the user to tell the Packager to ignore the absence of this value by typing in

  USE_PIN_GROUP OFF;

  in the directives file. The directive is on by default.

The anomalies report for the Packager has been revised to include a description of two problems that have been detected: a) nets consisting only of bidi-nodes, and b) pin swapping utilizing feedback_netlist. See the accompanying anomalies document for additional information.

DIAL

The DIAL program now incorporates the following changes:

- The backannotation files DIALPRTB, DIALSIGB, DIALSTAT and DIALBACK are no longer output by default.

- The userunit.pas file has been changed to fix the reference to dialint.obj.

- The capability of the system to make physical part names equivalent to location properties in logical DIAL has been restored.

- Formerly there were two error messages that referenced the same problem. Error #137 retains its meaning of an absence of pins on a logical part, but error #80 has been changed to indicate a parse stack overflow.

- There is a new routine called REASSIGN_SINGLE_PHYSICAL_NET that affects a single physical net, as opposed to REASSIGN_PHYSICAL_NET_NAMES, which reassigns all the physical nets in the physical net table. This function renames a specified net with a specified name. The syntax of the function is: **function reassign_single_physical_net *(net: net_ptr; name: string): boolean;***. It returns TRUE if the net was renamed, and FALSE if the net being reassigned does not exist in the physical net table or if the new net already exists in the physical net table. A net called 'ONE' can be renamed 'A1', for example. The user then can call this function with the net pointer pointing to a physical net called 'ONE' and use the 'A1' string as the name.

RSCS

RSCS has been changed:

- All uses of **vmcopy** for transferrals between a SCALDsystem and an IBM host operate properly regardless of the size of a file. Formerly files with a size that was a multiple of 25 or 24 eighty-character records required a change in the header.

7.25.8 SOFTWARE ANOMALIES

24 June 1985

7.25 SOFTWARE ANOMALIES

The following anomalies have been identified in the 7.25.8 release of the SCALD-system software. The anomalies are grouped according to program and, where possible, include "workarounds."

SCREEN/HARDCOPY DISPLAY OF DATA

Problem: Deleting a window that still has a process in it, particularly a background process, causes problems with the continuing processes and with subsequently-created windows.

Workaround: Create a small window for putting a process in the background and leave that window on the screen until the process is complete. If it is not possible to leave a window on the screen, the UNIX **at** command should be used rather than starting a background job. The preferred way for starting a background process is from a maintenance console or UNIX terminal (i.e. a terminal that is not a cluster workstation).

Problem: For the reasons listed above, a user has problems restarting RSCS.

Workaround: Always restart RSCS from a maintenance console or UNIX terminal, not a window. If the system does not have a console, use the **at** command.

Note: The vgb mode is the default hardcopy mode in GED for monochrome workstations.

Problem: The emulation of the visual editor **vi** scrolls one line at a time during moves through a file.

Workaround: Within **vi** it is possible to emulate jump scrolling by using **z** followed by a carriage return (to make the current cursor position the first line displayed on the screen), **z.** (to make it the middle line) and **z-** (to make it the bottom line) commands. Within **vi** it is also possible to go to any line by typing in a line number followed by a **G** (not g). For instance, typing **100G** takes one halfway through a two hundred line file, and control-G while within **vi** shows the line number of the current cursor position. The **vi** command '' returns the user to the previous cursor position. To look at an existing file, the **more** command allows one to view the file without delay for scrolling; once **more** goes to the desired location, it is possible to enter **vi** by typing **v**. Note that **more** has a help facility that is available from within **more** by typing the letter **h**.

GRAPHICS EDITOR

Problem: Copying a group and concatenating the copy to the original can modify the original group.

Workaround: Bear in mind that **co**py works on vertices, not objects.

Note: Using **set capslock_on** passes upper-case letters to the Simulator and to the UNIX operating system.

Explanation: Do not use **set capslock_on**. If one does use **set capslock_on**, all UNIX commands need to be issued from a separate window and all script files for the Simulator need to have upper-case letters.

Problem: Deleting or moving bodies leaves holes in the grid, especially when **grid dots** is used. Also, a hole can appear in a body that has been moved off the grid. The problem is visual only, not functional.

Workaround: Refresh the screen with the **win**dow; command.

Note: GED maps SCALD directory file names to lower case. In the event that one uses mixed case letters for directory file names, an error message appears. It says that GED cannot open a file name with the name in lower-case letters and also says that there is no SCALD directory by that name (in upper-case letters).

Workaround: Do not create a file with upper-case letters in the pathname. Be especially cautious about using upper-case letters in host names. For example, the system Ferd should be renamed /net/ferd in /etc/configure.

Note: The current algorithm for naming unnamed nets can cause a problem if a part is deleted from an unnamed net. If a part that was used to generate the unnamed signal name is deleted, then GED changes the name of the net. The renaming is local, and only the altered net is renamed, not the entire database.

Note that these net names are part of the database for physical design systems such as Scicards. For further information, see the relevant section of the Software Changes Information document.

Note: If a user draws a large circle, uses the **win**dow; command for a section on the upper right hand side of the screen, and then moves the circle around, the pixels in the GED screen, including the menu, are turned off. The difficulty can wipe out other windows.

Workaround: This procedure should not be attempted.

Problem: Placing a body outside the borders of the grid may crash GED.

Workaround: Turn on the grid to see the extent of the GED screen.

Problem: Maximum zoom in the lower left corner may cause GED to crash.

Problem: Using the **wire** command to create a 4-way tie may later cause the **move** command to treat the wires as unconnected segments.

Workaround: Put a dot at the center of a 4-way tie.

Problem: Overlapping wires with the **wire** command in a horizontal position, followed by **move** commands in the orthogonal mode, may cause a separation of segments.

Note: The vgb mode is the default hardcopy mode in GED for monochromatic systems.

Note: There is no way to turn off the display of the number of bodies, properties, etc. that appear in the upper left corner of the screen while executing a group command.

Note: In rare instances, the asterisk (*) lags behind by one string while using control-X in the **change** command.

Workaround: Refresh the window by issuing the **window;** command.

Problem: GED does excessive canonicalization during an edit, thereby slowing the system down.

Note: By default, the Graphics Editor provides a 0.6 text size for backannotated pin numbers in order to make a distinction between numbers that pertain to pins and other numbers/text on the drawing.

Workaround: To change the size to 0.8, use the **find $pn** command, followed by **display 0.8** group_name.

Note: A body with a pin at the origin cannot be rotated or changed into different versions.

Workaround: Do not place a pin at the coordinates (0,0).

Problem: The update package for the 7.25 Reference manual incorrectly lists the pathname for /u0/scald/section/section as /usr/bin/section. The correct pathname is /u0/scald/section/section.

Note: GED cannot be started with a remote login, since GED has to run on a 68010 connected to a local Video Graphics Board.

Problem: While running the Simulator under GED, it is possible to zoom out the GED drawing to the smallest possible depiction. In this state, the user then cannot pick things in the Simulator window with the puck.

Workaround: Zoom in on the GED drawing.

Note: The user cannot abort the **dir**ectory command when the output is more than one screenful.

Workaround: See instructions in the manual for the use of metacharacters in combination with the **dir**ectory command.

Problem: A beta version of 7.25 GED produced corrupted versions of body connectivity and ASCII files.

Workaround: Any drawings produced with that version can be read into a later release of GED and written back out again.

LIBRARIES

Problem: BACKANNOTATION causes the pin numbers on a drawing to be different from the notes on the corresponding body. The numbers do not affect connectivity file formats. Section swaps cause a wider divergence in numbering.

Workaround: The problem can be disregarded without impairing functions. If the numbering is confusing, it is possible to create a body without subscripts on the part.

COMPILER

Problem: Due to Pascal implementation differences, a compilation that is successful on a Valid S32 may not compile on an IBM 370 VM machine. The difficulty occurs when the compilation surpasses the IBM virtual memory capacity while processing drawings. When the memory is exceeded, neither the CMPEXP.DAT or CMPSYN.DAT files are created on the 370 VM machine.

Workaround: Use **sepcomp** and **seplink** whenever virtual memory is exceeded and one encounters a limit on the maximum drawing size that can be used on an IBM 370 VM machine.

Problem: Low asserted signals for the Compiler, Timing Verifier, and the Simulator are indicated by a leading "_" instead of a trailing "*" in the listing files.

Explanation: The Compiler, Timing Verifier, and the Simulator currently report the polarity of a signal, not its assertion. The Compiler interprets the assertion and passes both -A and A* on to other tools as -A; similarly, A and -A* are passed on as A.

TIMING VERIFIER

Problem: The Timing Verifier, at present, cannot interpret user-defined syntax in the case file.

Workaround: Consult explanation above in the Compiler section. This problem is temporary and will be remedied in 8.0.

Problem: Using the 40105B part from the CMOS library causes the 7.25 Verifier to crash.

Workaround: This problem exists as of April, 1985, and it will be fixed shortly thereafter. If in doubt, call the Hotline for the latest status before using this part.

PACKAGER

Problem: If a new section is allocated to a package using the LOCATION property, the Packager may reassign the sections already assigned to this part rather than leaving them as they were.

Workaround: Rather than using the LOCATION property to assign a new section, reassign sections manually for sections whose location and section are important.

Problem: If a comment is typed on two lines during entry in the physical part table, a syntax error results.

Workaround: Confine all comments for this table to one line.

Problem: The use of tabs within comment lines for the phsycial part table results in syntax errors.

Workaround: Do not use tabs in the comments for the physical part table.

Problem: If a net contains only bidirectional nodes, the net-input-loading is reported as zero.

Problem: When the file feedback_netlist is used for pin swapping, the Packager does not report an error for the swapping of two pins with different PIN_GROUP property values.

Workaround: Check to see that pins to be swapped by feedback have matching PIN_GROUP property values in the library. After packaging, check the output files to be certain that the pinswap was done properly.

SIMULATOR

Note: New enhancements to the Logic Simulator are not documented in the Reference Manual Update Package.

Workaround: Refer to "Release 7.5 of the Logic Simulator" in the "Software Changes Information" document for descriptions of new enhancements.

Problem: On an IBM host, the use of certain script files may cause error messages indicating that the logical record length has been exceeded.

Workaround: The problem does not affect simulator operation; it may cause problems in extracting data from output files.

Note: The "--MORE---" feature for simulator output to the screen causes the user to be unable to input additional commands to the simulator until the scrolling process is completed.

Workaround: Enter carriage returns until all scrolling is completed before issuing additional commands.

Problem: It is possible for the Simulator to go into an infinite loop while evaluating a circuit containing PASS TRANSISTOR or RES (resistor) primitives. In addition, it is not possible to deposit to a bidirectional net.

Workaround: Insert a buffer into the circuit at the input(s) to the RES or TRANSISTOR primitive. Values can then be deposited to the input of the buffer rather than directly to the RES or TRANSISTOR.

Problem: While using Library Format 4, a user cannot open signals containing a minus sign "-" in the name.

Workaround: Synonym the signal name to another name which does not contain a "-".

Problem: The user should not attempt the **trace_a**ll procedure.

INTERFACES

Problem: The Scicards interface cannot handle a Scicards LATCH4 function unless precautions are taken. Otherwise, the result is an error "209): Scicards pin not found on the SCALD part." This condition causes the program to end.

Workaround: Do not attach the SCI_PART property to library parts which are modelled by LATCH4 functions in the SCICARDS library. Omitting this property ensures that an entry will be generated in the scilib.dat file for that part.

COMMUNICATIONS

Problem: The **rwho** and **ruptime** commands do not work properly, and **/etc/rwhod** uses an excessive amount of system resources.

Workaround: In 7.25 **rwhod** is not started by default in **/etc/rc**, but the user can manually start it if desired by typing **/etc/rwhod** from a maintenance console or UNIX terminal. This command should not be issued from a window; if it is issued from a window, use the **at** command as described in the first entry for the Screen/Hardcopy Display section.

Problem: Typing a control-C during a remote login sometimes generates the message "Lost connection."

Workaround: Avoid using multiple control-C commands in rapid succession during **rlogin**. The first control-C has the desired effect, but the second can lose the connection.

Problem: When **rsh** takes a long time to execute and the user issues a control-C, a forked process is left on the remote machine and the local shell has an unusual stty state.

Workaround: Type control-D before typing control-C to terminate the remote shell processes. If the control-D step is ever omitted, use **rsh ps** and **rsh kill** (with the process id number) to kill off the forked process; then use **stty** to reset the shell.

Problem: The **rsh** command gives an "address already in use" error in certain instances. For example, **simlab** may generate that message after an **rsh simlab date** command.

Workaround: Kill the **rshd** daemon on the remote system and then type **/etc/rshd** on the remote system to restart it. The **/etc/rshd** command should not be issued from a window; instead, use the console or UNIX terminal.

Note: GED cannot be started with a remote login, since GED has to run on a 68010 connected to a local Video Graphics Board.

Note: All network communications commands (**rlogin**, **rsh**, **rwho**, etc.) require both systems to have the same revision level for the kernel. For instance, version 6 software systems can only communicate with version 6, version 7 with version 7, and 7.25 with other 7.25 systems.

UNIX KERNEL

Problem: The system clock slows down relative to real time when processes are running in a heavy context switching environment. The amount of time lost depends on the system loading, particularly if an intensive program such as **drc** runs overnight.

Problem: The **tar cv /u0** command does not work if run from /u0 or a subdirectory of /u0.

Workaround: Change to the root position and then type **tar cv /u0** to archive the data. Within the /u0 directory, the command **tar /u0/*** also works.

LANGUAGE COMPILERS

Problem: A Fortran program that has been compiled on a system with software before 7.0 produces the message "illegal instructions" and dumps the core.

Workaround: Recompile the program with 7.0 or later software.

Problem: The C Compiler refuses to cast a function to type "void," thereby creating an infinite loop.

Workaround: Either delete the keyword *void* or use a typedef to equate it to int (or some other legal typename).

Problem: Limit on size for single arrays in C.

Workaround: Do not create any single arrays larger than 32K.

Problem: The comma operator in a C program argument list is erroneously evaluated.

Workaround: Avoid using this construct.

Problem: Compiling anything with !n where n is double (not long) causes the Compiler to do a core dump.

Workaround: Do not use !n. Use (0===n) instead. Cast the variable to long before using the not operator.

Problem: If units are defined in a procedure in the interface section but are not defined in a body in the implementation section, a Pascal program can be compiled without complaint, but its executed version crashes.

Workaround: Define everything in the implementation section.

Problem: The **ulinker** program creates a core dump if it is given more than 128 .obj files. The file names, in this case, are entered interactively by giving no command line arguments to /u0/fortran/ulinker.

Workaround: Put the .obj files into a file to be used as input for **ulinker**.

RSCS/VM

Problem: RSCS messages follow the log-in name through multiple windows and appear on the one that the user has most recently logged-in on, not necessarily the current window. Some messages, consequently, may not be read.

Workaround: Create a special user for the purpose of sending RSCS data. Logon as that user to use RSCS transfer. Then the messages will also go to the window using the RSCS login.

Problem: Starting RSCS in a SCALDsystem window causes that window and the one after it to display a message about continuing processes.

Workaround: Start RSCS from the maintenance console or UNIX terminal. Be certain to kill the **bisync** and **rdaemon** processes prior to restarting RSCS.

Problem: On occasion RSCS crashes and the restart program does not always work.

Workaround: First attempt to kill the **bisync** and **rdaemon** processes prior to restarting RSCS. If this does not work, do a cold boot of the entire system.

Note: RSCS does not start if there is no entry in /u0/spool/rscs with the system machine name as a directory. If the entry is omitted, the system generates the warning message "rdaemon: cannot initialize.--could not initialize MSGWORK" in errorlog.

Workaround: Make a directory in /u0/spool/rscs with the system machine name. A VM system expects upper-case letters for machine names.

Problem: The RSCS commands are not documented on the on-line manual.

Workaround: See the Valid RSCS User's Manual.

Problem: GETRDR changes all files in a CMS disk to variable length format. The problem occurs when a file with file name of * and file type of * is sent to the VM machine, which does not define * as a legal file name or file type. The EXEC uses this combination while generating a rename instruction; consequently, all files are changed.

Problem: IBM to Valid transfers of binary files corrupts some of the data; however, Valid to IBM binary transfers do not corrupt the files.

Workaround: Avoid transferring binary files. See the RSCS manual published by IBM for additional instructions (DISKDUMP) for binary file transfer from IBM to Valid machines.

Problem: The soft key assignments for PF Keys 1 through 9 are inoperable within the 3277 emulation mode. PF Keys 10 through 12 do work, and in XEDIT mode all keys function. The problem is in the simulation code on the S-32.

UNIX UTILITIES

Problem: All user .login, .profile and .cshrc files are owned by root and have the permission code set at -rw-rw-rw.

Problem: The procedure for using **restor xv** directory_name has problems with restoring selected files or directories from a multi-volume dump tape. It tries to restore files onto parent directories that do not exist and then prints the message "unable to create file."

Workaround: When **restor** asks which volume to mount, begin with volume one. This procedure takes longer, but it guarantees that the directory structure is extracted. For additional information, see Valid Technical Bulletin No. 31.

Problem: The Graphics and Layout Editors ignore the umask setting in the C-Shell.

Problem: The **ps -au** command sometimes erroneously reports users with more the 100% usage of CPU resources. The problem typically occurs during heavy system load.

Problem: The **adb** utility does not assign the correct breakpoints for two byte instructions. For example, any C program that has been compiled into a.out can then be checked by **adb**. Once within **adb**, the user can give the following instructions:

```
_start+ Ox16:b
:r
:c
:c
```

and find that segmentation is at _start+ Ox1c. Consequently, the breakpoint is set at most two byte instructions.

Problem: Using **adb** in conjunction with wraplib.o is difficult. If one compiles C and Fortran together, and then wishes to set a breakpoint at "open" (as called by C) in the resulting a.out, the command

```
open:b
```

sets a breakpoint at _open, not "open."

Problem: Rebooting in single user mode can cause a file that was being edited to come up empty after a **reboot -s** is issued.

Workaround: The **shutdown -r** command ensures that all files maintain their integrity during reboot procedures. Those who wish to use **reboot** should type **sync** before issuing the **reboot** command.

Problem: Certain tty modes, especially those used by **edt** and other screen-oriented editors, cause some characters to be dropped during editing. The problem occurs on a sporadic basis in non-cooked modes, particularly when the user specifies **setenv TERM vt100** in the C-Shell or **set terminal/device_type=vt100** in VMS.

Workaround: If there is no alternative to using vt100 emulation and the problem is severe, use **ed**, a line-oriented editor for text input.

Problem: The **dump** procedure may generate an error message about a bad unit number or block number. For example, typing

dump 0u /dev/rim0b

produces the error message

Rim 01: Bad unit or block , block 71759

and a core dump.

Workaround: If this message appears, abandon the **dump** procedure and use **tar** to backup the file system.

Problem: Using control-Z in the Bourne Shell while in the single-user mode hangs the system.

Workaround: Reboot the system.

Problem: Using **cat** to display a binary file on a design station can cause unpredictable results that require the user to reload the Video Graphics Board.

Workaround: It is helpful to use the **file** command to identify binary files before attempting to **cat** any file. However, **file** does not correctly identify the file type for core files, and a user should not **cat** a core file without using the **strings** procedure first and then using **cat** to examine the output of the **strings** program.

Note that the **ls -F** command indicates directories with a slash mark and executable files with an asterisk. Many users find it helpful to **alias ls** to **ls -F** in the .cshrc file or .profile (for those who do not use the C-Shell).

Note: The **ls -c** command operates as **ls**.

Note: The table formatter **tbl** for **nroff** is not on the system.

Note: The **uucp** utility is not supported.

Note: The disk usage command (**du**) reports only the number of blocks in the size of directories, not the number of kilobytes as stated by the documentation. The actual disk storage used may be considerably less than **du** reports.

Note: The system accounting utilities (**sa**, **accton**) are not supported.

Problem: During **walknet** procedures, the **tp** program reads only MTA0 instead of the logical device designated by the command file.

Workaround: Use **tp -f** to force **tp** to force the program to use the first-named file rather than tape as the archive.

Note: If the user specifies both the user name and the terminal number while using the **write** command, the message is sent to that terminal regardless of whether or not another user is logged in.

Workaround: Use the **who** command to find out who is logged in before sending a **write** message.

Note: The tutorial program **learn** is not on the system.

Note: The on-line manual for Section 8V UNIX commands can be called to the screen only if an upper-case V is included as part of the command name. Typing **man 8V ether** is successful, but **man ether** and **man 8v ether** results in the message "not found."

Problem: The **mkusr** and **restor** manual pages as printed in the update package are each missing one option.

Workaround: See the pages attached at the end of this document and substitute them for the corresponding pages in the update package for the UNIX manual.

Problem: If a system manager specifies a non-existent file system as a user's home directory during the **mkusr** program, the program terminates and creates an erroneous entry in /etc/passwd. For example, a single-disk system has /u0 as the only file system; **mkusr** terminates if a user then specifies /u1.

Workaround: Be certain of the UNIX file system in which the user files are to be placed. If an error is made when specifying the location of the home directory, enter "No" in response to the prompt that queries the accuracy of the user data.

Note: The /etc/mknetusr program is not supported.

Problem: A line printer cannot be set for 2400 baud with the **stty** command. The code for the line printer daemon is hard-coded for 9600 baud.

Problem: The C shell terminates whenever the user types in 'echo' followed by a name that is also a command name in acute accents ('). The machine responds with a segmentation error message and a core dump.

Workaround: When giving 'echo' instructions, use double quotes ("whoami") or grave accents (`whoami`) to describe what is to be echoed.

SCALDstar

Problem: If an object is very large relative to the scale factor, the object may disappear. That object can include the cursor.

Workaround: Zoom out enough to make the object reappear.

Workaround: Instead of getting rid of a region by folding it over itself, either delete it or use the erase command.

Problem: Dynamically dragging an even number of objects, all of which share an identical edge, across the screen causes the coinciding line to disappear. Occasionally, small bits of the ghost image are left on the screen. This description pertains only to 8.0 software for **led**.

Workaround: Redraw the screen to eliminate any bits of images. The problem is purely visual during the act of moving the objects and in no way affects the functionality of the objects once they are deposited in a new location.