



## **PERQ SYSTEM D/L**

**Reference Manual**

**November 1983**

**Copyright © 1983  
PERQ Systems Corporation  
2600 Liberty Avenue  
P.O. Box 2600  
Pittsburgh, Pennsylvania 15230**

05 Nov 83

This document is not to be reproduced in any form or transmitted in whole or in part, without the prior written authorization of PERQ Systems Corporation (the Company).

The information in this document is subject to change without notice and should not be construed as a commitment by the Company. The Company assumes no responsibility for any errors that may appear in this document.

PERQ and System D/L are trademarks of PERQ Systems Corporation.

Copyright PERQ Systems Corporation 1982, 1983.  
All rights reserved.

TABLE OF CONTENTS

|            |   |
|------------|---|
| CHAPTER 1  | INTRODUCTION                              |
| CHAPTER 2  | INSTALLATION OF PERQ SYSTEM D/L           |
| CHAPTER 3  | PERQ SYSTEM D/L HARDCOPY FACILITIES       |
| CHAPTER 4  | THE PERQ SYSTEM D/L GRAPHICAL EDITOR - DP |
| CHAPTER 5  | SCHEMATIC WIRELISTER - SL                 |
| CHAPTER 6  | DESIGNER'S ASSISTANT - DA                 |
| CHAPTER 7  | MIASMA                                    |
| CHAPTER 8  | VAX PROGRAMS                              |
| CHAPTER 9  | SCOPE                                     |
| CHAPTER 10 | PERQ SYSTEM D/L DESIGN EXAMPLE            |
| CHAPTER 11 | COMPONENT DESIGN                          |
| CHAPTER 12 | SYSTEM D/L LIBRARIES                      |

**CHAPTER 1**  
**INTRODUCTION**



CHAPTER ONE  
INTRODUCTION

|                                      |   |
|--------------------------------------|---|
| THE WORKSTATION                      | 1 |
| SIMULATION                           | 2 |
| DESIGNING THE CIRCUIT                | 2 |
| PERQ SYSTEM D/L ACTION AND DATA FLOW | 4 |



The PERQ Systems Corporation digital design system, PERQ System D/L, is an integrated set of hierarchical design tools that automate the design of digital systems (see Appendix I of this chapter). The systems can be comprised of both gate arrays and off-the-shelf TTL components. The schematic editor, simulator, and other tools facilitate a hierarchical methodology of circuit design. PERQ System D/L accommodates new symbol libraries, additional vendors, simulators, rules checkers, and distinct design styles.

### THE WORKSTATION

A minimal design station consists of a PERQ Systems Corporation PERQ with a microprogrammed processor, a high-resolution graphics display, a digitizing tablet, a megabyte of main memory, and a 24 megabyte Winchester disk. The PERQ hardware includes RS232 and GPIB interfaces to provide compatibility with many printers; you can connect hardcopy output devices to the PERQ's RS232 port or to the GPIB interface. PERQ Systems Corporation offers many peripheral options for the PERQ, for example: PERQ Laser Printer 10 (PLP-10), Ethernet (TM of Xerox Corporation) local area network, and a 300 megabyte disk drive.

Most design stations include an optional Ethernet interface to permit use of remote file and print servers as well as communication of shared design information between designers. The PERQ can communicate with the VAX (TM of Digital Equipment Corporation) by either RS232 or Ethernet for file transfer. The PERQ can also emulate a terminal through the RS232 interface.

SIMULATION

PERQ System D/L runs on the PERQ Systems Corporation PERQ computer. The computation-bound task of circuit simulation is performed on a large mainframe computer. Tegas (TM of Comsat General Integrated Systems) may be used with one or more PERQ System D/L workstations. By running the simulation on a mainframe computer, the PERQ workstation is freed to perform other interactive tasks. PERQ System D/L produces input required by the Comsat General Integrated Systems Tegas simulator which runs on a Digital Equipment Corporation (DEC) VAX computer.

DESIGNING THE CIRCUIT

The logic designer begins with the schematic editor, DP. DP is a general-purpose drawing program that supports lines, text strings, circles, arcs, and cubic splines. DP supports both normal design and hierarchical design. The designer calls parts (e.g. gates) from one of the parts libraries. Several parts libraries are supplied by PERQ Systems Corporation, and it is easy to create new libraries or to add parts to an existing one. This combination of DP and the parts libraries supports hierarchical design.

For hierarchical design, a section of the design consists of a detailed drawing and an icon. The detailed drawing is a logic circuit and the icon is a small picture that can be used in other circuits to represent the detailed drawing. A particular icon may be used in several places to reduce the size and time spent on the schematic. Detailed drawings may themselves contain other icons. This allows arbitrarily complex drawings to be represented in a simple manner.

Once the circuit is drawn, the designer uses SL to extract electrical information from the graphical schematics. SL builds hierarchical wirelists to describe the logic design. SL performs some simple checking of the design, but is restricted to those checks that can be performed on the basis of graphical information only. For example, SL verifies that all wires are connected at both ends.

The designer then uses DA, the designer's assistant, to perform more rigorous checking of the design. DA checks for electrical errors such as unused inputs and networks that have multiple drivers. Once the design passes these tests, DA produces a description of the circuit which is used as part of the input to the Tegas simulator. The other required input is test data input. Tegas is a seven-state simulator containing support for

normal and hierarchical designs. For simple simulations, the test input may be prepared by hand using the PERQ text editor. If the simulation is more complex, a test data generator program, Miasma, may be used. Miasma is capable of complex generation schemes such as grouping signals together into decimal, octal, or hexadecimal buses.

After Tegas simulation, the designer can examine the textual output from Tegas. The Scope program provides the function of a logic analyzer. Scope allows the designer to group signals into buses and to display data numerically or as waveforms.

When the circuit is fully debugged, PERQ System D/L produces the wiring lists that are used to fabricate the design. For gate-array designs, PERQ System D/L currently provides wiring lists compatible with Fujitsu Ltd., and for discrete design it provides compatibility with Algorex Corporation, a printed-circuit board manufacturer. PERQ System D/L can produce various types of documentary material: gate switching rates, power dissipation estimates, board area estimates, bills of materials, gate cell usage, cross references, and back-annotation of the schematics with physical circuit information (for example, board locations).

Because PERQ System D/L incorporates support for other vendors' products, separate licenses are necessary for the following products:

DEC VAX with VMS operating system and DEC Pascal support

3COM (TM) VAX Ethernet interface board and driver software

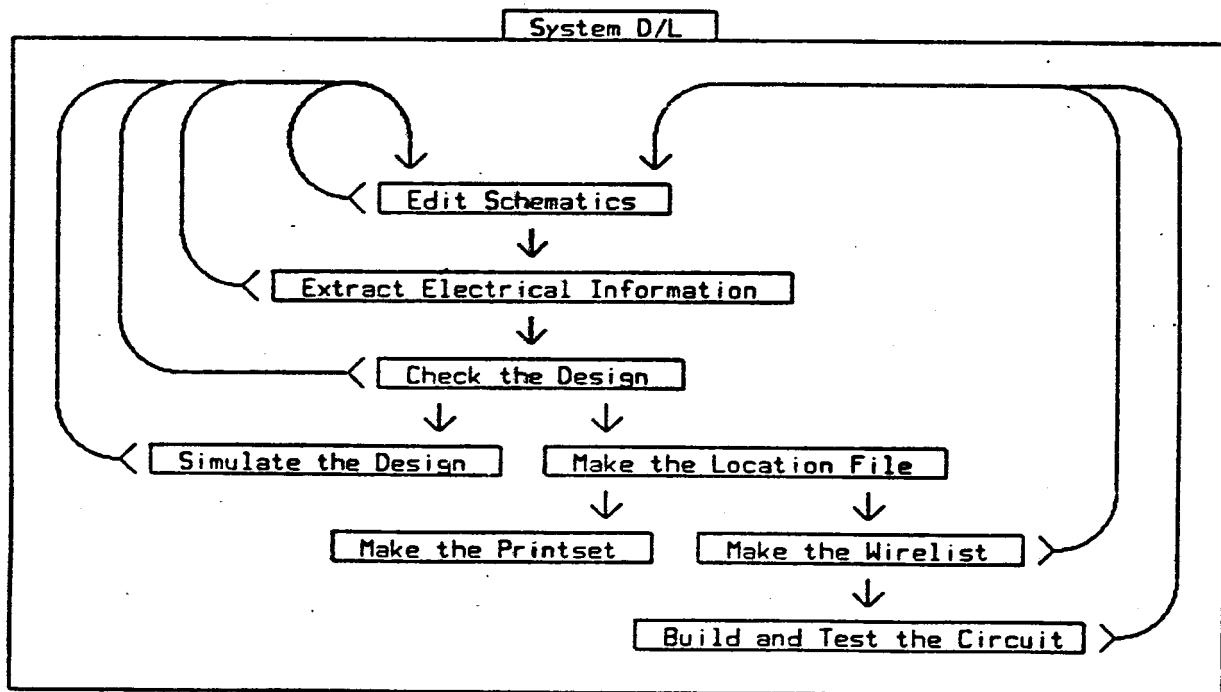
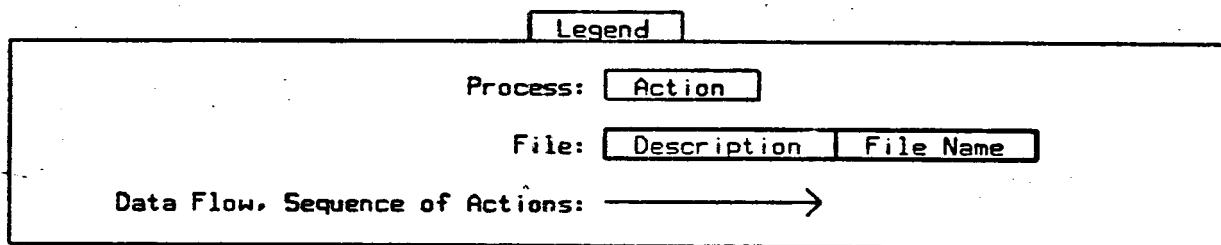
COMSAT General Integrated Systems Tegas simulator

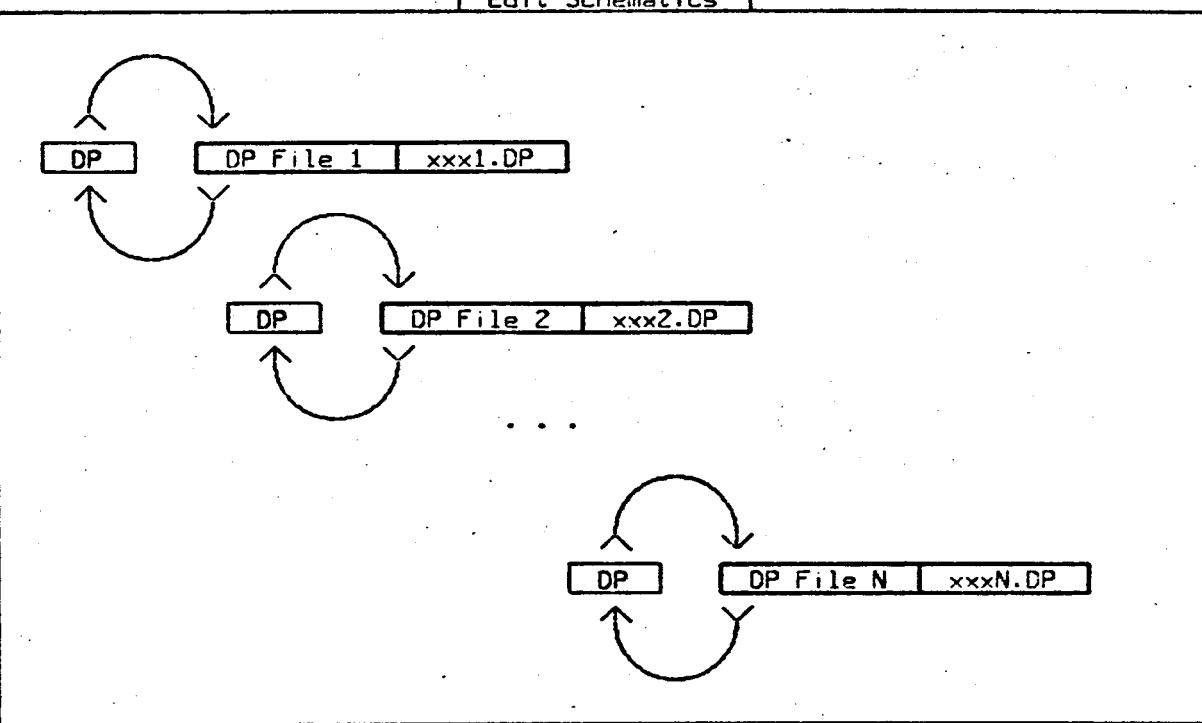
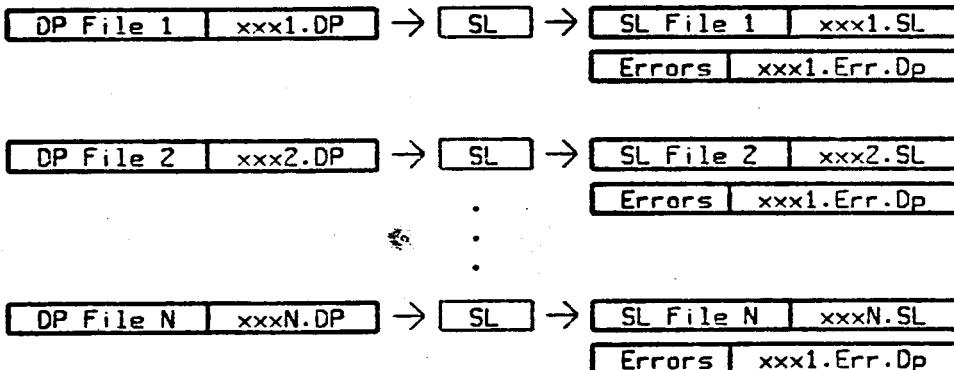
Fujitsu gate array design library

PERQ System D/L includes software to produce listings of text files and printed hardcopy of the schematics. The system can also reproduce the PERQ screen on a laser printer.

The following chapters discuss PERQ System D/L in detail.

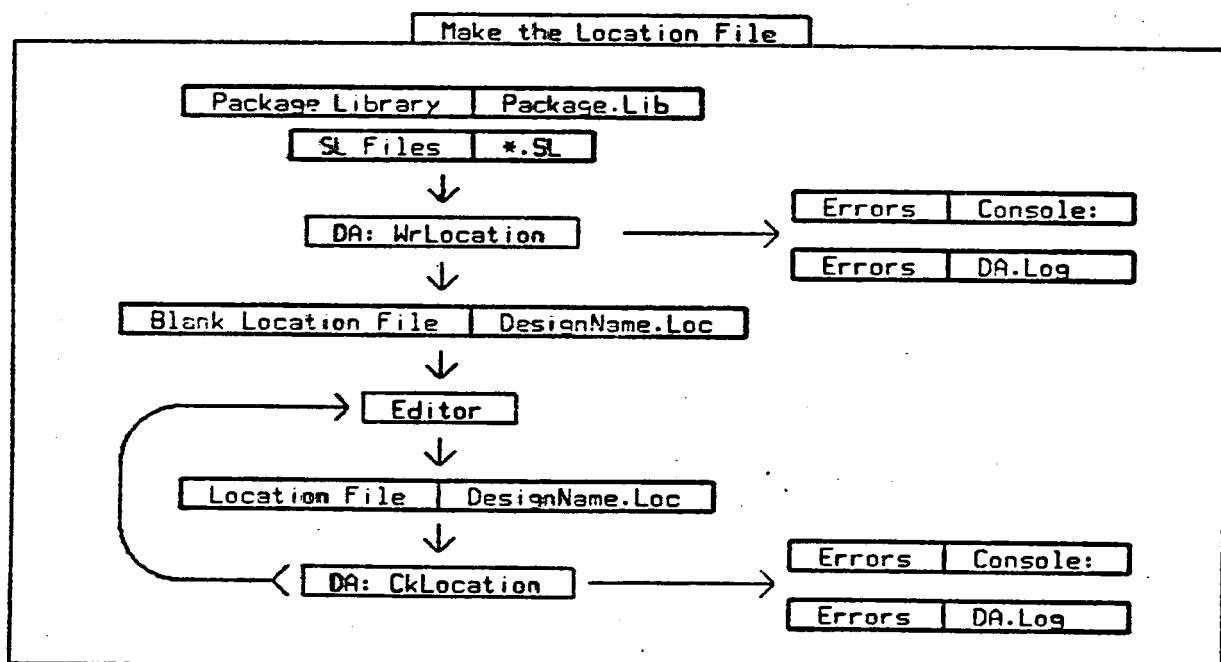
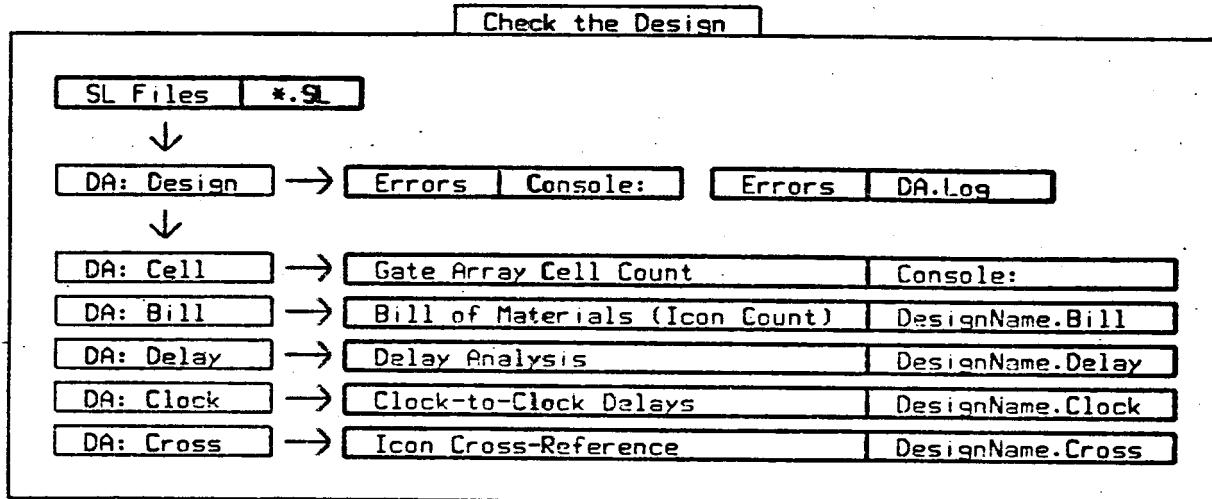
## PERQ SYSTEM D/L ACTION AND DATA FLOW

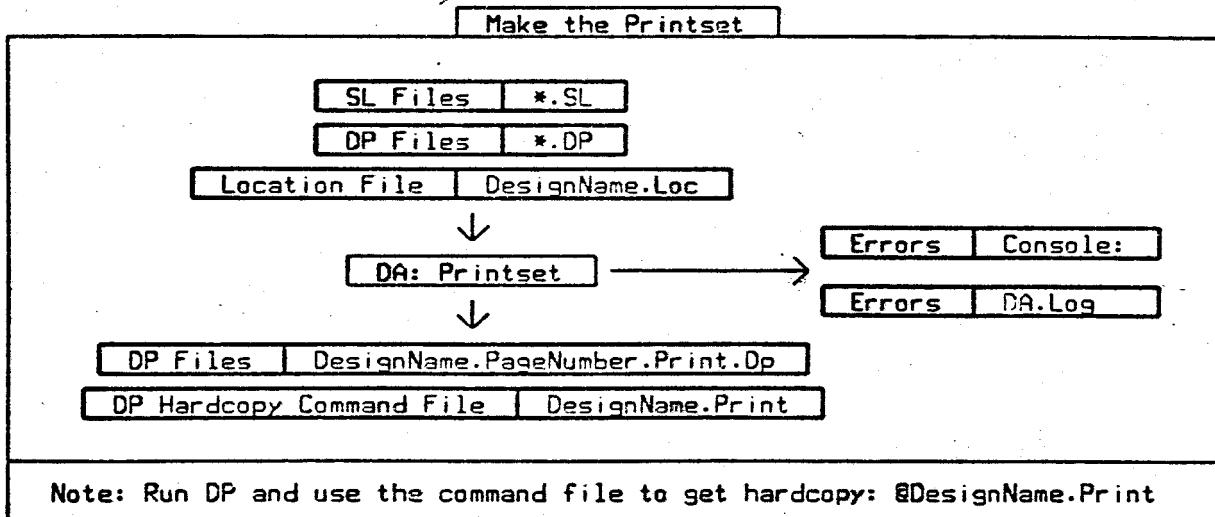
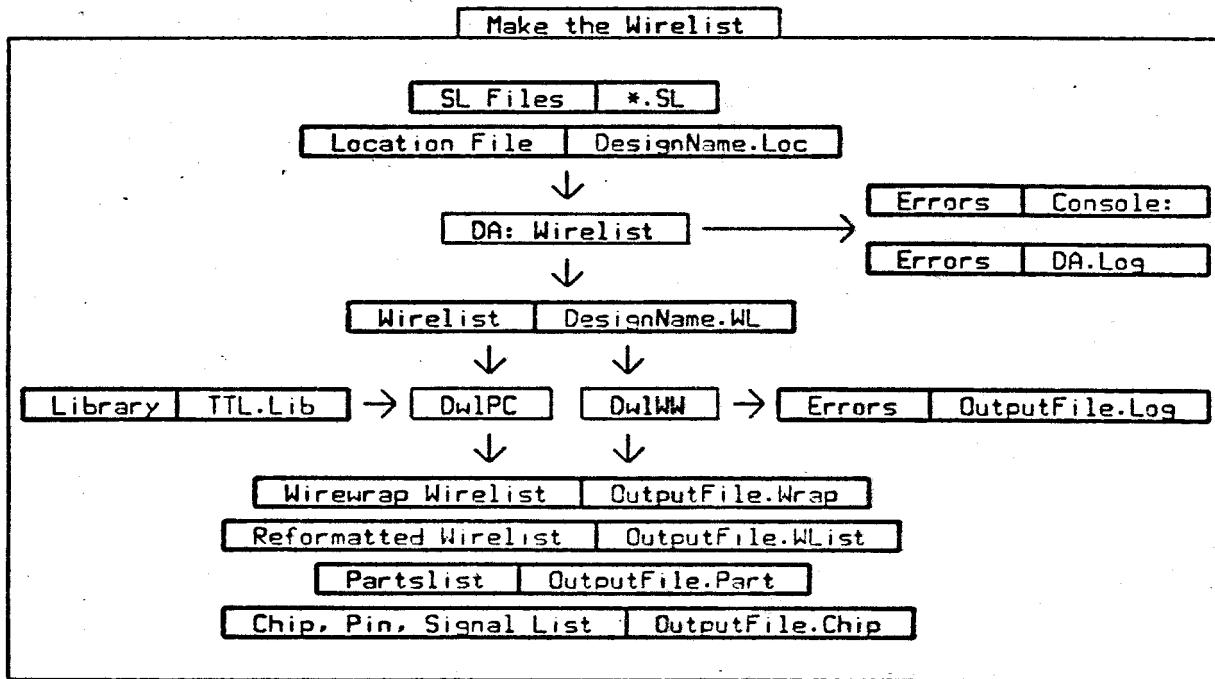


**Edit Schematics****Extract Electrical Information**

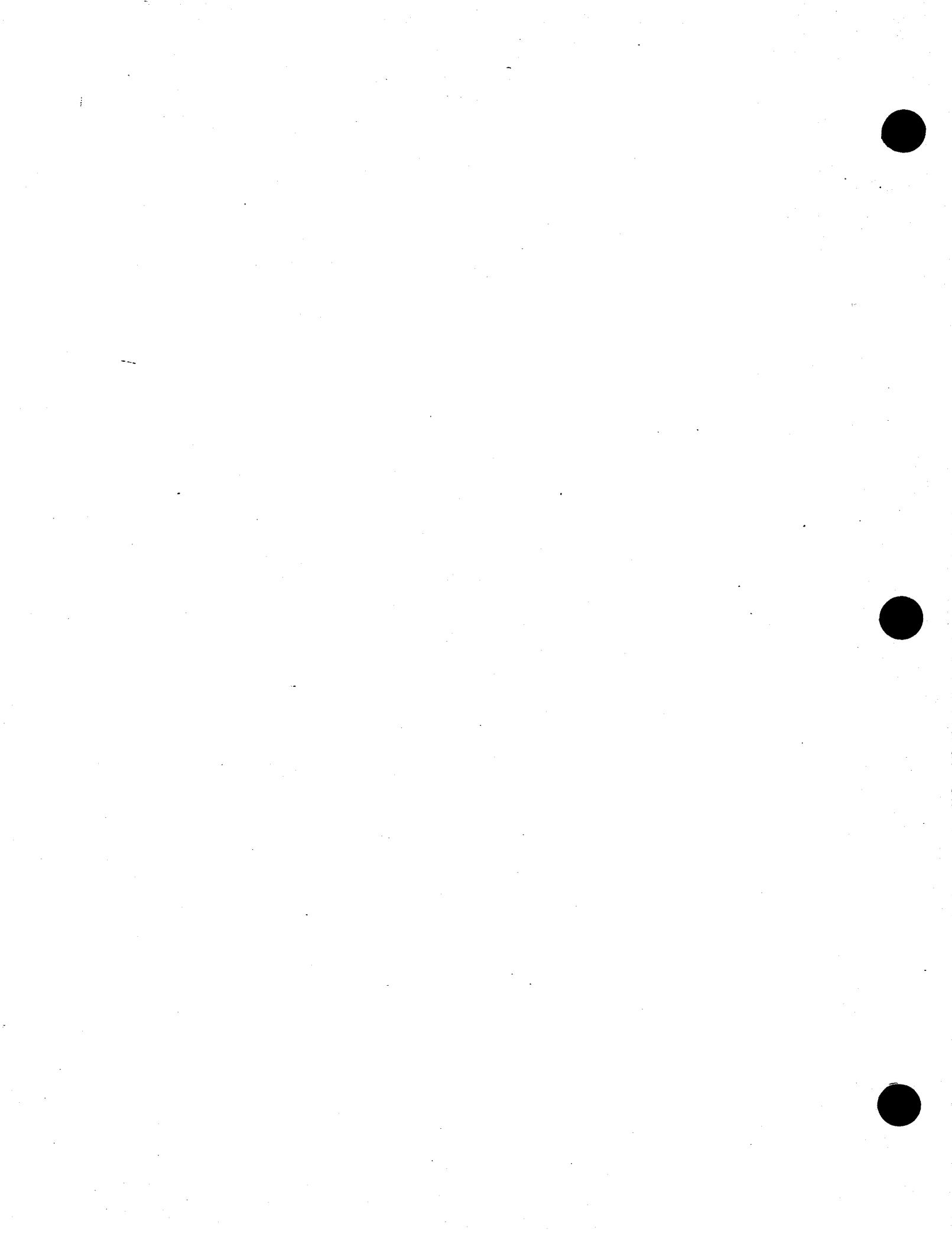
Note: Examine the error messages with DP:

I xxx.Dp  
I xxx.Err.Dp





**Note:** Run DP and use the command file to get hardcopy: &DesignName.Print



**CHAPTER 2**  
**INSTALLATION OF PERQ SYSTEM D/L**



## CHAPTER TWO

## INSTALLATION OF PERQ SYSTEM D/L

This chapter describes the steps used to install PERQ System D/L. This chapter assumes familiarity with the PERQ POS operating system. Specifically, you should have a working knowledge of the file system, file directories, user profiles, search lists, and use of floppy disks. Refer to the PERQ Software Reference Manual for this information.

PERQ System D/L consists of two parts, each of which is loaded into its own directory. The first part consists of programs, help files, and character set files which make up the System D/L software. The second part of System D/L is the component design library.

We recommend that you create a master directory for System D/L named SystemDL and that within this directory you create two sub-directories: one named Software for the software and the other named Library for the component library. At this printing the two directories take up a little less than 3100 disk blocks.

The installation package for System D/L consists of four floppy disks:

SYSTEM D/L  
SYSTEM D/L DP FILES 1 OF 2  
SYSTEM D/L DP FILES 2 OF 2  
SYSTEM D/L SL FILES

The SYSTEM D/L floppy disk contains the programs, help files, and character set files which make up System D/L. The SYSTEM D/L DP FILES and SYSTEM D/L SL FILES floppy disks contain the component design library.

To load the System D/L software, change your path to the System D/L Software directory, insert the SYSTEM D/L floppy, and execute the following commands.

```
Floppy Get Get.Cmd /NoConfirm
Floppy @Get.Cmd
Floppy Get Link.Cmd /NoConfirm
@Link
```

To load the System D/L library, change your path to the System D/L Library directory. For each of the three library floppies, insert the floppy and execute the following commands.

Floppy Get Get.Cmd /NoConfirm  
Floppy aGet.Cmd

Now, modify your user profile to contain a Login/SetSearch command naming the System D/L Library directory. This will make the System D/L library available to you at all times. The following shows the format of the Login/SetSearch command. ParentPathName is used to represent the name of the path containing the SystemDL directory.

#Login  
/SetSearch=ParentPathName>SystemDL>Library

You must now login again to take advantage of your modified user profile.

After you have loaded System D/L, modified your user profile, and logged in again, the System D/L software and library are available for your use.

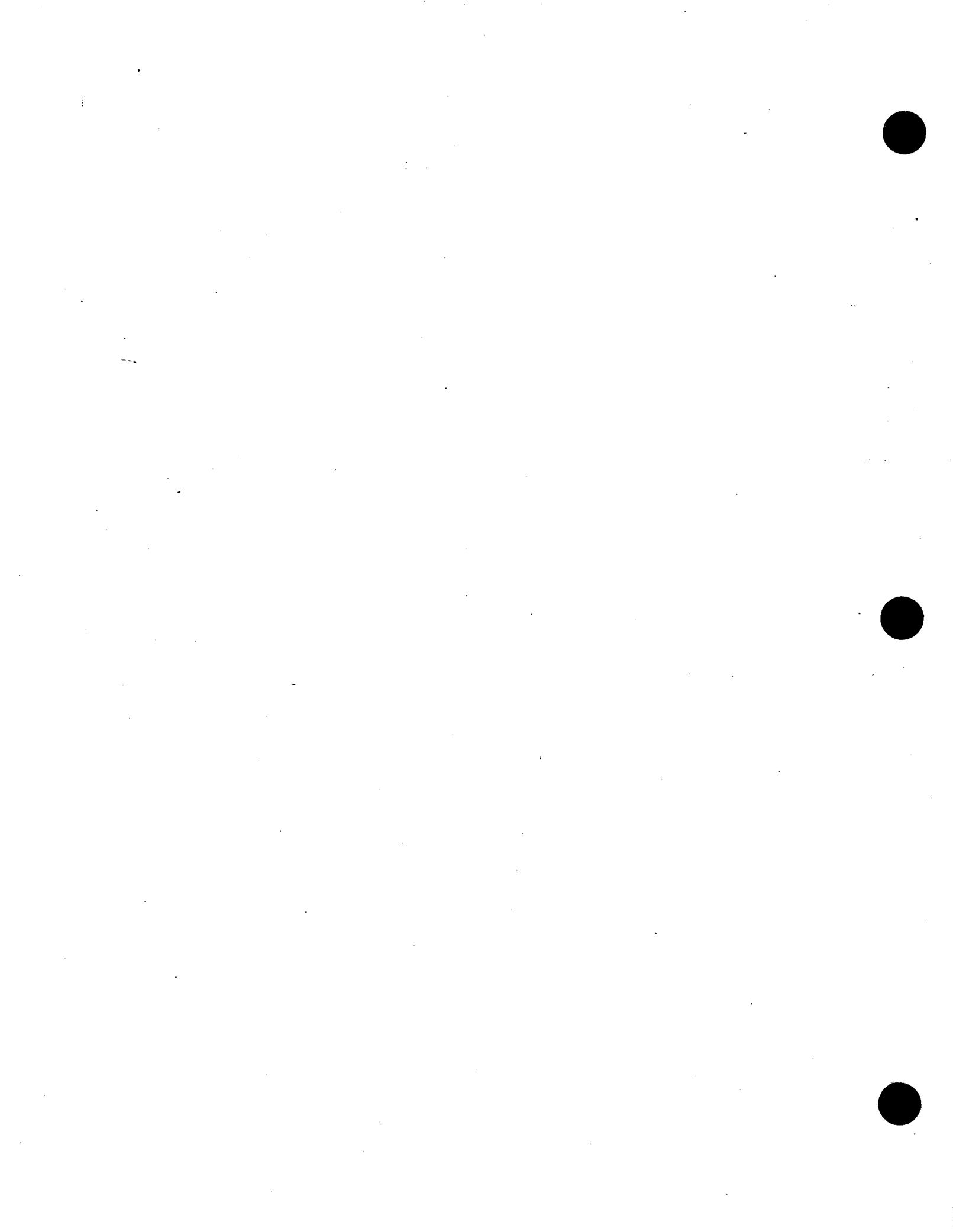
**CHAPTER 3**

**PERQ SYSTEM D/L HARDCOPY FACILITIES**



CHAPTER THREE  
SYSTEM D/L HARDCOPY FACILITIES

|                          |   |
|--------------------------|---|
| PRINTERS AND CONNECTIONS | 1 |
| SCREENDUMP               | 2 |
| TEXTUAL DOCUMENTS        | 2 |
| DP DRAWINGS              | 3 |
| PRINT SERVER             | 4 |



PERQ System D/L provides the ability to print paper copies of drawings and textual documents from the PERQ. Different methods are used for these two types of printing. Drawings are printed by displaying them on the screen and printing an image of the screen. This is called screendump printing. DP and Scope provide commands for screendump printing, and the sole purpose of the ScreenDump program is to print an image of the screen. Textual documents are printed by printing programs that read the text file and send ASCII characters or special commands to the printer.

This chapter describes the way PERQ System D/L uses the PERQ printing software. The various screendump printers, the PERQ Laser Printer 10 (PLP-10), the PERQ Ethernet interface, the RemotePrint program, and the CPrint program are PERQ options which must be ordered separately from PERQ System D/L.

### PRINTERS AND CONNECTIONS

The PERQ operating system optionally supports several printers. The following printers are supported for screendumps:

PERQ Laser Printer (PLP-10)                          \  
Hewlett-Packard 7310A thermal printer                  > or equivalent  
Paper Tiger 560 dot matrix printer                          /

The following printers are supported for textual printing:

PERQ Laser Printer (PLP-10)  
Hewlett-Packard 7310A thermal printer  
Paper Tiger 560 dot matrix printer  
Texas Instruments 810 dot matrix printer  
Diablo 630 daisy wheel printer  
any RS232 compatible ASCII printer                  or equivalent

Printers are connected to the PERQ in different ways. RS232 printers (Paper Tiger 560, TI 810, Diablo 630, ASCII printer) are connected directly to the RS232 port on the rear of the PERQ. GPIB printers (HP 7310A) are connected to the GPIB bus connector on the rear of the PERQ. The PLP-10 is connected to the I/O option connector on the rear of the PERQ.

Optional PERQ software is available that allows many machines to print drawings and textual documents on a single PLP-10 without moving the printer from machine to machine. This is done by assigning a single PERQ as the PrintServer machine. All PERQs including the PrintServer are connected together with the optional Ethernet local area network. The individual PERQs can print by sending messages to the PrintServer. This is the preferred method of connecting a printer for PERQ System D/L.

### SCREENDUMP

DP, DA, and Scope provide commands for printing a screendump of what is currently displayed. In addition, the G.3 and all subsequent versions of the operating system provide a special keystroke, Control-Shift-P, to print a screendump when using other programs. Under the G.3 and all subsequent versions of the operating system, screendump is done in such a way that programs need not know which printer is being used. The printer is selected at the time the operating system is linked by selecting the appropriate version of the SID module. Refer to the PERQ Software Reference Manual for more information.

### TEXTUAL DOCUMENTS

Three programs are optionally available for printing textual documents. The first is the standard PERQ Print program. This is used for printing on the Hewlett-Packard 7310A thermal printer, the Paper Tiger 560 dot matrix printer, the Texas Instruments 810 dot matrix printer, the Diablo 630 daisy wheel printer, or any RS232 compatible ASCII printer. See the PERQ Software Reference Manual for information on how to run this program.

The second printing program, RemotePrint, is used to print documents on the PLP-10 by sending them across the Ethernet to the PrintServer machine. The form of the RemotePrint command line is:

RemotePrint <FileName>{,<FileName>}{/<Switch>}

The valid RemotePrint switches are:

|                   |  |
|-------------------|--|
| KSet= <Kset file> | Set the default character set.         |
| Prose             | Set to prose mode (KSet=Can40P).       |
| Doc               | Set to Doc file mode (KSet=Can40).     |
| Help              | Give help on a command.                |
| WrapAround        | Wrap around long lines.                |
| NoWrapAround      | Don't wrap long lines.                 |
| BackToFront       | Print last page first.                 |
| FrontToBack       | Print first page first.                |
| Copies= <number>  | Number of copies to print.             |
| Delete            | Delete files after printing.           |
| NoDelete          | Do not delete files after printing.    |
| Wait              | Wait until the server is ready.        |
| NoWait            | Give up if the server is not ready.    |
| Server= <Name>    | Set the preferred server to be <Name>. |
| Scribe            | Set to Scribe mode.                    |

The default switch settings are:

/Doc/WrapAround/BackToFront/Copies=1/NoDelete/Wait

If the Scribe switch is used, the following switches are ignored:

Doc                    KSet                    Prose  
WrapAround            NoWrapAround            BackToFront            FrontToBack

To use any available server, use the name "ANY" as the server name. You can eliminate the need to use the Server switch by add the following line to your user profile:

CPrint /Server=<Name>

The third program is CPrint. CPrint performs the same functions as RemotePrint except that it is run on the PERQ owning the PLP-10 printer. The general form of a CPrint command line is:

CPrint <FileName>{,<FileName>}{/<Switch>}

The switches and defaults are the same as RemotePrint.

DP DRAWINGS

DP drawings are printed with the "H" hardcopy command. This prints a screendump of the drawing that is being edited. You can use the "H" command to print intermediate drawings as you are editing them or to print a set of drawings. Although you can print a set of drawings by typing the commands to read the drawings and print them one at a time, it is more convenient to use a DP command file. To use a command file, you should prepare a text file (using the system text editor) which contains a set of DP commands exactly as you would type them to DP. This means that the commands are not always separated by carriage returns. A DP command file to print three drawings named Drawing1, Drawing2, and Drawing3 would be prepared in the following way:

```
IDrawing1  
H  
SDDIDrawing2  
H  
SDDIDrawing3  
H  
SDD
```

In this command file, the "I" command reads in the first drawing. A carriage return terminates the name of the drawing. The "H" command prints the drawing. A carriage return follows because DP asks for verification of the "H" command. "SDD" selects and deletes the drawing before reading the next one.

This command file is executed by using the "a" command in DP. By using a command file, you can save time and trouble by eliminating the need to type the DP commands whenever you want to make a set of drawings. DP command files also make it easier to print the drawings in the order that you want. Because the PLP-10 reverses the order of the pages printed, your command file should print the drawings in reverse order.

PRINT SERVER

EServer is a program that runs on a PERQ that is connected to a PLP-10 printer. This program responds to requests from other machines on the network and performs any printing that is needed. The program listens for both ScreenDump and Document Print requests.

EServer has no command interface. To run the program simply type EServer to the command interpreter.

**CHAPTER 4**

**THE PERQ SYSTEM D/L GRAPHICAL EDITOR - DP**



## CHAPTER FOUR

## THE PERQ SYSTEM D/L GRAPHICAL EDITOR - DP

DP (Drawing Program) is a highly interactive graphics editor which runs on the PERQ computer. It is a general purpose graphics editor containing powerful features such as multiple layers, hierarchically nested symbols, and multiple windows.

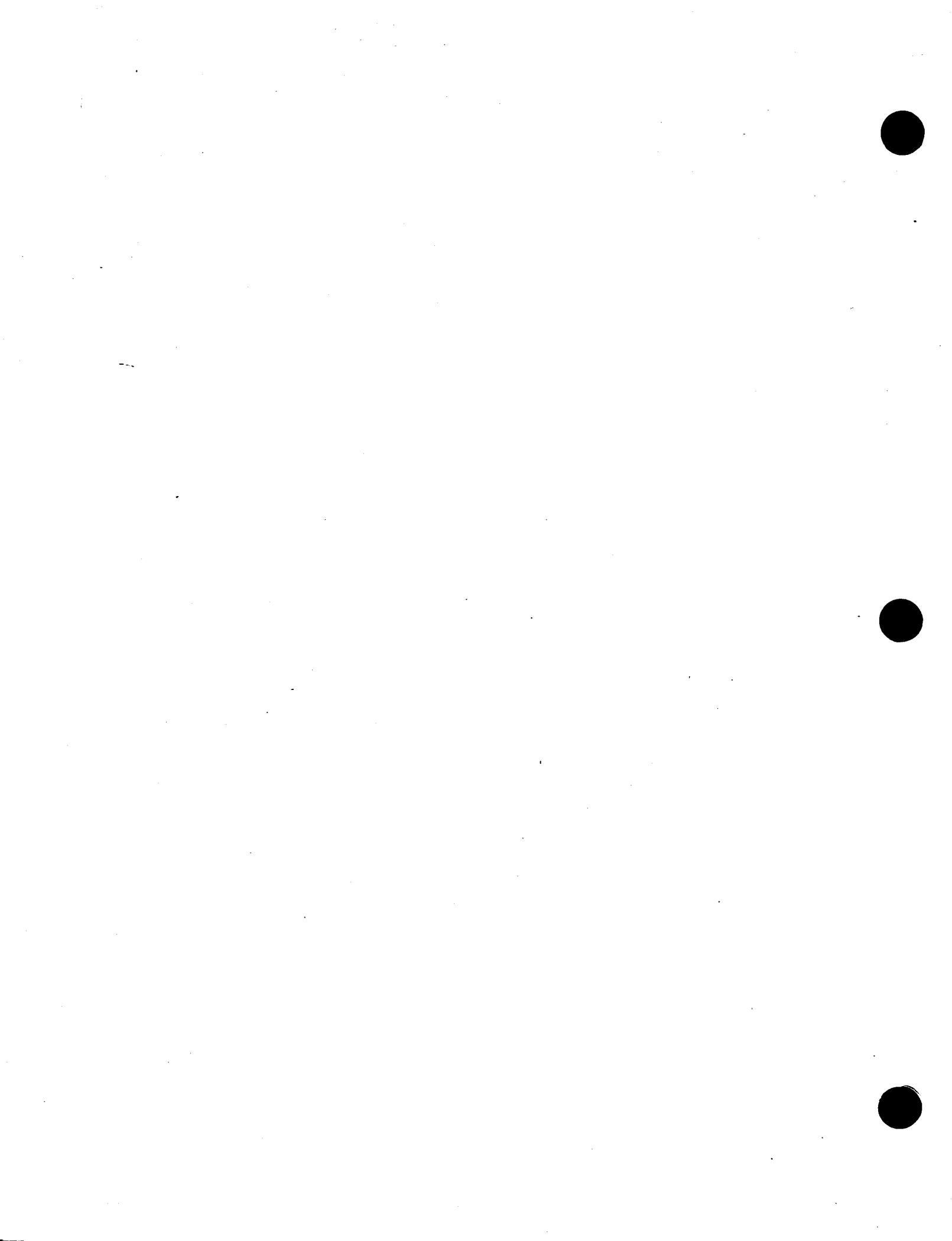
Some typical drawings produced with DP are diagrams, flow charts, graphs, architectural layouts, organizational charts, forms, and circuit designs. In creating drawings, you can copy parts of other drawings from DP files.

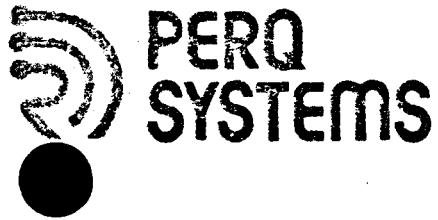
DP was developed by Dario Giuse of the Robotics Institute at Carnegie-Mellon University, Pittsburgh, PA, as part of a circuit design system to reduce the turnaround time between the conception and the implementation of prototype electronic circuits.

When used as an electronic circuit drawing tool, DP allows a designer to create the description of an electronic circuit in a graphical form that corresponds to the way circuits are normally represented in logic diagrams. Although DP has several features that are especially useful for this purpose, DP is purely a graphic editor; it does not try to "understand" the drawing. However, its output may be used as input to post-processors that are able to extract electrical information, perform error checking, and generate a set of lists, such as a wire list, a stuffing list, and a wrap list for wire-wrapping a board.

DP runs under the POS operating system with 256 Kbytes to 1 Mbyte memory. However, 256K memory is not recommended since many of the fast operations must be replaced by slower functions. Landscape monitor support is provided.

A separate DP User's Manual is available from PERQ Systems. It covers all of the features of DP and is written both for those who will use DP for circuit design and for those who will use it for general purposes.





PERQ Systems Corporation  
2600 Liberty Avenue  
P.O. Box 2600  
Pittsburgh, Pennsylvania 15230  
412/621-6250

## Product Release Notes

---

Product: DP Version B.2  
Release date: October 3, 1983

### 1. Product Description

---

DP (Drawing Program) is a highly interactive graphics editor which runs under the POS operating system on the PERQ. It is a general purpose graphics editor containing powerful features such as multiple layers, hierarchically nested symbols, and multiple windows.

The non-source DP product consists of a single floppy disk, the DP User's Manual, DP Command Summary, a sample bug report form, a document response form, and a package (either a three-ring binder or a rectangular box).

### 2. Changes from Previous Version

---

Version B.2 is the first commercially available version of DP. Previous versions had been made available to customers on an informal basis. The following is a list of changes from version 5.11 (the most widely distributed pre-release version) to version B.2:

- 2.1 In the DP user profile, the user may point to another file which contains the commands that are to be displayed in the Top-Level Commands menu.
- 2.2 In the DP user profile the user may set symbol-path and input-path parameters to specify the directories where the user expects to find most of the input and symbol library files. If the argument is /SearchList, the searchlist will be put on the list of directories to be displayed. When the i (input symbol) or I (input drawing) command is given, a popup menu listing these directories is displayed. Pressing at one of the directory names displays another menu showing the .DR and .DP files, if any, in that directory. Therefore a user can "walk through" the directory structure. Pressing outside the menu pops the user up one level to the parent directory. With the i command, after the user points to a .DP file in one of the menus another menu appears, listing the symbols in that file whose names begin with the filename (for example, symbols S240, S240R, and S240L would be shown for file S240). The user then points to the symbol name. If no symbol matches are found, the message "<NO SYMBOLS>" is given.
- 2.3 If the symbol-path and input-path parameters explained above are not placed in the profile, DP will prompt as it did in previous versions. However, if the user types a directory name (ending in ">"), a popup menu is displayed listing all of the .DR and .DP files in that directory (or the message "<NO FILES>").

- 2.4 The grid command (g) now steps through a circular list of mouse grid values. The values are set through the k command from the Unusual Commands menu. The user may add, delete, and change grids and may arrange them in the order desired.
- 2.5 Two new profile options (which also can be set with the k command for the Unusual Commands menu) are replace-grids and append-grids. These are relevant to the I (input drawing) command. Replace-grids ON indicates that the user wants to dispose of the current grids list and replace it with the grids list of the file which is being read in. Append-grids ON indicates that the user wants to keep the current grids list and add all of the grids found in the file which is being read in. Replace-grids and append-grids cannot both be ON; however, both may be OFF, in which case no grids are read from the input file. All grids in the grids list (up to 80 characters worth) are saved in the .DP file when the o (output) command is executed.
- 2.6 If a key is pressed rather than a button when a popup menu is on the screen, the menu is aborted and, if the key represents a command, that command is executed. For example, if the user types p while a menu is displayed, the menu will disappear and pin mode will be current.
- 2.7 Typing <CTRL-C> will abort any menu.
- 2.8 The Help facility has been enhanced.
- 2.9 The arrow cursor does not have to be positioned on the string the whole time the string is being edited.
- 2.10 DP no longer reads in grids and pagemark information when a symbol is read. (This had caused problems resulting in hundreds of pagemarks appearing in .DP files.)
- 2.11 The display of pin positions, pin numbers, and diamonds can be toggled with the P, ~, and # commands respectively, as well as from the Unusual Commands menu.
- 2.12 Layers can now be deleted.
- 2.13 The profile can specify whether a character is displayed next to the command description in the Top-Level Commands menu. The default is that the character be shown.
- 2.14 The profile can specify whether the current function is displayed with the cursor. The default is that the function not be shown.
- 2.15 In the Dp.commands file, an asterisk (\*) at the beginning of the line indicates a customer with a CSDX/Sidnet facility. If a plus sign (+) follows the asterisk, the time stamp is printed at the bottom of the page of a screendump; otherwise, it is not. The user may edit the file to add or remove the plus sign.
- 2.16 Landscape monitor support is provided for POS Versions G.2 and G.3.

- 2.17 If the profile contains errors, DP prints warning messages as it is reading the profile and prompts the user for permission to continue.
- 2.18 Warning and error messages left at the bottom of the screen are erased when a new command is executed. Notice is given in the bottom left corner of the screen when reading or writing a file.
- 2.19 The Magnify command (M) has been added to allow the user to zoom up and down at a small section of the screen.

### **3. Installation Procedure**

---

- 3.1 Turn on your PERQ and log in. If you need assistance in this step, refer to Appendix B of the DP User's Manual.
- 3.2 Type "MAKEDIR SYS:USER>DP" to create a DP directory in your user disk partition. Make sure that you have at least 800 free blocks in the user partition before continuing to the next step.
- 3.3 Type "PATH SYS:USER>DP" to make SYS:USER>DP your current directory.
- 3.4 If you have a NON-SOURCE version of DP:

- \* insert the DP floppy in the floppy drive
- \* type "FLOPPY GET GETBIN"
- \* type "FLOPPY @GETBIN"

If you have a SOURCE version of DP:

- \* insert the first DP floppy in the floppy drive
- \* type "FLOPPY GET GET1"
- \* type "FLOPPY @GET1"
- \* remove the floppy and insert the second floppy
- \* type "FLOPPY GET GET2"
- \* type "FLOPPY @GET2"
- \* compile all the .PAS files which are now on your PERQ.

You now have all DP files on your Perq.

- 3.5 Link the program as follows:

- \* if you are running version F.1 of the POS operating system, type "LINK DP,JPSDYNAMIC"
- \* if you are running version G.2 or G.3 of the POS operating system, type "LINK DP"
- \* In either case, if you wish to use the DP "H" (screendump) feature to send a screen image to a device such as a laser printer or a plotter, you need to link DP with the correct CSDX or SID file (depending on whether your POS operating

system version is F or G, respectively). This is done by adding a comma followed by the CSDX or SID filename at the end of the link command line. If, for example, you have a laser printer connected to your PERQ via an Ethernet and you are running POS version F.1 you would type

"LINK DP,JPSDYNAMIC,CSDXNET"

See your PERQ system manager for details.

- 3.6 You now have a runnable DP program on your PERQ. To run the program, just type "DP" or "DP <filename>", provided that your current directory is SYS:USER>DP. If you want to be able to run DP from any directory, you must edit your profile file to add SYS:USER>DP. See the PERQ Software Reference Manual for information on how to do this.

#### 4. Outstanding Restrictions

---

The following is a list of known restrictions and limitations in DP:

- 4.1 If the scale is changed, the items as they appear on the screen may no longer be aligned as they were originally. However, when the user returns to the original scale, the items are in their original positions.
- 4.2 There is a limit of 25 characters to filenames which you specify in your profile file. This limit pertains only to the actual name of the file, not to the entire pathname.
- 4.3 An ascii string which is vertical cannot be rotated to horizontal. This applies even to strings which were originally horizontal and have been rotated to vertical.

#### 5. Configuration Requirements

---

##### 5.1 Operating System

Versions of DP exist which run under POS versions F.1, G.2, or G.3. In order for DP source files to compile under F.1 the following constants must be set:

| File        | Constant            | Value |
|-------------|---------------------|-------|
| DP.Data     | GVersion            | false |
| Magnifier   | GVersion            | false |
| FindSymbols | FindSymbolsGVersion | false |
| DRScan file | DRScanGVersion      | false |

## 5.2 Support Software

The following files must always be present and available to DP:

```
Perq.Qcodes.dfs  
dphelp  
dp.commands  
Circle.micro (Circle.bin)  
Circle.dfs  
Gacha7  
Gacha7.90  
Gacha7b  
Gachaovb7  
Gachaovb7.90
```

The following files must be available in order to compile DP source files:

```
RealFunctions.Pas  
Csdx.Pas  
Sail_String.Pas  
PopUp.New.Pas  
Magnifier.Pas  
PopWindow.Pas  
JpsDynamic.Pas  
DRscan.pas  
FindSymbols.Pas  
RealString.Pas
```

The following files must be available in order to link DP binary files:

```
RealFunctions.Pas  
Csdx.Pas  
Sail_String.Pas  
PopUp.New.Pas  
Magnifier.Pas  
PopWindow.Pas  
JpsDynamic.Pas  
DRscan.pas  
FindSymbols.Pas  
RealString.Pas
```

## 5.3 Hardware Requirements

DP runs on either the PERQ or the PERQ2, with 256 Kbytes to 1 Mbyte memory. However, 256K memory is not recommended since many of the fast operations must be replaced by slower functions. Landscape monitor support is provided. Either a three-button or four-button mouse can be used (with a four-button mouse, the yellow and blue buttons perform the same function).

#### **5.4 Peripherals**

The DP program can send a screendump to a laser printer or a print request to a plotter, if those devices are connected to the PERQ directly or via an Ethernet print server and if the proper CSDX or SID modules are linked to DP.

### **6. Compatibility with Previous Versions**

#### **6.1 Upward compatibility**

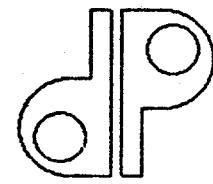
All .DP files created by previous versions of DP can be used without difficulty by this new version.

#### **6.2 Downward compatibility**

All .DP files created by this version of DP can be used without difficulty by previous versions.

### **7. Support**

Customer support for DP users is provided by the PQS Customer Service Department. Please direct all inquiries and send all bug reports to Customer Service, attention Software Support.



Version 8.2

## Command Summary

**perlog**

The word "perlog" is written in a stylized, lowercase font. The letter "p" is tall and narrow, while the other letters are shorter and wider. There is a horizontal line underneath the letters "e", "r", "l", and "o".

© PERQ Systems Corporation  
Prolog Software Ltd.



## COMMAND SUMMARY

Mouse  Yellow  
Green  
White

1

|     |   |   |  |
|-----|---|---|--|
| a   | Enter text strings                              |    | single string entry<br>multiple strings entry<br>single string entry   |
| b   | Pack a collection of items into a single symbol |    | one item<br>selected items<br>delimited items  |
| B   | Unpack a symbol into its components             |    | one item<br>selected items<br>delimited items  |
| c   | Copy items                                      |    | one item<br>selected items<br>delimited items  |
| d   | Delete items                                    |    | one item<br>selected items<br>delimited items  |
| D   | Delete ALL selected items                       |    | one item<br>selected items<br>delimited items  |
| DEL | Delete the previous marker.                     |   | Reverse of the INS command   |
| e   | Edit items                                      |   |  |
|     | LINE:   |  | change length, Hor/Vertical/45 deg<br>change length, Generic, NO gravity<br>change length, Generic, WITH gravity |
|     | CIRCLE:   |  | change radius<br>change to an arc<br>change to an arc  |
|     | ARC:  |  | change radius<br>change length of arc<br>change length of arc  |
|     | SPLINE:   |  | reveal spline nodes, edit nodes<br>close spline after editing<br>reveal spline nodes, edit nodes                 |
|     | TEXTS:  |  | position cursor before character<br>position cursor & delete next char.<br>position cursor before character      |
|     | PACKED SYMBOLS:                                 |  | display name<br>rename<br>display name   |

## Command Summary



2

- f Choose a font, enter a new font  
in the table, or delete a font

- font type (e.g. gacha, cou20, etc)
- font type
- new font (enter new font in table)
- delete font (from table)

Max. no of fonts in table = 40.



### Font properties:

family: gacha, german, cou20, etc.  
size: size of the font (1 - 50)  
face: r - roman,  
i - italic,  
b - bold.  
rotation: orientation (0 or 90 deg).  
Perfont: font filename (e.g. cou20.kst)

Properties depend on the font when  
it was created.

- G Go to the next mouse grid in buffer

- G Go to next marker in the  
circular buffer

Markers are inserted by the INS command;  
deleted by the DEL command.

- H Help command

- H Make a hard copy of the drawing on  
the screen (Screen dump)

- I Input a symbol from another drawing file

DP prompts user for the symbol name.

- I Input a drawing file

DP prompts user for the file name.

- INS Insert mark at center of current window  
and place the marker in circular buffer.

- j Input a text file starting at  
the current cursor position.

DP prompts user for the file name.

## Command Summary



3

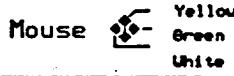
k Multiple menu to select:



} for menu selection

- |  |   |
|--|---|
| <input type="checkbox"/> change checkpoint value                       | DP prompts for value (default = 100)  |
| <input type="checkbox"/> change display grid value                     | DP prompts for value (default = 8)  |
| <input type="checkbox"/> change list of mouse grids                    | DP prompts for number and position in buffer.   |
| <input type="checkbox"/> change gravity                                | DP prompts for new value (default = 5)  |
| <input type="checkbox"/> use default ext. DP (YES/NO)                  | If YES, all drawing files saved have the filename extension .DP.  |
| <input type="checkbox"/> center file (YES/NO)                          | Center a drawing when it is input (read) or output (saved).   |
| <input type="checkbox"/> rename redefined (YES/NO)                     | If YES, symbol is renamed if same name as a symbol on incoming drawing. If NO, the symbol is replaced by incoming symbol. |
| <input type="checkbox"/> toggle cursor shape                           | Change NW cursor to NE cursor or vice versa.  |
| <input type="checkbox"/> display in black mode (YES/NO)                | Display all overlapping parts of items.   |
| <input type="checkbox"/> display diamonds (YES/NO)                     | If YES, show diamonds at junctions of lines (except corners).   |
| <input type="checkbox"/> display pin numbers (YES/NO)                  | If YES, show pin numbers on screen.   |
| <input type="checkbox"/> display pin positions (YES/NO)                | If YES, show pin positions on screen.   |
| <input type="checkbox"/> send to plotter (YES/NO)                      | Upon specifying YES, drawing in current window is sent to plotter connected to PERQ.                                      |
| <input type="checkbox"/> clip to size of window when printing (YES/NO) | If YES plotter print will include only portion of drawing shown in current window.  |
| <input type="checkbox"/> diamonds when printing (YES/NO)               | If YES, diamonds are printed at the junctions of lines (except corners).  |
| <input type="checkbox"/> read grids (YES/NO)                           | If YES, mouse grids on incoming drawing are read in.  |
| <input type="checkbox"/> replace grids (YES/NO)                        | If YES, mouse grids on incoming drawing replace current list of grids.  |

## Command Summary



4

1 Draw lines

Horizontal, Vertical, 45 deg lines  
Generic lines, no gravity  
Generic lines

L Draw rectangles

draw rectangles

M Move items

move one item  
move all selected items  
move delimited items

H Magnify items

White button to increase, green to decrease.

N Set symbols to new parameters:

set one item  
set selected items  
set delimited items

for menu selection

- new thickness <number>  
Specify a thickness.
- new thickness:  
Change the thickness to the no. shown.
- font <name>  
Specify a font.
- new font:  
Change to the font shown.
- layer <name>  
Specify a new layer.
- new layer:  
Change to the layer shown.
- color <number>  
Specify no. of colors on plotter.
- new color:  
Change to the no. of colors shown.

O Output (save) the drawing to a file

DP prompts for file name.

## Command Summary

Mouse  Yellow  
Green  
White

5

|  |  |   |
|--|--|---|
| P Create a pin   |   | prompt for pin number<br>no prompting (default no. used)      |
| P Toggle display of pins   |   | prompt for pin number   |
| Q Quit (exit from) DP.   |  |   |
| F Refresh current window   |  |   |
| R Refresh screen   |  |   |
| S Select items   |   | one item<br>All items<br>In Rectangle                         |
| S Select ALL items in the current window.  |  |   |
| t Transform symbol(s).   |  | one item<br>selected items<br>In Rectangle                    |
|  |  | for menu selection  |
|  Rotate <angle>           |  | Change to angle shown.  |
|  Rotate Counter-Clockwise |  | DP prompts user for value in degree.                          |
|  Rotate Clockwise         |  | DP prompts user for value in degree.                          |
|  Mirror X (-)             |  | Produce mirror image of symbols about the x-axis.             |
|  Mirror Y (1)             |  | Produce mirror image of symbols about the y-axis.             |
|  Scale                    |  | Scale up or down in size.                                     |
|  Scale X                  |  | Same as Scale, but for x axis only; has no effect on circles. |
|  Strip transformations    |  | Remove all transformations from designated items.             |
|  Read parameters          |  | Display current parameters of designated items.               |

## Command Summary

Mouse  Yellow  
Green  
White

6

- U      Undelete items deleted during previous Delete mode. (Upon reentry to Delete mode, all deleted items are erased from memory.)

- V      Layer command. When initiated, the following menu is displayed:



} for menu selection

| Display                             | Alter                               | Output                              | Current                             |
|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| <input checked="" type="checkbox"/> |                                     |                                     |                                     |
| <input checked="" type="checkbox"/> |                                     |                                     |                                     |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| <input checked="" type="checkbox"/> |                                     |                                     |                                     |

(See below)

<layer name>

<layer name>

STANDARD

INVISIBLE (optional)

|           |
|-----------|
| NEW LAYER |
| RENAME    |
| DELETE    |

add a new layer

rename an existing layer

delete a layer

indicates ON or enabled.

Display = display all items in the selected layer.

Alter = allow items to be changed in the selected layer.

Output = allow the items to be saved (output) to a file.

Current = Change the current layer. All items added after you leave the menu will go onto this layer.

## Command Summary

Mouse      Yellow  
Green  
White

7

H Move (scroll) the whole drawing



} move window (green button  
is fastest)

X 'Sticky' move. When a line connected  
to other lines is moved, these lines  
have a rubberband effect so as to  
stay connected to the line that is  
moved.



} one item  
selected items  
in rectangle

Z de-select items



} one item  
selected items  
in rectangle

Z de-select ALL items in the current  
window

G draw splines (curves)



} add point  
connect all points & close spline  
add point

O draw circles or arcs.

Arcs are drawn counter-clockwise.  
start = start point of the arc.  
end = end point of the arc.



} draw circle (center & radius)  
draw arc (start, end, center)  
draw circle (center & radius)

- (hyphen) Change current thickness

DP prompts user for the thickness.  
(1 - 7).

R use a command (transcript) file as  
an alternative to keyboard command  
input

DP prompts user for the filename:  
<filename> <delay>  
The larger the value, the greater  
the delay.

## Command Summary

Mouse  Yellow  
Green  
White

8

- = change the scale (DP prompts for a value)
- ' (backquote) enable/disable grid display
- ? display DP internal information:
  - \* coordinates of the cursor position.
  - \* current scale value.
  - \* current grid size.
  - \* coordinates of the current window:
    - bottom left corner
    - top right corner
  - \* coordinates of the viewport'
  - \* current thickness'
  - \* current color'.
  - \* no. of free segments.
  - \* list of deleted symbols, if any.
  - \* names of defined symbols, if any.
- \* enable/disable pin number display
- # display diamonds (black squares)  
at junctions (not corner) of lines
- CTRL C abort command prompt or menu
- SPACE display Top-Level menu

DP

USER'S MANUAL

Joyce Swaney

July 1983

This manual is for use with DP Version B.2, which is to be used with POS Release F.1 and subsequent releases until further notice.

Copyright(C) 1983  
PERQ Systems Corporation  
2600 Liberty Avenue  
P. O. Box 2600  
Pittsburgh, PA 15230  
(412) 621-6250

This document is not to be reproduced in any form or transmitted in whole or in part, without the prior written authorization of PERQ Systems Corporation.

The information in this document is subject to change without notice and should not be construed as a commitment by PERQ Systems Corporation. The company assumes no responsibility for any errors that may appear in this document.

PERQ Systems Corporation will make every effort to keep customers apprised of all documentation changes as quickly as possible. The Reader's Comments card is distributed with this document to request users' critical evaluation to assist us in preparing future documentation.

PERQ and PERQ2 are trademarks of PERQ Systems Corporation.

## TABLE OF CONTENTS

|                                     | <u>Page</u> |
|-------------------------------------|-------------|
| 1. INTRODUCTION                     | 1           |
| 2. FEATURES                         | 3           |
| 2.1 Basic Elements                  | 3           |
| 2.2 Sets, Symbols, and Instances    | 4           |
| 2.3 Layers                          | 5           |
| 2.4 Windows                         | 5           |
| 2.5 Gravity                         | 6           |
| 2.6 Scale                           | 6           |
| 2.7 The Coordinate System           | 6           |
| 2.8 Display Grid                    | 7           |
| 2.9 Mouse Grid                      | 7           |
| 2.10 Size of the Drawing            | 7           |
| 2.11 Files                          | 7           |
| 3. ENTERING DP                      | 8           |
| 4. EXITING DP                       | 9           |
| 5. GETTING HELP                     | 10          |
| 6. WORKING WITH DP                  | 11          |
| 6.1 The Display                     | 11          |
| 6.2 Using the Mouse and Mouse Grids | 14          |
| 6.3 Issuing Keyboard Commands       | 17          |
| 6.4 Aborting Commands               | 17          |

|   | <u>Page</u> |
|---|-------------|
| 6.5      Using Menus                          | 17          |
| 6.5.1    Changing the Top-Level Menu          | 18          |
| 6.6      Responding to Prompts                | 19          |
| 6.7      Editing Commands and Text Strings    | 20          |
| 6.8      Getting Information from the System  | 20          |
| 6.9      Changing the Display                 | 21          |
| 6.10     Automatic Saves                      | 24          |
| 7.        BASIC ELEMENTS                      | 25          |
| 7.1      Creating Elements                    | 25          |
| 7.1.1    Line                                 | 25          |
| 7.1.2    Rectangle                            | 27          |
| 7.1.3    String of Text                       | 27          |
| 7.1.4    Circle or Arc                        | 28          |
| 7.1.5    Spline                               | 29          |
| 7.1.6    Pin                                  | 30          |
| 7.2      Editing Elements                     | 31          |
| 7.2.1    Line or Rectangle                    | 31          |
| 7.2.2    String of Text                       | 32          |
| 7.2.3    Circle or Arc                        | 33          |
| 7.2.4    Spline                               | 34          |
| 7.2.5    Pin                                  | 34          |
| 8.        SETS, SYMBOLS, AND INSTANCES        | 35          |
| 8.1      Forming Sets (Selecting)             | 36          |
| 8.2      Forming Symbols (Packing)            | 37          |
| 8.2.1    Symbol Definition                    | 37          |
| 8.2.2    Symbol Instances                     | 39          |
| 8.3      Displaying or Changing a Symbol Name | 39          |
| 8.4      Transforming Instances               | 40          |

|   | <u>Page</u> |
|---|-------------|
| 8.5 Unpacking an Instance                       | 42          |
| 8.6 Redefining a Symbol                         | 43          |
| 9. MANIPULATING ITEMS                           | 44          |
| 9.1 Moving                                      | 44          |
| 9.2 Copying                                     | 44          |
| 9.3 Stretching                                  | 45          |
| 9.4 Rotating                                    | 46          |
| 9.5 Magnifying                                  | 46          |
| 9.6 Deleting and Undeleting                     | 47          |
| 9.7 Specifying Parameters                       | 48          |
| 9.7.1 Thickness                                 | 48          |
| 9.7.2 Font                                      | 49          |
| 9.7.3 Setting New Parameters on Existing Items  | 52          |
| 9.8 Printing                                    | 53          |
| 10. INPUT AND OUTPUT FILES                      | 54          |
| 10.1 Writing to a DP File                       | 54          |
| 10.2 Looking for Files                          | 55          |
| 10.3 Reading in Another Drawing in Its Entirety | 56          |
| 10.4 Reading in a Symbol from Another Drawing   | 57          |
| 10.5 Reading in Text from a File                | 58          |
| 10.6 Reading in Commands                        | 58          |
| 11. UNUSUAL COMMANDS                            | 59          |
| 12. TRANSCRIPTS                                 | 63          |

|   | <u>Page</u> |
|---|-------------|
| 13. LAYERS  | 64          |
| 13.1 Changing Settings or Changing to Another Layer | 65          |
| 13.2 Creating, Deleting, or Renaming a Layer        | 66          |
| 14. MULTIPLE WINDOWS                                | 67          |
| 15. PROFILE FILES                                   | 69          |
| 15.1 General Commands                               | 70          |
| 15.2 Window Commands                                | 74          |
| APPENDIX A COMMAND SUMMARY                          | A-1         |
| APPENDIX B INTRODUCTION TO THE PERQ                 | B-1         |

## DP

I. INTRODUCTION

DP (Drawing Program) is a highly interactive graphics editor which runs on the PERQ computer. It is a general purpose graphics editor containing powerful features such as multiple layers, hierarchically nested symbols, and multiple windows. Drawings produced with DP may be inserted into documents produced by the Mint or Scribe text formatting programs.

Some typical drawings produced with DP are diagrams, flow charts, graphs, architectural layouts, organizational charts, forms, and circuit designs. In creating drawings, you can copy parts of other drawings from DP files or you can place a hard copy of a drawing on the tablet or bitpad and trace it.

DP was developed by Dario Giuse of the Robotics Institute at Carnegie-Mellon University, Pittsburgh, PA, as part of a circuit design system to reduce the turnaround time between the conception and the implementation of prototype electronic circuits.

When used as an electronic circuit drawing tool, DP allows a designer to create the description of an electronic circuit in a graphical form that corresponds to the way circuits are normally represented in logic diagrams. Although DP has several features that are especially useful for this purpose, DP is purely a graphic editor; it does not try to "understand" the drawing. However, its output may be used as input to post-processors that are able to extract electrical information, perform error checking, and generate a set of lists, such as a wire list, a stuffing list, and a wrap list for wire-wrapping a board.

DP is written entirely in Pascal. It runs under the POS operating system with 256 Kbytes to 1 Mbyte memory. However, 256K memory is not recommended since many of the fast operations must be replaced by slower functions. Landscape monitor support is provided.

This manual is a complete user's manual, covering all of the features of DP. It is written both for those who will use DP for circuit design and for those who will use it for general purposes. The only prior knowledge assumed is an understanding of the basic features of the PERQ. (If you have never used a PERQ before, first read Appendix B which is a basic introduction to the PERQ. If you'd like more detailed information than that given in the appendix, see the PERQ Systems manual PERQ System Overview.) Section 2 explains the basic features of DP. Sections 3 through 6

explain how to enter, exit, get help, and work with DP. Section 7 covers the creating and editing of basic elements, and Section 8 tells how to combine basic elements into sets and symbols. Section 9 explains the functions you can perform on items, such as moving them or copying them. Section 10 tells how to write a drawing to a file or to read in other files. The rest of the manual explains features that you will find useful as you gain experience with DP (such as creating a drawing in layers and working with more than one drawing). Appendix A is a summary of all commands.

The following conventions have been used in this manual:

underlining - input to be typed by the user

< > - information to be supplied by the user, such as a filename

[ ] - an optional item, such as a command switch

| - a choice between the items on each side of this mark

For those who require technical information on connecting to plotters or printers or information on the format of DP files, the following publications are available:

Interfacing Plotters to PERQ DP

Interfacing Printers to PERQ DP for Screen Dumping

PERQ DP - Format of the Drawing Files

## 2. FEATURES

In order to give you an overall picture of DP, the following paragraphs briefly explain the features around which DP is built. Details on using these features are given in the referenced sections.

### 2.1 Basic Elements

In DP, drawings are composed of five basic elements, shown in Figure 1:

1. lines (finite length) with any slope--the endpoints have gravity, explained later;
2. strings (sequences of printing ASCII characters);
3. circles and arcs;
4. splines (generic curves, with the shape determined by a set of control points); and
5. pins (connection points with gravity, used mostly on electronic circuits). Each pin has a number associated with it.

Elements can be edited, deleted, and formed into larger units called sets and symbols. Elements also can be moved and copied, either within a drawing or from one drawing to another.

Details on creating and editing elements are given in Section 7, and details on working with elements are given in Section 9.

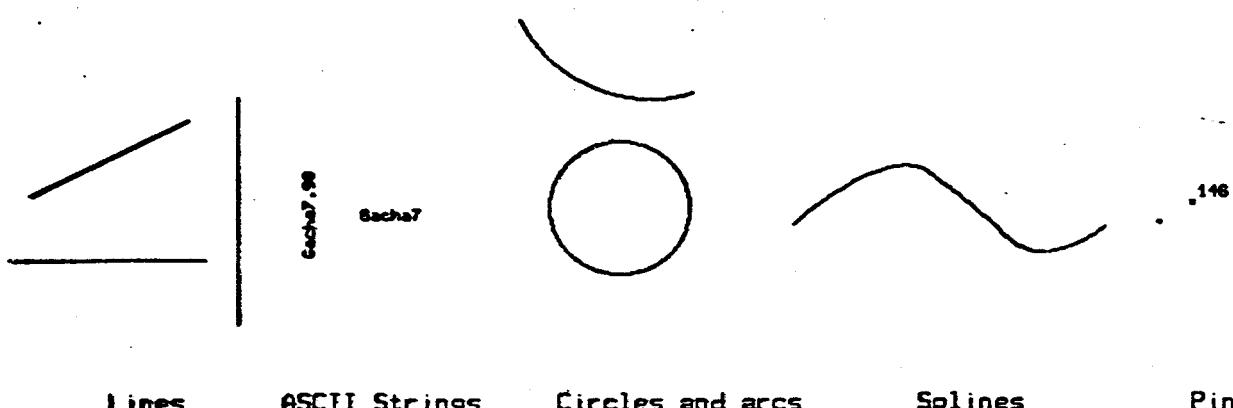


Figure 1. Basic Elements

## 2.2 Sets, Symbols, and Instances

A set is a group of items which have been designated with the Select function. The set is formed so that the same function (for example, delete) can be performed on all the items with one command. There can be only one set per window. The set is temporary, that is, the set will be broken up when you exit DP (or sooner if you choose).

A symbol is a permanent unit consisting of one or more basic elements or other symbols. Each symbol is identified with a unique name, assigned by you or the program. Instances (copies) of the symbol can be easily produced as often as needed on the same or another drawing. Because symbols may be nested inside other symbols, drawings may be created in a hierarchical manner. A symbol instance can be transformed (rotated, mirrored, or scaled), unpacked into its component parts, or deleted--all without affecting the symbol definition.

Like basic elements, the selected set and symbols can be moved and copied, either within a drawing or from one drawing to another.

Details on forming sets, symbols, and instances are given in Section 8, and details on working with these items are given in Section 9.

### 2.3 Layers

Drawings can be divided into layers. This feature provides the same effect as multiple transparencies containing parts of a drawing. Just as transparencies may be added or removed, modifying the drawing itself, each layer can be turned on (displayed) or off (made invisible). When displayed, a layer can be defined as read-only so that it will not interfere with other layers and cannot be changed unintentionally.

Two typical applications are 1) putting construction lines into a separate, read-only layer which can be made visible as needed, and 2) grouping items in a very complex drawing onto different layers so that you can choose to display only certain items at one time.

Details on layers are given in Section 13.

### 2.4 Windows

You may have two or more windows open on your screen at the same time, each containing a different drawing. Each window has a separate coordinate space and is comparable to a sheet of paper. Only the sheet corresponding to the current window (the window in which the cursor is located) may be written on, changed, or manipulated in any way. For example, a "Select All" operation selects all the objects associated only with the window in which the cursor is located. The only exceptions are the move and copy commands, which allow interaction between windows; these commands erase the object from the source window and write it into the destination window.

Details on windows are given in Section 14.

## 2.5 Gravity

The endpoints of lines, pins nested within symbols, circles, and splines all have gravity--that is, they attract the endpoints of lines close to them if those lines are drawn with the gravity option. Gravity serves two purposes: to aid in circuit design and to make lines meet intersecting elements neatly without gaps.

The strength of the gravity can be controlled, and gravity can be turned off.

Details on using gravity with Lines are given in Section 7.1.1, and details on changing the gravity strength or turning off gravity are given in Section 11.

## 2.6 Scale

The image on the current window can be "zoomed" so that it appears larger or smaller. Changing the scale does not affect the true size of the items, and it does not affect the scale of other windows on the screen. A scale of 1 is the real size of the drawing.

It is useful to use a large scale when creating and editing items and a small scale to view large chunks of the drawing or the whole drawing at once.

Details on changing the scale are given in Section 6.9.

## 2.7 The Coordinate System

DP uses a standard-oriented, cartesian coordinate system to represent the objects in a drawing. Sixteen-bit integer arithmetic is used throughout, and therefore the integers in the range -32767...32767 are valid coordinates. The only instance in which you will need this information is in setting your profile to open more than one window when you enter DP (explained in Section 15).

DP initially places point 0,0 at the center of the window and uses a scale of 1. Both the window and the drawing may be moved and the scale changed. You can return the drawing to scale 1 and its initial position by specifying the special case scale of 0.

## 2.8 Display Grid

DP contains a Display Grid, which simulates graph paper and can be used as an aid in drawing. You can have the grid visible or invisible, and you can alter the distance between the dots in the grid.

Details on making the grid visible or invisible are given in Section 6.9, and details on changing the grid are given in Section 11.

## 2.9 Mouse Grid

Cursor movements with the mouse lie on an imaginary grid. When you move the mouse on the tablet or bitpad, DP moves the cursor to the closest grid point. The setting for the mouse grid determines the distance between points. DP will keep several mouse grids in memory for each drawing and allow you to switch among them.

Details on switching among settings are given in Section 6.2. Details on changing the list of settings are given in Section 11.

## 2.10 Size of the Drawing

You may create drawings that are about 64 x 96 times the size of the PERQ screen. This means that a drawing can be as big as 6,000 sheets of paper. Large drawings should be printed on a large-page output device such as a plotter. If the whole drawing is to be printed on a device like the PERQ Systems PLP-10 laser printer, the drawing must be switched to a small scale or printed in several parts.

See Section 9.8 for details on printing a drawing.

## 2.11 Files

A drawing can be stored in a DP file which can be edited in subsequent sessions.

You may access other files to read in a symbol, text, an entire drawing, or DP commands.

Details on input and output files are given in Section 10.

### 3. ENTERING DP

Before you enter DP, you may wish to path to the desired directory (the directory containing an existing DP file you want to access or the directory where you want to create a DP file). However, this is not necessary because you can specify a pathname when you read or write a file within DP. Then type dp.

This command starts DP, using default values for all the variables and switches. If you have not set default values in your profile, as explained in Section 15, the program supplies the defaults listed below. These variables are explained in detail later.

```
current function = line
mouse grid = 3
display grid = 6
gravity field = 5
display scale = 1
current thickness = 1
current font = 1 (Gacha7)
current layer = "STANDARD"
color = 1 (black)
pin display = OFF
```

You may specify the name of an existing file when you enter DP by typing dp <filename>. The extension .DP may be omitted. Alternately, after you enter DP you may access an existing file by typing I. To create a new file, enter DP, create the drawing, and then write it to a file (explained in Section 10.1).

Likewise, DP transcripts (explained in Section 12) can be invoked either when you first enter DP, by typing dp e<filename>, or after you are within DP.

You may add switches to the command line. At present, only two switches are recognized:

|                 |  |
|-----------------|--|
| /profile=<name> | The profile may be used to completely set the initial status of DP (number and position of windows, font sets, default options, grid, etc.) Profile files are covered in detail in Section 15. |
| /help           | prints a short help message about DP and returns you to the Shell  |

#### 4. EXITING DP

To exit DP, type **q** (for quit). (Or get the Top-Level menu as explained in Section 6.5 and select "quit.") DP will ask for confirmation that you wish to exit. Type **y** (for yes) to exit or any other character to stay in the program.

DP keeps track of whether changes have been made to the file(s) on your screen since the last output command. If a window has been modified since the last output command, an asterisk (\*) is appended to the filename. When you give the quit command, DP checks this flag for all existing windows and issues a warning if any window is marked as modified. The algorithm for determining if a window has been modified is conservative, so a window may be flagged even if its contents have not actually been changed. The following rules determine the status of the window:

the output command (to write the changes to a file) clears the Modified flag;

input commands (to read in a file) do not alter the status of the flag;

all non-immediate functions (those that require further action from you, such as "copy") set the Modified flag after the function has been completed;

immediate functions (those that require no further action after you've given the command) do not alter the flag, except Delete and Undelete.

If you have more than one window open and want to remove a window but stay in DP, do not type quit. Instead, remove the window as explained in Section 14. However, only an empty window can be removed.

## 5. GETTING HELP

To obtain help, either type h or press the HELP key. A menu of DP help topics appears. Select the desired help message by pressing any button.

If the help message is more than one page long, press any button or key to get the next page.

After you have read the message, press any button or key. The Help menu will return; to remove the menu, either move the cursor outside the menu and press any button or type CTRL C.

## 6. WORKING WITH DP

The following sections give overall procedural information, such as what the display screen looks like in DP, how to use the mouse, keyboard commands, and pop-up menus, and how to edit text that is typed in response to commands or as input to the drawing.

### 6.1 The Display

Figure 2 shows a display screen in DP. The screen in this example contains only one window. The lower right corner of the window contains the status line, which shows the following information:

command character of the current function (l for Line);

name of the current function (Line); and

the actions associated with the mouse buttons for the current function (Generic | Hor/Ver | No grav. -- explained in the next section)

The area above the status line shows the current mouse grid and the name of the current layer.

The filename is shown in the lower left corner of each window. An asterisk (\*) is appended to the filename if the window has been modified since the last output.

The cursor is an arrow pointing northwest. The position of the cursor (arrow) is controlled by the mouse. The "current" window is the window in which the cursor is located. (If the cursor is not located within a window, you cannot issue commands.) If you choose, the command character and the name of the current function appear with the cursor, as a constant reminder of the current function (see Section 15 for details on how to set this up in your profile).

When the program is waiting for keyboard input in response to a prompt, a Prompt Area will appear on the screen. You must either reply or abort the command (with CTRL c).

The rectangles in the border are used to invoke the Top-Level Commands menu, and the corners of the border and the border itself are used to create, delete, move, or change a window (shown on Figure 2 and explained in detail in Section 14). Items within a window can be moved with the w command explained in Section 6.9.

Warning and error messages will be given at the bottom of the

screen and erased when a new command is given.

Section 6.9 explains ways in which the display can be changed without affecting the drawing.

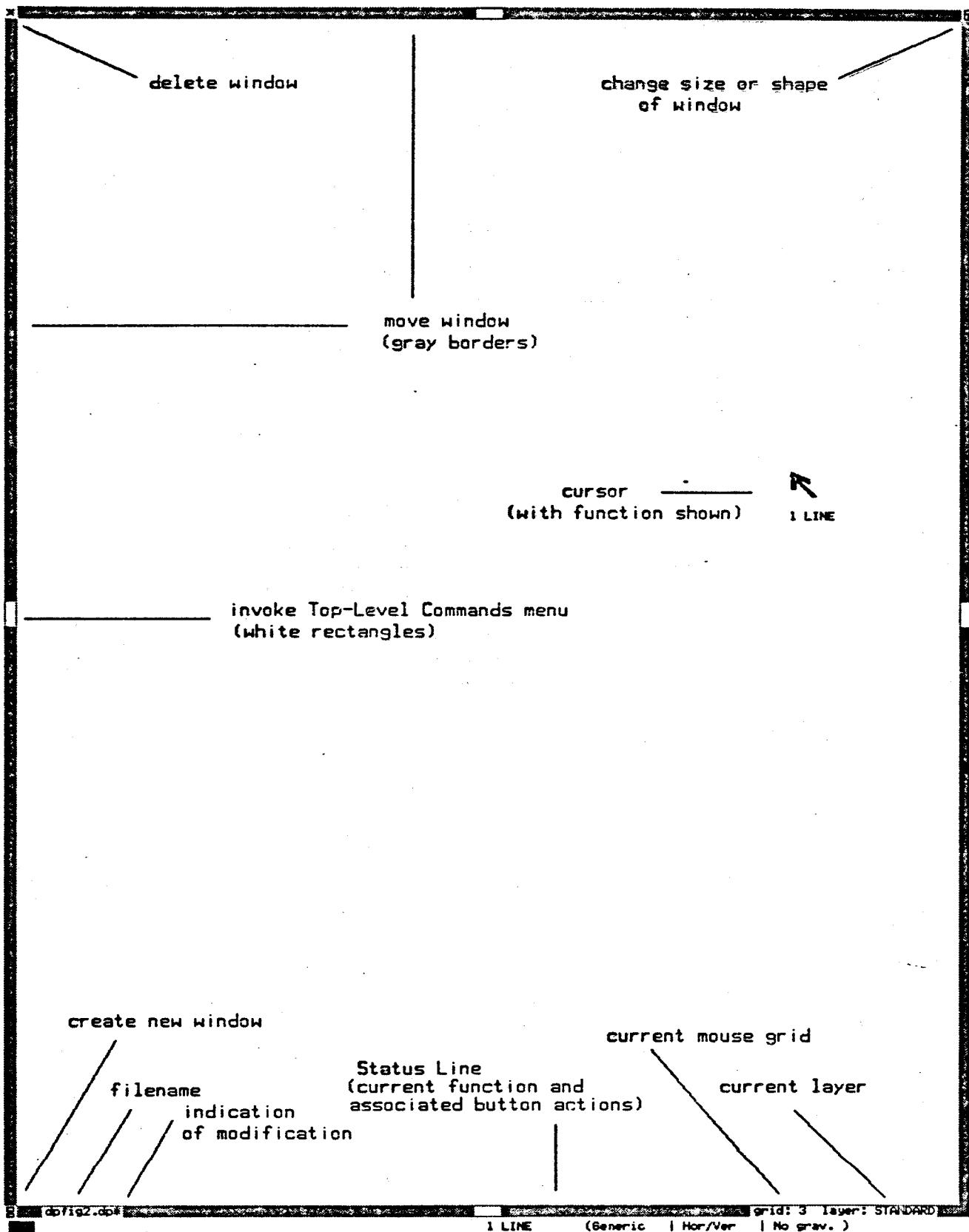


Figure 2. The Display Screen

## 6.2 Using the Mouse and Mouse Grids

In most functions DP assigns different meanings to each mouse button. DP uses only three buttons; if you have a four-button mouse with different colored buttons, the blue button performs the same function as the yellow button. The status line at the lower right of the screen always shows how the buttons perform in the current function. Each item separated by a straight line represents a button (for example, in Figure 2, the actions for Line function are "Generic" for the left or white button, "Hor/Ver" for the middle or yellow or blue button, and "No grav." for the right or green button). Details on the mouse actions for each function are given in this manual; however, after you have worked with DP a short time you'll probably find that the descriptions at the bottom of the screen will be sufficient.

Many of the functions use the buttons as follows:

left (white) - Rectangle (to choose a group of items identified by forming a rectangle around them). Position the cursor to the left or right of the group of items you want to manipulate. Press the button, and while holding the button, move the cursor in any direction to extend the rectangle until it encompasses all the items in their entirety. To freeze the rectangle, release the button. (If an item was not completely contained within the rectangle, the item will not be affected by the function you are performing.)

middle (yellow or blue) - One item (to choose one item at a time). Press the button and point the cursor at the item.

right (green) - Selected (to choose all the "selected" items in the current window). Section 8.1 tells how to select and deselect items with the S command. Selected items are displayed with a black square in the center or, if the item is text, highlighted. You do not need to point to the items with the cursor; pressing the right (green) button automatically performs the function on all Selected items.

As an example of the above, Figures 3, 4, and 5 show the effect of a Move command on the same drawing when the three different buttons are used. The small black arrow pointing NW represents the cursor.

Cursor movements with the mouse lie on an imaginary grid. When you move the mouse on the tablet or bitpad, DP moves the cursor to the closest grid point. A small grid allows you to draw fine details easily. A large grid (such as 6) makes it easier to line up items. For each drawing DP keeps several grid settings in memory in a circular list. The list can be changed, as explained in Section

11. The current grid is shown at the lower right of the screen. To switch to the next grid on the list, type g (for grid). If you switch grids while you are working on a drawing, you may experience difficulty in pinpointing items to be moved, deleted, etc.--if so, return to the grid that was current when the item was created.

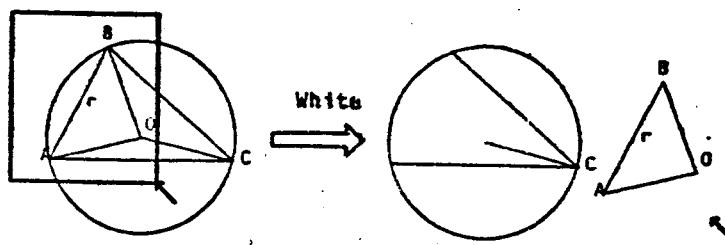


Figure 3. left (white) button: Moving the items within a rectangle

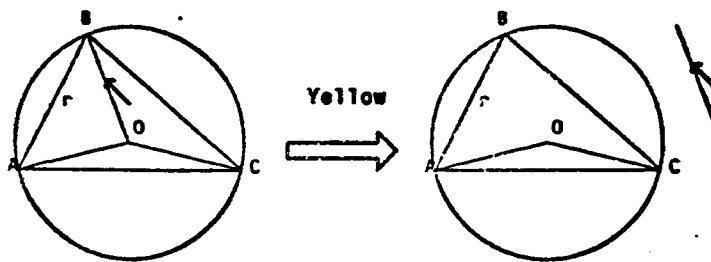


Figure 4. middle (yellow or blue) button: Moving one item at a time

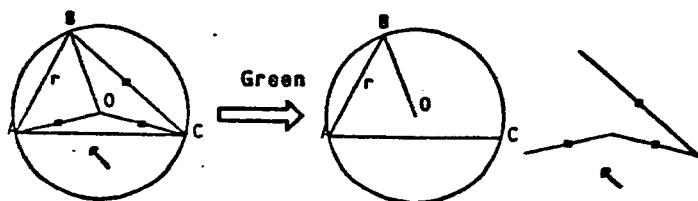


Figure 5. right (green) button: Moving the selected items

### 6.3 Issuing Keyboard Commands

All keyboard commands are single letters or special keys. You do not have to hold CTRL or any other key while you give the command. Upper and lower case command letters are considered different.

Some commands (such as Redraw) are Immediate; that is, they are immediately executed when the command is typed and then you are returned to the previous function. Non-immediate commands (such as Line and Copy) do not change the drawing without some further action from you, and after the procedure has been completed, you are left in the new function.

### 6.4 Aborting Commands

Commands that result in the display of a prompt or a pop-up menu (both explained below) can be cancelled by typing CTRL c. However, commands that are executed immediately, such as a change in function from Line to Circle, or commands involved in drawing or editing cannot be aborted. (Instead you would have to switch back to the previous function, or in the second case, edit or delete the item.)

### 6.5 Using Menus

A pop-up menu is a rectangular area which appears on the screen when requested or in response to certain commands. The menu will be centered at the location of the cursor and will contain several commands. A command on the menu can be executed by pointing to it with the cursor and pressing any button. The menu does not interfere with the underlying area--when the menu is exited, the previous contents of the area reappear.

If all of the commands do not fit on the menu, a scroll bar is shown at the bottom of the menu. To scroll, point the cursor at the bar and, while holding a button down, move the cursor left or right.

The most frequently used menu is the Top-Level menu, which gives you the option of invoking Top-Level Commands from the menu rather than typing them. To get the Top-Level menu, either press the space bar or point the cursor at a white rectangle in the border of the window and press any button.

Some menus disappear after you have made a choice, whereas others return to allow you to make another choice.

In DP any menu can be aborted with CTRL C. If a key is pressed while the menu is on the screen, the menu will be aborted and, if the key represents a command, that command will be executed.

#### 6.5.1 Changing the Top-Level Menu

You may change the Top-Level menu in several ways. You may specify that command characters not be displayed with the command description (for example, to have just "Copy" rather than "C Copy"), or you may specify that the Top-Level menu not be presented at all. These changes are accomplished with the "menu-character" and "menu-old" options in your profile, as explained in Section 15.

You may also delete commands from the Top-Level menu or change the descriptions of the commands. To do either of these, you must create a file which contains the contents of the menu as you want it to be presented. Do the following:

- 1) Create a file and type the commands, one per line, as follows:

character text-string-describing-command

For example: k Unusual Commands

You must type each command you want to appear on the menu. You may use upper or lower case letters. If you would prefer not to have the command character displayed with certain command descriptions, insert two spaces before the description. (Or use the profile option "menu-character OFF," explained in Section 15, to suppress the command characters for all commands.)

Regardless of the order in which you type the commands, DP will present the menu in alphabetical order by the command characters. Commands with no character listed will precede other commands.

- 2) Use the "commands-file" option in your profile to point to this file (see Section 15).

### 6.6 Responding to Prompts

Commands that require you to provide input use a Prompt Buffer, a small rectangular area which will appear in the middle of your screen. The buffer will contain a header which tells what type of information is required (such as "Filename:") and perhaps a default, shown either in square brackets following the header or on the next line.

To accept the default, press RETURN or the right (green) button.

If you type in a reply, note that the Prompt Buffer contains a pencil cursor (a thin line to the right of the text as you are typing or between two characters if you move it back within the string). You can use BACKSPACE to delete a character or OOPS to delete the whole line. Terminate the reply by pressing RETURN.

To abort the command without replying to the prompt, type CTRL c or position the cursor outside the Prompt Buffer and press any button.

### 6.7 Editing Commands and Text Strings

If you need to make corrections as you type a reply to a command or as you enter text for the drawing, use the following keys and mouse buttons to position the pencil cursor and to make the change:

left (white) button - instruct DP to position the pencil cursor before the character you are pointing to with the arrow cursor

middle (yellow) button - instruct DP to position the pencil cursor before the character you are pointing to with the arrow cursor and delete the character following the pencil cursor. Holding the button deletes several characters.

right (green) button - terminate the editing (same function as the RETURN key)

BACKSPACE - delete the character immediately preceding the pencil cursor

OOPS - delete all the characters preceding the pencil cursor

RETURN - terminate the editing (close the string)

After you use the arrow cursor to position the pencil cursor, you may move the arrow cursor out of the way if you wish.

### 6.8 Getting Information from the System

You can request information from DP at any time by typing ?. The following items will be displayed for the current window:

parameters;  
list of deleted items, if any;  
segment allocation table and number of free segments; and  
names of symbol definitions, if any, and number of instances of each.

### 6.9 Changing the Display

The commands listed below affect only the way the drawing is displayed on the screen; they do not change the internal representation of the items. If a command is listed below as "immediate," the command is executed as soon as you type it and then you are returned to the current function.

Redraw the screen

Type R. This is helpful if the display contains an error message you've taken care of or some other extraneous material. Immediate.

Redraw the current window only

Type r. Immediate. (This is also referred to "refreshing" the window.)

Change the scale

Type =. You will be prompted for a number. Immediate. The default ("actual size") is 1. A scale of 0.5 displays the objects half size, 0.25 quarter size, etc., and a scale of 2 doubles the size, 3 triples it, etc. A special case is a scale of 0, which returns a drawing to scale 1 and to its original position (point 0,0 at the center of the window).

It is useful to use a large scale (3 or 4) when creating or editing items and a small scale when viewing a large drawing. After you have switched to a large scale, you can bring the desired items into view with the w command described below.

Be aware that the scale on the screen may not correspond exactly to that of the printing device (see Section 9.8).

Turn the display grid on or off

Type ' (back quote). The display will be toggled to ON or OFF. Having the grid on is useful for aligning items. Immediate. (To alter the distance between the dots, see the Unusual Commands in Section 11.)

Move (scroll) the whole image in the current window

Type w. Then hold down any button and move the cursor. The image follows the cursor. Release the button when the image is where you want it. Although any button will work, the right (green) button uses a faster algorithm that requires less time to start up. You can move parts of the drawing off-screen and then bring them back, which allows you to work on parts of large drawings.

Insert a mark at the center of the current window

Press the INS key. Immediate. Each window has a circular buffer of marks that can be used to remember positions in the drawing. When you start a drawing, the buffer is empty. To insert a mark, use the w command to move the desired item to approximately the center of the screen and press INS. There is no limit to the number of marks you can insert. To use the marks, see the G command below.

Go to the next mark

Type G. Immediate. The next mark moves to the center of the window; thus this command allows you to jump between specified points in a drawing. (Marks are inserted with the INS key, described above.)

Delete the previous mark Press the DEL key. Immediate.

Display diamonds

Type a number sign (#). Diamonds will be displayed at "T" connections of two lines, at butting connections of two lines, and at connections of three or more lines. This is a way of annotating the drawing; the diamonds are not items on the drawing. They are not connected to any of the lines—if you move the lines, the diamonds stay in place. The diamonds remain until the screen is redrawn or until you leave DP. If you want diamonds to appear in a printout, use the "Diamonds (press)" option on the Unusual Commands menu explained in Section 11.

Display pins

Type P. The display of pin positions will be toggled to ON or OFF.

Display pin numbers

Type ~. The display of pin numbers will be toggled to ON or OFF.

### 6.10 Automatic Saves

Since typical DP sessions tend to be fairly long, DP automatically saves your drawing periodically. Each window has a counter associated with it, which is incremented every time a new function is entered while the window is active. After a given number on the counter, the contents of the window are written out to a "checkpoint" file and the counter is reset. This ensures that a recent copy of the drawing is always available, even if the system crashes. The checkpoint filename is formed by appending .CKP to the filename associated with the window (for example, the file "latch.dp" is checkpointed to "latch.dp.CKP").

The default for the number of commands between check points is 100. The number may be modified with the Unusual Commands menu (see Section 11).

Also you may write out the changes to the original file or to any other file at any time by typing o (for output). This is explained further in Section 10.1.

## 7. BASIC ELEMENTS

As explained previously, all drawings in DP are composed of five basic elements--lines, circles and arcs, splines, strings of text, and pins. In addition, you may create rectangles--which are not considered basic elements but merely four lines.

You may find it helpful to use a large scale (3 or 4) while you are creating elements (see Section 6.9).

### 7.1 Creating Elements

#### 7.1.1 Line

Lines are more complicated than other elements because they can be drawn with or without gravity, depending on the gravity field you set and which button you use to draw the line. All lines (as well as pins nested in symbols, circles, and splines), regardless of how they are drawn, have gravity--the ability to attract other lines. But the gravity field and buttons determine whether lines are affected by gravity as they are drawn. If the gravity field is strong (for example, 10), the endpoint of a line as it is being drawn is attracted to nearby elements unless the line is drawn with the right (green) button. The first endpoint is attracted when the button is pressed, and the second endpoint is attracted when the button is released. This attraction allows you to close corners and other intersections neatly. If the gravity field is 0 (off), none of the lines you draw will be attracted to other lines, regardless of the button used.

Section 11 explains how to set the gravity field. If you normally do not want to use gravity, put a statement gravity 0 in your profile, as explained in Section 15.

To create a line, do the following:

- 1) Type the letter l to enter Line mode.
- 2) Position the cursor where you want the line to begin; then press and hold one of the buttons:

left (white) - Generic (a line which can go in any direction)

middle (yellow or blue) - Hor/Ver (a line which DP adjusts to a true horizontal, vertical, or diagonal position--0, 45, or 90 degrees)

right (green) - No grav. (a generic line without gravity). The line will not be attracted to other elements, regardless of the current gravity field.

- 3) Move the cursor to the position where you want the line to end and release the button.

The current thickness (see Section 9.7.1) is used for the line.

If two lines are drawn exactly on top of each other, they become invisible. They can be displayed with the "Black mode" option in the Unusual Commands menu (Section 11).

### 7.1.2 Rectangle

To create a rectangle, do the following:

- 1) Type L to enter Rectangle mode (the letter L was chosen because rectangles are just a special case of lines).
- 2) Position the cursor where you want the upper left corner of the rectangle. Press any button and hold it down. As soon as you start to move the cursor, a rectangle will appear.
- 3) Move the cursor up or down to increase the length, left or right to increase the width.
- 4) Release the button to "freeze" the rectangle.

### 7.1.3 String of Text

To create an ASCII string of text, do the following:

- 1) Type the letter a to enter ASCII String mode.
- 2) Position the cursor where you want the text to begin; then press and release one of the buttons:

left (white) or middle (yellow or blue) - Single (to enter a single line of text). Terminate the string by pressing RETURN.

right (green) - Automatic (to enter a sequence of strings which will be aligned below the first string). Position the cursor where you want the first string to begin. Type each string, terminating each by pressing RETURN. The sequence must be single spaced; pressing RETURN twice terminates the action.

You may use upper and lower case characters. If you need to make changes, use the keys and buttons explained in Section 7.2.2. Strings cannot exceed 80 characters.

The current font is used for all characters in the string (see Section 9.7.2).

#### 7.1.4 Circle or Arc

To create a circle or an arc, do the following:

- 1) Type 0 (zero) to enter Circle mode.
- 2) To make a circle, position the cursor where you want the center of the circle. Press and hold the left (white) or middle (yellow or blue) button. Move the cursor away from the center of the circle to where you want the circumference; the circle will grow as you move the cursor. Release the button to freeze the circle. This is shown in the Status Line as "Center,r"--indicating a circle with the center and radius determined by pressing and releasing the button.

To make an arc, press the right (green) button at the beginning point of the arc. Going in a counterclockwise direction, press the right (green) button again at the end point; and finally press it again in the approximate area of the center of the arc. This is shown in the Status Line as "1,2,center."

The circumference of the circle or the arc is drawn in the current thickness (see Section 9.7.1).

### 7.1.5 Spline

To create a spline, do the following:

- 1) Type 9 to enter Spline mode.
- 2) Position the cursor where you want the first control point and press any button except the right (green) one. A black rectangle will temporarily mark the location of the control point. (Each control point will not necessarily be located exactly on the completed spline; a smooth curve will be calculated which will run close to each point.)
- 3) Repeat Action 2 for each control point. You must specify at least two control points; the endpoints will be the first and last control points you specify.
- 4) Press the right (green) button to create a curve connecting all of the control points.

The current thickness (see Section 9.7.1) will be used.

### 7.1.6 Pin

A pin is a connection point, used mostly within symbols on electronic circuits. Each pin has a unique number assigned to it.

Pins that are nested within symbols have gravity unless the gravity field is set to 0 (see the discussion of gravity under Lines in Section 7.1.1).

To create a pin, do the following:

- 1) Type **p** to enter Pin mode.
- 2) Position the cursor where you want the pin.
- 3) Press one of the buttons:

right (green) - to insert a pin with no prompting for the number (the program will assign the pin number, incrementing the previous pin number by 1 regardless of whether the previous number was assigned by the program or you).

any other button - to insert a pin and to be prompted for the pin number

Pin numbers may range from 0 to 4095. If you supply a larger number, you will receive an error message.

## 7.2 Editing Elements

Edit mode is used to change the shape and size of any existing basic element. (To move, copy, delete,, change the parameters, or perform other functions on elements, see Section 9.)

Regardless of the element you are editing, you must first enter Edit mode by typing e. Then you must "open" the desired item by pointing at it with the cursor and pressing any button. Circles and splines are opened by pointing at one of their endpoints; lines, strings, pins, and symbols are sensitive over the entire item. The exact procedure then depends on the type of element being edited. The descriptions of the actions associated with each button displayed in the Status Line in Edit mode are meaningful only when editing lines. As you edit, the new shape is redisplayed dynamically.

It is helpful to use a large scale (3 or 4) while you are editing elements (see Section 6.9 for instructions on changing scale).

### 7.2.1 Line or Rectangle

To edit a line or rectangle, do the following:

- 1) Type e to enter Edit mode.
- 2) Position the cursor precisely on the line, near the endpoint you want to change; then press and hold one of the buttons. (The whole line is "sensitive"--that is, you can choose the line by pointing at any part of it. However, the endpoint nearest the cursor is the end that will follow the cursor.) Each line of a rectangle is considered a separate item.

left (white) - Generic (to lengthen or shorten a line or move it in any direction)

middle (yellow or blue) - Hor/Ver (to lengthen or shorten a vertical, horizontal, or diagonal line)

right (green) - No grav. (to lengthen or shorten a line in any direction, with no effects from gravity)

- 3) Move the cursor to the desired new location of the endpoint. The line is dynamically adjusted and appears to follow the cursor in a rubberband fashion. Release the button when the endpoint is at the desired location.

### 7.2.2 String of Text

To edit one or more strings of text, do the following:

- 1) Type e to enter Edit mode.
- 2) Identify the string of text you wish to edit by positioning the arrow cursor within the string and pressing any button except right (green). A pencil cursor (a straight line) will appear within the string.
- 3) Position the pencil cursor where you want to insert or delete characters, as follows:

left (white) button - instruct DP to position the pencil cursor before the character you are pointing to with the arrow cursor

middle (yellow or blue) button - instruct DP to position the pencil cursor before the character you are pointing to with the arrow cursor and delete the character following the pencil cursor. Holding the button down deletes several characters.

- 4) To insert text, just type the text. To delete, use the following keys:

BACKSPACE - delete the character immediately preceding the pencil cursor

OOPS - delete all the characters preceding the pencil cursor

- 5) Terminate the editing by pressing RETURN or the right (green) button.

To edit another string, repeat the procedure.

Note that a string cannot be entirely deleted this way. If you delete all of the characters, the original string returns when you press RETURN or the right (green) button. To delete an entire string, use the procedure explained in Section 9.6.

Multi-line strings must be edited one line at a time.

### 7.2.3 Circle or Arc

To edit a circle or an arc, do the following:

- 1) Type e to enter Edit mode.
- 2) Press one of the buttons as follows:

middle (yellow or blue) button - to change the radius. Open the circle or arc by moving the cursor near one of the endpoints and then pressing and holding the button. (The endpoints of a circle are located at the rightmost point of the circumference.) Move the cursor to the desired new location of the circumference and release the button.

left (white) or right (green) button - to move the endpoint, changing the subtended angle but keeping the radius constant. (A circle can be shortened to an arc in this manner.) Open the circle or arc as explained above; then move the cursor to the desired location for the endpoint and release the button.

The endpoints must be pinpointed precisely in order for these procedures to work.

If the order of the two endpoints is swapped, the circle changes abruptly, for example, from a short arc to an almost full circle or vice versa. If the circle is embedded in a symbol and that symbol is rotated or mirrored, the endpoints of the circle are also transformed. Therefore, when the symbol is unpacked (see Section 8.5), the endpoints of a full circle may no longer be at (Radius,0); they may be at (0,Radius).

#### 7.2.4 Spline

To edit a spline, do the following:

- 1) Type e to enter Edit mode.
- 2) Position the cursor over one of the endpoints and press any button except right (green). The control points will become visible (displayed as black rectangles).
- 3) Position the cursor over the endpoint you want to move; then press and hold the left (white) or middle (yellow or blue) button. Move the cursor to the desired location for the point; release the button.
- 4) Repeat #3 for any other points you want to move.
- 5) Press the right (green) button to close the spline and erase the control points.

#### 7.2.5 Pin

Editing allows you to change the position of a pin number, as follows:

- 1) Pin Numbers Display should be ON. (If not, invoke the Unusual Commands menu explained in Section 11 and change the display setting.)
- 2) Type e to enter Edit mode.
- 3) Position the cursor precisely over the pin itself (the two small lines). Press and hold any button.
- 4) Move the cursor to the desired location for the pin number (it can be located in any of the four quadrants around the pin); release the button.

The display position of new pins is computed automatically when they are used in a symbol.

## 8. SETS, SYMBOLS, AND INSTANCES

Basic elements can be combined either into a set or into symbols. A set is a group of items (basic elements or symbols) that you temporarily want to handle as a unit so that you can perform the same function on all of the items with one command. A set is formed by using the Select function to designate the desired items. There can be only one set per window. The set will be disbanded when you leave DP (or sooner, if you choose). A symbol is one or more items to which you assign a name and which will be treated as a unit permanently. A symbol consists of a definition (that is, a template) and one or more instances (occurrences) of that symbol. A symbol definition is created with the Pack function.

The advantages of a symbol over a set are:

- a) If the symbol is composed of several items, the group will not be broken up when you leave DP.
- b) Because a symbol is treated as a unit, it can be moved, etc., without affecting intersecting items.
- c) A symbol can be copied very easily within the same drawing or from another drawing on your screen.
- d) The symbol is identified with a name and therefore you can copy a symbol from another drawing without bringing up the whole drawing on your screen.
- e) Because symbols can contain other symbols, drawings can be created in a hierarchical manner.

To make maximum use of DP's power, make into a symbol any item or group of items that is repeated on the same drawing or on more than one drawing. An instance of a symbol can be transformed for parameters such as rotation and scale, and an instance can also be broken down into its components, edited, and defined as a new symbol. Therefore it is useful to make a symbol even when creating items that are similar.

### 8.1 Forming Sets (Selecting)

A set is a unit consisting of all the "selected" items in a window. Combining basic elements into sets is useful for tasks in which you want to perform the same operation on more than one item, such as deleting several items or moving a group of items that are located near each other. Items that are "selected" as explained below can be treated as a group for the functions transform, pack, unpack, move, copy, stretch, delete, and set new parameters. However, remember that sets are temporary designations. If the items should always be handled as a unit, make them into a symbol.

To form a set, do the following:

- 1) Enter Select mode and identify the items to be included by doing one of the following:

- a) To select all of the items on the current window, type S. (See the note below.)
  - b) To select certain items, type s.

- 2) If you typed s, press one of the buttons:

left (white) - Rectangle (to identify the items by enclosing them in a rectangle)

middle (yellow or blue) - One item (to choose the items for the set, one at a time)

right (green) - Selected (to select all of the items on the current window). This action is the same as typing S.

Strings of text that have been selected are inverted on the display (that is, if your screen background is white, the selected string will be displayed as white text on a black background). Other items which have been selected are displayed with a small square near the center of the visible portion of the item.

Items remain selected until you exit DP or deselect them. To deselect all of the items on the current window, type Z. To deselect certain items, type z and then press the appropriate button. The buttons perform as they do in Select mode.

Note: If you want to select a large number of items, it is faster to select all items (s) and then deselect certain ones (z), than it is to select the items one at a time.

## 8.2 Forming Symbols (Packing)

A symbol is a unit containing one or more basic elements or other symbols. The symbol consists of a definition (that is, a template that specifies how to draw a graphical entity); a unique name assigned by you or DP; and one or more instances (occurrences) of the symbol. If all instances are deleted, the definition is deleted.

You cannot tell from looking at a drawing which items are symbols. To get a list of the symbol names and the number of times each appears on the current drawing, type ?. Section 8.3 tells how to find out the name of a particular symbol or to change a name.

### 8.2.1 Symbol Definition

Creating a new symbol definition from a set of items is called "packing" a symbol.

To pack a symbol, do the following:

- 1) Draw the components of the symbol either in a location where you want it to appear or any place (you can move it later).

When you are creating a symbol, we recommend a mouse grid of 3 or 6; a high scale (3 or 4), especially when drawing lines for a shape; and that very short lines be drawn with gravity off. (Mouse-grid and Gravity are Unusual Commands, explained in Section 11. Scale is a display command, explained in Section 6.9.)

You will find it useful (especially for circuit schematics) to copy an existing symbol that is similar, unpack it, edit the components, and then repack it into a new symbol. (When creating a circuit, copy the shape and then add the strings and pins using a scale of 1.)

- 2) Type b.
- 3) Press one of the buttons:

left (white) - Rectangle (to identify all of the items to be included by enclosing them in a rectangle)

middle (yellow or blue) - One item (to make a symbol with a single item)

right (green) - Selected (to include all of the items that are Selected on the current drawing)

- 4) Reply to DP's prompt for a symbol name. You may give the symbol a name or you can let the system generate a name. The advantage of giving a name is that you can then use that name to retrieve the symbol.

To name a symbol, type the name (up to 80 characters). If you type the name of an existing symbol, DP asks whether the old definition should be deleted. If you reply no, DP will again prompt for a name. If you reply yes, the old symbol is redefined, as explained in Section 8.6.

To have DP generate a name, press RETURN. DP will assign a name such as \$\$12:28:32 (denoting the time that the symbol was created). Symbols with automatic names should be used inside other symbols. In an integrated circuit symbol, for example, "pins" are usually complex items consisting of a pin, a string (the visible pin name), and possibly other items. Automatic names are very handy for symbols like this.

To change a symbol name, see Section 8.3. To change a symbol definition, see Section 8.6.

You cannot explicitly delete a symbol definition. It is automatically deleted when all instances of the symbol have been deleted.

### 8.2.2 Symbol Instances

You may use a symbol as many times as you wish, by creating instances of it. To create an instance, copy the symbol as explained in Section 9.2. Because the Copy command works across windows, you may copy a symbol from any drawing on the screen.

If you wish to copy a symbol from another drawing without bringing the whole drawing to the screen, use the i (input symbol) command explained in Section 10.4.

### 8.3 Displaying or Changing a Symbol Name

To display or change the name of a symbol, do the following:

- 1) Type e to enter Edit Mode.
- 2) Point to the symbol and press one of the buttons:
  - right (green) - to change the name
  - any other button - to display the name  
(in the lower left corner of the screen)
- 3) If you pressed the right (green) button, DP will display the current name in the prompt area in the center of the screen. Edit the name as desired.

If you change the name, all of the instances of that symbol will be renamed.

#### 8.4 Transforming Instances

Symbol instances may be rotated, mirrored, or changed to a new scale without affecting the original symbol definition. Do the following:

- 1) Type **t** to enter Transformation mode. A pop-up menu will appear with the following items:

Rotate 90

Rotate counter-clockwise

Rotate clockwise

Mirror X(-) (mirror the item around the X axis)

Mirror Y(+) (mirror the item around the Y axis)

Scale (scale an item along both axes, relative to the current size. For example, if you specify 0.5, pointing to an item (#4 below) will reduce that item in half. Pointing to it again will reduce it in half again.)

Scale X (same as Scale, except it works only on the horizontal axis. To scale along the Y axis only, rotate the item, scale X, and rotate the item back to its original position.)

Strip transformations (remove previous transformations and return a symbol to its original parameters)

Read parameters (display in the lower left of the window the current parameters on chosen symbols)

- 2) Move the cursor to the desired transformation and press any button.
- 3) Reply to DP's prompt for information, if any (such as angle of rotation).
- 4) Indicate the instance(s) to which the transformation is to be applied, by pressing a button as follows:

left (white) - Rectangle (to identify the instances by enclosing them in a rectangle)

middle (yellow or blue) - One item (to identify one instance at a time)

right (green) - Selected (to transform all the instances that have been selected on the current window)

### 8.5 Unpacking an Instance

Unpacking an instance of a symbol refers to breaking it up into its components. The item will no longer exist as an instance; in its place will be the elements and nested symbols that made up the symbol. The symbol definition is not affected (unless this was the only instance, in which case the definition is deleted).

To unpack a symbol instance, type B and then identify the instance you wish to unpack by pressing the appropriate button: left (white) - Rectangle; middle (yellow or blue) - One item; right (green) - Selected.

To unpack a nested symbol, repeat the procedure at each level of nesting.

### 8.6 Redefining a Symbol

To redefine a symbol, do the following:

- 1) Draw the symbol with its new configuration.
- 2) Pack the new symbol definition (as explained in Steps 2 and 3 in Section 8.2.1, type b and then use the buttons to identify the elements to be included in the symbol).
- 3) When DP prompts for the symbol name, type the old name. DP will warn that the name is in use and ask if you want to delete the old definition. Reply yes.

All instances on the current drawing and on any other drawing on the screen are changed to the new definition. However, other drawings that contain instances of the old symbol, but which are not on the screen, are not affected.

## 9. MANIPULATING ITEMS

The following sections describe all the operations that can be performed on elements, sets, or symbol instances.

In all cases, the selected set or a symbol instance is treated as one item. Items on write-protected layers are not affected by any of the commands.

### 9.1 Moving

You may move one or more items within a window or from one window to another. (Move and Copy are the only commands that work across window boundaries.)

To move one or more items, do the following:

- 1) Type m (for Move).
- 2) Position the cursor on the item to be moved; press and hold one of the buttons:
  - left (white) - Rectangle (to enclose in a rectangle and move a group of items)
  - middle (yellow or blue) - One item (to move one item at a time)
  - right (green) - Selected (to move as a group all the selected items)
- 3) Move the cursor to the desired new location for the item(s) and release the button.

### 9.2 Copying

You may copy one or more items within a window or from one window to another. The procedure is the same as that explained above for the Move function, except in Step 1 type c for Copy.

### 9.3 Stretching

The Stretch function (also called the Sticky Move function) resembles the Move function, but it is especially adapted for circuit design--lines, strings of text, and symbols can be moved along with connecting lines. Strings and symbols can be moved in any direction; vertical lines can be moved only horizontally and horizontal lines can be moved only vertically. When vertical lines are moved, connecting lines are stretched to the new location.

A line is considered to be connected to a symbol if the line ends exactly over a pin of the symbol. If possible, all existing connections are preserved and no new connections are created.

To stretch an item, do the following:

- 1) Type x.
- 2) Use one of the buttons to identify the item(s) to be stretched:
  - left (white) - Rectangle (enclose the items to be stretched in a rectangle)
  - middle (yellow or blue) - One item (to stretch one item at a time)
  - right (green) - Selected (to move all Selected items in the direction of the cursor movement; however, connecting lines are not stretched)
- 3) Move the cursor to the desired new location and release the button.

#### 9.4 Rotating

Symbol instances can be rotated in Transformation mode, as explained in Section 8.4. Strings of text can be rotated as they are typed or afterward, by specifying the font as one that contains the desired rotation angle. Details are given in Sections 9.7.2 and 9.7.3.

#### 9.5 Magnifying

The Magnify command allows you to zoom up and down at a small section of the screen (useful for testing connectivity in circuits). To use this function, type M. A magnifying glass will appear, which can be moved by moving the mouse (the glass temporarily replaces the cursor).

Pressing the left (white) button increases magnification, and pressing the right (green) button decreases it. Pressing the HELP key displays a pop-up window with help for the M command.

To remove the glass, type CTRL c or press any key.

## 9.6 Deleting and Undeleting

To erase one or more items, do the following:

- 1) Type d to enter Delete mode.
- 2) Press one of the buttons, as follows:

left (white) - Rectangle (to delete a group of items identified by forming a rectangle around them)

middle (yellow or blue) - One item (to delete one item at a time) A circle or arc can be deleted by pointing to the line or any area inside. If you delete within overlapping items, all of those items are deleted.

right (green) - Selected (to delete all Selected items in the current window)

Typing D results in the same action as pressing the right (green) button.

If you have trouble deleting an item, it may be because you are not using the same mouse grid used to create the item. Type the g command to switch grids, as explained in Section 6.2.

As a safety feature in case you unintentionally delete one or more elements, deleted items are not physically removed from memory until you re-enter Delete mode. (The items are deleted immediately upon re-entry, even before you delete any additional items.) To see what deleted items are being held, type ? Each item will be listed with its coordinates and name (circle, line, etc.).

To undelete an item, type u (for undelete). Deleted items are returned to the drawing in their original positions and are selected. The fact that they are selected is useful in case you want to keep only one of them--deselect the desired item; then delete the selected items.

If you want to intentionally erase the latest deleted items to save memory space, type another d (to in effect re-enter Delete mode) immediately after you have finished deleting. However, remember that if you do this you can no longer retrieve those items.

## 9.7 Specifying Parameters

You can control the following parameters for each item:

thickness (of lines, circles, and splines) - see  
Section 9.7.1 below

font (for strings of text) - see Section 9.7.2 below

the layer on which the item appears - see Section 13

color (applies to all items except symbol instances).  
In a black and white system, this parameter is  
meaningless (a default of 1 is used). However, the  
program has allowed for systems that use color.

As explained in the following sections, you can change the default  
thickness or font before you create a new item and you can change  
any of the parameters on existing items.

To see what parameters have been used on an existing item, type t  
to enter Transformation mode, as explained in Section 8.4.

### 9.7.1 Thickness

To change the current thickness (the one in effect as you create  
new items), do the following:

- 1) Type a minus sign (-).
- 2) Type a number in the range 1 to 7 (1 is a very thin  
line, and 7 is very thick). If you type a number  
higher than 7, No. 7 thickness will be used.

The new value will be used for all new lines, splines, and circles.

To change the thickness of existing items, see Section 9.7.3.

### 9.7.2 Font

DP supports multiple fonts and maintains a menu of fonts for each drawing. The font specification is device-independent (it describes an "abstract" font such as Helvetica8 or TimesRoman12, which is approximated as closely as possible on output devices). See your support personnel to find out what fonts are available on your printer and plotter and to get the font files on your PERQ.

For each font, the description (which appears on the font menu) consists of five items:

- family** - the ASCII name of the family. A family is a set of fonts with similar characteristics, such as Helvetica, Gacha, TimesRoman, etc.
- size** - an integer indicating the size of the font, usually in the range 6 to 16 (small numbers mean small fonts)
- face** - one or two characters that describe the particular face used. Valid characters are r (Roman, the standard face), b (boldface), i (italics), and bi (boldface italics).
- rotation** - an integer indicating the rotation of the font, expressed in degrees from the positive X axis (0 for standard fonts like the printing on this page, 90 for fonts running vertically up the page). In converting the rotation, DP may round off the number (for example, to 89.999992370605).
- PERQ font** - the name of the file containing the font that will be used to display the string on the screen. Such files usually have the extension .KST. If the font file is not in the directory in which you will be using DP, you must add the directory containing the font to SETSEARCH before you enter DP.

The PERQ font does not have to match the above specifications which will be used when the drawing is printed. You can use

the same PERQ font for more than one DP font; although the items will appear on your screen in the same font, the specified font for each item will be used when printed on a plotter.

For each output device, the best approximation of the specified font will be used.

If you want to rotate a string of text 90 degrees, you must have two fonts of the same family and size. One should have a rotation of 0 degrees and the other 90 degrees.

To change the current font (the one in effect as you create new strings), to enter a new font, or to delete an existing font, do the following:

- 1) Type f (for font). A menu will be displayed, showing all of the active fonts plus options to add or delete fonts.
- 2) Point the cursor to the desired font or option and press any button.
- 3) If you selected a font, no further action is necessary. All new strings you create will be in the selected font.

If you chose to delete a font, another menu appears showing the current fonts. Position the cursor over the font and press any button. An error message is generated if the font is still in use.

If you chose to add a font, a prompt buffer appears with a default font which you may edit. You must specify the five pieces of information shown above. In some cases you can define a new font merely by specifying a different size, face, and rotation and using the same family and PERQ font file used for an existing font; but in some cases you must specify a different font file which precisely defines the font you are adding. When you add a new font, it is automatically chosen as the current font. If you specify a PERQ font that doesn't exist, DP will not accept it. DP will list a font that is on file, and you must press CTRL c and start over.

Section 10.5 explains a method for accessing from within DP a file

containing the names of font files. If all of your font files are in one directory and if all have the extension .KST, you can create such a file at the command level by pathing to the appropriate directory and typing dir \*.kst <filename>.

### 9.7.3 Setting New Parameters on Existing Items

Changing parameters on existing items is a three-step process. First enter the New Parameters mode by typing n. A menu appears with two items for each of the four parameters (thickness, font, layer, color).

The second step is to specify a new parameter. The first block in each set on the menu is the last value specified in New Parameters (it is not the current parameter). Thus DP keeps a parameter in memory, which is useful if you are switching between two parameters frequently. Pressing any of the first blocks sets the parameter as shown. If you wish to change the parameter, select the second block of the set and DP will prompt for a new value. After you reply, the prompt area will disappear.

After you have set a new parameter, the third step is to indicate which items you wish to change to the new parameter. Point the cursor to the item; then press one of the buttons:

left (white) - Rectangle (to identify the items to be changed by enclosing them in a rectangle)

middle (yellow or blue) - One item (to choose one item at a time)

right (green) - Selected (to change all the selected items on the current drawing)

The New Parameters mode does not affect the current parameters. That is, if the current thickness is 2 and you enter New Parameters and change some lines to a thickness of 6, new lines which are drawn subsequently will have a thickness of 2.

Note: To find out the current parameters on any item, type t to enter Transformation mode, as explained in Section 8.4.

### 9.8 Printing

If you are connected to a PERQ Systems PLP-10 laser printer, you may request a copy of the drawing as it appears on your screen, without leaving DP. Your dp.commands file, which comes with the software, contains one line stating the DP version number (example: \*+DP 5.26). The line will contain an asterisk (\*) if you are connected to the laser printer. If a plus sign (+) follows the asterisk, a time stamp will be printed at the bottom of each page. You may edit the file to add or remove the "+" as you desire.

To get a copy of the screen, do the following:

- 1) Adjust the scale and position of the drawing as desired with the z and w commands (Section 6.9).
- 2) Type H (Hard-copy).
- 3) DP will ask if you want a screen dump; reply yes. A black bar will run over the drawing, indicating that a "screendump" (a copy of the screen) has been made.

Text strings will be printed in the font shown on the screen. The border of the window, the status line, and other DP information will not be printed.

If you want the drawing reproduced on a plotter, use the "Send to plotter" command (see Unusual Commands in Section 11).

If you were to request a copy of the drawing file without entering DP, the copy would not show your drawing as it appears on the screen when you enter DP. The file contains instructions to DP and becomes a "drawing" only when you enter DP.

The scale of the drawing on the screen may not correspond exactly to the scale of the copy produced on the printing device. You may wish to develop a ruler or grid which shows the relationship and incorporate it as a layer on each drawing.

## 10. INPUT AND OUTPUT FILES

You can write out all of the items to a DP file which can be used in subsequent DP sessions. You can read in another drawing in its entirety, a symbol definition or text from another drawing, or commands contained in a separate file.

When a file is being read or written, a message appears at the lower left corner of the screen, telling what action is being taken.

### 10.1 Writing to a DP File

A DP file stores the drawing along with instructions to DP so that the drawing can be edited in subsequent DP sessions. As you are working, DP automatically saves the drawing periodically in a checkpoint file (which is a DP file), but if you wish, you may write to the original file (or any other file) at any time. Before you leave DP, you should write the drawing to a DP file of your choice if you wish to keep any items created and any changes made in the current session.

To write all of the items in the current window to a file, do the following:

- 1) Type o (for output).
- 2) Either type a filename (the program will assign the extension .DP) or press RETURN to use the default name shown in brackets. (Include the pathname if you do not want to put the file in the current directory.)

If a file of that name already exists (a previous version of the drawing), the previous file will be saved with a \$ appended to the name. This follows the usual practice on a PERQ of making a backup copy before overwriting a file.

### 10.2 Looking for Files

In order to find the names of files you wish to copy, you can have DP give you a list of the .DR and .DP files in certain directories. When you type the i or I command to read in a symbol or drawing, reply to the prompt for a filename by typing a directory name followed by >.

You may use your profile to specify which directories to search and also to have displayed a list of symbols that contain the filename. Then when you type i or I, as explained in the following sections, you will receive a pop-up menu of filenames and symbols from which to choose.

In either case, typing any character will abort the menu and redisplay the prompt for a filename.

Your profile also can specify whether the mouse grids on the incoming drawing will be brought in and, if so, whether they'll replace the current grids.

### 10.3 Reading in Another Drawing in Its Entirety

To read in another drawing file and merge it with the items in the current window, do the following:

- 1) Type I (for input).
- 2) If you have not set your profile to search for files, type in either the name of the existing file or a directory (example: :user>swaney>). If you type a filename and the file does not exist, DP will generate an error message. If you type a directory name, you will receive a popup menu listing all of the .DP and .DR files in that directory.

If you have set the "input-path" command in your profile to search for directories, DP will display a pop-up menu listing all of the files in that directory with a .DR or .DP extension. If you want to look at one of the subdirectories, select it and DP will again give a list of .DR and .DP files. If there are no .DR or .DP files, you will receive a message <NO FILES>.

Depending on which "Center files" option you've chosen (see Section 11), the items from the other drawing will be either centered in the window (intermingled with the items, if any, on the current drawing) or entered in the same position they occupied on the original drawing. As explained in Section 15, your profile can determine whether the mouse grids from the other drawing are imported.

If the incoming drawing contains a symbol with the same name as a symbol on the current drawing, the symbol on the current drawing will be either replaced or simply renamed, depending on the "rename-symbols" commands in your profile (see Section 15) and the "rename redefined" command on the Unusual Commands menu (see Section 11).

#### 10.4 Reading in a Symbol from Another Drawing

To create an instance of a symbol which has been defined in another drawing, do the following:

- 1) Position the cursor where you want the instance to be located.
- 2) Type i (for input symbol).
- 3) Reply to the prompt for the symbol name. (If the name is the same as the name of a symbol in the current drawing, you will receive an error message.)
- 4) Reply to the prompt for a filename by typing either a filename or a directory. DP supplies as a default filename the name of the symbol, which you can edit (helpful if you have given symbols and files similar names).
  - a) If you type a filename and the file does not exist or the symbol is not defined in that file, you will receive an error message. If the symbol is found, an instance of that symbol will be created, centered around the cursor position.
  - b) If you type a directory name, you will receive a pop-up menu listing all of the .DP and .DR files in that directory; choose one.

If you have set the "symbol-inputs" command in your profile to search for files, a menu will appear listing all .DP and .DR files in the specified directories. If you choose a .DP file, another menu appears displaying the symbols, if any, in that file whose names begin with the filename chosen (for example, parts S240, S240R, and S240L in file S240). Choose the desired symbol. An instance will be created, centered around the cursor position.

### 10.5 Reading in Text from A File

Sometimes it is useful to be able to read in a text file (usually to a window which you have opened for just that purpose). For example, you might want to look up the specifications for a design, or you might want to look at a file you have created previously which contains the names of all of the font files on your PERQ. You might also want to read in text to be included as part of the drawing you're working on.

To read in text from a file and have it displayed as ASCII strings in the current file, do the following:

- 1) Position the cursor where you want the text to begin.
- 2) Type j.
- 3) Type the filename, including the extension.

The text will begin at the cursor position, in the current font. Each line of the file will be considered a separate text string.

This procedure can be very useful if you are using a post-processor, as it will allow you to examine the output of the processor with the drawing from which the output was obtained.

### 10.6 Reading in Commands

You can store commands to set various options in a DP profile file, as explained in Section 15. The commands will take effect automatically when you enter DP.

You can store any of the commands that set options, as well as commands to actually create a drawing, by creating a transcript of a DP session as explained in Section 12. To read in the transcript, type @ and then reply to DP's prompt for the filename. DP will take control, executing the commands contained in the file.

## 11. UNUSUAL COMMANDS

A group of commands which most users will need infrequently have been designated as Unusual Commands and combined in a special pop-up menu. To use these commands, do the following:

- 1) Type k to obtain the menu.
- 2) Choose the desired option by pointing the cursor at it and pressing any button. (The options are described below.)
- 3) Reply to DP's prompt for information, if any. If information is required, a default (the previous value) is shown. The default can be selected by pressing RETURN or it can be edited. If the command requires a Yes or No argument, pressing any button will reverse the argument.
- 4) To remove the menu, either move the cursor outside the menu and press any button or press any key.

Many of these commands with the desired values can be placed in your profile (see Section 15).

Here are the options that are available:

**checkpoint**

Specify the number of commands between automatic saves. After the specified number of commands has been typed to a window, the contents are written to a checkpoint file.

**display grid**

Alter the distance between dots in the Display Grid, simulating graph paper. This is useful for aligning items. Grid 2 provides dots that are extremely close together; Grid 200 provides dots far apart. In Grid 1 the dots are so close that the result is a black background. (The grid is made visible or invisible with the \_ command.)

|                     |  |
|---------------------|--|
| mouse grids         | Add, delete, or change grids on the circular list of mouse grids. The use of mouse grids is discussed in Section 6.2. If you choose to add a grid to the list, DP will prompt for the grid after which the new grid should be placed. Therefore you can arrange the circular list to your liking.  |
| gravity             | Specify the size of the gravity field, in screen units. Smaller fields have weaker gravity; a value of 0 means that gravity is not used. A setting of 10 is strong. On very crowded drawings, 0 gravity greatly increases the speed with which things are handled.   |
| default extension   | If YES, the extension .DP is automatically affixed to filenames being written in or out. If NO, you must provide the extension if any.   |
| center files        | Specify whether items read in from another drawing will be centered on the current drawing (YES) or located in the same position they occupy on the other drawing (NO).  |
| rename redefined    | If YES, a symbol in the current drawing is renamed if a symbol with the same name is read in with the <u>I</u> command (a \$ is appended to the symbol name on the current drawing). If NO, the symbol on the current drawing is replaced by the incoming symbol. (This also can be set with the "rename-symbols" command in your profile, as explained in Section 15.) Note: This does not apply to the <u>i</u> command because that command will not permit you to read in a symbol with the same name as an existing symbol. |
| toggle cursor shape | Toggle the direction of the cursor between northwest and northeast.  |

|                       |   |
|-----------------------|---|
| display in black mode | Display in black those items that are normally inverted (such as the intersection where lines overlap or two lines drawn on top of each other). When you give this command, only existing items are affected, not the items drawn subsequently. |
| display diamonds      | Specify that a black diamond be displayed on the screen at the "T" connections of two lines, butting connections of two lines, and connections of three or more lines. This command works exactly like the # command discussed in Section 6.9.  |
| pin numbers           | Enable or disable the displaying of pin numbers. The screen is redrawn when this command is given.  |
| pins                  | Enable or disable the displaying of pins. The screen is also redrawn when this command is given.  |
| send to plotter       | Print the drawing on a plotter (requires that your PERQ be connected to a plotter)  |
| clip (Press)          | If YES, a print (press) from the plotter will include only the portion of the drawing that is visible in the window (it is clipped--cut off--at the window boundaries). If NO, the whole drawing is printed.                                    |
| diamonds (Press)      | If YES, diamonds will be printed when you print (press) the drawing (whether or not the diamonds are displayed on the screen).  |

## read grids

If YES, when a drawing is read in with the I command, the current mouse grids list is kept and all of the grids in the incoming file are added. If NO, grids are not brought in with the incoming drawing. See the note below.

## replace grids

If YES, when a drawing is read in with the I command, the current mouse grids list will be discarded and replaced with the grids list on the incoming file. If NO, grids are not brought in with the incoming drawing unless "read grids" above is YES.

NOTE: "Read grids" and "Replace grids" cannot both be YES. However, both can be NO, in which case no grids are read in with the incoming file.

## 12. TRANSCRIPTS

If you would like to create a transcript of a DP session (to use, for example, in a demonstration), within DP type \$. Execute the desired commands and then close the file by typing another \$. All of the commands you type between opening and closing the file will be saved in a file called DpScript.

To play back the file, enter DP and type @; DP will prompt for the filename. After you type the filename, DP will execute all of the commands in that file.

A transcript may be stopped during replay by pressing and holding any mouse button; release the button to resume the replay.

You may specify a few seconds delay between commands. When you type the filename, follow it with a number from 1 to 32767. The default is 0 (no delay).

If you want to keep more than one transcript on file, rename the transcript after you leave DP. Otherwise, it will be written over the next time you create a transcript.

Transcripts are not supported in all versions of DP.

### 13. LAYERS

In DP a drawing may be divided into layers, providing the same effect as multiple transparencies. Each layer is independent and can be made visible or invisible. A layer can also be set to read-only, so that changes are not made to it accidentally while you are working on another layer. You can control whether or not a layer appears in output files.

The layer mechanism is especially useful for complex drawings that contain logically separated parts. For example, when one is working on a printed circuit board, only one layer at a time is usually worked on, but it is essential that all of the layers be visible. Another common use of layers is to have a layer which contains only construction lines which can be made visible as needed. Construction lines may be kept even after the drawing is finished so that they can be reused if modifications to the drawing are made later. The layer mechanism is also used by circuit post-processors to quickly discard useless information, such as items in a "Comment" layer.

Invisible layers result in faster operations, since various algorithms that search DP items (such as the Gravity algorithm and the One-item-at-a-time algorithm associated with the middle or yellow button) scan only visible items.

A symbol can be made from items on different layers. Layers and symbols interact in the following way:

- If the symbol is on an invisible layer, all of the items that make up the symbol are invisible (even if some of them are on visible layers).
- If the symbol is on a visible layer, the visibility of the parts depends on the layer on which each is located.

The following sections explain how to change the settings on existing layers, how to switch from one layer to another, and how to create, delete, or rename a layer.

### 13.1 Changing Settings or Changing to Another Layer

To change the settings on existing layers or to switch to working on a different layer, do the following:

- 1) Type **y** to get the Layers menu. The top portion of the menu displays all the layers and the settings of the parameters for each. If the box is black, the parameter is ON. The parameters are:

Display - If ON, the items in the layer are visible. If OFF, the items in the layer are invisible, do not have gravity, and are write-protected. (Changing "Display" changes "Alter" but not vice versa. This means that items not displayed can't be changed, of course, but displayed items can be write-protected.)

Alter - If ON, the items are affected by DP operations. If OFF, the items are write-protected.

Output - If ON, items are written to a file when the **o** command is given. If OFF, you will receive a message that some items have not been written out.

Current - If ON, this is the layer that will be current when you leave the menu.

- 2) To change a parameter of any level, point to the appropriate box in the menu and press any button. The parameter will be switched from its previous setting.
- 3) To change the current layer (the layer into which new items will be placed), point to the "Current" rectangle of the desired layer and press any button.

Like regular pop-up menus, the Layer menu can be terminated by positioning the cursor outside the menu and pressing any button.

### 13.2 Creating, Deleting, or Renaming a Layer

To create a new layer, rename an existing layer, or delete a layer, do the following:

- 1) Type **y** to get the Layers menu.
- 2) Point the cursor to the desired option at the bottom of the menu (New Layer, Rename, Delete) and press any button.
- 3) To create a new layer, reply to the prompt by typing a name (up to 80 characters). (All layers must have unique names; if you specify the name of an existing layer, you will receive an error message.)

To rename a layer, choose the layer from the next menu DP provides; then reply to the prompt for a new name. (The default name for the first layer created in a drawing is "Standard," which you may change if you wish.)

To delete a layer, reply to DP's prompt by typing the layer's name.

#### 14. MULTIPLE WINDOWS

You may work on several drawings at a time by opening a window for each one. You may also find it useful occasionally to open a "scratch" window on which to try out things which can be moved to the "good" drawing if desired.

To switch from working in one window to another window, just move the cursor. The only commands that work across windows are Move and Copy.

A region of the border near the lower left corner is used to display the filename associated with each window. If no filename is given when DP is invoked or when the window is opened, DP assigns a default name.

Windows are opened, closed, and manipulated by positioning the cursor over the border of the window in a particular spot and pressing any button. Different areas of the border have different functions, as listed below and shown on Figure 2:

Change the shape or size of the window - square at upper right corner (icon showing a reduced window within the screen). The corner follows the cursor until you release the button.

Create a new window, using one half of the area of the current window - square at lower left corner (icon showing two windows). The current window is divided in half, and the new window appears above the current window. The new window is empty initially. The original window is unaffected but, because it is smaller now, not all of the items may be visible (use the w command to scroll the window or the = command to change the scale). The maximum number of times you can divide a window is determined by the amount of usable space that is left. Four windows of equal size is the practical maximum.

Delete the window - square at upper left corner (icon showing an X). The space that had been occupied by the window will be empty after you delete the window. In order to delete a window, it must be empty (type SDD if you have not made any changes you wish to keep). The last window cannot be deleted; you must exit DP in the regular manner by typing q.

Move the window in the screen, without changing its

size or shape - gray border. This command works only when you have two or more windows; with one full-size window, there is no room for movement.

There is only one status line regardless of how many windows you have opened. Changing the current function (from circle to line, for example) changes the current function for all windows.

Many commands cannot be invoked if the cursor is outside a window. In that case, the Status Line will not show the new command and the PERQ will "beep."

Changes made in a drawing are not written to a file until you give the o command (see Section 10.1). If you bring in a file but do not make any changes you wish to keep, you can remove the drawing from your screen if you'd like by a) typing SDD to select, delete, and erase from memory all items and then b) deleting the window itself, as explained above. The original file will remain unchanged.

You cannot work on different parts of the same drawing by opening multiple windows. Although a drawing can be displayed in more than one window, only the changes made in one window will be kept.

## 15. PROFILE FILES

A DP profile file contains defaults which are put into effect each time you enter DP. You may override the defaults after you are within DP if you wish. Adapting a profile to the tasks you perform most often will greatly enhance your efficiency.

To set up a DP profile, do the following:

- 1) In your >default.profile file, make an entry in either of these forms:

```
#DP  
[path]<profile-name>  
#END
```

or

```
#DP [path]<profile-name>
```

The DP profile file can have any name and extension.

- 2) Create the DP profile file and type the commands as explained below.

As an alternative to Step 1 above, you can access the DP profile when you enter DP, as explained in Section 3. (You might want to do this if your profile contains options that you want to be in effect for only certain tasks.)

Throughout this manual "profile" refers to the DP profile file.

The profile file is a text file. Case is ignored, and one or more blanks may be used to separate items on a line. Comments may be inserted by beginning the line with an exclamation point (!); everything after ! is discarded. The commands may be entered in any order.

The general format of a line is a keyword (which must be spelled out completely), usually followed by one or more values. Options that have a Boolean value are listed as ON or OFF (ON, of course, corresponds to true).

You may specify that more than one window be opened, by including begin-window and end-window statements. Between those statements you may insert only certain commands that pertain to windows, as explained in Section 15.2.

If errors are encountered in the profile, DP will print a warning and ask for permission to continue.

New values override old ones. Therefore if you put a command into the profile more than once, the last value is retained until you change it within DP.

The following sections give the keyword, description of the values, and an example for each command which may be placed in the profile. For your convenience, unless stated otherwise the example shows the default value that DP will use if you do not put the command in your profile. Of course, if that is the option you want you do not need to put the statement in your profile. Although ON and OFF are capitalized in the following sections, you do not have to capitalize them in your profile.

### 15.1 General Commands

The following commands are recognized:

|                |   |
|----------------|---|
| pin-numbers    | If ON, pin numbers are displayed.<br>Default: pin-numbers OFF   |
| pin-positions  | If ON, pin positions are displayed.<br>Default: pin-positions OFF   |
| use-extension  | If ON, DP will use the .DP extension when looking for input files and will append the .DP extension to output files. If OFF, DP first tries the name without any extension, and it appends no extension to output files.<br>Default: use-extension ON |
| center-files   | If ON, a file which is read in will be centered in the current file. If OFF, items will appear in the same location as on the original drawing.<br>Default: center-files ON   |
| rename-symbols | If ON, a symbol is renamed if a file containing a symbol with the same name is read in with the <u>I</u> command (a \$ is appended to the name of the symbol in the current file). If   |

OFF, the symbol on the current drawing is replaced by the incoming symbol with the same name. (This is the same as the "rename redefined" command on the Unusual Commands menu.)  
Note: This does not apply to the i command because that command will not permit you to read in a symbol with the same name as an existing symbol  
Default: rename-symbols ON

**input-path**

When the I command is given, DP will display a menu which contains a list of all of the files in the specified path which have a .DP or .DR extension. If you have this command in your profile more than once, when you type I you will receive a menu listing all the specified paths. If the argument is /Searchlist, the directories in your searchlist will be used (see the PERQ Software Reference Manual for a description of Searchlist). To specify the current path (the directory in which the current drawing is located), use >.

Example: input-path :user>swaney>

**symbol-inputs**

This command is like the "input-path" command above, except that it works with the i command and, after you choose a file, DP lists symbols which begin with the filename. See discussion in Section 10.4.

Example: symbol-path :user>swaney>

**font**

DP will append the specified font to the font table on each new drawing. The argument should be in the same format as that required for the f command.

Example: cmss 8 r 0 cmss8r10.kst

**def-font**

This statement performs the same function as font explained above, except that the specified font is then used as the default font.  
Default: def-font gacha 7 r 0 gacha7

|               |  |
|---------------|--|
| layer         | DP will append the specified layer to the layer table on each new drawing. The argument is a layer name followed by the attributes--a maximum of three characters from the set [r w o] meaning Readable, Writeable, Output-able. If the letter is present, the attribute is ON.<br>Default: layer Standard rwo |
| def-layer     | This statement performs the same function as <u>layer</u> explained above, except that the specified layer is then used as the default layer.<br>Default: def-layer Standard rwo   |
| def-thickness | DP will use the specified thickness as the default for lines, circles, and splines.<br>Default: def-thickness 1  |
| def-color     | DP will use the specified number of colors as the default. Only "1" is meaningful on a black and white system.<br>Default: def-color 1   |
| checkpoint    | After the specified number of keystrokes in a window, DP will write the window to a checkpoint file.<br>Default: checkpoint 100  |
| gravity       | DP will use the specified gravity as the default. Higher numbers mean stronger gravity. A setting of 0 means do not apply gravity at all.<br>Default: gravity 5  |
| replace-grid  | If ON, all mouse grids contained in a file that is read in will replace the current grids.<br>Default: replace-grids OFF   |
| append-grid   | If ON, all mouse grids contained in a file that is read in will be appended to the current set of grids. <u>Replace-grid ON</u> and <u>append-grid ON</u> cannot be used at the same time.<br>Default: append-grid OFF   |

function-in-cursor

If ON, the current function is displayed with the cursor. If OFF, it is displayed only at the bottom of the screen.

Default: function-in-cursor OFF

commands-file

The argument is a file containing a list of the commands which are to be displayed in the Top-Level Commands menu. See Section 6.5.1 for details on how to set up the file. If you do not have this command in your profile, DP uses its own default menu.

Example: commands-file Dpcmds.txt

menu-old

If ON, the Top-Level Commands menu will not be displayed and you will have to enter all commands from the keyboard.

Default: menu-old OFF

menu-character

If OFF, the Top-Level Commands menu will show only command keywords, not command characters.

Default: menu-character ON

use-raster

If OFF, raster-op will not be used.

Default: use-raster ON

begin-window

DP will create a new window. The argument consists of four numbers which represent the coordinates of the lower left and the upper right corners. Coordinates are x,y pairs. The default is a full-size window--(0,0) at the bottom left of the screen and (767,1023) at the top right. Between this statement and the end-window statement, you may place only the commands shown in Section 15.2.  
Default: begin-window 0 0 767 1023

end-window

This statement ends the special window mode invoked with the above statement.

Example: end-window

15.2 Window Commands

Between a begin-window and an end-window statement, you may place the following commands if you'd like. Other commands will not be recognized. Each command will apply only to that window.

**scale**

DP will use the specified scale for the new window. A real number must be used.

Default: scale 1.0

**mouse-grid**

DP will use the specified mouse grid for the new window.

Default: mouse-grid 3

**display-grid**

DP will use the specified display grid in the new window.

Default: display-grid 6

**show-grid**

If ON, the display grid will be visible in the new window.

Default: show-grid OFF

## APPENDIX A. COMMAND SUMMARY

Enter: dp [<filename> | @<filename>] [/<switch=value>] (Section 3)

Exit: q (for quit) (Section 4)

Top-Level menu: rectangles in border or press space bar (Section 6.5)

Window (Sec14): Change shape--upper right square New window--lower left square  
Delete window--upper left square Move window--gray border

|   |  |
|---|--|
| a enter ASCII String mode (7.1.3)   | s enter Select mode (8.1)                                |
| b create (pack) a symbol definition(8.2.1)                                  | S select all items in current window (8.1)               |
| B unpack a symbol instance into its components (8.5)                        | t transform symbol instance (8.4)                        |
| c copy items and move them (9.2)  | u undelete items deleted since last delete command (9.6) |
| d enter Delete mode (9.6)   | v layers menu (13.1)                                     |
| D delete all Selected items (9.6)   | w move the image inside the current window (6.9)         |
| e enter Edit mode (7.2)   | x stretch items (9.3)                                    |
| f change the current font or change the font table (9.7.2)                  | z enter Deselect mode (8.1)                              |
| g go to the next mouse grid (6.2)   | Z deselect all items in current window (8.1)             |
| G go to the next mark in the circular buffer (6.9)                          | 0 enter Circle mode (7.1.4)                              |
| h (or press HELP) type a Help file (5)                                      | 9 enter Spline mode (7.1.5)                              |
| H print hard-copy of current window (9.8)                                   | ~ toggle display grid (6.9)                              |
| i read a symbol from another file and put it at the current position (10.4) | # display diamonds (6.9)                                 |
| I read a drawing from another file, merging it with current window (10.3)   | = change current scale (6.9)                             |
| j read a text file (10.5)   | - change current thickness(9.7.1)                        |
| k present Unusual Commands menu (below)                                     | @ read a transcript file (12)                            |
| l enter Line mode (7.1.1)   | ? present internal information (6.8)                     |
| L enter Rectangle mode (7.1.2)  | INS insert mark at center of current window (6.9)        |
| m move items (9.1)  | DEL delete last mark (6.9)                               |
| M magnify items (9.5)   | CTRL c abort command or menu (6.4 and 6.5)               |
| n set new parameters for existing items (9.7.3)                             | SPACE Top-Level menu (6.5)                               |
| o write out contents of window (10.1)                                       |  |
| p enter Pin mode (7.1.6)  |  |
| P toggle display of pins (6.9)  |  |
| r redraw current window (6.9)   |  |
| R redraw whole screen (6.9)   |  |

Profile options: see Section 15

### k Unusual Commands menu: (Section 11)

set no. of commands between checkpoint saves

alter distance between dots in display grid

change list of mouse grids

specify strength of gravity

default extension (.dp), Y/N

center incoming files, Y/N

rename symbol if same name as symbol on incoming dwg--YES=rename; NO=replace  
toggle cursor (between NW and NE)

display in black mode the marks at overlapping existing items, Y/N

display diamonds at intersections of lines, Y/N (same as # command)

display pin numbers, Y/N

display pins, Y/N

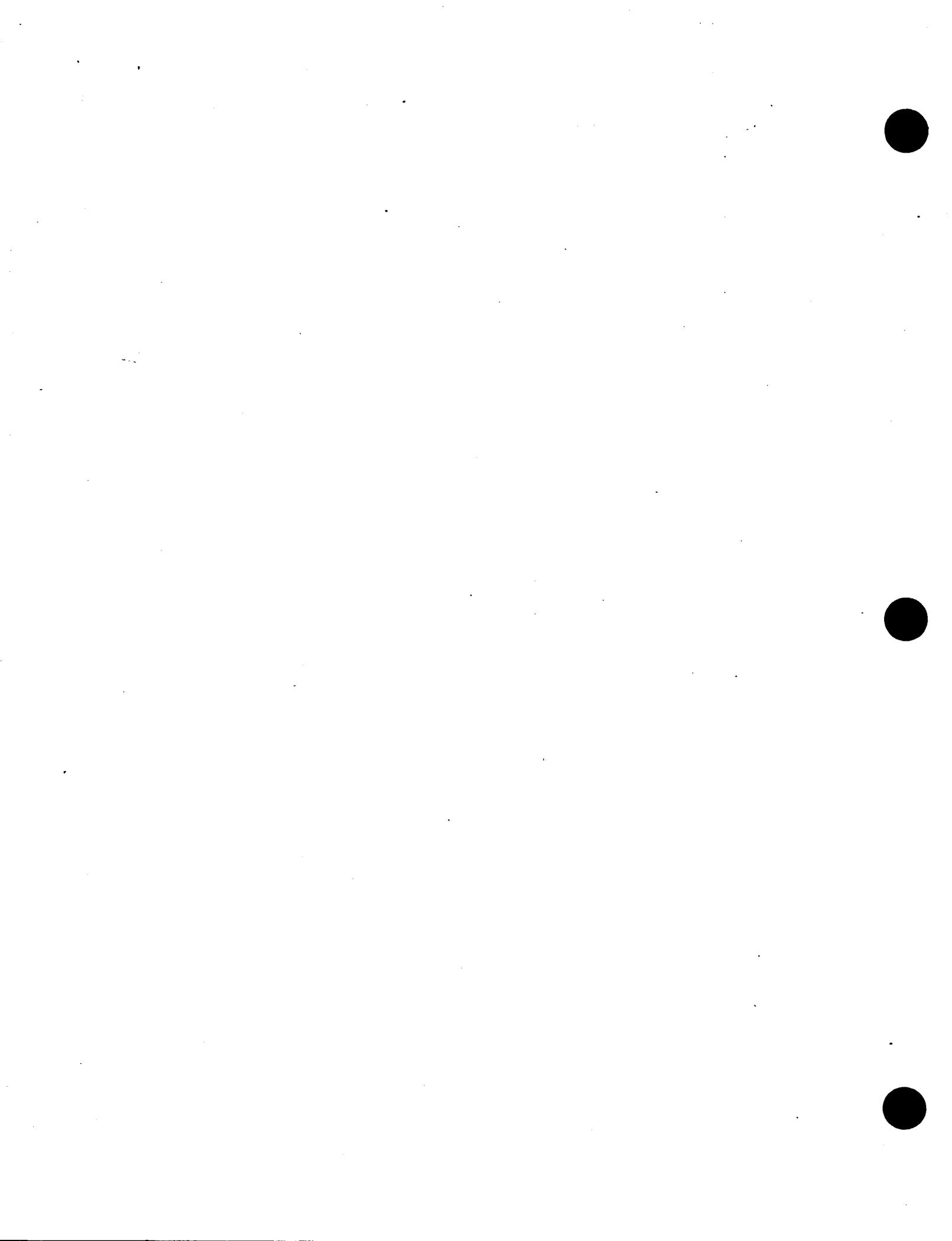
send to plotter

clip the drawing to the size of the window when printed on plotter, Y/N

print diamonds when drawing is printed, Y/N

read in mouse grids with incoming dwg, Y/N

replace current mouse grids with grids on incoming dwg, Y/N



**CHAPTER 5**  
**SCHEMATIC WIRELISTER - SL**



## CHAPTER FIVE

## SL - SCHEMATIC WIRELISTER

|                                     |    |
|-------------------------------------|----|
| HIERARCHICAL DESIGN                 | 1  |
| HIERARCHICAL DESIGN WITH SYSTEM D/L | 2  |
| DEFINING A PART                     | 4  |
| DEFINITION RECTANGLE                | 6  |
| The Icon                            | 7  |
| Inputs                              | 8  |
| Outputs                             | 9  |
| Bidirect                            | 9  |
| Tristate                            | 10 |
| Structured Nets                     | 10 |
| Other Text                          | 11 |
| IMPLEMENTATION DRAWING              | 11 |
| Parts and Instance Names            | 12 |
| Wires and Signal Names              | 13 |
| Structured Nets                     | 13 |
| Connections                         | 15 |
| SPECIAL SYMBOL NAMES                | 15 |
| Comment                             | 16 |
| TitleBox                            | 16 |
| Literal                             | 16 |

|  |    |
|--|----|
| Text   | 17 |
| SN   | 17 |
| SPECIAL PREDEFINED SYMBOLS                   | 17 |
| NC   | 18 |
| GND  | 18 |
| VCC  | 18 |
| ERRORS                                       | 18 |
| Inaccurate Signal Name Placement             | 19 |
| Incorrect Symbol Definitions                 | 19 |
| Extraneous Lines or Other Items              | 19 |
| Bad Names                                    | 19 |
| Missing Instance Names                       | 20 |
| QUICK GUIDE TO CREATING A NON-PRIMITIVE PART | 20 |
| RUNNING SL                                   | 21 |

The designer uses the DP program to draw the schematics that represent an electronic circuit. DP is a general-purpose graphical editor. It is not specific to electronics design and can be used to make any kind of drawings. DP has no understanding of electronics.

The SL program examines each drawing to extract the electrical information and writes a file which describes this electrical information. So while DP is used to draw schematics, the designer must follow certain guidelines in order that the schematics are correctly interpreted by SL. SL provides the ability to do hierarchical circuit design. This chapter explains how to use DP to make drawings that will be understood by SL.

This chapter assumes that the reader has some familiarity with DP. In particular, the reader should know how to draw lines, make strings, pack symbols, and call symbols from other files.

### HIERARCHICAL DESIGN

A completed electronic design is an interconnected collection of electronic components. Traditional design methods impose little or no structure on the design. A design may be divided into several main units (e.g. arithmetic, input/output, memory, etc.) but left unstructured beyond that. Hierarchical design is a method in which the design is divided into several main units, each of those is divided into simpler units, and so on, until the units are simple enough that each contains only a few electronic parts. This results in a design which may be viewed at varying levels of detail. Thus it is called hierarchical design.

In this chapter we will use the term primitive part to mean an electronic component which is not defined in terms of other components; the term non-primitive part to mean one of the intermediate units in a hierarchical design; and the term part to mean a primitive or non-primitive part.

Primitive parts are the raw materials that the designer uses. In TTL design, these are intergrated circuits or sections of ICs. In gate-array design, these are cells or groups of cells whose definition is provided by the gate-array vendor.

Hierarchical design may be done in a top-down manner, a bottom-up manner, or a combination of the two. In top-down design, the designer begins with a single description of the circuit as a whole--this is called the top-level part. The designer then defines this part by inventing several simpler parts and interconnecting them as though they were primitive parts. This defines the interface between the new parts. The designer repeats this job with each of the second-level parts, the third-level parts, and so on, until, through some expansion, each non-primitive part has been fully defined in terms of primitive parts.

In bottom-up design, the designer begins with primitive parts. The designer connects relatively small groups of these together and defines the resulting sub-circuits to be parts. These parts are interconnected to make more complex parts, until the desired circuit is defined as a single part.

It is rare for a designer to use either top-down design or bottom-up design method exclusively. Usually the designer uses both methods at the same time, working simultaneously on different parts of the design and at different levels. Often a part is defined by intermixed primitive and non-primitive parts.

### HIERARCHICAL DESIGN WITH SYSTEM D/L

Hierarchical design, top-down design, and bottom-up design are abstractions that are used to help understand the process of design. They are independent of specific circuit design tools such as System D/L. It is possible to do hierarchical, top-down design with a pencil and paper. System D/L provides an environment which aids the designer in hierarchical design. The SL program provides a certain mechanism for interpreting hierarchical schematics.

The basis of System D/L design is that every part may be viewed at two levels of detail: the part as a single unit and the definition of that part. A small drawing called the icon is used to represent the part as a whole. The implementation is a logic circuit which uses other parts. A particular icon may be used in several places to reduce the size and time spent on the schematic. The difference between primitive and non-primitive parts is that primitive parts have no implementation.

Icons are usually drawn to look like components. They are usually rectangles and have connection points. These are points where lines (representing wires or buses) may be drawn to interconnect different icons. The connection points of an icon are the input/output connections to the part.

The implementation shows the division of the part into simpler parts and their electrical interconnections. The implementation contains icons of other parts. For example, the icon in Figure 1 could be used to represent an octal D-latch named Lat8.

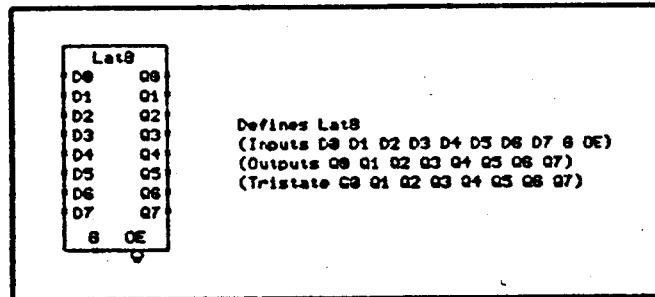


Figure 1

The implementation of Lat8 shown in Figure 2 uses eight one-bit D-latches named Lat, two inverters named IV, and eight tristate buffers named BF.

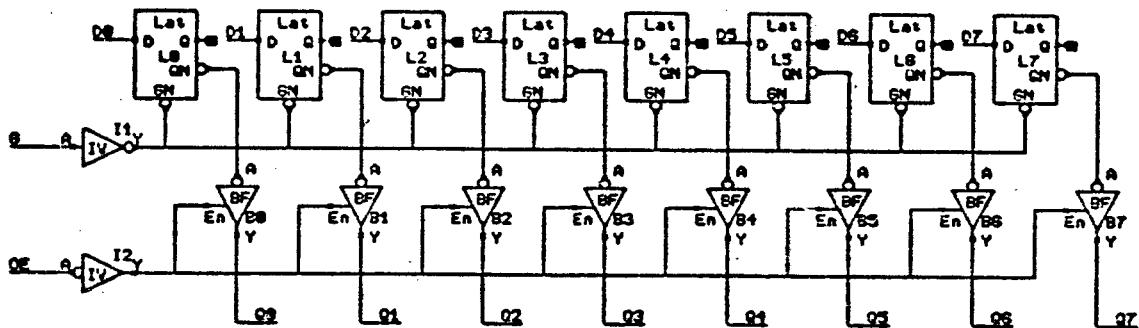


Figure 2

Figure 3 shows an implementation of Lat using inverters (IV) and nand gates (ND).

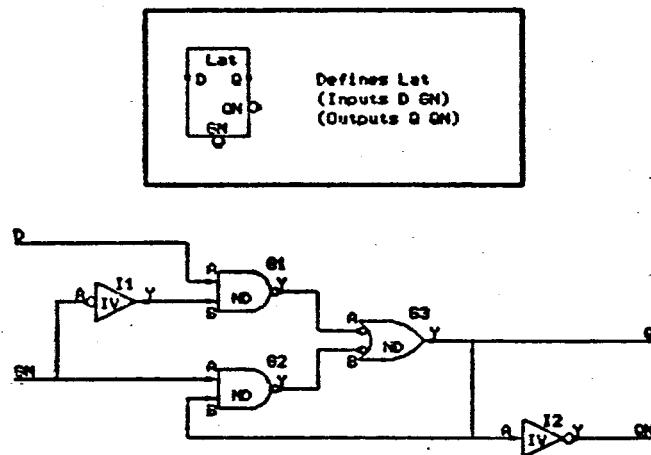


Figure 3

DEFINING A PART

A new part is defined by drawing it with DP. The icon and the implementation are combined in a single drawing. The icon and other information about the part are surrounded with the definition rectangle. This is a rectangle drawn with lines of width two. The definition rectangle contains all information which is needed to understand the interface to the part. The implementation drawing is everything that is outside the definition rectangle. Other non-electrical information may be included in the drawing by using comment symbols. Comments are described in detail later.

Figures 4 and 5 show the DP drawings for Lat8 and Lat.

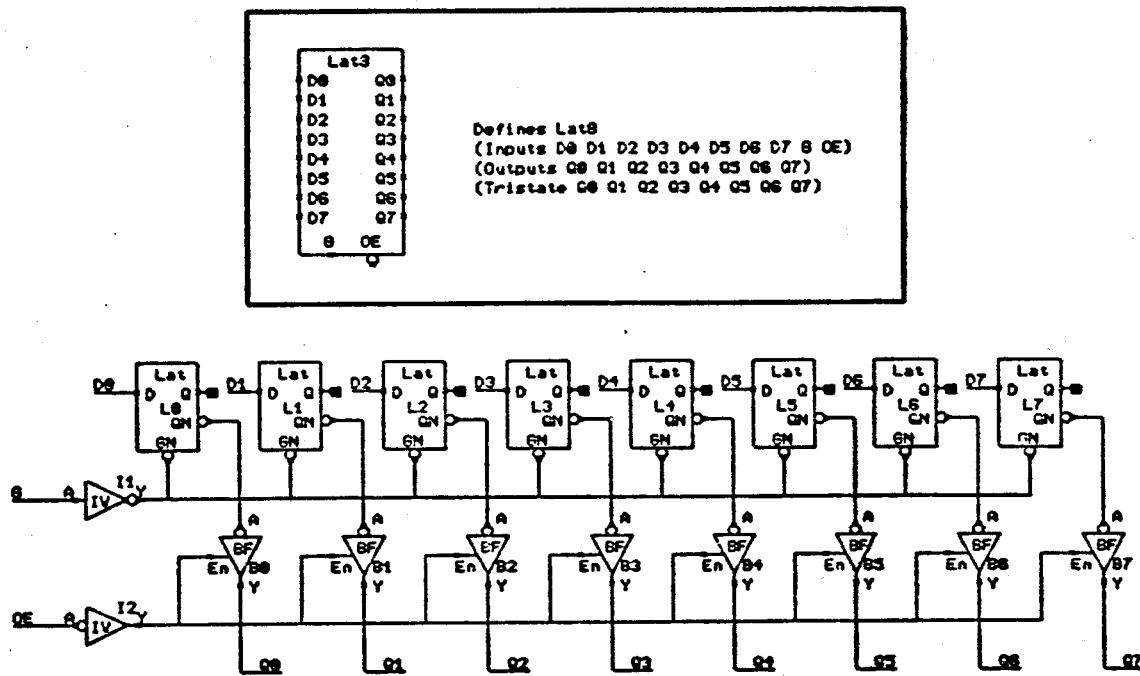


Figure 4

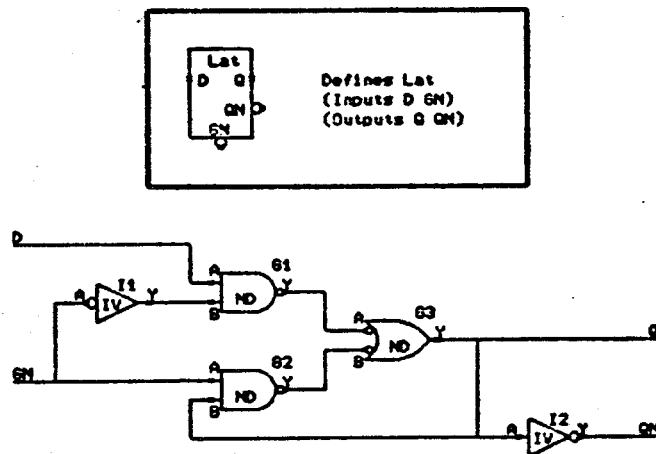


Figure 5

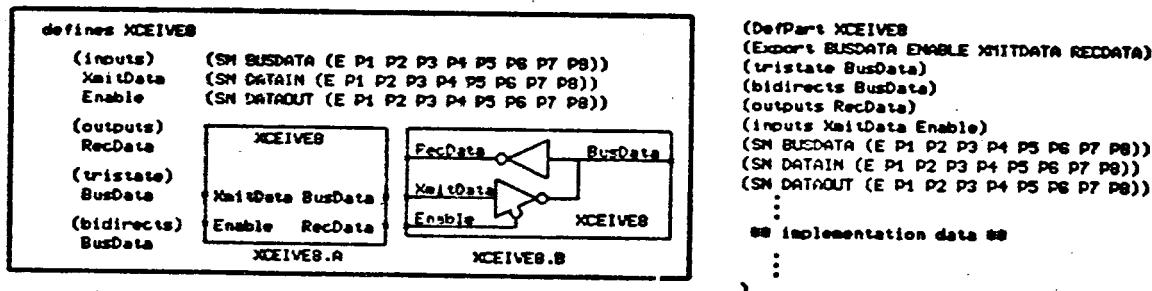
The implementation drawing describes the inner workings of the part as an interconnected set of parts. The icons of these parts are added to the implementation drawing by using the "i" command to read them from their DP files.

Lines are drawn between parts to define their electrical connections. Lines may represent single wires or buses (groups of wires). Buses are normally used to group related signals together. A bus may be manipulated as a whole or as individual signals.

A single wire is sometimes called a signal because it carries an electrical signal and sometimes called a net because it represents a wiring network. A bus is called a structured net because it represents a structured group of wiring networks.

## **DEFINITION RECTANGLE**

The definition rectangle contains the icon and text strings which declare some of the attributes of the part. Figure 6 shows an example of a definition rectangle which includes an example of several declarations.



## **DEFINITION RECTANGLE**

## **SL OUTPUT**

**Figure 6**

The definition rectangle must contain the string

Defines PartName

where PartName is the name of the part being defined. This name should match the name of the icon symbol and the DP file. Other text strings in the definition rectangle are written into the output file produced by SL. These are used to impart information needed by other post processors. The minimum required is a description of the connection points of the part.

Each connection point is characterized as an input, output, bidirectional, tristate, or structured net connection point. Each of these is described in greater detail below.

### The Icon

The icon must be built in a specific way (see Figure 7). The shape of the icon should be drawn first, and the name typed within the shape. Connection points are placed on the border of the icon.

A connection point is a DP packed symbol consisting of a single DP pin and a single string which represents the pin name. Connection points are sometimes called pin symbols.

Pins are made with the DP "p" command. DP will request a pin number for each pin. This number is ignored by SL--you may use the "~~" command to turn off the display of the pin number. A pin is drawn by DP as four dots in a small square. The connection point represented by the pin is at the center of the square.

A pin name must be associated with each connection point. This name is used within the implementation and is also used during simulation. Each pin and its name should be packed into a DP symbol using the "b" command. The name of this symbol is ignored by SL, so the easiest thing to do is to type RETURN to make an unnamed symbol.

The connection points are positioned so that each pin lies on the border of the icon, and the entire icon is packed into a DP symbol using the "b" command.

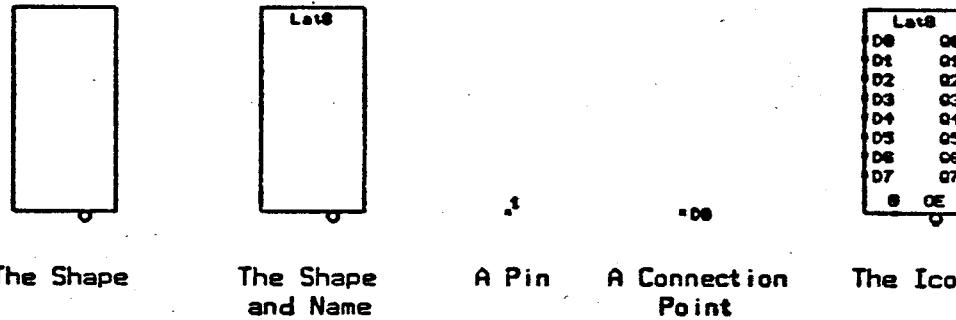


Figure 7

The name of an icon symbol should be

PartName -or- PartName.IconName

where PartName is the name of the part in the Defines string and IconName uniquely identifies the icon. This naming convention allows a certain part to have more than one icon. For example, Figure 8 shows a Nand gate drawn as an inverting And gate and as a negative-logic Or gate. The symbols are named ND and ND.Or.

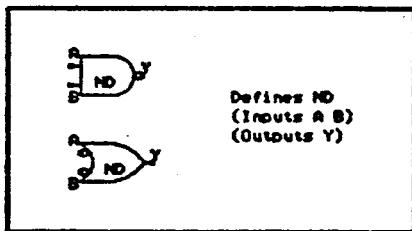


Figure 8

### Inputs

Each connection point of a part must be declared to be an input, output, or bidirectional connection. The inputs are listed together in the definition rectangle of the part. This is done in one of two ways. The simplest is to type a string of the form

(Inputs Name1 Name2 ... NameN)

which contains all inputs to the part. Listing all the inputs in a single string may be inconvenient, particularly if there are many of them. In this case, the names of the connection points are typed as separate strings. The string "(Inputs)" and the names of the input signals are packed into a symbol:

(Inputs)

Name1

Name2

...

NameN

The resulting symbol must be named Literal.<AnyString>. The convention is to use Literal.In. In Figure 6, XmitData and Enable are inputs to XCEIVE8 and are included in its inputs declaration.

Outputs

Each connection point of a part must be declared to be an input, output, or bidirectional connection. The outputs are listed together in the definition rectangle of the part. This is done in one of two ways. The simplest is to type a string of the form

(Outputs Name1 Name2 ... NameN)

which contains all outputs from the part. Listing all the outputs in a single string may be inconvenient, particularly if there are many of them. In this case, the names of the connection points are typed as separate strings. The string "(Outputs)" and the names of the output signals are packed into a symbol:

(Outputs)  
Name1  
Name2  
...  
NameN

The resulting symbol must be named Literal.<AnyString>. The convention is to use Literal.Out. In Figure 6, RecData is an output of XCEIVE8 and is included in its outputs declaration.

Bidirect

Each connection point of a part must be declared to be an input, output, or bidirectional connection. The bidirectional connections are listed together in the definition rectangle of the part. This is done in one of two ways. The simplest is to type a string of the form

(Bidirect Name1 Name2 ... NameN)

which contains all bidirectional connections to the part. Listing all the names in a single string may be inconvenient, particularly if there are many of them. In this case, the names of the connection points are typed as separate strings. The string "(Bidirect)" and the names of the signals are packed into a symbol:

(Bidirect)  
Name1  
Name2  
...  
NameN

The resulting symbol must be named Literal.<AnyString>. The convention is to use Literal.Bi. If a connection point is declared as bidirectional, it need not be declared as an input or an output. In Figure 6, BusData is both an input and output of XCEIVE8 and is included in its bidirect declaration.

### Tristate

Normally a wire may connect any number of input pins and exactly one output pin. If the outputs are tristate drivers, however, more than one output may be connected to a wire. In this case, each output connection point must be declared to be a tristate driver in addition to its declaration as an output. The tristate connections to a given part are listed together in the definition rectangle of the part. This is done in one of two ways. The simplest is to type a string of the form

(Tristate Name1 Name2 ... NameN)

which contains all tristate connections to the part. Listing all the names in a single string may be inconvenient, particularly if there are many of them. In this case, the names of the connection points are typed as separate strings. The string "(Tristate)" and the names of the signals are packed into a symbol:

```
(Tristate)
Name1
Name2
...
NameN
```

The resulting symbol must be named Literal.<AnyString>. The convention is to use Literal.Tri. In Figure 6, BusData is a tristate signal of XCEIVE8 and is included in its tristate declaration.

### Structured Nets

If the connection point of a part is a structured net but is not expanded into individual signals in the implementation part, it must be declared to be a structured net in the definition rectangle. This is done by typing a string of the form

(SN PinName (E P1 P2 ... Pn))

where n is the number of signals in the structured net--PinName!P1 through PinName!Pn are the names of the signals which make up the structured net. Listing all the elements in

a single string may be inconvenient, particularly if there are many of them. In this case, several strings may be typed:

```
(SN PinName (E P1 P2 ... Pn))  
(SN PinName (E Pn+1 ... Pm))
```

so that each element appears once. The elements correspond to pin numbers in a structured net symbol in the implementation part. These are described in greater detail in the IMPLEMENTATION DRAWING section. In Figure 6, BusData, XmitData, and RecData are 8-bit structured nets which are not expanded in the implementation part of XCEIVE8. They have therefore been declared in the definition rectangle as follows:

```
(SN BusData(E P1 P2 P3 P4 P5 P6 P7 P8))  
(SN XmitData(E P1 P2 P3 P4 P5 P6 P7 P8))  
(SN RecData(E P1 P2 P3 P4 P5 P6 P7 P8))
```

#### Other Text

In some cases additional information may need to be added to the definition rectangle. For the Component Library, this additional information must follow a strict declaration syntax.

When the order of the strings in the definition rectangle is important, these strings should be packed into a symbol named Text.<AnyString>. SL does not guarantee that strings placed in the definition rectangle will be written to the .SL file in a fixed order. However, strings packed into a Text symbol inside the Definition Rectangle are written to the .SL file in reading order. That is, strings are written starting with the upper left-most string and ending with the lower right-most string. The coordinates determining string position are the upper left-most bit of the first character of the string.

#### IMPLEMENTATION DRAWING

The portion of a drawing outside the definition rectangle is the implementation drawing. The implementation part of a drawing shows the internal detail of a part definition. It contains the icons of each sub-component, as well as the electrical interconnections between sub-components. These icons are called into the implementation as symbols using the "i" command. Interconnections are indicated by lines and through matching pairs of signal names.

The implementation must define the internal interconnections of each terminal declared in the definition rectangle. These interconnections are indicated using the connection mechanisms described in detail below. Figure 9 shows the implementation corresponding to the definition rectangle in Figure 6. This example will be used throughout the remainder of this section to illustrate the various items appearing in an implementation.

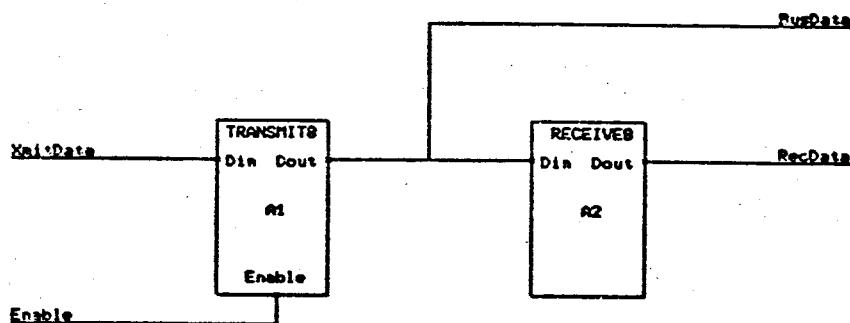


Figure 9

### Parts and Instance Names

Primitive parts are obtained from a directory on your PERQ. This directory should be in your searchlist while creating a drawing using DP. Special directories on the PERQ and on the VAX are used to keep similar primitive parts in one place. If you use DP to examine a primitive part, you should take care not to accidentally create a copy of the primitive part in your current directory.

Each part used in the implementation, whether a primitive or non-primitive part, must be given a unique instance name. This name is used to identify the gate during simulation. Each instance name should be as mnemonic as possible. For individual gates, instance names such as "a1", "c3" and so on may prove adequate. SL treats any string whose lower-left or lower-right corner appears within the bounding box of a symbol as the instance name of that symbol. In Figure 9, A1 and A2 are the instance names of the parts TRANSMIT8 and RECEIVE8 respectively.

DP will permit almost anything as a valid string. However, the post-processors, and in particular the Tegas simulator, place serious restrictions on names. Tegas permits only 12 characters in names. The first character must be alphanumeric while the characters following can be alphanumeric, "." or "-". Capitalization is ignored so that names which are identical

with the exception of capitalization are considered to be the same.

Some programs check the names, truncating to 12 characters and replacing invalid characters with valid ones. These methods may cause names which are different in the DP drawings to become identical. Tegas also confuses signals named "IN" and "OUT" with Tegas reserved words.

In addition, you should avoid using the same name for a symbol and a symbol instance within the same drawing. Avoid the names listed in the section Special Predefined Symbols (see below). You should never use "!" or "/" in signal names. To be safe, use only 12 or fewer characters, alphanumeric or "-" or "." characters only, and always a letter for the first character.

### Wires and Signal Names

A named signal is represented in a drawing as a line with an endpoint coincident with the lower-left or lower-right corner of a string. The string is considered to be a name and the line is treated as a wire. Connections are described in detail below.

Named signals are used to define the terminals of a part. Pin names in an icon of the part being defined are matched with signal names of wires in the implementation part. Each pin name which appears in an icon of a part MUST appear in the implementation of that part.

Signals in the implementation of a part whose names do not appear in an icon of the part are treated as local signals of that part. Local signals are accessible only within the implementation that uses them.

### Structured Nets

A particular signal used by a part may be declared to have internal structure. Such a signal is called a structured net. A structured net may consist of individual signals or other structured nets. Figure 10 shows two examples of structured nets. In one example, Enable consists of 8 bits, and in the other example, Bus consists of two structured nets Data and Address, which consist of 8 bits each.

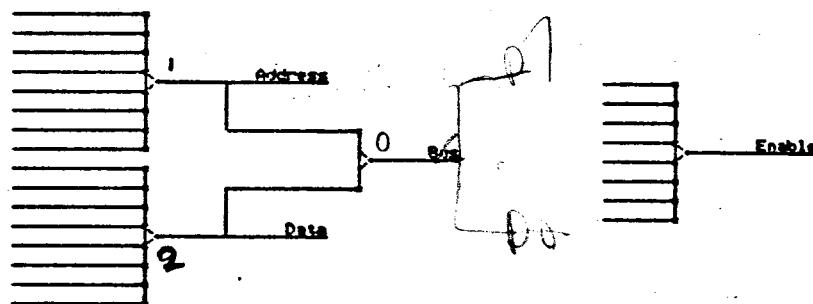


Figure 10

The conversion between a structured net and its components is done with a structured net symbol. Any symbol named SN or SN.<AnyString> will be interpreted as a structured net. When creating an SN symbol, you must put pins on the symbol for each element in the structured net. Pin 0 must be the bus itself, with pins 1 through n denoting the n elements of the structured net. By convention an SN symbol has an angle and a horizontal or vertical bar, with pin 0 on the point of the angle and pins 1 through n on the side of the bar opposite the angle. After all pins are placed on the bar, the symbol should be packed with the DP "b" command and given a symbol name starting with "SN.". Note that the pins are not packed with names as they are in icons. Figure 11 shows two SN symbols.

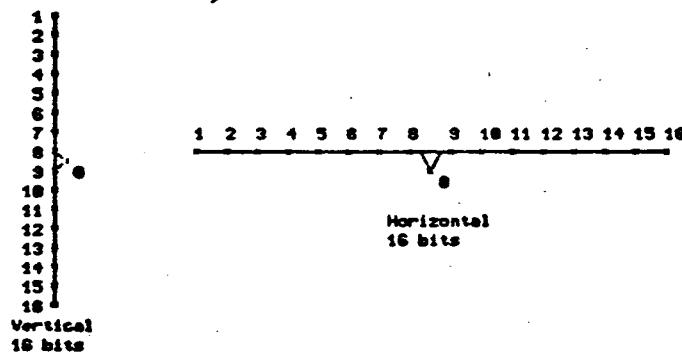


Figure 11

Structured nets may appear as inputs or outputs of a part. Any two structured nets with the same number of elements may be connected using the connection mechanisms described in detail below. When connecting structured nets together, elements are

paired by element number (pin number on the bar of the SN symbol). If a structured net signal is used in the implementation drawing but is never expanded using an SN symbol, it must appear in an SN declaration in the definition rectangle. This declaration is described in the DEFINITION RECTANGLE section.

### Connections

SL implements both symbolic and graphic connection mechanisms. A symbolic connection is made whenever two signals have identical names. A graphic connection is made whenever a line ends on another line or on a connection point of an icon. Lines which cross are never considered to be connections. The DP "/" command displays small black squares at all points where lines are connected. Figure 12 shows how connections are indicated.

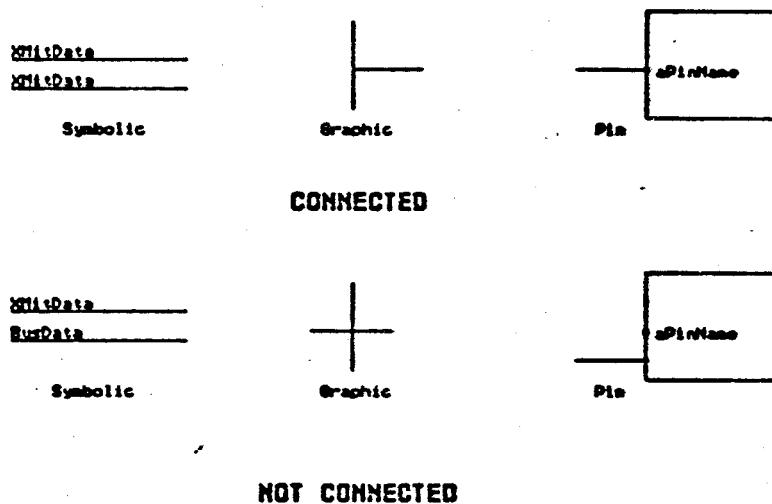


Figure 12

### SPECIAL SYMBOL NAMES

Certain symbol names are reserved by SL for special processing. These are used to create graphical comments and drawing titles and to cause SL to do special processing of strings in the definition rectangle. You should not use any of these names for your icons.

Comment

Any symbol whose name is Comment or Comment.<AnyString> is ignored by SL. This mechanism allows both symbolic and graphic comments to be inserted into a drawing without altering the wirelist.

TitleBox

Any symbol whose name is TitleBox or TitleBox.<AnyString> is ignored by SL. In addition, SL ignores any string within the bounding box of a TitleBox symbol. This provides a mechanism for inserting fields within a standard form without altering the wirelist. For example, many companies specify a standard graphical format for engineering drawings, including the drawing name, the designer, revision history, and so on. This format can be drawn, packaged into a TitleBox symbol, and stored in a library directory for use in each drawing.

A standard set of TitleBox symbols which create a schematic page format is included with System D/L in the file Page.Dp. The use of a standard page format is a convention that you may or may not wish to follow. In order that your schematics have a consistent appearance, it is suggested that you adopt the page format supplied with System D/L or one which better suits your needs.

Literal

Within the definition rectangle, a symbol named Literal or Literal.<AnyString> may be used to create declaration strings. Literal symbols must contain only text string. One of these text strings should have a left parenthesis as its first character and a right parenthesis as its last character. SL inserts all other strings into this string immediately before the right parenthesis. The strings are inserted in no particular order and are separated by spaces.

A symbol called Literal.C in the black box which consisted of the strings:

(Clocks)  
DClock  
MasterClock  
Phi2

would result in the string

(Clocks DClock Phi2 MasterClock)

Text

Normally, the order of declarations in the definition rectangle is not significant, and System D/L imposes no fixed ordering. For some libraries, however, the order is significant. The Text symbol is used to control the ordering of text in the definition rectangle.

Strings which are packed into a symbol named Text.<AnyString> are interpreted in reading order. Reading order is the order that the user would read text from the screen. That is, text is read left-to-right, top-to-bottom. The upper-left-most string is first, followed by strings on the same line, followed by the string which is the next lower on the screen, and so on. The upper-left most bit of the string determines the position of the string on the screen.

Text symbols may contain other Text symbols and Literal symbols. In this case, the symbols themselves are interpreted in reading order (by their upper-left corner). That is, the upper-left-most symbol is first, followed by symbols on the same line, followed by symbols which are lower on the screen. You should use multiple Text symbols if you have several columns of text. You should pack each column into its own Text symbol, and then pack the Text symbols into another one.

SN

Any symbol named SN or SN.<AnyString> will be interpreted as a structured net. See the Structured Nets section for an explanation of how to use these symbols.

SPECIAL PREDEFINED SYMBOLS

Each connection point of each part must be wired to something. An input which is not wired to another part should be wired to logical 1 or 0. An output which is not wired to another part must be wired to a symbol which indicates that the output is not connected. Several special symbols are supplied as part of the standard library for ground, Vcc, and no-connect. These symbol names are recognized by SL and so you should not use any of these names for your icons.

NC

A special symbol named NC should be used for outputs that are intentionally not connected to anything. Any symbol named NC or NC.<AnyString> will be interpreted as a special no-connect symbol. You may define an NC symbol to have a single pin which you connect to your unused outputs. This will tell the post-processors that these pins are intentionally not connected. This mechanism ensures that every pin connects to some symbol, even if the symbol is an NC. A pin which is not connected to any symbol at all is declared to be an error.

GND

A special symbol named GND, denoting electrical ground (logical 0), should be used for signals that are intentionally grounded. Any symbol named GND or GND.<AnyString> will be interpreted as a ground symbol.

VCC

A special symbol named VCC should be used for signals that are intentionally connected to Vcc (logical 1). Any symbol named VCC or VCC.<AnyString> will be interpreted as a connection to Vcc.

ERRORS

When SL detects errors in a DP drawing, it writes an error file in the form of a DP drawing. To view this drawing, it is overlayed on the original DP drawing. For example, if a DP drawing named Test.DP has errors, SL writes a drawing named Test.Err.DP. To view the errors, first read Test with the DP "I" command, then overlay Test.Err with the "I" command. The order is significant--if you read the drawings in the opposite order, the error overlay will not match properly.

Errors in declarations or logical errors in the wiring may not be detected by SL. These are usually detected by DA, but some may cause problems when the circuit is simulated. Several common errors are described in this section. Check this list first if you have trouble with a post-processor, if the wirelist seems incorrect, or if the simulation does not make sense.

Inaccurate Signal Name Placement

A common error is the inaccurate placement of signal names on lines. Signal names must be placed so that their lower-left or lower-right corner is within 5 dots of the endpoint of the wire. Failure to do so will result in an incorrect wirelist or error messages from the post-processors. SL usually catches these errors, but sometimes will report a misleading error message. This can happen if a signal name is correctly placed on a line, but is also partly within the bounding box of a symbol--the string may be interpreted as the instance name of the symbol.

Incorrect Symbol Definitions

Erroneous symbol definitions can cause errors. The connection symbols may be incorrect--usually because they are not defined as symbols containing only a pin and a signal name. The icon itself may be incorrectly packed. You can check this by decomposing the icon symbol and the connection symbols with the DP "B" command. The DEFINITION RECTANGLE section describes in detail how to create connection symbols and icon symbols.

Extraneous Lines or Other Items

A less frequent source of errors is an extraneous line, string, or other item in the drawing. This will usually result in the error "net with too few pins". The extra item may be hard to spot if several identical items are overlayed. When an item is copied in place, it disappears because DP uses exclusive-or to draw the screen. To determine if this is your problem, display the DP file along with the error file generated by SL and use the DP "m" (move) command to see if you can move an item in the area pointed to by the error message.

Bad Names

Improper naming of signals, instances, or symbols may not be detected by SL. Such errors may not show up until DA is run or the circuit is simulated.

Missing Instance Names

If an instance name is omitted, SL generates an instance name for you. This instance name consists of an integer and a "\$". When SL creates instance names, it creates an overlay file containing the names and prints the message

Automatic locations; DrawingName.Loc.DP written.

This overlay file may be examined in the same way as the error file.

QUICK GUIDE TO CREATING A NON-PRIMITIVE PART

1. Enter DP or, if already in DP, write the drawing you have been working on to a file.
2. Use the "SDD" commands to delete the drawing you have been working on.
3. Use the "I" command to read Page (the standard page format).
4. Use the "o" command to write this blank page to your DP file with the same name as the new part you are creating. This step establishes the default output file for DP and avoids accidentally overwriting Page.

5. Create the new icon (see Figure 6).
  - a. Draw the shape of the new icon.
  - b. Type the name of the new part into the icon.
  - c. One at a time, create the connection points and place them on the border of the icon. Connection points are made by packing a pin and a pin name into an unnamed symbol.
  - d. Pack the entire icon into a symbol and assign it the name of the new part.
6. Type the string "Defines PartName" near the icon. PartName should be the name of the new part.
7. Type the appropriate input, output, bidirect, tristate, and structured net declarations near the icon.
8. Draw a rectangle made of lines of width 2 around the icon, the defines string, and the declarations.
9. Draw the implementation, using primitive and non-primitive parts as necessary.
  - a. Remember that the names of connection points are used as signal names within the implementation.
  - b. Remember to name all signals which are externally visible.
  - c. Remember to assign an instance name to each part used.
10. Output the finished drawing using the "o" command.
11. Run SL on the drawing and examine any errors it reports.

#### RUNNING SL

SL may be started by typing

SL <cr>

SL prompts the user for the names of files to be examined; no extension can be typed, since the default drawing-file extension ".DP" is automatically appended. To terminate the SL program, type a <cr> in response to the "DP file:" prompt.

A typical sequence might be:

```
SL ver. 1.2
DP file: cmb6
.....
cmb6.sl written

DP file: adder
.....
adder.sl written
*** 1 Errors : adder.Err.dp written

DP file: <cr>
```

Instead of typing all the DP file names it is possible to let SL read them from a file. The file should contain a list of file-names, one per line; the name of this command file should be typed after "SL". If "namesList" is the name of such a file, for instance, the command line would be

```
SL namesList <cr>
```

All the conversation with the user is recorded in a file called "logStream".

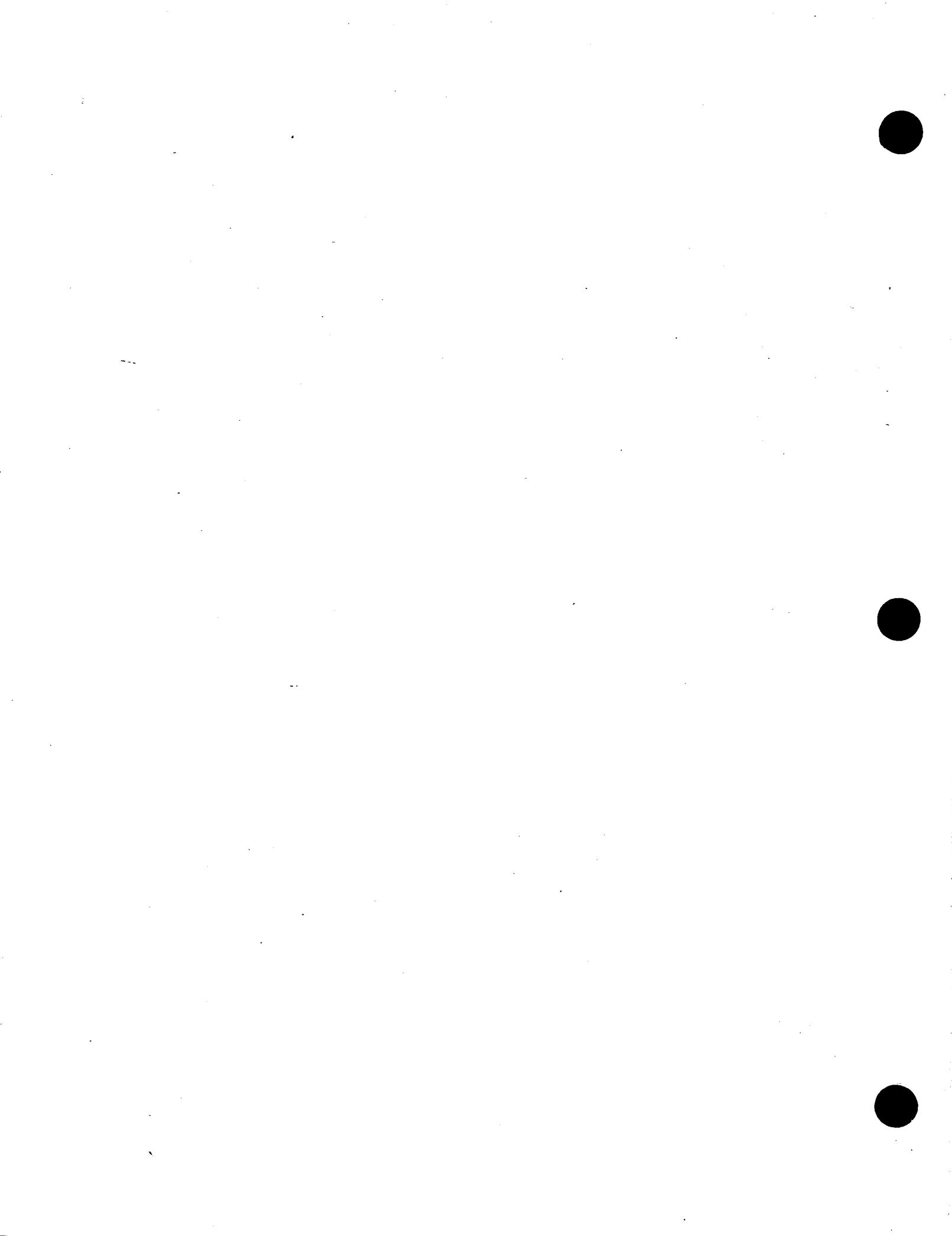
**CHAPTER 6**  
**DESIGNER'S ASSISTANT - DA**



CHAPTER SIX

THE DESIGNER'S ASSISTANT - DA

|                        |   |
|------------------------|---|
| ERROR NOTIFICATION     | 1 |
| COMMAND DESCRIPTIONS   | 2 |
| Command Summary        | 2 |
| ADDITIONAL INFORMATION | 4 |
| Generating IDL         | 4 |
| Switch Summary         | 5 |



The Designer's Assistant serves as a backend to the PERQ System D/L design database, providing a number of services to the designer. DA generates a network description language, TDL, required by the logic simulator Tegas. DA can provide bills of material, cross reference listings, gate array cell use summaries, and delay and timing information.

DA is an interactive program which parses the hierarchical wirelist files produced by SL, the drawing post-processor. DA reads each SL file in the hierarchy beginning with the current design given as input. Simple design rule and consistency checking are performed. DA warns of suspect usage, missing symbols, inconsistent use of signals, signals which are not declared to be either inputs, outputs or bidirectionals, and nets which have multiple sources.

DA accepts commands from the keyboard, command files, or through the use of pop-up menus. DA will prompt for the design name and then read it before performing a requested function.

Like the other PERQ System D/L tools, DA uses only the design name to specify the design. All PERQ filenames are of the form "Root-Name.Extension.Extension...". DA makes consistent use of filename extensions by using the same Root\_Name in all of the files relating to a particular design. DA output files are always named after the design from which they were compiled. Thus, as SL creates the file FlipFlop.SL when asked to post-process FlipFlop, DA will read in FlipFlop.SL when asked to process FlipFlop.

#### ERROR NOTIFICATION

DA will sometimes print error or warning information. In addition to being displayed on the screen all such messages are also written to a file called DA.Log.

COMMAND DESCRIPTIONS

DA is an interactive program which responds to commands from either command files, keyboard input, or PopUp menus. PopUp menus can be activated by pressing any puck button. DA then displays a menu of functions. Pointing to the desired function and pressing a puck button invokes that function.

Each function requires a design name. This defaults to the most recently used design. A response of carriage return uses the default name. Typing a design name followed by a carriage return will use the specified design name.

Command Summary

The following are valid DA commands.

**Help**

The Help command displays a help message describing switches and commands.

**Read Design File**

This command reads a design from the database, doing some consistency and rules checking along the way. DA will print the name of each symbol it encounters in the hierarchy as it reads the design. Read Design File requires the name of a design. It can be typed on the same line as the command or DA will prompt for it. If it is omitted DA will suggest as a default the last symbol you used in the current session.

**IDLGen**

This command is used to create a Tegas simulation TDL file. It requires the name of a design. The result is a file with an extension of '.TDL' and the same root name as the design. See the section on generating TDL (page 4) for greater detail.

**Cell Usage**

This command prints a summary of gate array cell usage in the design. In order to use this command the primitives must be gate array primitives and must include cell counts. This command requires the name of a design.

### Clock Delays

This command produces a text file containing a list of all combinatorial delays between clocked elements of the design. The format of the file is described in the file. It also includes loading and delay information. The file created has an extension of '.Clock' and the same root name as the design. This is useful for detecting clock and timing problems.

### Bill of Materials

This command produces a text file with an extension of '.Bill' with a complete list of all primitive parts used in the design and the number used.

### Delay Information

This command produces a file with an extension of '.Delay' containing for each output signal the number of loads it drives and the complete delay specification (min, max, typ rise and fall times). The format of this file is described within the file. It can be examined by the designer, but is also used as input to the GenTCF program which augments a TCF file with actual delay information so that the simulator can simulate the design with accurate timing.

### Cross Reference

This command produces a file with an extension of '.Cross' containing a cross reference list for the design. The cross reference is in two parts. The first is a hierarchical display with proper indentation illustrating the nesting in the hierarchy. Each symbol called by another is displayed slightly indented from its parent. The second part contains a list of each symbol followed by the modules which instantiate it.

### Quit

The Quit command causes DA to exit, writing the DA.LOG file.

ADDITIONAL INFORMATIONGenerating TDL

One purpose of DA is to generate TDL, the network description language required by the Tegas simulator. Because TDL is such an essential part of the simulation, the user is assumed to have some knowledge of the Tegas simulator and its use, although explicit knowledge of the TDL syntax is unnecessary. TDL is generated by the TDLGen command.

DA starts with an arbitrary design supplied by the user (in the '.SL' file format) and generates a TDL file with the same name as the design with '.TDL' appended. DA will generate a TDL file containing TDL for the entire design except for circuit elements which are primitives. Primitives are stored in the master directory of the TDL master library on the VAX.

When the TDLGen command is invoked, DA prompts the user for information regarding the simulation. Defaults are enclosed in square brackets. (Note: These defaults may be changed by using the switches defined below.) DA asks for:

Master Directory Name, defaulting it to Master.

User Directory Name.

Whether to interactively specify lower level terminals. Usually, DA generates the full hierarchical design from the given symbol down to primitives. Interactively specifying lower level terminals allows the user to selectively specify modules in the hierarchy to expand. This is intended only as an efficient way to generate new TDL for designs which have changed in minor ways. Only modules whose descendants have changed since DA last produced TDL should be regenerated. This results in faster runtime in both DA and the TDL compiler and shorter TDL files to transfer to the VAX.

TDL options. The default option is REPL for replace. For a complete list of options, consult the TDL preprocessor manual.

Whether a TCF option is needed. Unless you want to perform a simulation with timing information, the correct response to the TCF question is a carriage return. TCF is the mechanism used to include real delays in a Tegas simulation. For design verification, you typically use unit delays until the module is debugged. You then rerun the simulation with estimated or computed delays to ensure

that there are no timing problems. TDL can generate a TCF file, an alternate flattened description of the network, to which delay information can be added. Tegas is capable of using this description instead of the one produced by the TDL compilation to simulate the design with actual delays.

Comments can be included in the .TDL file by creating a text file called Preamble.TDL. DA then outputs this as a comment to the TDL file followed by the date and time and the version of DA which created the file. The preamble file is specified with the '/TDLPreamble' switch.

DA creates an FTP command file to transfer to the VAX all of the .TDL files produced during a session. A header can be included in the FTP command file by creating a text file called Preamble.FTP (or using the /FTP\_Preamble switch in the user profile to use a different file name) which would contain any FTP commands to be executed before those generated by DA. This is useful for setting up remote directories, initializing the connection, and specifying the desired transfer device. For more information, consult the FTP documentation in Chapter 8.

#### Switch Summary

DA recognizes a set of switches which allows the user to change global values used by DA. The switches are:

**/TDL\_Preamble = <filename>**

The TDL\_Preamble switch is used to tell DA what filename, if any, it should use to include preamble information in the TDL files it creates for simulation. The preamble is included as a comment in the .TDL file as a means of generating standard preambles with the company name, the designer, or a copyright notice, for example. If not specified, the preamble is assumed to be in the file Preamble.TDL.

**/TDLMasterDirectory = <string>**

The MasterDirectory switch is used to tell DA in which master directory in the TDL master library the primitives may be found. This should be Master unless your DA system administrator changes it.

**/TDLUserDirectory = <string>**

The UserDirectory switch is used to tell DA into which user directory in the user library to place compiled IDL. This is often the name of the project the designer is working on. See the Tegas documentation for more information.

**/TDLTCF = 'YES' | 'NO'**

The TCF switch tells DA whether or not to cause IDL to generate a TCF file. The TCF file is used to drive a simulation with actual delays instead of unit delays. In order to use the TCF, you must generate a TCF file and a Delay file, and then run GenTCF to merge the two. Tegas can then use the augmented TCF file to simulate your design with accurate timing information. The default is NO.

**/TDLOptions = <string>**

The Options switch is used to tell IDL what to do with compiled modules. The default is REPL (for replace) and is generally the correct option. For a complete list of valid options and their effects consult the Tegas/TDL documentation.

**/FTPPreamble = <FileName>**

The FTPPreamble switch is used to include any preamble in the FTP command file which DA will create when generating IDL. The preamble should include any FTP commands needed to connect to the remote machine, set up directory paths, select the transfer device, etc. The default is FTP.Preamble.

## CHAPTER 7

### MIASMA



## CHAPTER SEVEN

## MIASMA

|  |   |
|--|---|
| OVERVIEW                                   | 1 |
| MIASMA USES                                | 1 |
| Logic Simulator Test Programs              | 1 |
| Bit Slice or Finite State Machine Programs | 2 |
| MIASMA INPUT                               | 2 |
| SYMBOLS                                    | 2 |
| NUMBERS                                    | 2 |
| EXPRESSIONS                                | 3 |
| DIRECTIVES                                 | 4 |
| Align                                      | 4 |
| Bitsperword                                | 4 |
| Cycle                                      | 4 |
| Endfields                                  | 5 |
| Endmacros                                  | 5 |
| Endrepeat                                  | 5 |
| Endtext                                    | 5 |
| Fields                                     | 5 |
| Include                                    | 7 |
| Leftfirst                                  | 7 |
| Loadmem                                    | 7 |
| Loc  | 7 |
| Macro                                      | 8 |
| Maxaddress                                 | 9 |

|                               |           |
|-------------------------------|-----------|
| Radix                         | 9         |
| Quad                          | 9         |
| Repeat                        | 9         |
| Rightfirst                    | 10        |
| Sequence                      | 10        |
| Settime                       | 10        |
| Skip                          | 11        |
| Specify                       | 11        |
| Start                         | 13        |
| Testpat                       | 13        |
| Text                          | 13        |
| <b>USING MIASMA</b>           | <b>13</b> |
| <b>BINARY FILE FORMAT</b>     | <b>14</b> |
| <b>TYPICAL MIASMA PROGRAM</b> | <b>16</b> |

NOTE: If you want to prepare by hand test inputs to your circuit simulation, you needn't read this chapter.

### OVERVIEW

Miasma is a general purpose microcode assembler. It supports the definition of one or more fields which can be combined into instructions (or microwords). These instructions can then be combined into user programs. The length of each field is independently programmable, so that Miasma can be used to assemble microcode of arbitrary instruction width.

A flexible syntax, using symbols, expressions and default values, allows fields to be assigned values. A macro-expansion facility allows field assignment statements to be combined into user-defined macros with optional parameters. The use of these macros helps to increase the clarity of user programs.

### MIASMA USES

#### Logic Simulator Test Programs

The field-assignment facility of Miasma provides a convenient way of creating test vectors while preserving for the designer the symbolic naming of groups and assignment of a numeric value to each group. Many simulators (including Tegas) require test vectors which set each signal individually. Most designers, on the other hand, group signals together into buses, registers, or other logical sets. Using Miasma, the normally laborious translation process which is required at this interface is made simple and straightforward.

In many synchronous designs, various signals change at different times in the cycle. To support this, Miasma provides the ability to specify a cycle which results in a number of output patterns per instruction.

### Bit Slice or Finite State Machine Programs

Many designs have PROM (or RAM) driven state machines instead of random logic. Miasma provides an easy way to write programs for such designs with complete freedom to specify path widths, symbolic names, default values and labels. The resulting code can be downloaded into a PROM programmer or loaded into the Tegas simulation of the PROM/RAM.

### MIASMA INPUT

Miasma processes an ASCII-formatted input text file. It outputs a variety of files whose format depends on the options selected in the input. The Miasma source files are free-field, case insensitive text. The input is divided into statements separated by semicoions. A recommended practice is to have only one statement per line. A comment is heralded by an exclamation point ("!"). Text between the "!" and the end of the line is ignored. Blank lines are always ignored. Multiple spaces are treated as a single separator. Normally, one input statement to Miasma results in one output instruction.

### SYMBOLS

Symbols in Miasma are alphanumeric. They may be up to 12 characters long and may not start with a numeral. Symbols may represent Keywords (reserved names to Miasma), Labels, Field names, sequence values or macro names. Examples of legal symbols are:

Foo  
FooBar  
Test\_one  
A99  
TestLevel1

### NUMBERS

Numbers are signed 32 bit quantities. The syntax for a number is:

[<radix> "#"] ["+" | "-"] <digit> {<digit>}

Radix is an optional decimal (base 10) integer. If specified, it must be in the range 2-36. The radix is separated from the number by a "#" character. The sign is also optional. Digits are 0-9, a-z or A-Z. The default radix is used when no radix is included in the number. The default radix may be changed by the RADIX directive (see page 9). All radices are written in decimal. A number that does not start with a radix or a sign must begin with a digit in the 0-9 range. This rule means that 16#FF is always the number 255 (decimal), but even if you set the RADIX to 16, FF is a symbol and OFF is a number. Examples of legal numbers are:

```
1  
124  
-246  
+112256  
8#177570  
16#3FF  
2#101101110  
10#-25  
21#1FH2K
```

## EXPRESSIONS

Miasma supports the interchangeable use of numbers and expressions. Expressions may utilize the following operators:

|   |                |
|---|----------------|
| + | Add            |
| - | Subtract       |
| * | Multiply       |
| / | Integer Divide |
| & | Bitwise AND    |
|   | Bitwise OR     |
| % | Bitwise XOR    |

Expressions are evaluated from left to right. 32 bit arithmetic is performed. Miasma does not currently support expressions which contain parentheses.

An example of a legal expression is:

5 xor 1 \* 2 !is 8 (decimal)

## DIRECTIVES

"Directives" are commands to Miasma that modify the state of the assembly, but do not result in an instruction. Each directive is invoked by its keyword. Parameters, when needed, follow the keyword on the same line. One directive per line is allowed. This means that the end-of-line is a statement separator for directives, unlike instructions, which are separated by semicolons.

Miasma directives are presented alphabetically here. The examples indicate how and in what order they typically appear.

### Align

The Align directive sets the alignment of the next instruction constructed by Miasma. Align accepts one parameter and forces the next instruction address to be the next even multiple of the parameter.

**WARNING:** It is not advisable to use more than one Loc, Align, or Quad directive in a single instruction as the results will be determined by the order of the directives.

### Bitsperword

Miasma must be configured for the organization of the design or state machine for which testpatterns or code is to be created. The first item that must be specified is the width, in bits, of output that is needed. The width is specified with the Bitsperword directive. For example:

**BITSPERWORD 32**

Bitsperword may specify less than the number of bits per instruction, in the case where instructions consume more than one word of program memory. The default radix for Bitsperword is decimal, even if a Radix directive precedes the Bitsperword directive (but it can be overridden by specifying the radix with the parameter as in BITSPERWORD 16#1F).

### Cycle

The Cycle directive establishes the time increment used while constructing Tegas test patterns (see the Testpat directive). The Cycle directive accepts one parameter, the time increment (in arbitrary units).

When making Tegas test patterns, the time increment per cycle should be specified with the Cycle directive, which has a default radix of decimal. While the time units are unspecified, the delay calculation routines assume a time base measured in units of tenths of nanoseconds.

#### Endfields

The end of a block of field definitions is heralded by an Endfields directive. The Endfields directive accepts no parameters.

#### Endmacros

The end of a block of macro definitions is heralded by an Endmacros directive. The Endmacros directive accepts no parameters.

#### Endrepeat

The Endrepeat directive terminates the group of statements included in a repeat block. The Endrepeat directive accepts no parameters.

#### Endtext

The Endtext directive terminates a text block. The Endtext directive accepts no parameters.

#### Fields

The fields of the output data are defined with the Fields directive. The Fields directive accepts no parameters; however, it is followed by a block of field definitions. Each definition takes the form:

```
<FieldName> ":"  
    <Bit position of lsb> "," <Width> "," <Default Value>
```

An arbitrary number of definitions may be present. The list of definitions is terminated by the Endfields directive.

For example:

AluF: 0,4,\$17

specifies a 4 bit wide field called AluF whose least significant bit is bit 0 in the output. Bits are numbered from 0, with 0 the least significant bit. Bit numbers and widths use a default radix of 10, even if the Radix directive specified a different default. The <Default Value> field can be a numeric value (like the example, octal 17), or it may be OLD (use the value in the last instruction). The <Default Value> field uses the current default radix (set by the Radix directive) unless overridden with the "#" convention (as shown above). X or Z values can only be used if testpattern output is being generated; X and Z are meaningless in the state machine mode.

Field definitions may not overlap, but may extend beyond the length imposed by the Bitsperword directive. This is done when the instruction requires more than one word on the target machine.

It is possible to specify more than one format for the output word by having more than one Fields entry. The field boundaries in each of the Fields definitions do not have to correspond. For example:

```
BITSPERWORD 6
FIELDS
  Left: 0, 3, 0
  Right: 30, 3, 0
ENDFIELDS
```

```
FIELDS
  Big: 0,5,0
  Little: 5,1,0
ENDFIELDS
```

In this example, we have two formats for a 6-bit word, two 3-bit halves, and a 1-bit field next to a 5-bit field.

The same field name may be defined for use in more than one format. If so, however, the field definitions must be equivalent.

A "\$" symbol, when optionally prepended to the field name, will cause the field not to be displayed in the list file. This symbol is not part of the name and should not appear in subsequent field assignment statements.

Include

The Include directive allows a Miasma source program to Include a group of Miasma statements from another file. For example, the user can create one common set-up file and Include it into each test program. The filename of the file containing the Miasma statements is accepted as the only parameter to the Include directive. The input source scanner is redirected to take input from <filename> as if the contents of the Included file had been copied into the original source file at the point of the Include statement. Includes can be nested (i.e. an Included file may Include another file).

Leftfirst

The default order of output words is right-most (least significant word) first. The Leftfirst directive inverts this order, causing the most significant word to appear first in the output, followed by the least significant word. Bit 0 of each word is always the least significant bit.

Loadmem

If the state machine output is needed in a Tegas simulation, the Loadmem directive (with no parameters) will produce a correctly formatted output file with extension ".TS". If no Testpat or Loadmem is given, the output will be placed, in binary form, in a file with ".BIN" extension. A listing file with extension ".LIST" is generated in all cases. The Loadmem directive accepts no parameters.

Loc

The Loc directive sets the location of the next instruction constructed by Miasma. It accepts one parameter, the desired address.

**WARNING:** It is not advisable to use more than one Loc, Align, or Quad directive in a single micro instruction as the results will be determined by the order of the directives.

Macro

Macros can be used to make the source code more readable. Macros are defined by a Macro directive. The syntax of the Macro directive is:

```
"MACRO" <cr>
  {<Macro Name> ":" "" <replacement string> "") <cr>
"ENDMACRO" <cr>
```

During instruction processing, each occurrence of a <Macro Name> is replaced by the <replacement string> in a manner that allows for nested macros. Note that there may be a maximum of 255 characters in a source line at any time. If there are more than 255 characters when all the macros are expanded, a run time error will occur. Macros are restricted to one line. The word "NULL" is replaced by nothing so macros can be defined just for readability. The end of macro definitions is heralded by Endmacros, just like the Endfields directive ends the field definitions. Symbols with values may be declared with macros and then used in expressions.

Macros may have parameters. In the definition of the macro, the points where parameters are to be substituted are indicated with a "\$" and a digit (e.g. \$1, \$2, \$3). In an invocation of a macro, the actual parameters are supplied enclosed in parentheses, separated by commas. The parameters are substituted as strings into the body of the macro. For example:

```
MACROS
Zero:    '0'
OutFifo: 'YSel = 7, OSel = 0'
DMAReq:  'YSel = 0, OSel = 0'
True:    'CC = 0'
False:   'CC = 1'
IF:      'NULL'
GoTO:    'JumpL = 3, JumpH = 1, Label = $1' !Note parameter
T:       'OutTime = 1, Time = $1'
ENDMACROS
```

Using these macros:

```
OutFifo, Data = Zero;          !Write data to FIFO
IF True GoTo(NextWord);       !Parameter substitution
DMAReq, T(2);
```

Is the same as:

```
YSel = 7, OSel = 0, Data=0;
CC = 0, JumpL = 3, JumpH = 1, Label = NextWord;
YSel = 0, OSel = 0, OutTime = 1, Time = 2;
```

Maxaddress

The maximum address that will be emitted (i.e. the maximum length of the program) is specified using the Maxaddress directive (with a parameter). Using the Maxaddress directive, Miasma can verify that a given program will fit in the target memory.

Radix

The Radix directive is used to specify the default radix for numbers and expressions. When the following line is included in the source file:

RADIX 8

Miasma assumes a default radix of octal for each number and expression appearing in the source program (not including arguments of directives) until the next Radix directive is encountered. The Radix directive accepts one parameter, the desired value for the new default radix. All values of Radix from 1-27 are accepted although the most common are 2 (binary), 8 (octal), 10 (decimal) or 16 (hexadecimal).

Quad

The Quad directive forces the low two bits of the micro address to be the specified value. The address is incremented until the lower two address bits match the value specified in the Quad directive.

The syntax of the Quad directive is:

"QUAD" "(" <value> ")"      where  $0 \leq \text{value} \leq 3$ .

**WARNING:** It is not advisable to use more than one Loc, Align, or Quad directive in a single micro instruction as the results will be determined by the order of the directives.

Repeat

Repeat causes the following lines up to the next Endrepeat to be replicated a number of times. Repeat accepts one parameter, the number of repetitions to make. Nested Repeats currently are not supported.

Rightfirst

The default order of output words is right-most (least significant word) first. The Rightfirst directive restores the output word ordering to its default order. This is only needed after a Leftfirst directive has been given. Bit 0 of each word is always the least significant bit.

Sequence

The Sequence directive provides a convenient way of sequencing a field through a set of values, possibly repeating values periodically. For example,

```
SEQUENCE foo(1,2,4,8#10,8#20,8#40,8#100,8#200)
```

will create a sequence. If you then use foo as a value (right side of an field=value assignment), it will first give you 1, then the next time you get a 2, then a 4, etc.

Sequences are more useful with a Repeat loop, where you may repeat a number of statements multiple times. For example:

```
REPEAT 5
    LoadIt(foo);
    Count;
ENDREPEAT
```

using the definitions of the counter and foo above results in the equivalent of

```
Loadit(1);
Count;
LoadIt(2);
Count;
LoadIt(4);
Count;
LoadIt(10);
Count;
Loadit(20);
Count;
```

Settime

The Settime directive establishes the time used in constructing the next Tegas test pattern (see the Testpat directive). The Settime directive accepts one parameter - the desired time (in arbitrary units).

Skip

The Skip directive establishes a time offset used while constructing the next Tegas test pattern (see the Testpat directive). The Skip directive accepts one parameter - the time increment (in arbitrary units).

Specify

The Specify directive allows the user to control the time at which signals change within a cycle. A Specify directive must be present if a Testpattern is being generated.

The lines in the Specify directive have the field name terminated by a ":" , and value/time pairs separated by commas. The times specify durations. For example, "name:+/50,Z/150,\*100" means that the cycle lasts for 300 time units and is divided into 3 sections of 50, 150 and 100 time units respectively.

The values which may be assigned to a signal are:

0: logic 0

1: logic 1

X: Undefined

Z: TriState

\*: The value specified in the source line for this instruction

+: The value specified in the source line for the previous instruction

The times for a given field must add up to the Cycle time. Specify times use a default radix of decimal.

Let us take, for example, a simple counter. The counter has a Clock signal which defines the period of the "machine". The other input signals (Reset, Enable, Load, Data) have various set-up and hold times within the cycle which must be maintained. The Miasma source for this example might be:

```
RADIX 8
MAXADDRESS 16
BITSPERWORD 8
TESTPAT
CYCLE 300      !clock cycles at 30.0 ns
FIELDS
Clock: 0,1,0
Reset: 1,1,0
Enable:2,1,1
Load:  3,1,0
Data:  4,4,0
ENDFIELDS
```

```
SPECIFY
Clock: 1/150, 0/150  !Go high at start of cycle,
                      !low at 15.0 ns
Reset: +/50, */250   !Hold time of 5.0 ns
Enable:+/50, */250
Load:  +/50, */250
Data:  +/50, Z/150, */100 !5ns hold, tristate for 15,
                           !then data
ENDSPECIFY
```

```
MACROS
Clear: 'Reset=1'
LoadIt: 'Load=1,Data=$1'
Count: 'Enable=1'
Wait:  'Enable=0'
ENDMACROS
```

```
Start: Clear;      !expands to Reset=1.
                      !Enable and Load default to 0
Wait;
Wait;
Count;           !Enable=1
Count;
Count;
Count;
Count;
LoadIt(7); !Load=1, Data=7 (octal)
Count;
LoadIt(17); !Load=1, Data=17 (octal)
Count;
```

The example shows the counter clearing, waiting 2 cycles, incrementing to 4, loading a 7, incrementing to 10 (octal), loading a 17, and incrementing to 0. There will be 4 patterns output for each line: one at 0 (the start of a cycle), at 50 time units later (Reset, Enable, Load and Data change here), at 150 (the clock changes), and at 200 (Data changes to its specified value).

Start

The Start directive establishes the time used in constructing the first Tegas test pattern (see the Testpat directive). The Start directive accepts one parameter - the desired time (in arbitrary units).

Testpat

Miasma defaults to a mode which produces binary output words for a state machine. If Tegas test patterns are required, the Testpat directive (no parameters) should be used. The Testpat directive causes the output to go to a correctly formatted output file with extension ".T5". If no Testpat or Loadmem is given, the output will be placed, in binary form, in a file with ".BIN" extension. A listing file with extension ".LIST" is generated in all cases. The Testpat directive accepts no parameters.

Text

When outputting testpatterns or Loadmem (i.e. Tegas T5 files), it may be necessary to intermix other T5 file commands with the patterns that Miasma generates.

The TEXT command is used to pass lines unchanged to the T5 file. All lines between Text and Endtext are written. See the example at the end of this document for a typical prologue/epilogue sequence using Text/Endtext.

USING MIASMA

The fundamental instruction format is:

```
[<label> ":" ] <field> "=" <value> { "," <field> "=" <value> } ";"
```

This means you specify a label, separated from the rest of the statement by a ";", and assign values to fields. The <field=value> phrases are separated by commas, and the statements are separated by a semicolon. A statement may be spread out over more than one line. The field assignments can come in any order. The values may be numbers or, if Testpat was specified, may be "X" or "Z". If you do not specify an assignment for the field, the default given in the <Default Value> of the Field definition is used.

BINARY FILE FORMAT

The binary file is a file of integers. The integers are grouped as address and data. The first integer is the address. The data is contained within one or more integers. Each address-data pair is followed immediately by other address-data pairs. Each integer contains up to 16 bits of data for the specified address. The number of data words depends on the length of the instruction (number of bits in the Field specification) and the word length in the target machine (parameter of Bitsperword).

$$N = (\text{BITSPERWORD} + 15) / 16.$$

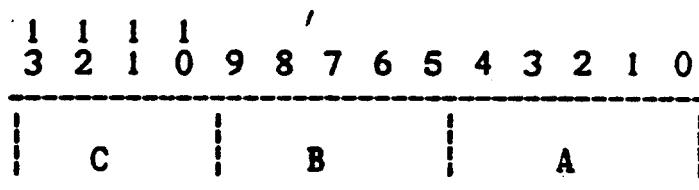
If the length of the instruction exceeds the Bitsperword, several address-data pairs are necessary for a single instruction.

The order of the Bitsperword group is controlled by the Leftfirst and Rightfirst directives. In Leftfirst the leftmost (highest numbered) bits are emitted first. In Rightfirst the rightmost (lowest numbered) bits are emitted first.

Example:

Radix 10  
BitsPerWord 9  
Field  
A: 0,5,0  
B: 5,5,0  
C: 10,4,0  
Endfields

microword is:



if Rightfirst, it is broken down into two target words:

location 0: | x x x x x x x x 8 7 6 5 4 3 2 1 0 |

1: | x x x x x x x x x x x x 13 12 11 10 9 |

if Leftfirst, it is broken down into two target words:

location 0: | 13 12 11 10 9 8 7 6 5 x x x x x x x |

location 1: | 4 3 2 1 0 x x x x x x x x x x x |

TYPICAL MIASMA PROGRAM

The following Miasma source programs show a Tegas initialization sequence for a typical gate array design. The first program ("DEFINITIONS.MAS") is a set of definitions which are then used by the second program ("INITIALIZATION.MAS").

## DEFINITIONS.MAS

TestPat

Cycle 170

BitsPerWord 95

Radix 8

|   |    |    |      |    |    |      |    |     |    |    |     |    |
|---|----|----|------|----|----|------|----|-----|----|----|-----|----|
| ! | 24 | 8  | 3    | 24 | 8  | 1    | 8  | 4   | 2  | 4  | 8   | 1  |
| ! | AR | X  | ASel | BR | Y  | BSel | Z  | AOp | F  | SF | IOD | NI |
| ! | 71 | 63 | 60   | 36 | 28 | 27   | 19 | 15  | 13 | 9  | 1   | 0  |

## Fields

| ! | Name  | Pos, | Len, | Def |
|---|-------|------|------|-----|
|   | AR:   | 71,  | 24,  | 0   |
|   | X:    | 63,  | 8,   | 0   |
|   | ASel: | 60,  | 3,   | 0   |
|   | BR:   | 36,  | 24,  | 0   |
|   | Y:    | 28,  | 8,   | 0   |
|   | BSel: | 27,  | 1,   | 0   |
|   | Z:    | 19,  | 8,   | 0   |
|   | AOp:  | 15,  | 4,   | 0   |
|   | F:    | 13,  | 2,   | 0   |
|   | SF:   | 9,   | 4,   | 0   |
|   | IOD:  | 1,   | 8,   | 0   |
|   | NI:   | 0,   | 1,   | 0   |

EndFields

## Specify

|      |       |       |       |
|------|-------|-------|-------|
| IOD  | Z/40, | */25, | Z/105 |
| SF   | */65, | Z/105 |       |
| F    | */65, | Z/105 |       |
| AOp  | */65, | Z/105 |       |
| Z    | */65, | Z/105 |       |
| BSel | */65, | Z/105 |       |
| Y    | */65, | Z/105 |       |
| BR   | Z/65, | */105 |       |
| ASel | */65, | Z/105 |       |
| X    | */65, | Z/105 |       |
| AR   | Z/65, | */105 |       |
| NI   | 0/65, | */105 |       |

EndSpecify

## Macros

## ! Special Functions

|              |   |
|--------------|---|
| LongK:       | 'Y = \$1, Z = \$2, BSel = 1, F = 0, SF = 0' |
| StackReset:  | 'F = 0, SF = 2'                             |
| TOSgets:     | 'F = 0, SF = 3'                             |
| Push:        | 'F = 0, SF = 4'                             |
| Pop:         | 'F = 0, SF = 5'                             |
| LoadMul:     | 'F = 0, SF = 11'                            |
| LoadOp:      | 'F = 0, SF = 12'                            |
| BPCgets:     | 'F = 0, SF = 13'                            |
| IOB:         | 'Z = \$1, F = 0, SF = 17'                   |
| VictimLatch: | 'F = 1, SF = 0'                             |
| MulStep:     | 'F = 1, SF = 1'                             |
| MQGets:      | 'F = 1, SF = 2'                             |
| BaseGets:    | 'F = 1, SF = 3'                             |
| GetsMQ:      | 'F = 1, SF = 4'                             |
| PushLongK:   | 'Y = \$2, Z = \$1, BSel = 1, F = 1, SF = 5' |
| Fetch4R:     | 'F = 1, SF = 10'                            |
| Store4R:     | 'F = 1, SF = 11'                            |
| Fetch4:      | 'F = 1, SF = 12'                            |
| Store4:      | 'F = 1, SF = 13'                            |
| Fetch2:      | 'F = 1, SF = 14'                            |
| Store2:      | 'F = 1, SF = 15'                            |
| Fetch:       | 'F = 1, SF = 16'                            |
| Store:       | 'F = 1, SF = 17'                            |
| NextInst:    | 'NI = 1'                                    |

**! Arithmetic**

|           |            |
|-----------|------------|
| A:        | 'Aop = 0'  |
| B:        | 'Aop = 1'  |
| notA:     | 'Aop = 2'  |
| notB:     | 'Aop = 3'  |
| and:      | 'Aop = 4'  |
| andNot:   | 'Aop = 5'  |
| nand:     | 'Aop = 6'  |
| or:       | 'Aop = 7'  |
| orNot:    | 'Aop = 10' |
| nor:      | 'Aop = 11' |
| xor:      | 'Aop = 12' |
| xNor:     | 'Aop = 13' |
| sum:      | 'Aop = 14' |
| sumCarry: | 'Aop = 15' |
| dif:      | 'Aop = 16' |
| difCarry: | 'Aop = 17' |

**! ASelect Sources**

|         |                       |
|---------|-----------------------|
| Shift:  | 'ASel = 0, AR = \$1'  |
| NextOp: | 'ASel = 1, AR = \$1'  |
| IORead: | 'ASel = 2, IOD = \$1' |
| MDI:    | 'ASel = 3, AR = \$1'  |
| MDX:    | 'ASel = 4, AR = \$1'  |
| UState: | 'ASel = 5'            |
| ARAM:   | 'ASel = 6, AR = \$1'  |
| ETOS:   | 'ASel = 7'            |

**! BSelect Sources**

|           |                      |
|-----------|----------------------|
| BRAM:     | 'BSel = 0, BR = \$1' |
| Constant: | 'BSel = 1, Y = \$1'  |

**EndMacros**

(End of DEFINITIONS.mas)

The preceding program has been written in such a way that several programs may include it. This permits a change in a definition (if, for example, a change is made to the hardware) to be easily propagated to a number of test programs. A typical test program is given next. This program, "INITIALIZATION.MAS", uses the definition program given above.

## INITIALIZATION.MAS

include Definitions

Text

MODE 2;  
SUPPRESS 34, 35, 36;  
LOAD;  
IVECTOR

AR.24, AR.23, AR.22, AR.21, AR.20, AR.19, AR.18, AR.17, AR.16, AR.15,  
AR.14, AR.13, AR.12, AR.11, AR.10, AR.9, AR.8, AR.7, AR.6, AR.5, AR.4,  
AR.3, AR.2, AR.1,  
XAD.8, XAD.7, XAD.6, XAD.5, XAD.4, XAD.3, XAD.2, XAD.1,  
ASEL.3, ASEL.2, ASEL.1,  
BR.24, BR.23, BR.22, BR.21, BR.20, BR.19, BR.18, BR.17, BR.16, BR.15,  
BR.14, BR.12, BR.11, BR.10, BR.9, BR.8, BR.7, BR.6, BR.5, BR.4,  
BR.3, BR.2, BR.1,  
YAD.8, YAD.7, YAD.6, YAD.5, YAD.4, YAD.3, YAD.2, YAD.1,  
B,  
Z.8, Z.7, Z.6, Z.5, Z.4, Z.3, Z.2, Z.1,  
AOP.4, AOP.3, AOP.2, AOP.1,  
F.2, F.1,  
SF.4, SF.3, SF.2, SF.1,  
IOD.8, IOD.7, IOD.6, IOD.5, IOD.4, IOD.3, IOD.2, IOD.1,  
NI;

SIMSETUP:

CHANGE CLOCK 0 0;  
CHANGE CLOCK 1 60, 50000, 170;  
CHANGE CLOCK 0 165, 50000, 170;  
CHANGE ABORT 1 0;  
CHANGE RESET 1 0;  
CHANGE RESET 0 680;

END;

TESTPATT;

Endtext

ARam(0), A;  
ARam(0), A;  
ARam(0), A;  
ARam(0), A;  
ARam(0), A;  
ARam(0), A, StackReset;  
BPCGets, ARam(0), A;  
BaseGets, ARam(0), NotA;

```
text
ENDBLOCK "testpatt";
SAVE ALL;
SIMULATE 1359;
CHECKPOINT;
ENDBLOCK;
endtext
```

The use of the Text/Endtext directives allows arbitrary text to be inserted in the output file. In this case, Tegas command blocks have been created using this facility. These command blocks are not interpreted by Miasma; they are simply passed verbatim to the output file.

Finally, the output file (a ".T5" file, because of the Testpat directive) from this example is given below.

#### INITIALIZATION.T5

```
MODE 2;
SUPPRESS 34 35 36;
TRISTATE 78 87;
LOAD;
IVECTOR
```

```
AR.24, AR.23, AR.22, AR.21, AR.20, AR.19, AR.18, AR.17, AR.16, AR.15,
AR.14, AR.13, AR.12, AR.11, AR.10, AR.9, AR.8, AR.7, AR.6, AR.5, AR.4,
AR.3, AR.2, AR.1,
XAD.8, XAD.7, XAD.6, XAD.5, XAD.4, XAD.3, XAD.2, XAD.1,
ASEL.3, ASEL.2, ASEL.1,
BR.24, BR.23, BR.22, BR.21, BR.20, BR.19, BR.18, BR.17, BR.16, BR.15,
BR.14, BR.13, BR.12, BR.11, BR.10, BR.9, BR.8, BR.7, BR.6, BR.5, BR.4,
BR.3, BR.2, BR.1,
YAD.8, YAD.7, YAD.6, YAD.5, YAD.4, YAD.3, YAD.2, YAD.1,
B,
Z.8, Z.7, Z.6, Z.5, Z.4, Z.3, Z.2, Z.1,
AOP.4, AOP.3, AOP.2, AOP.1,
F.2, F.1,
SF.4, SF.3, SF.2, SF.1,
IOD.8, IOD.7, IOD.6, IOD.5, IOD.4, IOD.3, IOD.2, IOD.1,
NI;
```

```
SIMSETUP;  
  CHANGE CLOCK 0 0;  
  CHANGE CLOCK 1 60, 50000, 170;  
  CHANGE CLOCK 0 165, 50000, 170;  
  CHANGE ABORT 1 0;  
  CHANGE RESET 1 0;  
  CHANGE RESET 0 680;
```

**END;**

## TESTPATT;

MIASMA

28 Dec 82

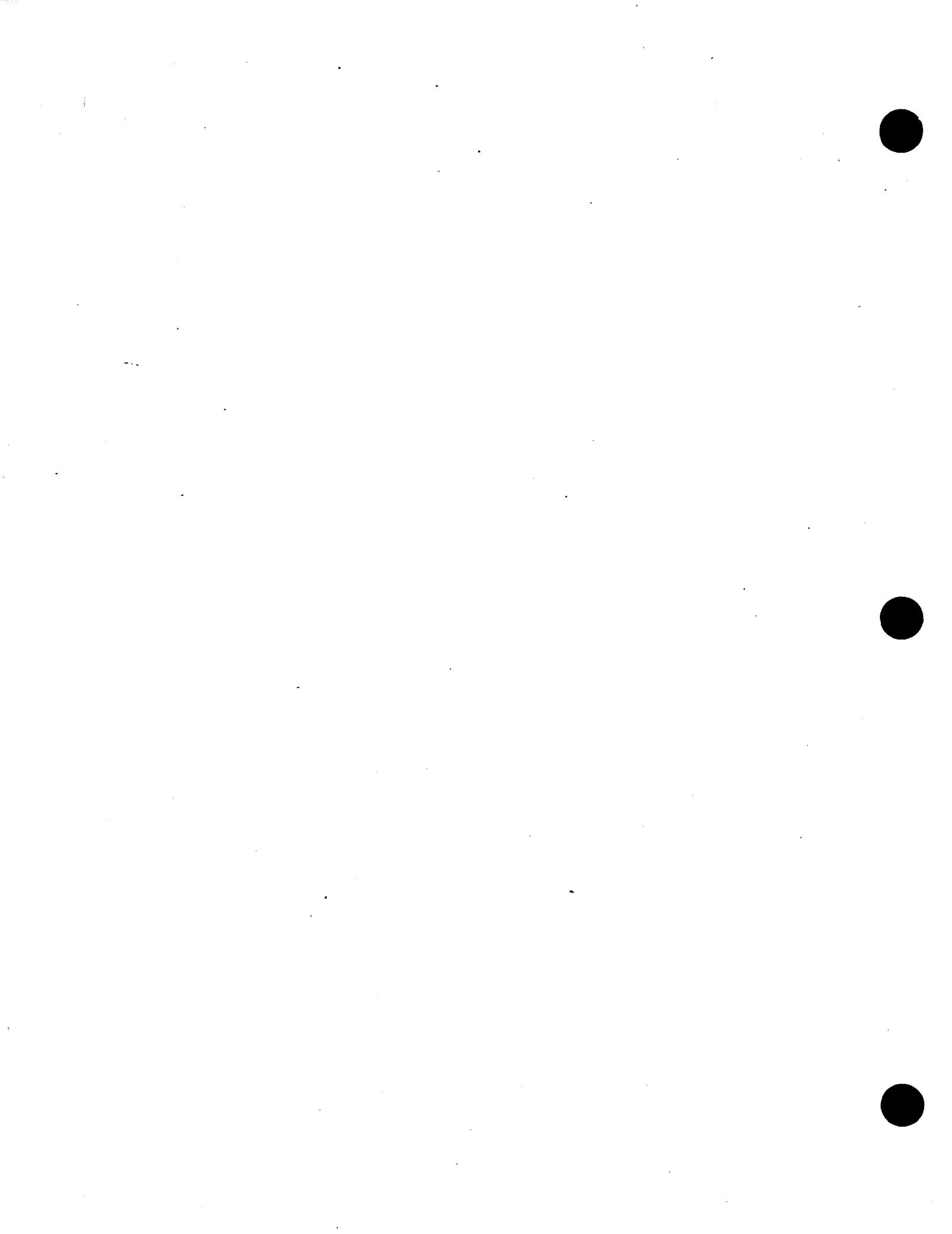
```
00000010010011ZZZZZZZ0 $  
ZZZZZZZZZZZZZZZZZZ00000000110ZZZZZZZZZZZZZZZZZZ0000000000000000 1230  
000000100100110000000000 $  
00000000000000000000000000000000ZZZZZZZZ00000000000000000000000000000000 1255  
ZZZZZZZZZZZZZZZZZZZZZZZZZ0 $  
ENDBLOCK "TESTPATT";  
SAVE ALL;  
SIMULATE 1359;  
CHECKPOINT;  
ENDBLOCK;
```

**CHAPTER 8**  
**VAX PROGRAMS**



CHAPTER EIGHT  
VAX PROGRAMS

|                        |   |
|------------------------|---|
| VAXFTP                 | 1 |
| TEGAS                  | 2 |
| The TDL Preprocessor   | 3 |
| The TDL Libraries      | 4 |
| THE SIMULATOR          | 4 |
| Tegas Files            | 5 |
| GENTCF                 | 7 |
| Files Used by GenTCF   | 7 |
| Running GenTCF         | 7 |
| CREATING A SIGNAL FILE | 9 |



This chapter provides an introduction to the use of VAX-related PERQ System D/L programs. It is neither a complete user's manual nor a tutorial; the chapter simply outlines the major data flow and files with which the user must be familiar.

### VAXFTP

This section describes the use of VAXFTP. VAXFTP is a VAX Pascal program which implements the PERQ FTP protocol. It can be used with a PERQ running PERQ FTP to transfer files in both directions.

VAXFTP runs on a VAX/VMS system and uses the terminal line for communicating with the PERQ. This imposes some severe restrictions on the program since all terminal output is going to be interpreted by the PERQ as FTP packets, while all terminal input is assumed to be FTP packets sent by the PERQ. Thus, no error messages are printed by the VAXFTP program. If a message might be of interest to the user, it is sent as an Abort packet so that the FTP running on the PERQ can print it on the PERQ display. In addition, the VAX terminal characteristics must be temporarily changed in order to pass the full 8-bit ASCII character set.

A command file, VAXFTP.COM, is used to set up the necessary terminal characteristics, and run VAXFTP. It places the terminal in NOECHO, PASSALL, and EIGHT\_BIT modes and then runs VAXFTP. After VAXFTP exits, the command file resets the terminal to its normal mode.

To run VAXFTP, make sure the RS232 port on the PERQ is connected to a VAX terminal line (either directly or through a patch panel or modem). Run Chatter on the PERQ and login to the VAX. Now type

**@VAXFTP**

This invokes the VAXFTP.COM command file. Now quit Chatter by typing CTRL/R, CTRL/Q. Do not type anything else after invoking the command file, since it will be interpreted by VAXFTP as FTP packets coming from the PERQ.

On the PERQ, type

FTP  
MODE VAX

This starts FTP and instructs it that the machine on the other end is a VAX. Setting Mode = VAX also sets the default baud rate to 4800 baud. If your terminal line to the VAX is not a 4800 baud line you must set the baud rate with the Baud command in FTP. Do this after you type MODE VAX.

Now, you can use FTP just as though there were a PERQ running FTP in POLL mode at the other end. When you're through say

QUIT

to FTP so that FTP can tell VAXFTP to finish up and return your VAX terminal line to the proper state. If you forget, and CTRL/C out, you can run FTP on the PERQ again, and say MODE VAX and then type QUIT.

If you want to do anything else on the VAX you may run Chatter and reconnect to your job, otherwise log it out when you're through.

Notes:

1. VAXFTP does not implement timeouts.
2. All files on the VAX end are assumed to be 'ordinary' files. They are not allowed to be comprised of fixed length records. To do this one must change the parameters to the Open procedure in VAXFTP so that the file's type is specified as RECORD\_LENGTH := n. For details on this see the VAX-11 Pascal Language Reference Manual (DEC Order No. AA-H484B-TE) Section 7.1.5.

TEGAS

One function of PERQ System D/L is to provide input for Tegas. Tegas is run on a VAX and provides its own set of documentation. The PERQ System D/L user should become acquainted with this documentation. The following is a short overview of how Tegas fits into PERQ System D/L and some hints on using Tegas.

The Tegas system is best viewed as two separate programs: TDL, a preprocessor which processes network descriptions provided by PERQ System D/L's DA program; and Tegas, which loads the compiled network description and performs the actual test generation, fault detection and simulation.

### The TDL Preprocessor

The first phase of Tegas is that of describing and compiling the description of the circuit network. This is the task of writing in TDL a module or set of modules which implement the desired function. A TDL module is a text file with an extension of '.TDL' which contains a list of inputs and outputs, other modules or primitive elements called by it, and some optional delay information. This '.TDL' file is created by the DA program on the PERQ and then transferred to the VAX via an Ethernet or RS232 connection.

Primitives used by these modules are those provided by Tegas. They are well documented and reside on the master library (in the [DL.tegas] directory) as TDLMAS.DAT. Other modules called by this module must reside on the user's library, called TDLUSR.DAT on the user's own VAX directory.

The '.TDL' file is compiled by the TDL preprocessor. TDL creates '.PIT', '.NAM', and '.OUT' files and updates the user library (TDLUSR.DAT). The '.OUT' file contains all of the information from the '.TDL' file as well as error notification and messages from the TDL program. All of the information in the '.OUT' file is for user notification and is not needed as input to another program. However, the other files ('.NAM', '.PIT' and TDLUSR.DAT) are used by the Tegas simulator.

To run the TDL preprocessor, copy the default Tegas user's login profile from [dl.tegas]TEGAS.PRO into the Tegas user's LOGIN.COM file.

The preprocessor can then be invoked via the TDL command. If it is invoked by 'TDL FileName' then TDL will look for input in FileName.TDL, and write all output to FileName.OUT. If it is invoked by 'TDL' followed by a carriage return, then TDL prompts at the terminal for input and writes output directly to the terminal. If it is invoked by 'TDL FileName' followed by anything on the same line, TDL will read from FileName.TDL and write to the terminal.

This last method is useful to quickly verify that a short .TDL file is correct.

The TDL Libraries

Using the "DIR:" construct in TDL, you can specify a directory within the user library where the compiled module (TDLUSR.DAT '.PIT', '.NAM') is to reside. This can be a useful mechanism for separating TDL modules, describing cell libraries of different vendors, project chips, etc.

The user can use the VAX/VMS directory structure to reflect design organization so that the TDL sources for a given module can be easily located. Thus, if the current project was partitioned into three separate designs (named ALU, Video, and I/O), the user could define three separate directories within the TDL user library on [dl.projectname]TDLUSR.DAT as well as three separate VAX/VMS subdirectories [dl.projectname.Video], [dl.projectname.ALU] and [dl.projectname.IO]. These would be the homes of the TDL files for each of these projects. In addition, a separate subdirectory for the TDL descriptions of a vendor's cell library could be defined.

THE SIMULATOR

After a successful TDL run (i.e. no compilation errors in the '.OUT' file), the Tegas simulator can be run to simulate a design, generate test vectors, or do fault detection. The Tegas simulation program uses the '.PIT', '.NAM', and the TDLUSR.DAT files (all created by the TDL program), the TDLMAS.DAT master library, and a '.TS' file (created by the PERQ System D/L) as input. The '.TS' file is a file of test vectors for the design. The output of the Tegas simulator is a '.SIM' file.

If the default Tegas profile has been copied to the user's LOGIN.COM file, the Tegas simulator can be invoked by typing 'Tegas'. Typing 'Tegas FileName' will read FileName.TS and write the file FileName.Sim. If the file name is omitted, then the terminal will be used for both input and output. In either case, Tegas will prompt for the name of the '.PIT' and '.NAM' files.

The '.PIT' and '.NAM' files that should be used for simulation are those produced by the TDL program for this design. Until the user is familiar with Tegas simulation (or has an extensive set of test patterns in a file), the simulator can be run interactively by typing 'Tegas' followed by a carriage return.

To verify that the design is logically correct, use a mode 1 simulation. This will use the nominal delays specified in the TDL files and is the fastest simulation mode. It is used to

make sure the network has been correctly coded. Fault detection and extensive timing simulation and test vector generation are performed in mode 2. Consult the Tegas User Manual and the Tegas Reference Manual for a complete list of Tegas simulation commands and functions.

The Tegas system usually resides on the [dl.TEGASI] directory on the VAX. Also on the [dl.TEGASI] directory is the master library (TDLMAS.DAT) which contains definitions for the Tegas primitive elements.

### Tegas Files

The following summarizes the various files with which a Tegas user might need to be familiar.

#### [DL.TEGAS]TEGAS.PRO

This is the Tegas user's default profile. It should be copied into the user's LOGIN.COM file before using Tegas.

#### [DL.TEGAS]TEGAS.COM [DL.TEGAS]TDL.COM

These files are command files which make running Tegas and TDL much easier. They hide the FORTRAN file assignments which are necessary and allow the user to specify as little information as possible to run the preprocessor or simulator. These are invoked automatically when you type 'Tegas' or 'TDL' if you have the Tegas profile installed on your account.

#### [DL.TEGAS]TDLMAS.DAT

This is the TDL master library containing the TDL primitive elements. It is never written into and is shared by all Tegas users.

#### TDLUSR.DAT

This is the TDL user library which resides on the user's own directory (or that of the project). It contains all modules that the user defines and compiles with TDL. It can be partitioned into directories as described in the introduction above.

**\*.TDL**

These are TDL preprocessor input files prepared by the user.

**\*.OUT**

These are TDL preprocessor output files generated by TDL. They contain the text of the .TDL input file along with the preprocessor's prompts and output.

**\*.TS**

These are Tegas simulation input files prepared by the user.

**\*.SIM**

These are Tegas simulation output files generated by the Tegas simulator. They contain input and output test vectors, fault information, timing information, etc.

**\*.PIT****\*.NAM**

These are written by the TDL preprocessor and read by the Tegas simulator. If you use the standard Tegas login profile these files will have the same names as the .TDL files with which they belong.

In addition, there are a number of FORTRAN (FOR0xx.DAT) files which are written and read by the simulator and preprocessor. These are never referred to by name and are disguised from view by the TDL and Tegas command files. For reference purposes they are listed below along with an indication of their purpose.

|            |   |
|------------|---|
| FOR001.DAT | Default PIT File                                    |
| FOR007.DAT | Used for Checkpoint/Restart                         |
| FOR007.DAT | Equivalent Faults File                              |
| FOR008.DAT | Master Fault File                                   |
| FOR010.DAT | Alternate Fault File                                |
| FOR011.DAT | Detected Faults File                                |
| FOR012.DAT | TCF File  |
| FOR013.DAT | Used for MergeSave & Written by<br>SaveDump command |
| FOR014.DAT | Used for MergeSave                                  |
| FOR014.DAT | Fault Catalog File                                  |
| FOR015.DAT | Control File  |
| FOR018.DAT | RAM/ROM File  |

FOR019.DAT  
FOR020.DAT  
FOR021.DAT

PLA Module Library  
Save File  
Hierarchical Names File

The Save File (FOR020.DAT) is used by the Display, Save and Fileinput commands. (FOR013.DAT) is written by the Tegas Savedump command.

### GenICF

GenICF is a VAX program that is used to get delay information from the PERQ System D/L timing calculators into a Tegas simulation. GenICF reads a ".Delay" file from the PERQ, ".NAM" and ".TCF" files from the VAX and generates a new ".TCF" file. For details on the form of a ".TCF" file and the Tegas and TDL commands needed to use and generate TCF see the Tegas documentation. This section describes how to use the GenICF program.

#### Files Used by GenICF

GenICF needs to have a number of files on the VAX to perform its function. The required files are:

- The NAM file for the design.
- The original TCF file generated by TDL.
- The file of delay information from the PERQ.

GenICF defaults any extensions for filenames without extensions. The default extensions used by GenICF are:

- .TCF A TCF file.
- .DLY A PERQ .Delay file.
- .NAM A Tegas .NAM file.

#### Running GenICF

To run GenICF execute the VAX command:

GenICF

GenTCF is a command, defined in the PERQ System D/L Login.Com, that will run the program GenTCF.

Once the program starts to execute it will ask a number of questions. The default answers to these questions will be found enclosed in square brackets "[ ]". To use the default answer type a single <cr>. Answers supplied to questions (in either UPPER or lower case) will override the defaults.

The first question asked by GenTCF is the Root Name of the files. This name will be used to generate all other default file names. There is no default for this question. For the rest of this document we will assume that the root name is "FOO".

The second question is the name of the TCF file that was generated by Tegas. This would default to FOO.TCF.

The third question is the name of the output TCF file. This will default to the same value as the name of the input file. If you do not want to overwrite the original file you can supply a different output file name.

The fourth question is the name of the file that contains the delay information. This file will have a PERQ name of FOO.Delay. When you FTP FOO.Delay to the VAX you must provide a remote name of FOO.DLY.

The fifth question asked by GenTCF is the name of the Tegas .NAM file. This .NAM file MUST match the input .TCF file. This means that you CANNOT change the TDL for the design between the time that you run TDL and GenTCF.

The sixth question is whether or not you would like to have a data file. A positive response will result in a prompt for the file name. This file will contain information about the number of gates used, the different types of gates and other circuit information. It is not usually necessary to produce this file.

The next question is which set of delay numbers is to be used. DA provides MIN, MAX, and TYP numbers for all gates in a design. Your answer to this question will determine which of these to use.

The last question determines which time base to use for the simulation. At the current time the values that are in the .DLY file, 'DesignName.Delay' on the PERQ, are in tenths of nano-seconds.

After all of the questions have been answered, GenTCF will start to create the NewTCF file. As it reads each part of the input TCF file it will print a message about what it is doing. While reading the Pin Occurrence Table it will place the new delays into the output file. GenTCF will print the name of each pin as it processes it. GenTCF will also tell you what type of pin it is currently dealing with. If there is no information printed about the pin type then that pin is an output pin.

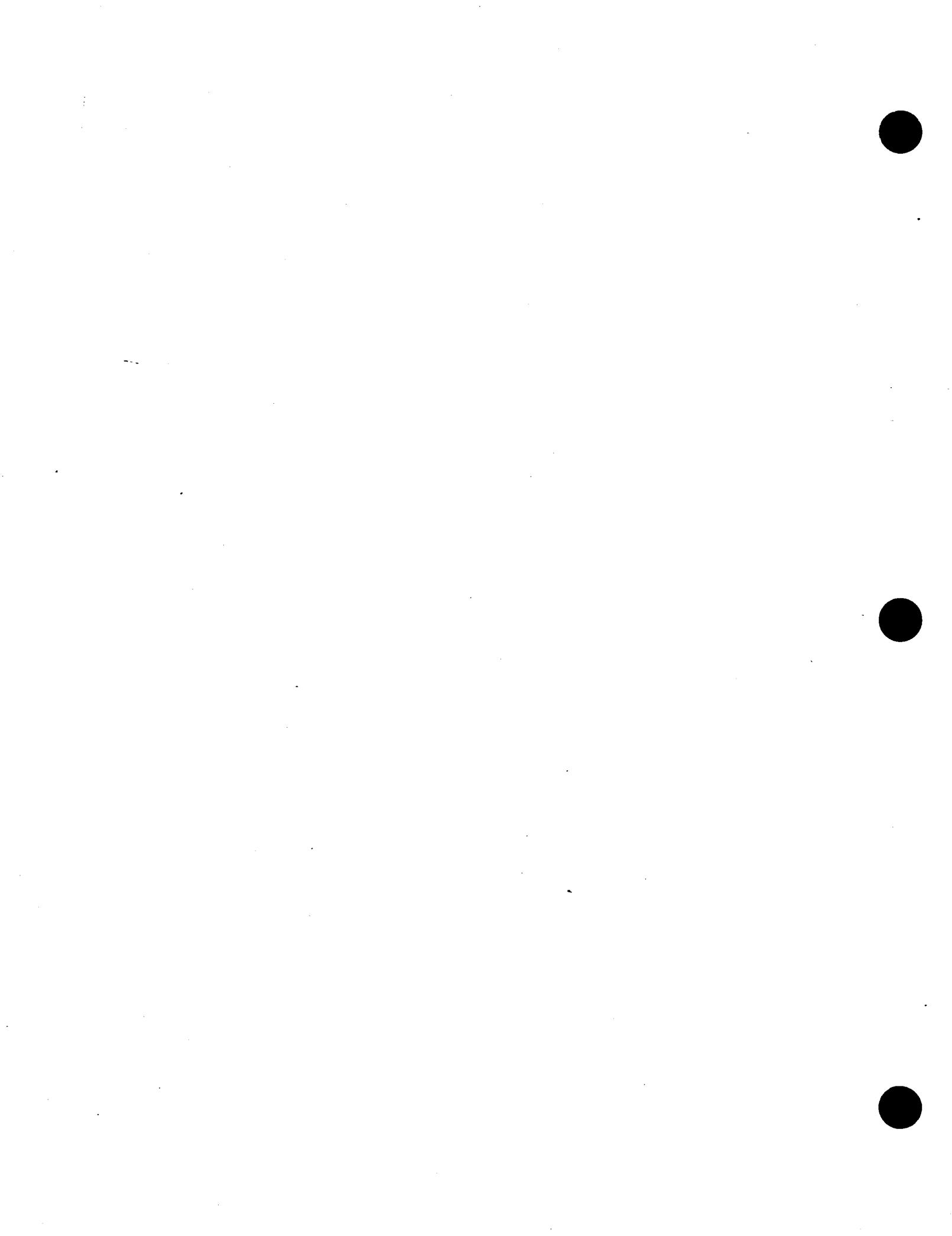
### CREATING A SIGNAL FILE

The program SIG on the VAX is used to convert the output of a Tegas run (a '.SAV' file) into a signal file (a '.SIG' file). A '.SAV' file contains the values of all selected signals within a specified time period of a simulation but does not always contain full signal names. Signal names saved in a '.SAV' file are limited to eight characters in length. Signal names in a '.SIG' file have no length limit.

The SIG program is run by typing

SIG <file name>

This will look for the file names <file name>.SAV, <file name>.NAM (the Tegas NameList) and <file name>.ERB. The ERB file is used to hold information about any errors that occurred in the conversion. If this file is empty, the conversion was successful. The '.SIG' file can then be transferred to the PERQ and used by the Scope program to examine the simulation results.



## **CHAPTER 9**

### **SCOPE**



## CHAPTER NINE

## SCOPE

|   |   |
|---|---|
| GENERAL CONCEPTS                                    | 1 |
| Signal and Save Files                               | 1 |
| Signal Groups                                       | 1 |
| Name Lists  | 2 |
| Signal Names and Group Names                        | 2 |
| Windows   | 3 |
| Getting Help  | 3 |
| True Time   | 3 |
| SCOPE COMMANDS                                      | 4 |
| SigFileRead <File Name>                             | 4 |
| SavFileRead <File Name>                             | 4 |
| Rename <From Wild Card Spec> <To Wild Card Spec>    | 5 |
| Prefix <Signal name prefix>                         | 5 |
| Group <Name> <Radix> <Signal>{,<Signal>} {High Low} | 6 |
| Names <Name List Specifier> <Wild Card Spec>        | 6 |
| Add <Wild Card Spec>                                | 7 |
| Remove <Wild Card Spec>                             | 7 |
| Display <From>   * <To>   * [<By>   *]              | 7 |
| Data <From>   * <To>   * [<By>   *]                 | 8 |
| CmdFile <File Name>                                 | 9 |
| HardCopy  | 9 |
| ChangeLog   | 9 |
| Type <File Name>                                    | 9 |

SCOPE

27 Dec 82

Quit

9

Help

10

Scope displays simulation results. These results can be in a ".SIG" file produced by the SIG program or in a ".SAV" file produced by Tegas.

### GENERAL CONCEPTS

The Scope program is used to examine the results of a digital simulation. After running the Tegas simulator, you can examine the results in several ways. The simplest way is to use the Tegas Display command to print the values of selected signals in the listing. Each signal is displayed as an independent value. You can use the Scope program to display the results as waveforms or as binary, octal, decimal or hexadecimal numbers.

### Signal and Save Files

Scope reads its input data from either a Tegas save file (.SAV extension) or a signal file (.SIG extension). Tegas produces save files when the Save command is used. Save files contain the values of all selected signals within a specified time period but do not always contain full signal names. For signal names longer than 8 characters, the save file contains a signal number which must be looked up in a Tegas NAMELIST. The Sig program converts from save file format to signal file format. Signal files always contain all known signal names. See Chapter 8 for information on the Sig program.

### Signal Groups

Scope provides a way of referring to a group of signals by a single name. When one signal is displayed numerically, it is always displayed as a binary number. When a group is displayed numerically, it may be displayed in binary, octal, decimal or hexadecimal. This is especially useful for buses.

### Name Lists

Scope maintains four lists of names. You can use the Names command to look at the lists.

The four lists are:

- Signals - This is the list of all valid signal names.
- Groups - This is the list of the currently defined groups. A group consists of one or more signals and a radix. When you display a group of signals using the Data command, Scope uses the signals in the group to form a value in the associated radix.
- Display - This is the list of the signals that will be displayed by the Display command. This list contains no group information.
- Data - This is the list of signals and groups that will be displayed using the Data command. This list is similar to the Display list except that it contains group information.

### Signal Names and Group Names

Several Scope commands take signal or group names as parameters. The names may be full names or may make use of the standard PERQ wild cards (see Chapter 2 of the PERQ System D/L Primer for a full description of wild cards). For example, "Clock" selects a single name, but "Clock\*" selects any name beginning with "Clock".

You can specify a range of names containing numbers or letters in sequence by using a pattern generator. A pattern generator is a pair of numbers or letters separated by a colon. For example, Data0:7 specifies the names Data0, Data1, ..., Data7 and CtrlM:PBar specifies CtrlMBar, CtrlNBar, CtrlOBaR, CtrlPBar. The pattern generator may be embedded anywhere within a name. When there is more than one pattern generator in a name, the rightmost one varies the fastest. Numeric generators always use decimal, and alphabetic generators are always one letter wide.

Scope provides a name prefix ability which adds a default prefix to the selected names. This reduces typing when many names have the same initial string.

Wild cards and prefixes may be applied to signal names and group names, but pattern generators may only be applied to signal names.

### Windows

The Scope display consists of two windows. The upper window is the display window. This window is used for waveform and other data displays. The lower window is the typescript window. This window is used for command and error processing. All commands and parameters are echoed in this window. In addition all error messages will appear in the lower window.

### Getting Help

Pressing the HELP key will always give help. You can get help on a specific command by typing the command followed by the HELP key or by typing the command followed by "/Help".

### True Time

When a wave form is displayed by Scope a vertical line is provided to line up signal transitions. The approximate time corresponding to the cursor is shown in the upper right hand corner of the display window.

Because of the limited resolution of the PERQ display and possible rounding errors in time calculations, a facility for getting the true time of a transition is provided. To get the true time of a transition, point to the desired signal transition and press the right button (green). The cursor will indicate the transition which is just prior to the vertical line within the waveform at which you are pointing. The true time of the transition can now be read in the upper right hand corner of the window. The line will stay at the transition until you release the button.

In addition to obtaining the true time of a transition, Scope provides a reference cursor. This cursor can be placed at a transition by pointing at the signal and pressing the left button (white). This will place the reference cursor at the prior transition. The time of the reference cursor is displayed in the upper window title line. Also displayed is the difference in time between the reference cursor and the standard cursor.

SCOPE COMMANDS

Scope can accept commands from the keyboard, from a popup menu, or on the Scope command line. You get a menu by pressing the middle button (yellow) on the puck. If you type commands on the keyboard, you may use any unique abbreviation of the command.

A number of Scope commands require a user action before beginning the next command. An example of this is the Names command. This command waits for a user action to keep the display from being redrawn before you get a chance to read it.

All Scope commands can take parameters on the command line. If you do not supply needed parameters, Scope will prompt for them.

Scope can accept commands from a command file. Scope command files contain commands, one command per line. All input is the same, independent of its source. To execute a command file type @<File name>. The CmdFile command can be used to create a Scope command file.

**SigFileRead <File Name>**

The SigFileRead command is used to read a signal file.

This command takes one parameter, which is the name of the signal file that is to be read. Scope first tries to find the file exactly as typed, then appends ".Sig" and tries again. If neither file is found, an error message will be printed. After the signal file is read the title line of the typescript window will contain useful information about the file.

**SavFileRead <File Name>**

The SavFileRead command is used to read a SAV file.

This command takes one parameter, which is the name of the SAV file that is to be read. Scope first tries to find the file exactly as typed, then appends ".Sav" and tries again. If neither file is found, an error message will be printed. After the SAV file is read the title line of the typescript window will contain useful information about the file.

**Rename <From Wild Card Spec> <To Wild Card Spec>**

The Rename command is used to change the names of signals.

The command takes two parameters. The first is a specification of the signals that are to be renamed. This specification can contain wild cards. It CANNOT contain pattern generators. The second parameter is used to determine what the new names of the signals will be. If the FROM specification contains wild cards then the TO specification MUST contain the same wild cards in the same order.

You CANNOT use the Rename command if there are any signals selected for display.

**Prefix <Signal name prefix>**

The Prefix command is used to bind a value to the Signal Name Prefix character. This facility is used to allow the user to provide long signal names in a short number of key strokes.

The Prefix command takes a single parameter on the command line. This is the string that is to be used if you type the prefix character, `'' (grave accent, or `~~` key) at the beginning of a signal name. Here is a simple example:

If we give the prefix command:

Prefix Top/Alu/Add/

we could then add a number of signal names by the command:

Add `A  
Add `B  
Add `C

This command would add the following signals:

Top/Alu/Add/A  
Top/Alu/Add/B  
Top/Alu/Add/C

**Group <Name> <Radix> <Signal>{,<Signal>} {High;Low}**

The Group command is used to define a group of signals. Once a group of signals has been defined the value of that group can be displayed by using the Data command. This is very useful for signals that form a value, e.g. the output of a counter.

The first parameter to Group is the name of the new group. Once a group has been defined, it can be added to or removed from the Display and Data lists by using this name.

The second parameter is the radix that is to be used when the values of the group are displayed. A radix may be binary, octal, decimal or hex.

Following the radix parameter is the list of signal names that will make up the group. The first name that is supplied will be the low order bit of the data value. Wild cards are accepted in the list of signal names.

The final parameter determines what is a 1 and what is a 0 when the Data command is used. If you specify High then signals that have High values are interpreted as 1. If you specify Low then signals that have low values are interpreted as 1. The default is High.

**Names <Name List Specifier> <Wild Card>**

The Names command is used to list the valid names in one of the Name Lists.

Names takes two parameters. The first parameter is the name of the list that you would like to examine. The valid list names are Signals, Groups, Display or Data.

The second parameter can be used to limit the amount of data that is provided by the command. Only names that match the <Wild Card Spec> will be displayed.

**Add <Wild Card Spec>**

The Add command is used to add a signal or group to the list of signals that will be displayed.

Add takes a single parameter on the command line which is the signal or group name that is to be added. You may specify a wild card spec to add a number of signals. If the name you supplied is not a valid signal or group, i.e. it is not in the Signal or Group lists, then an error message will be printed.

If the name supplied is both a Group name and a Signal name, the group will be added.

**Remove <Wild Card Spec>**

The Remove command is used to remove a signal or group from the list of signals that are to be displayed.

This command takes a single parameter on the command line. That parameter is the name of the signal or group that is to be removed. It may be a wild card spec, in which case all signals and groups that match the wild card spec will be removed.

**Display <From> | \* <To> | \* [<By> | \*]**

The Display command is used to display a wave form on the screen. It will display the values of all of the signals that are in the Display list. Display takes three parameters on the command list.

The first parameter is the starting time of the display. You may specify a valid time or \*. If you type \*, the entire waveform will be displayed.

The second parameter to display is the ending time of the display. You may specify a valid time or \*. If you type \*, the last time in the signal file is used as the ending time of the display.

The third parameter specifies the time interval that is to be used for the display. Let us assume that you supplied a starting time of 15, an ending time of 60 and an interval of 10. The signal would only be sampled and displayed at times 25, 35, 45, 55 and 65. If you do not provide a value for this parameter then all times will be displayed. Not supplying a By is equivalent to specifying 1.

Instead of specifying the parameters to Display on the command line you may use the puck to pick the starting and ending points of the display. To do this execute the display command without any parameters. When it asks for the starting time, point to the desired starting time on, the current wave form and press the right button (green). Then point to the new ending point of the display and press the right button (green). Scope will use the current value of By time as the By time of the new display.

Data <From> | \* <To> | \* [<By> | \*]

The Data command is used to display a table of signal values. It will display the values of all of the signals and groups that are in the Data list.

Data takes three parameters on the command list.

The first parameter is the starting time for the Data display. You may specify a valid time or \*. If you type \*, the entire time span of the signal file will be displayed.

The second parameter to Data is the ending time of the display. You may specify a specific time or \*. If you type \*, the last time in the signal file is used as the ending time of the display.

The third parameter specifies the time interval that is to be used for the Data display. It is the same as the By parameter to Display.

Instead of specifying the parameters to Data on the command line it is possible to use the puck to pick the starting and ending points of the display. This facility is only available if there is a valid wave form display on the screen. To do this execute the Data command. When it asks for the starting time, point to the the desired starting time on the current wave form and press the right button (green). Then point to the new ending point of the display and

press the right button (green). Scope will use the current value of By time as the By time of the new display.

#### CmdFile <File Name>

The CmdFile command is used to create a Scope command file. It will create a command file that can be used to restore the current state of Scope. This command will create commands that will: a) read the current SIG or SAV file, b) create any defined groups, c) add any groups that have been Added, d) add all signals that have been Added, and e) display the signals with a Display or Data command.

CmdFile takes a single parameter which is the name of the command file that is to be created.

#### HardCopy

The HardCopy command is used to make a hardcopy of the current Scope screen. Scope can make a hardcopy on a number of different devices. The device used is determined when the Scope program is linked. See Chapter 3 for an explanation of PERQ System D/L hardcopy.

#### ChangeLog

The ChangeLog command is used to type the Scope changelog. It will print a single changelog entry in the top window of the display. Type "E" to exit the ChangeLog command. An empty line will go on to the next entry.

#### Type <File Name>

The Type command is used to view arbitrary text files. It will type the file in the upper window of the display.

#### Quit

The Quit command is used to exit the program.

**Help**

The Help command is used to give general help in the use of Scope. For help on a specific command type:

<Command>/Help <cr>  
or  
<Command><HelpKey>

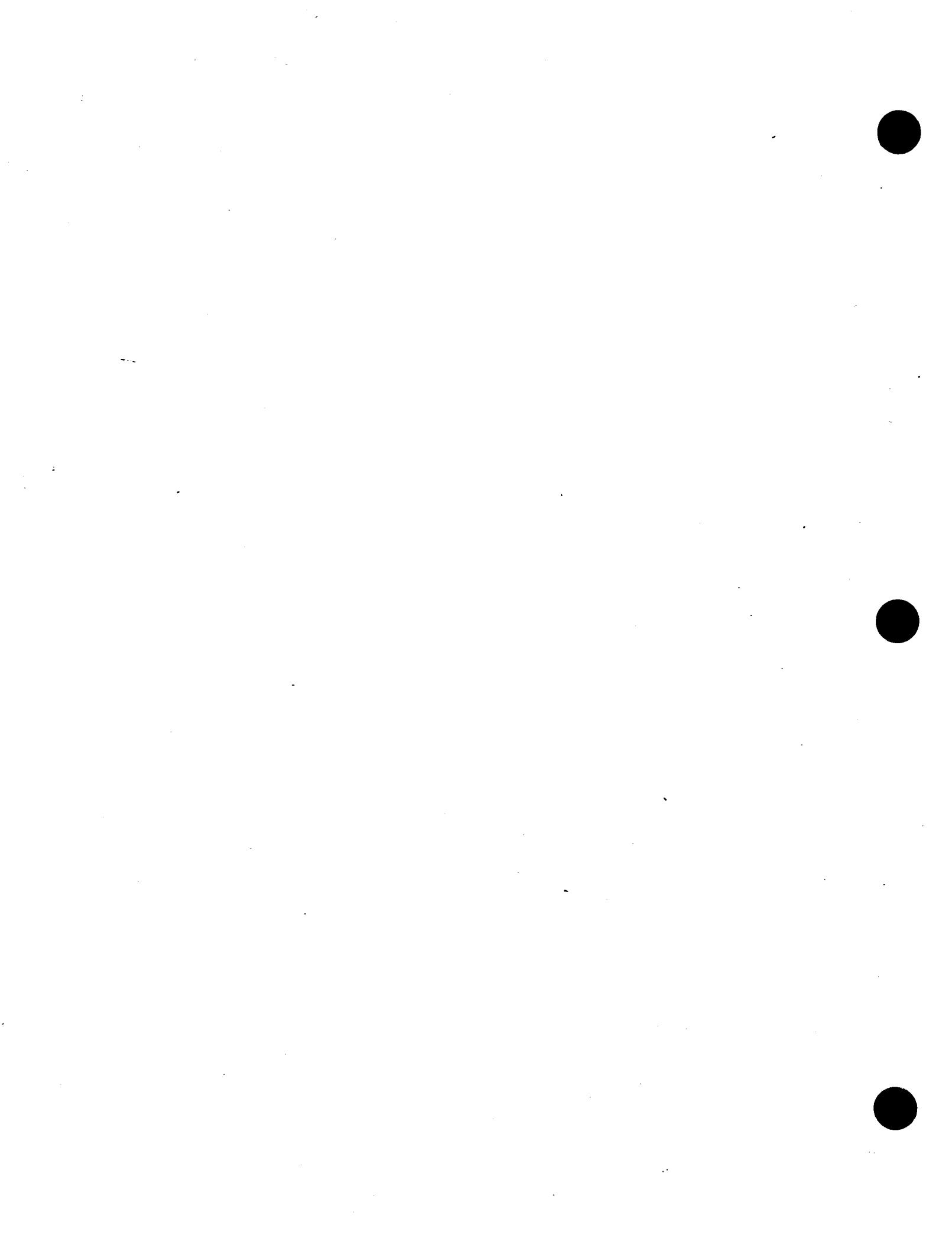
**CHAPTER 10**

**PERQ SYSTEM D/L DESIGN EXAMPLE**



CHAPTER TEN  
A PERQ SYSTEM D/L DESIGN EXAMPLE

|  |    |
|--|----|
| OVERVIEW   | 1  |
| CAPTURING THE DESIGN WITH DP                         | 2  |
| THE SL PROGRAM                                       | 12 |
| PREPARING FOR SIMULATION - THE DA PROGRAM            | 18 |
| PREPARING FOR SIMULATION - THE MIASMA PROGRAM        | 21 |
| PREPARING FOR SIMULATION - INPUT FILE TRANSFER       | 21 |
| SIMULATION WITH TEGAS                                | 22 |
| REVIEWING SIMULATION RESULTS - RESULTS FILE TRANSFER | 22 |
| REVIEWING SIMULATION RESULTS - THE SCOPE PROGRAM     | 22 |
| APPENDIX I   | 25 |
| APPENDIX II  | 26 |



OVERVIEW

PERQ System D/L is a Computer Aided Engineering System designed for the PERQ. It consists of discrete program components which allow the engineer to draw the design, perform rules checking and design verification, simulate (Tegas simulator running on a VAX computer), and view the results through a waveform display program, Scope. PERQ System D/L also provides wirelists and schematics for TIL design.

This chapter is intended for the first-time user who has no previous experience with PERQ System D/L but who has had design experience. A simple design is drawn and defined through the DP program. As DP is exited, a '.DP' file is written. This file is used as input to the SL program. The SL program is used to do simple design rule and syntax checking. If any errors are discovered by the SL program, an errorfile is written. This errorfile can be viewed through the DP program and, when overlaid with the original design in DP, will graphically point to the errors in the schematic capture of the design. The design can then be modified to correct these errors. These two activities continue until the SL program reports no errors.

When SL finds no errors, an '.SL' file for the design is written. This text file describes the component inputs, outputs, bidirectional signals, and signal networks for the design. The '.SL' file is used as input to the DA program. The DA program is a set of functions whose common denominator is the database built from the '.SL' files. One of the main functions of the DA program is to provide Tegas Design Language (TDL) for the TDL compiler of the Tegas simulator. This chapter will describe the steps for creating TDL through the DA program.

In order to run the simulator, the user must create a set of test vectors as well as TDL for the design. The Miasma program is used to generate the set of test vectors. These two files are then transferred to the VAX, and the TDL preprocessor and the Tegas simulator will be run.

After the design is simulated, the results are viewed through the Scope program.

This chapter will describe a TTL design although the same steps would be performed for a gate array design.

Additional information can be found in Chapters 1 through 9 of the PERQ System D/L Reference Manual.

### CAPTURING THE DESIGN WITH DP

The purpose of the DP program is to enable the engineer to schematically capture the design on the PERQ through the use of the bit pad and puck.

PERQ System D/L allows the engineer to use a hierarchical methodology for circuit design. This means that the circuit can be designed in small low level parts, these parts combined to make some higher level parts, these higher level parts combined, etc. The lowest level parts are called primitives and are stored in libraries. There are different libraries for different technologies and for different vendors for gate array design. An example of a TTL primitive is a particular gate in a chip.

The primitives have been previously defined to DP. Each primitive is stored in its own file with the file name Primitive\_Name.DP. In each of these files is an icon of the primitive. An icon is a single symbol used to represent a more detailed drawing, i.e. the implementation (which may contain many parts). An icon consists of an outline and the pins representing the external connections of the circuit being defined. The way to copy a primitive into a drawing will be explained in a later paragraph. Each of the drawings in a design must contain an icon, whether or not the drawing will be used as a part in a higher level design.

To begin the DP session type:

DP <cr>

This will invoke the DP program. If you wish to edit an existing DP file, type:

DP filename <cr>

and the file will be read and will automatically appear on the screen.

Generally, an engineer will want to input primitives and lower level drawings to design a circuit. However, all of the primitives supplied with PERQ System D/L are drawn with GRID=6.

To avoid complication, all drawings should be created with GRID=6. This means that items on the screen will be placed at the nearest grid point, with all grid points being six pixels away (horizontal and vertical) from the nearest grid points. To make sure that the GRID value is 6, type

g

A small box will appear in the lower right-hand corner of the screen with the prompt:

GRID:

Respond with

6 <cr>

The grid size is displayed at all times in the lower right-hand corner of the border surrounding the DP drawing area. A drawing created in several different sessions with a different grid each time may have problems with items not being connected to neighboring items.

One thing to note about DP is whenever a file is read into DP it is automatically centered. Therefore if a drawing is to be stored in a file and then read back in for another DP session, it must take up the whole screen in order for DP to restore it to its exact position on the screen. One way to make sure this happens is to draw lines at the edge of all sides of the DP drawing area.

If two lines are drawn one upon the other, they become invisible. The file Page.DP supplies a titlebox as part of the PERQ System D/L parts library. This is a file which defines invisible lines at the edges of the screen. (This file could be edited with DP to contain project information, dates, etc. The lower portion of what an edited DP file could look like is given in Figure 1.) Input the file into the current session with the "I" command.

I <cr>

DP will prompt for a file name. Answer:

Page <cr>

The file will be copied into the current session.

|  |         |     |          |           |                      |                |      |     |
|--|---------|-----|----------|-----------|----------------------|----------------|------|-----|
| THIS DOCUMENT IS NOT TO BE REPRODUCED IN ANY FORM,<br>OR TRANSMITTED IN WHOLE OR IN PART, WITHOUT PRIOR<br>WRITTEN AUTHORIZATION OF Three Rivers Computer. |         |     |          | NEXT ASSY | COMPANY CONFIDENTIAL |                |      |     |
|  |         |     |          | TITLE     |                      |                |      |     |
| teil exmpl   | DRAWN   | ONE | 11/10/82 | SIZE      | CODE                 | IDENTIFICATION | VAR  | REV |
|  Three Rivers Computer  | CHECKED |     |          | A         |                      |                |      |     |
|  | APPV'D  |     |          | PROJ      |                      |                | PAGE | of  |

14 Dec 82 11:18:02 ff.ds

Figure 1

There are two principal ways to bring items into the current DP session. These are the "I" and "i" commands. (In DP all commands are case sensitive.) The "I" command copies the entire contents of a file into the current DP session. The "i" command copies only the icon (if one has been defined) of a file into the current DP session. The icon symbol will be placed at the current cursor position.

Now that a titlebox has been copied to the session, everything that the engineer draws will remain at that exact position. Note that a titlebox is not necessary for a successful design. If there is no titlebox, items in the drawing will be recentered each time the file is brought into a DP session.

The next step is to actually create the design. In this case (for simplicity) the task is to design a register out of flip-flops. The flip-flops are primitives and therefore can be input via an "i" command. Type

i &lt;cr&gt;

DP asks for the symbol name. For this example a three-input nand gate, the 74S10 is appropriate. The user responds:

S10 &lt;cr&gt;

Then DP prompts for the name of the file where S10 is defined. The default file name is the same as the symbol.

Input DP file [S10]

Since the 74S10 is stored in the file S10 the user need type only:

&lt;cr&gt;

The icon for the 74S10 will appear at the current cursor position (see Figure 2). For this example a 74S00 (S00) and 74S04 (S04) are needed. They're input the same way. See the chapter for the particular technology you are using for part names. Turning the grid display on can help with the placement and alignment of parts and lines. By typing the ` (back quote) key, the grid display is toggled on and off. The grid display shows one dot for every point on the grid (see Figure 3).



Figure 2

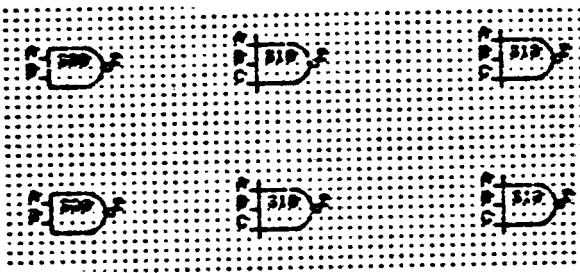


Figure 3

After copying and moving symbols, and drawing lines to connect the figures (see Figure 4) a flip-flop has been drawn. In order to be able to use this flip-flop as a part in another drawing, an icon representing the drawing must be created and the input and output signals must be defined. In addition, each symbol that was input (or copied from input) must be named.

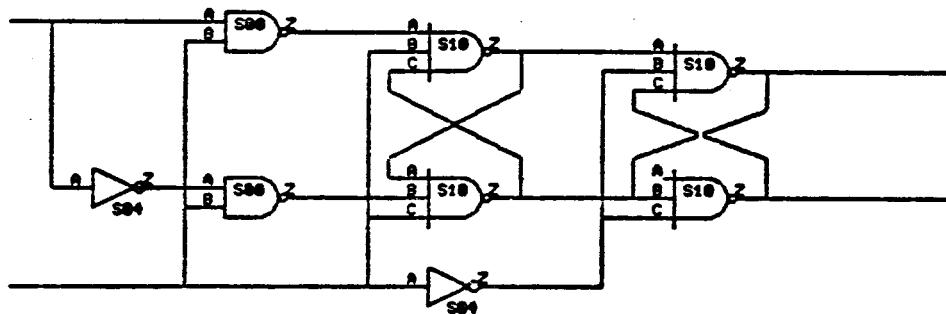


Figure 4

To name a symbol, choose a name which is unique to the drawing for each symbol. Use the 'a' (ASCII) command to name the symbol. Type:

a

Point to the position where the name is to appear in the symbol which is to be named. When naming a symbol, the symbol should be within the bounding box of the symbol. A bounding box for any figure is the smallest rectangle which can contain the entire symbol. Press the puck at the desired position. DP will prompt with:

string:

Enter:

Symbol-Name <cr>

When a <cr> is typed, the name will appear with the first character where the puck has been pressed. Each of the symbols (in this case there are eight) must be named this way. Likewise any signal line which is not connected to a symbol must be named. The name of the signal line should appear as close as possible to the end of the line. After the symbols and signal lines have been named the drawing should look like Figure 5.

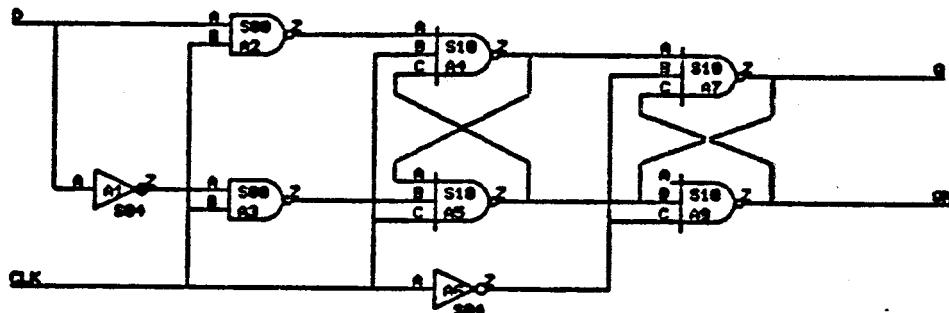


Figure 5

After the symbols and lines have been named an icon must be created. The icon for a drawing must be placed inside a rectangle of double width lines. To draw lines of width 2 change the line width with the '-' command. Type:

-  
DP will respond with:

new thickness:

Type:

2 <cr>

All lines drawn after this '-' command will be double the width of the default line width. Draw a rectangle about four inches by five inches or large enough to fit a small (one inch square) box, and input and output information. Then change the line width back to '1' by using the '-' command. Type:

-  
To the "new thickness" prompt type:

1

Next the figure which will represent the drawing must be drawn. In this case a small square about an inch on each side is appropriate. This figure is what the flip-flop will look like when this part is called. The inputs and outputs of the flip-flop must be defined. Type:

a

Point and press inside the double line box. In response to the 'string:' question respond:

(INPUTS D CLK)

(See Figure 6.) Point and press again at an area inside the double line box. In response to the 'string:' question respond:

(OUTPUTS Q QN)

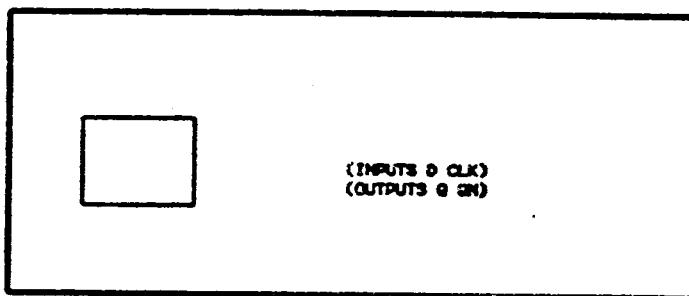


Figure 6

The last thing to do is to define the pins on the icon figure and pack the figure into a symbol. Using the 'p' (pins) command, define all of the signals seen by higher level modules using this part. In this case there are four signals: D, CLK, Q, and QN. Type:

p

Point and press at the position on the lines of the icon symbol where the signal will be defined. DP will prompt with:

pin number:

For each time a pin is created type a different number. In this case use '1', '2', '3', and '4'. Type:

1 <cr>

An '=' will appear where the puck was pressed and the number '1' will appear to the right of it. Repeat this process for pins 2 through 4 (see Figure 7).

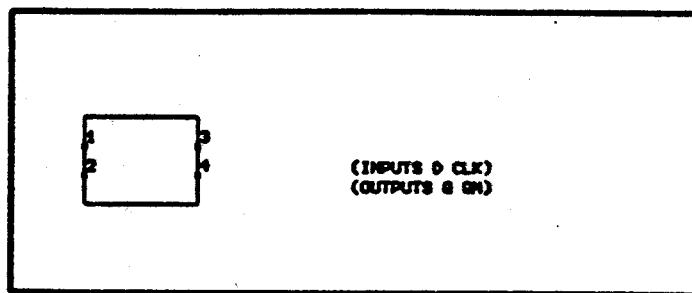


Figure 7

After the pins have been defined, each must be named. Using the 'a' command, place names close to the pin declarations and inside the icon box. These names should correspond to the names which were given in the INPUTS and OUTPUTS declarations and on the signal names. The icon symbol must also be defined. Again using the 'a' command, give the name by which this symbol will be known to higher level designs. In this case the name will be 'FF'. The name 'FF' is placed inside the square which defines the icon (see Figure 8).

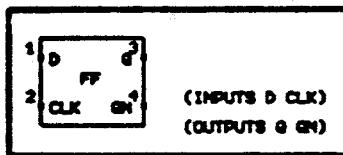


Figure 8

The last step in defining the icon is packing the pins and packing the icon symbol. Packing is a means of associating a group of items into one item. In this case, each pin definition now consists of three items: an '=', a pin number, and a pin name. After these items are packed, only one item (containing pin number, pin name, and '=') will exist for each pin. Likewise, when the icon is packed, one item will remain. This item will consist of the square, the icon name, and several pins (each of which is a packed item consisting of pin name, pin number and '='). To pack a pin, the 'b' command is used. Type:

**b**

Then point at the set of symbols which must be packed into one symbol. This can be accomplished in several ways. One way is to select all relevant items with the 's' command, and then after selecting the 'b' command, push the right button (green), which corresponds to 'selected items'. Another way is to select the 'b' command and draw a 'box' around the items which are to be packed with the left button (white), which corresponds to 'in rectangle'. After the puck press, the user will be prompted for

Name for the symbol:

It is not necessary to give a name for a symbol. A response of <cr> will allow DP to create a name for the symbol.

After the symbol is named, write the entire screen out to a file with the 'o' command. Type:

o

DP will respond with:

output file:

Type the name of the symbol. (Always name the file the same name as the symbol.) Type:

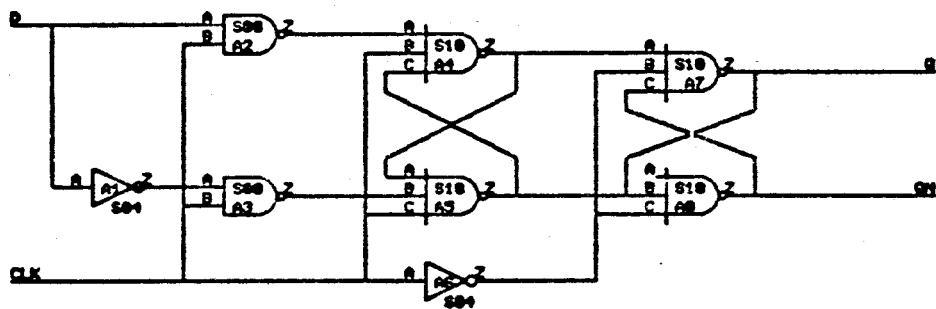
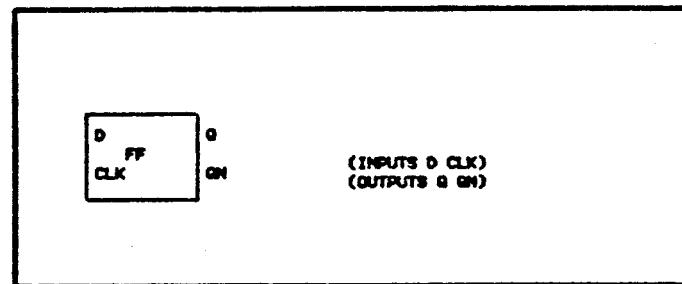
FF <cr>

After the screen has been written out to a file, type:

q

to quit DP. DP will ask for confirmation. If a 'q' has been typed and modified buffers exist (i.e. the drawing has been changed since it was last written out to file), DP will report this and will ask for additional confirmation.

The complete FF drawing is shown in Figure 9.



**Figure 9**

Now that the figure is drawn, the next step is to use the SL program to create '.SL' files and to do some preliminary rules checking.

THE SL PROGRAM

The SL program is used to do some simple syntax checking and to extract design information from the '.DP' file. SL must be run on all '.DP' files for a design before simulation or design analysis can begin. Type:

SL

The SL program will prompt for the DP file:

DP file:

Type:

FF <cr>

```
.RUN not found.  
►SL  
SL ver. 1.14  
DP file: FF  
FF.sl written  
*** 2 Warnings 1 Errors : FF.ERR.dp written  
DP file:  
(recorded in logStream)  
►screendump
```

Figure 10

Three files were written: FF.sl, FF.ERR.dp, and logStream. LogStream contains a log of all the commands given to and the responses given by SL. FF.sl is the file which contains the design information. In this case, errors were recorded and the '.sl' file contains only part of the circuit design information. The '.ERR.dp' file contains a graphic description of the errors in DP format. When this file is overlaid on the original FF.dp file, lines in the '.ERR.dp' file will point to the exact position of the error and text will describe the error. Note that the design '.dp' file must be displayed before the '.err.dp' file.

To examine the SL errors type:

DP

Input both the FF.dp file and the FF.ERR.dp file. Type:

I

DP will ask for the file name. Type:

FF.DP <cr>

Once the drawing appears on the screen, type:

I

When DP asks for the filename, type:

FF.ERR <cr>

If the entire drawing with the '.ERR.dp' file appended does not fit on the screen, the drawing can be reduced in size with the Equal command. Type '=' then type the magnification factor when prompted (.5 = half size, 2 = twice the size). (See Figure 11.)

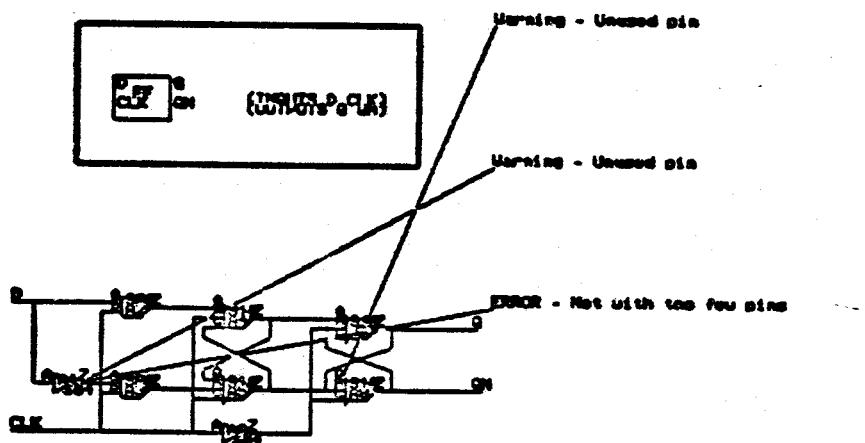


Figure 11

Figure 11 shows that several errors have appeared. SL reports that there are two unused pins and one net with too few pins. The FF drawing must be changed to correct these errors. Since the current drawing contains both FF.dp and FF.err.dp, the screen must be cleared and the FF file input again. Select the entire screen. Type:

S

All elements on the screen will be marked with a small black box. All of the elements can then be deleted with the 'D' command. Type:

D

The 'D' command performs a 'Delete Selected Items' operation. Since all deletes may be immediately un-deleted, the Delete buffer should be cleared before another is begun. This is done with a second 'D' command. Type:

D

Then bring back FF.dp with the 'I' command. Type

I

When prompted for the file name, answer:

FF

to bring back the flip-flop by itself.

Close inspection of the diagram shows that the 'A' pin of the S10 in the lower right-hand corner of the drawing is not connected to anything. In fact, it should be connected to the 'Z' output of the S10 above it. Line edit (the 'e' command) these lines to correct the drawing. Likewise, inspection of the leftmost S04 shows that the line connecting the 'Z' pin of this component to the 'A' pin of the closest S00 does not touch the S04 'Z' pin but extends into the S04 going past the 'Z' pin. Edit this line to just touch the 'Z' pin of the S04 (See Figure 12).

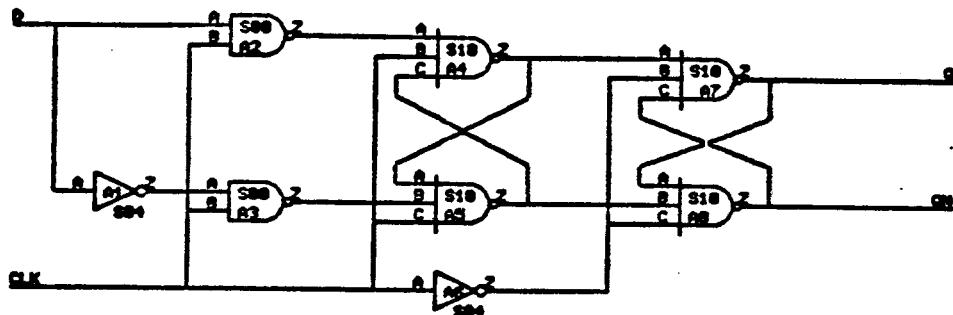


Figure 12

After writing the file out to the FF.dp, run SL again with FF as input. This time SL reports no errors and the file FF.SL contains a description of the circuit.

Now use the FF part to design a four-bit register. Note that the FF part has been finished before it is used in a design. DP does not maintain dynamic pointers to lower level parts, but actually makes a copy of the design information for that part in the higher level design which uses it. Therefore, if a lower level part is changed, the change will not automatically ripple through to higher level drawings. If the changes to a lower level primitive are wanted in a higher level drawing, the old instances of the relevant part must be deleted in the higher level drawing, the higher level drawing must be written out to a file, reread, and then the lower level primitive must be brought in with the 'i' command and copied to the appropriate places.

Use the 'i' command to input the icon for the FF part. (Remember, 'I' would input the entire file, 'i' will input just the icon of the part.) Type:

i

and answer 'FF' when asked for the symbol name and file. Copy the FF part 3 times to make a 4-bit register and draw lines for all of the input and output signals as well as a clock line (See Figure 13).

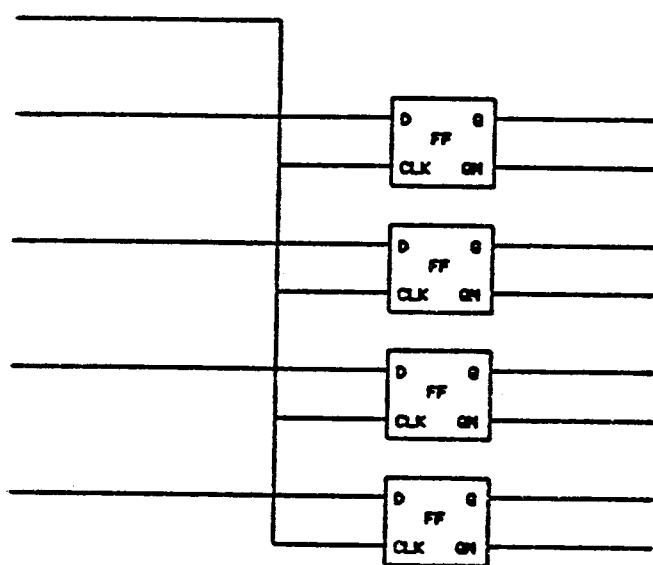


Figure 13

As was done for the FF part, name all of the input and output signals. Call the clock 'CLK', the inputs 'D0' through 'D1', and the outputs 'Q0' through 'Q3', and 'QN0' through 'QN3'. Then draw the 'magic box' with double width lines. Inside it define the inputs and the outputs.

Figure 14 shows that the '(INPUTS)' and '(OUTPUTS)' clauses are of a different form than those for the FF symbol. The DP user may opt to place the names of the input or output signals below the '(INPUTS)' or '(OUTPUTS)' keyword as in:

```
(INPUTS)
  CLK
  D0 D2
  D1 D3
```

instead of

```
(INPUTS CLK D0 D1 D2 D3)
```

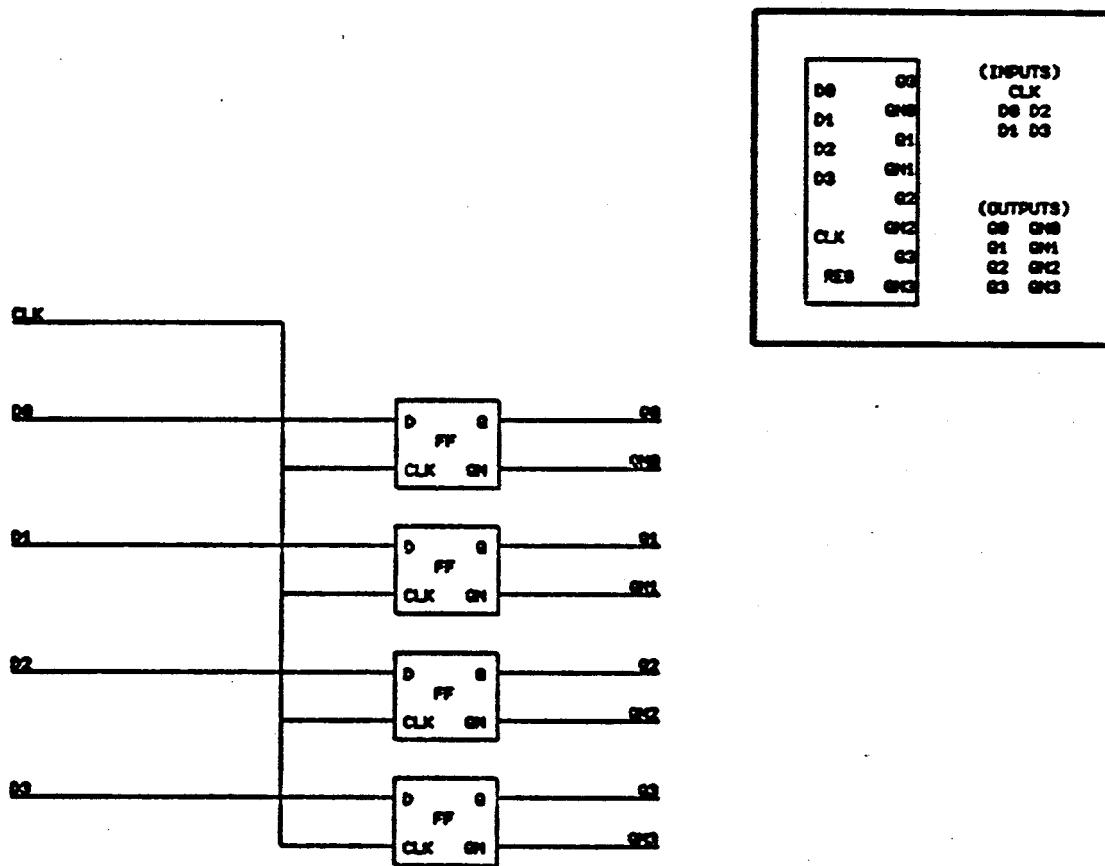


Figure 14

If the signals are listed below the keyword (as in the first case), the keyword and the signal names must be packed into a special symbol called 'LITERAL.{anything}.' To do this first use the puck to select the keyword and the signal names. Use the 'b' command to pack the selected items. Type:

b

Press the puck button which corresponds to the 'Pack Selected' command. DP will respond with:

Name for the symbol:

Answer:

Literal.IN (for (INPUTS)) or Literal.OUT (for (OUTPUTS))

Next define the symbol which will represent this register when it is called into a higher level design. This step is performed in the same way as the earlier example of FF. Draw

a single line width box to represent the register. With the 'p' command define the five input signals and the eight output signals. Name the signals with the same names that were used on the signal lines on the drawings. Name the icon itself, calling it 'REG'. (All of these names are defined with the 'a' command.) Pack each of the signal names individually, then pack the icon itself into the symbol name 'REG'. The file should then be output to the name 'REG'.

Finally, create an .SL file for REG.dp. Type:

SL <cr>

To the prompt for DP file answer:

REG <cr>

SL will check for errors and report them (as in the FF example above).

#### PREPARING FOR SIMULATION - THE DA PROGRAM

After '.SL' files have been sucessfully created, the next step is to create '.TDL' files necessary for simulation. A '.TDL' file is input to the Tegas TDL preprocessor on the VAX. All designs must be preprocessed through the TDL program before being simulated with the Tegas program.

The DA program performs a variety of design aids. These include functions which create TDL files, describe design cell usage, describe clock delays, provide a bill of materials, delay information, and cross reference listings. The user should become familiar with all of the functions. However, in order to simulate with the Tegas simulator, only the first function, creating TDL files, need be performed.

In order to create a TDL file, invoke the DA program. Type:

DA

DA then prompts for a command. If the 'HELP' key is depressed, a list of all valid DA functions and switches will be listed. To create TDL, the Simulate command is performed. The Simulate command will prompt for additional information. Type:

Simulate

If the Read Design File command has not been executed (in this case it hasn't), DA will prompt for the design file name:

Design Name[]:

Reply with the name of the SL file which contains the highest level design. In this case, REG.SL:

REG <cr>

DA then reads through the design file printing the names of all of the lower level parts and primitives. For example:

REG  
FF  
S00  
S10  
S04

DA then reports:

Creating REG.TDL

and asks:

Master Directory Name [MASTER]:

This is the name of the directory on the VAX where the master library of primitives resides. The engineer should consult with the local VAX/Tegas administrator for the name of this library. For the system for which the author has access, the name:

[DL.TEGAS]

is used. DA then prompts for:

UserDirectory Name [USER]:

This is the name of the directory on the VAX which belongs specifically to the engineer or project and where the user library for the Tegas simulation resides. Once again the engineer should consult with the local VAX/Tegas administrator for the name of this library. For this example, the author has access to:

[USER.SMITH]

DA then asks:

Do you want to interactively specify lowest level terminals [NO]

If the answer to this question is 'no' (i.e. the question is answered with a <cr>) TDL is generated for all of the design. If, however, the answer is 'yes' then DA will interactively prompt the user asking whether TDL is to be generated for each part of the design. In a case where a top level design calls on, say, five lower level parts, and the engineer must change only one of the lower level parts to correct a design, it is not necessary to regenerate TDL for the other four parts. By interactively specifying lower level terminals, the engineer could generate TDL only for the top level and the one lower level part that had changed.

In this example, TDL is being generated for the first time and so TDL must be generated for all of the parts. Answer:

<cr>

DA then prompts for which TDL options the engineer wants to use:

TDL Options [REPL]:

Consult the Tegas user's manual for information about options.  
Answer:

<cr>

to accept the default option of 'REPLACE'. DA then asks:

Do you want to generate TCF [NO]

In other words, does the engineer wish to simulate with real timing delays rather than unit delays. In this example answer:

<cr>

to accept the default answer of 'no'. If the engineer had answered 'Yes' to the prompt from the previous question of interactively specifying lowest level terminals, then the engineer would be prompted on whether to expand each of the lowest level terminals (but not primitives).

When the TDL file has been created, DA will report:

REG.TDL written

Type:

quit <cr>

to exit DA.

#### PREPARING FOR SIMULATION - THE MIASMA PROGRAM

The Tegas simulator requires that a '.TS' file, containing simulation commands and input test vectors, be available for simulation. This file may be created manually on the PERQ or the VAX or can be created through the Miasma program. The Miasma program is a general purpose microcode assembler which contains special purpose facilities for creating '.TS' files. The facilities and use of the Miasma program are explained in Chapter 7.

Creation of a '.MAS' file for input to Miasma requires knowledge of the Tegas '.TS' file format requirements and input test vectors applicable to the circuit which is to be simulated. The Miasma program generates a '.TS' file with simulation directives and input test vectors. Also generated by Miasma are a list file ('.LIST'), and an intermediate file ('.INT') which contains the macro substitutions performed while generating the '.TS' file.

Examples of the '.MAS', and '.TS' files for this example are found in Appendices I and II of this chapter.

#### PREPARING FOR SIMULATION - INPUT FILE TRANSFER

Before simulation the '.TS', and '.TDL' files must be transferred to the VAX. The engineer should consult with the local VAX/Tegas administrator for information regarding local procedures. RS232 and Ethernet communication facilities are available for use with PERQ System D/L.

SIMULATION WITH TEGAS

Methods and procedures for using the Tegas simulation system are described in the Tegas reference manuals.

REVIEWING SIMULATION RESULTS - RESULTS FILE TRANSFER

In order to review simulation results with Scope (an oscilloscope simulation program) the '.SAV' file produced by the Tegas simulator must be transferred to the PERQ from the VAX. Once again, the engineer should consult with the local VAX administrator for information regarding local procedures.

REVIEWING SIMULATION RESULTS - THE SCOPE PROGRAM

The Scope program is used to examine the results of a digital simulation. These results can be examined in tabular form with a listing of all signal values at each time in the simulation when there is any transition, or graphically with the signal wave forms being displayed on the PERQ display. Scope takes as input either a '.SAV' file produced by the Tegas simulator or a '.SIG' file created by the SIG program on the VAX.

(The SIG program takes a '.SAV' file which contains signal names produced by the Tegas simulator, and produces a '.SIG' file which contains the real signal names as given by the design engineer. The Tegas simulator limits signal names to eight characters. PERQ System D/L does not.)

To run the Scope program type:

Scope <cr>

The Scope program produces a two-window display. The top window is used for displaying signal wave forms. The bottom window is used for displaying input from the keyboard. Commands to Scope may be entered via the keyboard or chosen via a menu which is displayed when the middle button (yellow) is pressed. For this example the keyboard will be used.

The first step in examining the results of the Tegas simulation is to read the '.SIG' or '.SAV' file. All commands and functions are performed in the same way whether a '.SIG' or a '.SAV' file is read. If the file available is a '.SAV' file type:

ReadSav Reg <cr>

In this example, a '.SAV' file would be sufficient since all of the signal names used in the REG example are named with 8 characters or less. If a translation had been necessary, 'REG.Sig' could have been read with:

ReadSig Reg <cr>

To display a list of all signal names used in this simulation use the Name command. Type:

name \*

and all signal names are displayed in the top window of the screen. The Add command is used to add to or create a list of signals which are to be displayed. Type:

add \*

to include all signal names in the list of signals to be displayed. Likewise the Remove command is used to remove signal names from the list of those to be displayed. Once a list of signals has been created, these may be displayed in tabular form or in signal wave form. If a tabular format is desired type:

data \*

A list of all signal values for each transition time in the simulation will be displayed. To display the signal wave forms type:

display \*

The signals will be displayed in the top window. A broken vertical line will also be displayed. Movement of this line is controlled by the puck. The time position of the broken line is displayed at the top of the upper window.

To create a command file which recreates the current state and display of the Scope session type:

cmdfile file\_name

File\_name.cmd will be created. When executed from within the Scope program this command file will reproduce the actions which were performed to produce the current state.

Other functions of the Scope program are described in Chapter 9 of the PERQ System D/L Reference Manual.

## APPENDIX I

MODE 2  
SUPPRESS 34 35 36  
LOAD  
IVECTOR  
D0, D1, D2, D3;

SIMSETUP;  
CHANGE CK 0 0;  
CHANGE CK 1 20, 10000, 100;  
CHANGE CK 0 70, 10000, 100;  
END;

## TESTPATT

|      |      |
|------|------|
| 0001 | 0    |
| XXXX | 30   |
| 0010 | 100  |
| XXXX | 130  |
| 0011 | 200  |
| XXXX | 230  |
| 0100 | 300  |
| XXXX | 330  |
| 0101 | 400  |
| XXXX | 430  |
| 0110 | 500  |
| XXXX | 530  |
| 0111 | 600  |
| XXXX | 630  |
| 1000 | 700  |
| XXXX | 730  |
| 1001 | 800  |
| XXXX | 830  |
| 1010 | 900  |
| XXXX | 930  |
| 1011 | 1000 |
| XXXX | 1030 |
| 1100 | 1100 |
| XXXX | 1130 |
| 1101 | 1200 |
| XXXX | 1230 |
| 1110 | 1300 |
| XXXX | 1330 |
| 1111 | 1400 |
| XXXX | 1430 |

ENDBLOCK "TESTPATT";  
SAVE ALL;

SIMULATE;

ENDBLOCK;

## APPENDIX II

TestPat

Cycle 100

BitsPerWord 4

Radix 8

Fields

!Name Pos, Len, Def  
Data: 0, 4, X  
EndFields

Specify

Data \*/30, X/70  
EndSpecify

Macros

Load: 'DATA=\$1'  
EndMacros

Text

MODE 2  
SUPPRESS 34 35 36  
LOAD  
IVECTOR  
D0, D1, D2, D3;

SIMSETUP;  
CHANGE CK 0 0;  
CHANGE CK 1 20, 10000, 100;  
CHANGE CK 0 70, 10000, 100;  
END;

TESTPATT

EndText;

LOAD(1);  
LOAD(2);  
LOAD(3);  
LOAD(4);  
LOAD(5);  
LOAD(6);  
LOAD(7);  
LOAD(10);  
LOAD(11);  
LOAD(12);  
LOAD(13);  
LOAD(14);

A PERQ SYSTEM D/L DESIGN EXAMPLE

28 Dec 82

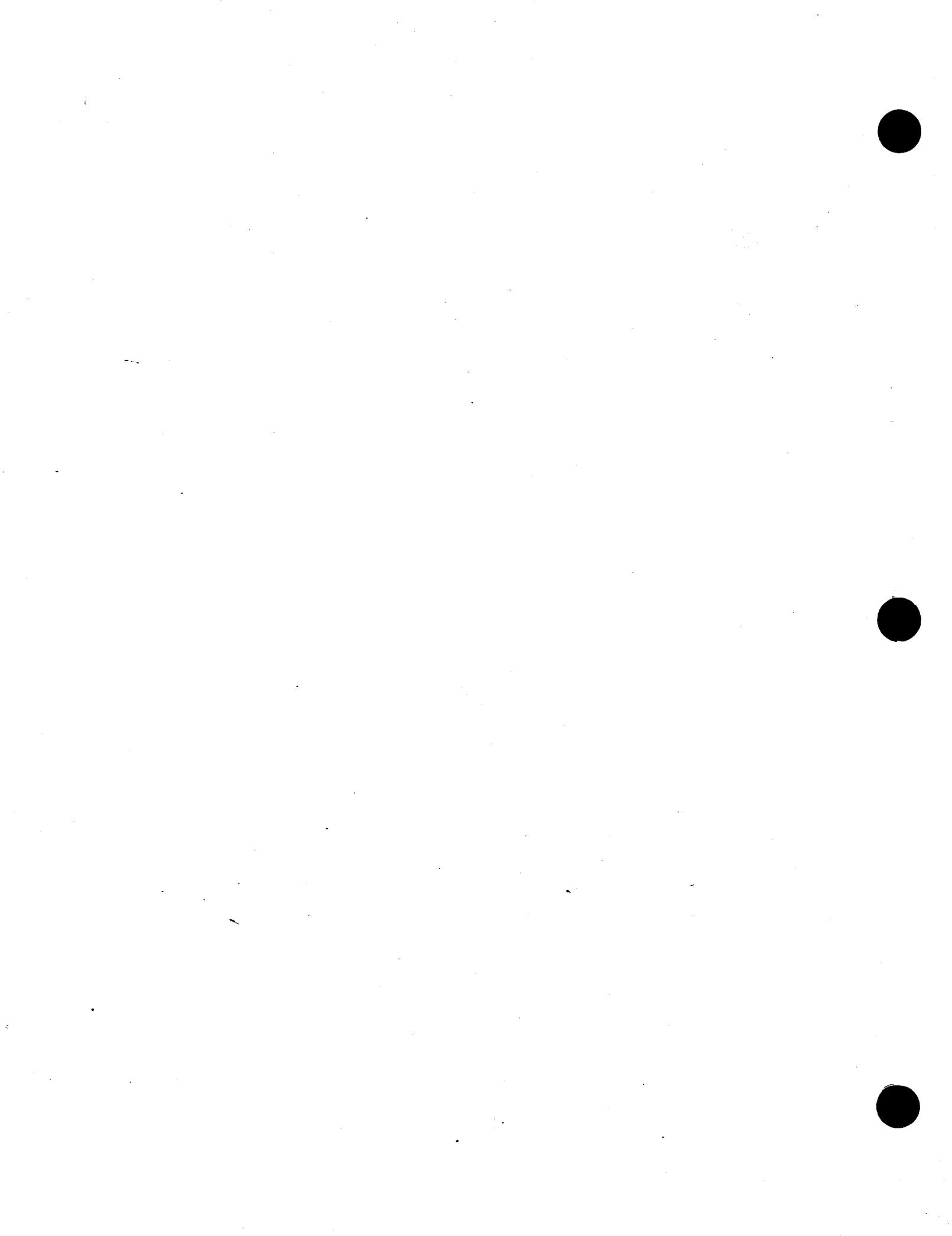
LOAD(15);  
LOAD(16);  
LOAD(17);

Text  
ENDBLOCK "TestPatt";  
SAVE ALL;

SIMULATE;

ENDBLOCK;

EndText;



**CHAPTER 11**  
**COMPONENT DESIGN**



CHAPTER 11  
COMPONENT DESIGN

|  |    |
|--|----|
| Characteristics of Component Design                | 1  |
| Component Design with System D/L                   | 2  |
| DESIGNING WITH COMPONENTS                          | 2  |
| Component Icons                                    | 3  |
| Parts and Part Names                               | 6  |
| ASSIGNING LOCATIONS                                | 7  |
| Making and Checking the Location File              | 8  |
| MAKING A WIRELIST                                  | 9  |
| Dwl - Design Wirelister                            | 9  |
| Running Dwl  | 12 |
| Dwl Prompts  | 13 |
| MAKING A PRINTSET                                  | 14 |
| The Annotated Schematics                           | 14 |
| Cross References                                   | 16 |
| APPENDIX A - BCDLoop - Sample Heirarchical Design. | 17 |
| APPENDIX B - BCDLoop - Location File Template.     | 22 |
| APPENDIX C - BCDLoop - Location File.              | 23 |
| APPENDIX D - BCDLoop - Wirelist.                   | 24 |
| APPENDIX E - BCDLoop - Wirewrap Wirelist.          | 26 |
| APPENDIX F - BCDLoop - Reformatted Wirelist.       | 27 |

|  |    |
|--|----|
| APPENDIX G - BCDLoop - Part List.              | 28 |
| APPENDIX H - BCDLoop - Chip, Pin, Signal List. | 29 |
| APPENDIX I - BCDLoop - Printset.               | 30 |
| APPENDIX J - Special DP Files for Printset.    | 47 |

The previous chapters of this manual discuss abstract digital design. The designer need not be concerned with physical characteristics of a circuit in order to create, simulate, and debug it. Component design means designing electronic circuits using physical components that have specific, predefined functions. The following characteristics of component design distinguish it from abstract design.

#### Characteristics of Component Design

- o The primitive parts correspond to physical devices.
- o Primitive parts must be associated with physical locations on a circuit board.
- o Pin numbers must be associated with each input and output of each primitive part.
- o Point-to-point wirelists may be created in order to build the circuit boards.
- o Annotated schematics showing actual locations and pin numbers may be produced to document the circuit and to aid its debugging.
- o In a heirarchical design, non-primitives may be instantiated more than once. Each primitive part in the detailed drawing of such a non-primitive represents several physical components--one for each instance of the non-primitive.

### Component Design with System D/L

PERQ System D/L aids the designer in performing the tasks of component design. The designer begins by creating the abstract definition of the circuit using parts from the component design library. This process is described in the initial chapters of this manual. To summarize: the designer prepares schematics with DP, extracts electrical information from the schematics with SL, uses DA to perform design checking, and if desired, uses a logic simulator to verify the design. Once the design is judged to be complete, the designer must perform three more tasks which are specific to component design:

1. Associate each part with a location on the circuit board by preparing a location file.
2. Use DA and DWL to create a wiring list.
3. Use DA to create the annotated schematics.

The following sections describing component design refer to a sample design shown in Appendices A through J. The sample design, BCDLoop, is a cascadable BCD counter which cycles between a minimum and maximum BCD value.

### DESIGNING WITH COMPONENTS

In this chapter we will use the term component to mean a whole physical part and the term section to mean a distinct portion of a component. For example, the LS161 contains a single counter, but the S04 contains six independant inverters. Thus an LS161 component contains a single section and an S04 contains six sections. The terms primitive part, non-primitive part, part, and icon are defined in Chapter 5: SL - SCHEMATIC WIRELISTER.

The component library contains icons which represent components or sections of components. For example, the LS161 icon shown in Figure 1 represents a single component and each S04 icon shown in Figure 2 represents a section of a component.

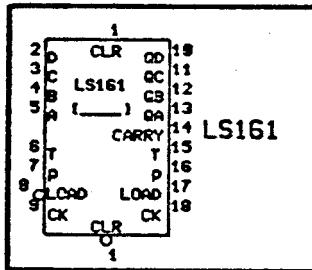


Figure 1 - LS161.DP

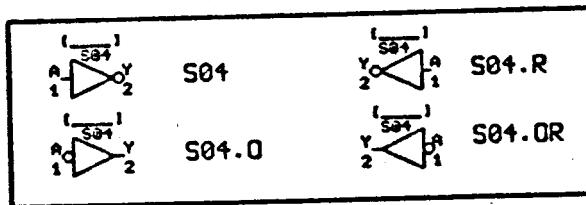


Figure 2 - S04.DP

### Component Icons

There are several significant things to observe about component icons.

- o The part name is often abbreviated by omitting the family prefix (54, 74, etc.). This abbreviation is determined by local convention when the library is created.
- o The DP pin numbers on the icon need not be the actual pin numbers of the physical device. Information in the library provides a mapping from pin name to pin number. This mapping is used by DA when it produces the annotated schematics.
- o Each icon contains the key string "[.....]" which is replaced by the circuit-board location name when DA produces the annotated schematics. The key string is shown in the icon so that the designer can be careful not to place other text or graphics nearby.

Many devices can be used in more than one form. For example, Figure 3 shows the S00 as a positive-logic NAND gate and as a negative-logic OR gate. These gates are equivalent under DeMorgan's Theorum and are physically identical. S00.DP contains both icons: S00 is the NAND form and S00.0 is the negative-logic OR form. Some other examples are shown in Figures 2, 4, and 5.

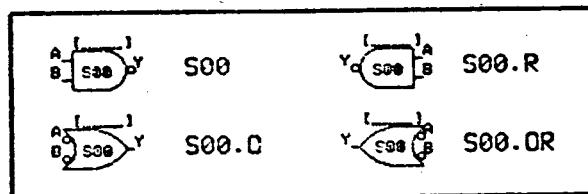


Figure 3 - S00.DP

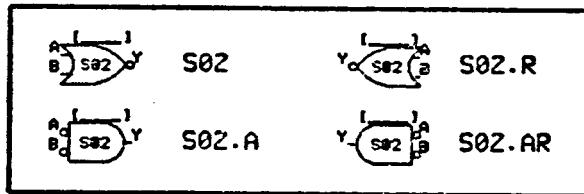


Figure 4 - S02.DP

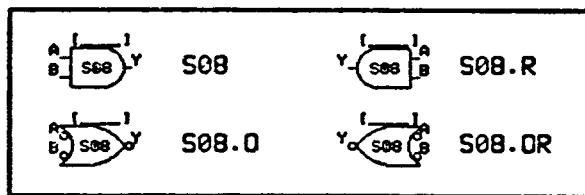


Figure 5 - S08.DP

Some devices may be used as one component or as several sections. For example, Figure 6 shows that the S373 may be used as one 8-bit latch or as 8 1-bit latches. The 8-bit form is physically different from the 1-bit form in the sense that the former is a component whereas the latter is a section of a

component. Primitive parts which are physically different must be defined in different DP files. Primitive parts which are physically identical may be contained in the same DP file. S373.DP contains a single icon and S373x1 also contains a single icon. Another example is the S240 which has the three forms shown in Figure 7.

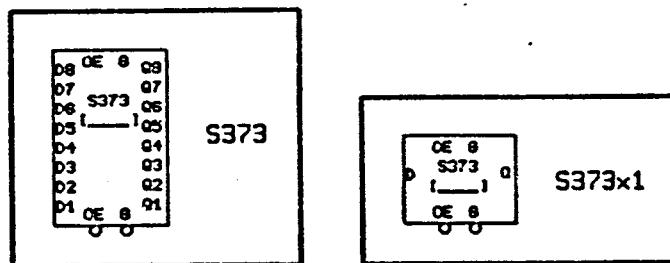


Figure 6 - S373.DP and S373x1.DP

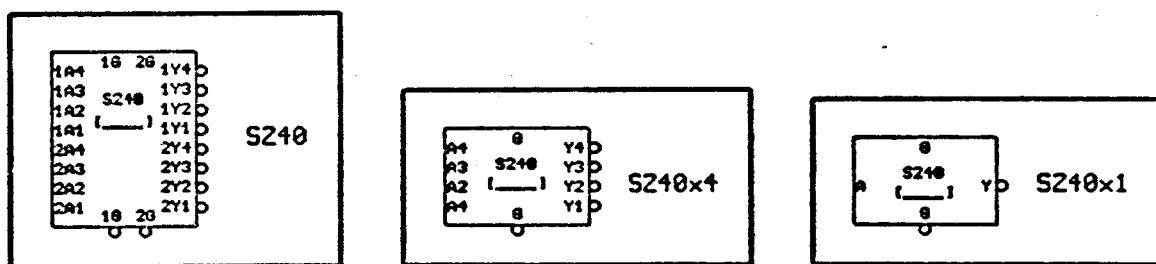


Figure 7 - S240.DP, S240x4.DP, and S240x1.DP

Parts and Part Names

Parts in System D/L must meet four requirements:

1. The portion of the icon name which precedes the first period is called the part name. This part name must be identical to the name of the DP file in which the part is defined.
2. All icons in a single DP file must have the same part name--they must be identical up to the first period.
3. All icons in a single DP file must be physically identical, but may be logically different if they are equivalent under DeMorgan's theorem.
4. All icons in a single DP file must have the same number of pins with the same names.

These requirements imply that S00 and S00.0 must be in the same DP file but S373 and S373x1 must be in different files. The icon naming conventions are largely arbitrary as long as they meet the requirements listed above. The default library uses certain rules which are determined by local convention. See Figures 1 through 7 for examples of the conventions described here.

- o The part number alone is used for positive-logic gates: S00, S02, S04, S08.
- o The part number alone is used for a whole component: S373, S374, LS161.
- o The part number followed by ".0" (Or) is used for negative-logic OR gates, negative-logic NOR gates, negative-logic inverters, and negative-logic buffers: S00.0, S04.0, S08.0.
- o The part number followed by ".A" (And) is used for negative-logic AND gates and negative-logic NAND gates: S02.A.
- o The part number followed by ".R" (Reverse) is used for gates which face to the left rather than to the right: S00.R, S02.R, S04.R, S08.R.
- o The part number followed by "x" and a number is used for n-bit portions of a multi-bit component: S373x1, S240x4.

- o There are, of course, exceptions to these rules. For example, the S74 is only available in a one-bit variety which is called S74 rather than S74x1.

In some cases of multi-section parts certain pins are shared among the sections. For example, all S373x1 sections share common clock and output-enable pins. System D/L requires that common pins of separate sections in a single component be connected to the same signal.

See Appendix A for an example of a hierarchical component design using whole components, sections of components, and gates which have different pictorial forms.

### ASSIGNING LOCATIONS

Once a design has been drawn and checked, the designer must assign physical locations to its primitive parts. Each primitive part which represents a whole component is assigned a location on the circuit board, and each primitive part which represents a section is assigned a location and a specific section number.

The designer prepares a location file which describes the location assignments of all primitive parts in the design. Using an auxiliary file allows creation of different releases of board layout (e.g. wire-wrap and printed-circuit) without changing the schematics themselves.

The location file is a text file with one line for each primitive part in the design. This line contains the full instance name of the part, its part name, its board location, and a section number (if appropriate). The following are several lines from the location file for the sample design shown in the appendices.

|       |        |       |
|-------|--------|-------|
| EQ/X3 | LS266: | U22-4 |
| LD    | S02:   | U23-1 |
| MM    | S374:  | U106  |

System D/L imposes no particular syntax on location names--they are determined by local convention. The section number is appended to the board location. The form of the section number is determined by the component library. The default library uses a "-" followed by a number. Some examples of board locations:

|        |  |
|--------|--|
| U106   | A whole physical part.                 |
| U23-1  | The first section of a multi-bit part. |
| X5Y7-3 | The third section of a multi-bit part. |

### Making and Checking the Location File

To aid the designer in creating the initial location file, DA provides the WrLocation command. This command writes a location file template for the design. All primitive parts are listed in instance name order, but the board location and section number are omitted. Appendix B shows the location file template for BCDLoop. The designer uses a text editor to complete the location file by adding location names and section numbers. Appendix C shows the completed location file for BCDLoop. The DA CkLocation command reads this file and checks for:

- o Primitive parts which have no location (or section if appropriate).
- o Primitive parts which have more than one location (or section if appropriate).
- o More than one primitive part with a certain location (and section if appropriate).
- o Different part types associated with sections of a single location.
- o Inconsistency of multiple sections which share pins.
- o Certain errors in the component library itself.

### MAKING A WIRELIST

A wirelist may be produced in order to build the circuit. The DA Wirelist command writes a wirelist file which consists of a list of all primitive parts used in the design and a list of all wiring networks. Appendix D shows an excerpt from the wirelist for BCDLoop.

The Parts section of the wirelist contains the board location, part name, and section number (if appropriate) for each component or section used in the design.

The \Runs\ section of the wirelist contains all wiring networks separated by blank lines. Each network has a net name followed by a "\\" and a list of all pins in the network. Each pin is described by the board location and pin number, separated by a dash. We call this the location-pin pair.

DA chooses net names by picking the full instance name of some pin in the hierarchical design which is connected to the network. DA first tries to find a pin on a high level non-primitive pin, but if there is none, it chooses an appropriate primitive pin.

### Dwl - Design Wirelister

The wirelist file is used as input to the Dwl processor.

A single module of Dwl, Dwl.Board, contains knowledge about the characteristics of the target circuit board. This knowledge includes:

- o valid location names.
- o the geometry of the board.
- o the power/ground characteristics of the board.

At Three Rivers there are two versions of Dwl.Board: Dwl.WWBoard (for wirewrap boards) and Dwl.PCBoard (for printed-circuit boards). When Dwl is linked, one or the other is chosen. Thus there are two versions of Dwl: DwlWW and DwlPC.

DwlWW makes wirelists for 3RCC wirewrap boards. It understands 3RCC location names and sorts the wirelist to minimize wire lengths for point-to-point wirewrapping. DwlWW also translates pin numbers from the actual numbers on the component itself to the corresponding numbers of a 20-pin socket. This translation aids wirewrapping because 3RCC wirewrap boards contain only 20-pin sockets. When the circuit board is viewed from the

wiring side it is not possible to determine the number of pins on a component. We use the terms pin number and pin for a pin number of the component, and the terms pad number and pad for a pin number of the circuit board.

DwlPC is used for printed-circuit boards. It has no restrictions on location names. Since path lengths are determined by the layout, DwlPC makes no attempt to minimize path lengths. DwlPC does not distinguish between pin and pad numbers. They are always identical.

Dwl produces four output files:

1. <OutputFile>.Wrap - Wirewrap wirelist (Appendix E). This wirelist is designed for fence-post style wirewrapping. Figure 8 illustrates two popular styles of wirewrapping. Dwl prints an entry for each wiring network. The first line has 25 dashes followed by the net name. Each subsequent line describes a single wire to be wrapped by specifying the location-pad pair of the two ends of the wire and L1 or L2 to describe whether the wire is a Level 1 or Level 2 wrap respectively. The .Wrap file is optional.
2. <OutputFile>.WList - Reformatted wirelist (Appendix F). Dwl assigns a number to each wiring network. Each network contains the net number in columns 1 through 4, a list of all location-pad pairs in columns 7 through 41, and the net name in columns 43 through 80. Blank lines separate nets. The .WList file is not optional.
3. <OutputFile>.Part - Part list (Appendix G). This file contains a list of all board locations used in the design and the type of part at that location. The .Part file is optional.
4. <OutputFile>.Chip - Chip, Pin, Signal list (Appendix H). An entry is printed for each component used in the design. The first line of this entry contains the location and part name. Each subsequent line describes a single pin of the component. Columns 1 through 3 contain the pin number, columns 6 through 27 contain either the name of the net that includes the pin or \*NC\* if the pin is not in any net, columns 28 through 35 contain the location-pad pair, and columns 36 through 80 contain a notation which describes the pin as a ground pin (GND), power pin (VCC), input pin (TI), output pin (TO), tri-state output pin (TOT), open-collector pin (TOC), bi-directional pin (ANA), or analog pin (ANA). The .Chip file is optional.

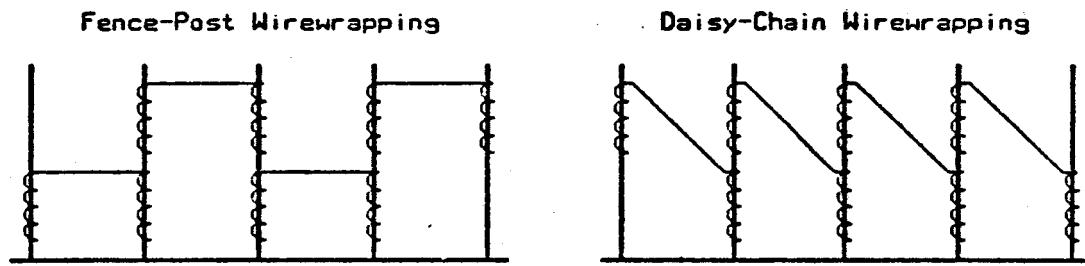


Figure 8 - Wirewrapping Styles

Running Dwl

Dwl is an interactive program. Figure 9 is a transcription of the Dwl session that created the appendices. Responses typed by the user are underlined.

DwlWW

This version of DWL is WW 2.0

Output files name = BCDLoop.WW  
:BCDLoop.WW

WL files (terminate with empty line)

File = BCDLoop

:BCDLoop

File = <CR>

:

Do you want a wrap file? [Y] <CR>

:

RUN format? [N] <CR>

:

Do you want CityBlock? [N] <CR>

:

Do you want to separate grounds? [Y] <CR>

:

Do you want a .CHIP file? [N] Y

:Y

Writing BCDLoop.WW.PART

Reading TTL.Lib

Writing BCDLoop.WW.CHIP

Writing BCDLoop.WW.WLIST

Writing BCDLoop.WW.WRAP

Figure 9 - Sample Dwl Session

Dwl Prompts

Output files name - The name which is used to form the output file names. Various extensions are added to this name:  
".PART" is added for the part file name.  
".CHIP" is added for the chip, pin, signal file name.  
".WLST" is added for the reformatted wirelist file name.  
".WRAP" is added for the wirewrap wirelist file name.

WL files - The name of the WL file should be typed in response to this question. Although DWL can combine many wirelist files together, this is not necessary when using System D/L since the DA Wirelist command combines many SL files into one WL file. Only WL file need be specified.

Wrap File - The .Wrap file is optional. If selected, a fence-post style wirewrapping list is written to this file (see Appendix E).

RUN Format - The .Wrap file may be written either in System D/L format or CMU RUN format. RUN format is not used at 3RCC.

CityBlock - Distances on the board may be measured two ways.

$$\begin{array}{ll} \text{Point-to-point:} & D = \text{Sqrt}( dX \times dX + dY \times dY ) \\ \text{CityBlock:} & D = dX + dY \end{array}$$

Dwl can minimize wire length for either wiring style. Note that this option is irrelevant for DwlPC.

Separate Grounds - If this question is answered with "no", DWL combines into a single network all signals which are connected to ground. If answered with "yes", DWL leaves the ground networks as they are drawn in the schematics.

Chip File - The .Chip file is optional. If selected, a chip-by-chip, pin-by-pin listing of signal names is written to this file (see Appendix H).

### MAKING A PRINTSET

The DA Printset command writes a set of DP files which provide an annotated printset for the design. The DP files which are written are assigned names of the form

<DesignName>.<PageNumber>.Print.Dp

where <DesignName> is the name of the top-level DP drawing. A DP command file, <DesignName>.Print is also written. This command file contains DP commands which read and print all the files in the printset. The files are printed in reverse order so that they are stacked properly by printer which reverse page order. Before executing this command file, you should turn on the displaying of pin numbers.

The printset for a design consists of annotated versions of the original schematics and several kinds of cross reference. Appendix A shows a sample design and Appendix I shows its printset.

### The Annotated Schematics

In a hierarchical design, non-primitives may be instantiated more than once. Each primitive part in the detailed drawing of such a non-primitive represents several physical components--one for each instance of the non-primitive. For example, the sample design shown in Appendix A uses the non-primitive part BCD four times. Each of the primitive parts in BCD represents four physical parts. In order that all physical components be represented in the printset, a page is printed for each instance of each non-primitive part.

Each page of the schematic is modified in several ways. The symbol name of the non-primitive, its instance name, and the page number are added to the drawing. The pins of the non-primitive parts are removed and the pins of the primitive parts are changed to the actual pin numbers for the section which is used. The location names from the location file are added to the primitive parts.

The annotated schematics are copied from the hierarchical design drawings. Pin numbers are modified and certain key strings are replaced with new information. These key strings are recognized by DA because they have a special form. The strings fall into one of three forms which determines the placement of the new string.

When a key string is replaced with other information, the position of the key string determines the position of the new string. If left justified, the new string will have its lower-left corner where the lower-left corner of the key string was. If right justified, the lower-right corner of the new string is placed where the lower-right corner of the key string was. If centered, the new string is centered around the position of the key string. The new string is written with the same font, color, and layer of the key string. This provides the designer with the flexibility to format a page in various ways.

- [KeyWord] Replaced with a left-justified string.
- [KeyWord] Replaced with a right-justified string.
- [KeyWord] Replaced with a centered string.

The following is a list of the keywords.

- \*Symbol\* Name of the non-primitive part defined.
- \*Instance\* Full instance name of the non-primitive.
- \*Page\* Page number.
- \*Date\* Date the printset was made.
- \*Time\* Time the printset was made.
- \*CDate\* Date the source DP file was last changed.
- \*CTime\* Time the source DP file was last changed.

Note that the CDate and CTime are updated every time the file is written. This means that moving the file from one machine to another will update the CDate and CTime.

The DP file "Page.Dp" shown in Appendix J provides a default page format. The component library includes this file.

A key string that is contained in the icon of a primitive part is replaced with the location name of that part. The primitive parts in the component library contain the key string "[.....]". If no key string is found, the location name is centered within the icon.

### Cross References

The printset includes pages which aid the designer in using the schematics. A table of contents, part list sorted by location name, part list sorted by part type, and part list sorted by icon name are printed at the beginning of the schematics. These lists indicate the page numbers on which the parts are found.

Two signal lists are printed at the end of the schematics to aid the designer in following signals from one page to another. The signal-to-net list, sorted by printset page number and location, shows all primitive pins and the names of their wiring networks. The net list, sorted by wiring network, shows all primitive pins which are members of each net. These two lists allow the designer to find all parts to which a certain signal is connected.

To create the cross references, DA copies certain DP files and adds the text to them. The following list describes the DP files which are used as templates. Appendix J shows several of these files.

|                |  |
|----------------|--|
| Contents.Dp    | - Used for table of contents.                |
| ByLoc.Dp       | - Used for part list by location name.       |
| ByPart.Dp      | - Used for part list by part type.           |
| ByIcon.Dp      | - Used for part list by icon name.           |
| SignalToNet.Dp | - Used for list of pins and their net names. |
| NetList.Dp     | - Used for list of nets and their members.   |

The key strings recognized within schematics are also recognized within the cross references, but they have slightly different meanings.

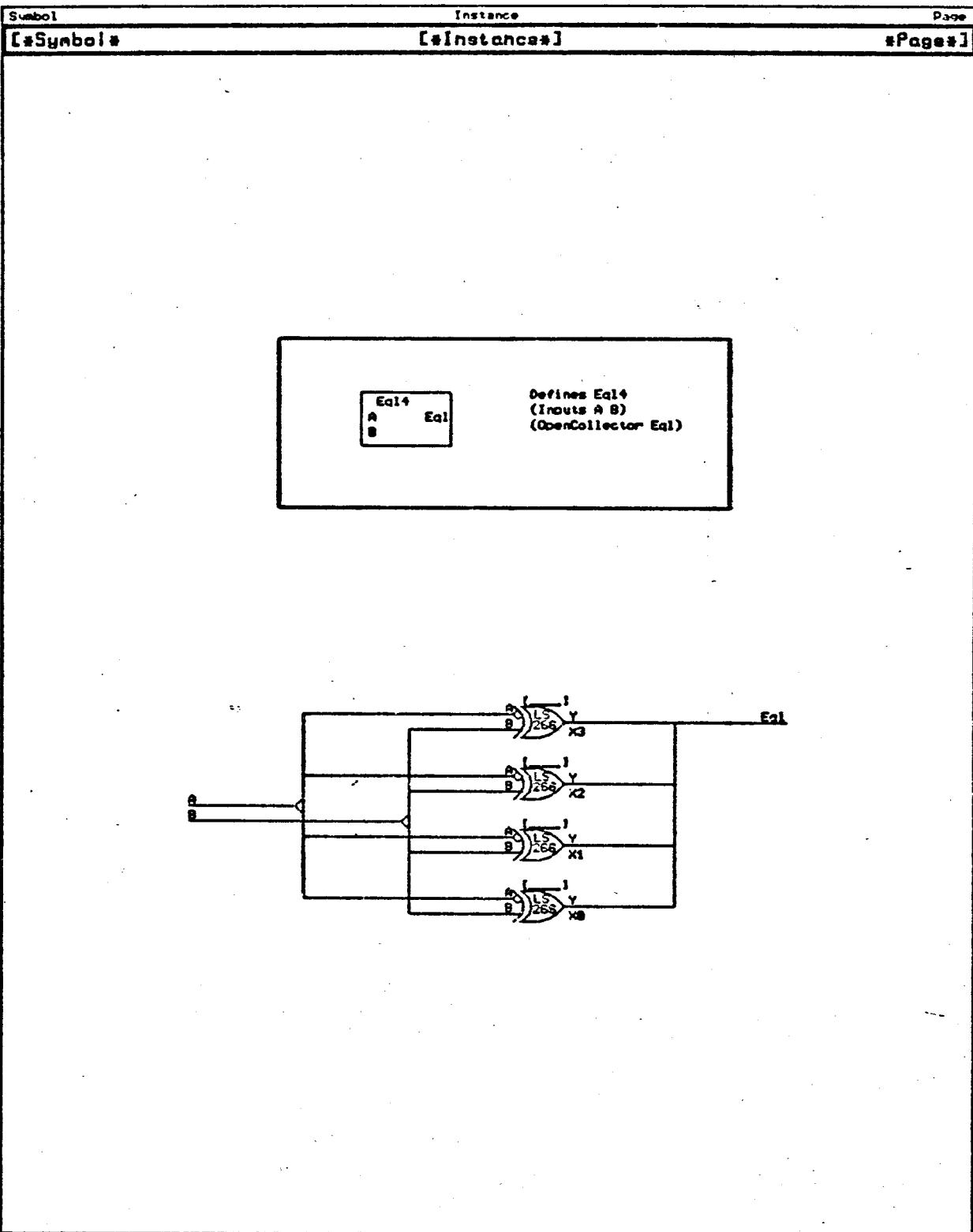
|            |   |
|------------|---|
| *Symbol*   | Name of the top-level drawing.                |
| *Instance* | Title of the page.                            |
| *Page*     | Page number.                                  |
| *Date*     | Date the printset was made.                   |
| *Time*     | Time the printset was made.                   |
| *CDate*    | Date the top-level DP file was last modified. |
| *CTime*    | Time the top-level DP file was last modified. |

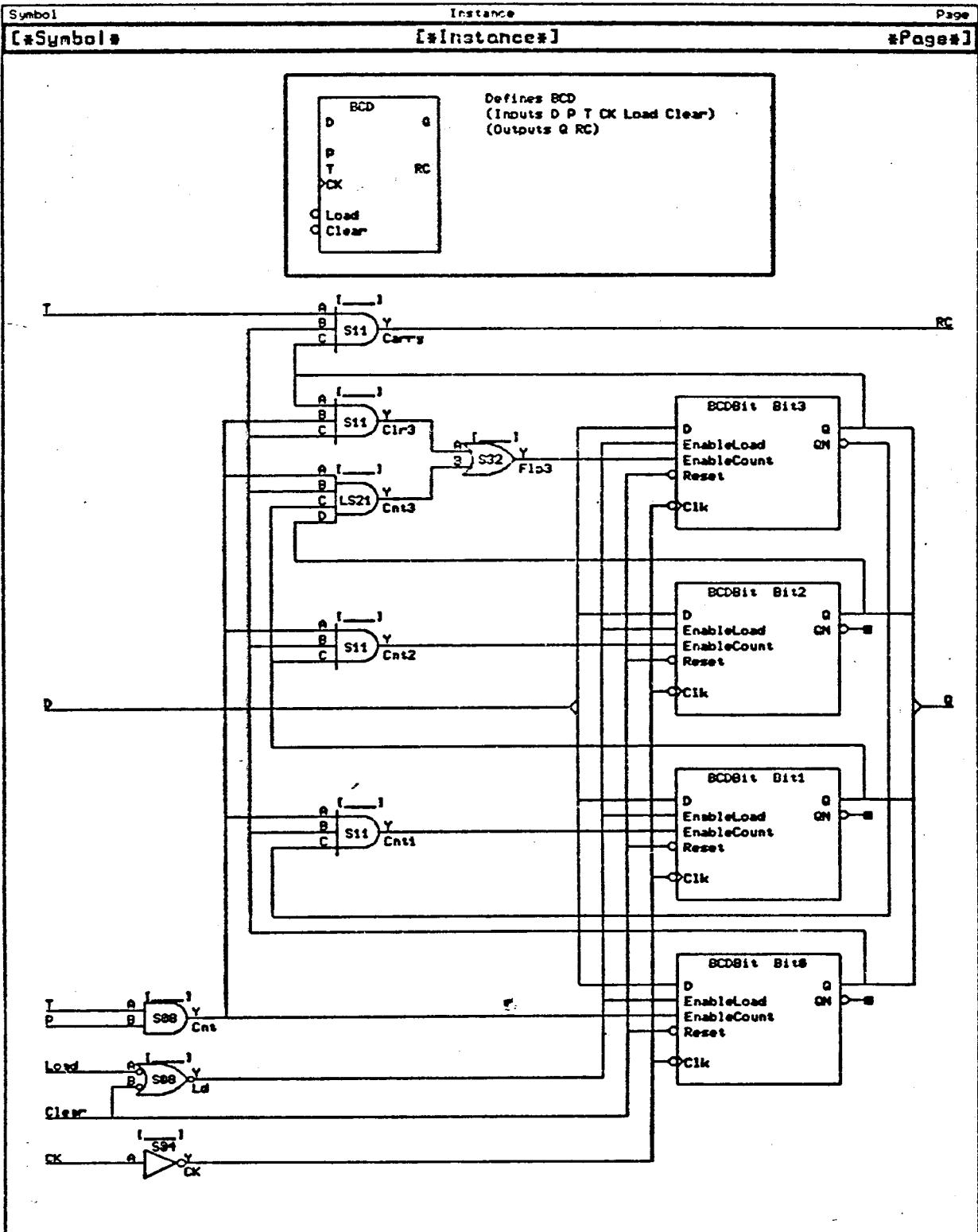
Text is added to these DP files. Up to 9 columns of text may be placed on each page. The font, color, layer, and placement of the text is determined by a set of key strings which define the columns. The keywords used in these key strings are \*Column1\*, \*Column2\*, ..., \*Column9\*. The placement of each column is determined by two key strings: one at its upper-left corner and one at its lower-right corner.

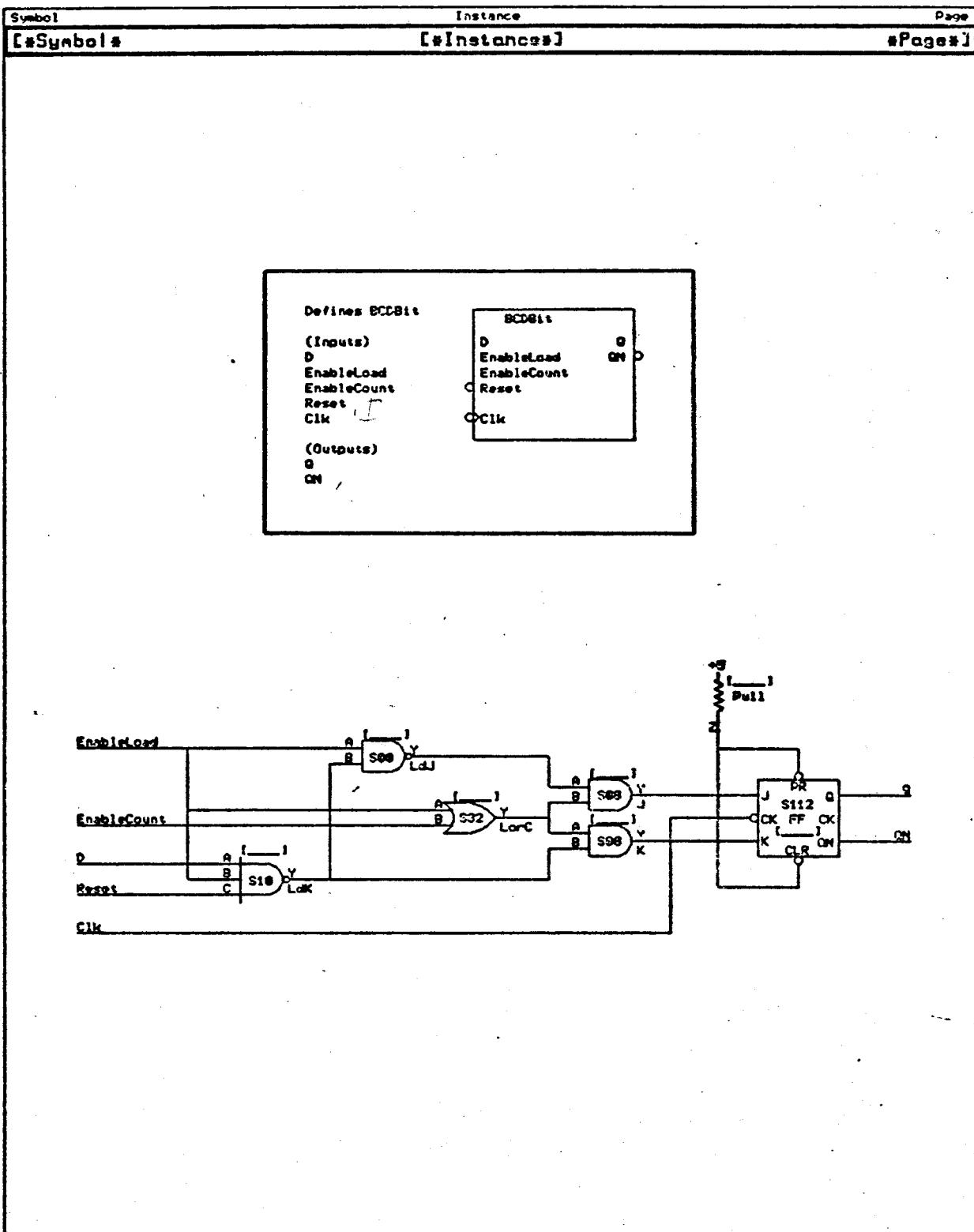
**APPENDIX A**

**BCDLoop - Sample Heirarchical Design.**

| Symbol  | Instance | Page          |
|---|----------|---------------|
| [*Symbols*]   |          | [*Instances*] |
|   |          | [*Pages*]     |
| <pre>Defines BCDLoop (Inputs Min Max LdMinMax Reset P T CK) (Outputs Q RC) (OpenCollector Eq1Max) (SN 2 (E P1 P2 P3 P4))</pre>  |          |               |
|   |          |               |
| <p>This circuit describes one digit of a synchronous BCD counter whose output ranges between a minimum and maximum value. Several digits may be used to make a multi-digit counter whose value ranges between a multi-digit minimum and maximum.</p> <p>Min - Minimum BCD value.<br/>     Max - Maximum BCD value.<br/>     LdMinMax - Clock which loads Min and Max into MM on rising edge.<br/>     Reset - Synchronous reset which loads Min into the counter.<br/>     P - Parallel count enable -- used for multi-stage counter ala S168.<br/>     T - Ripple count enable -- used for multi-stage counter ala S168.<br/>     CK - Clock which increments the counter on rising edge.<br/>     Q - Output of the counter.<br/>     Eq1Max - Open collector output which indicates that the Q = Max. Eq1Max must be pulled-up outside of LoopCtr and for multi-stage counters, all Eq1Max pins must be tied together with a single pullup.<br/>     RC - Ripple carry out ala S168.</p> |          |               |







## APPENDIX B

## BCDLoop - Location File Template.

BCDLoop.Loc as written by the DA WrLocation command.

|               |          |
|---------------|----------|
| CTR/BIT0/FF   | S112:    |
| CTR/BIT0/J    | S08:     |
| CTR/BIT0/K    | S08:     |
| CTR/BIT0/LDJ  | S00:     |
| CTR/BIT0/LDK  | S10:     |
| CTR/BIT0/LORC | S32:     |
| CTR/BIT0/PULL | SIP5VX1: |
| CTR/BIT1/FF   | S112:    |
| CTR/BIT1/J    | S08:     |
| CTR/BIT1/K    | S08:     |
| CTR/BIT1/LDJ  | S00:     |
| CTR/BIT1/LDK  | S10:     |
| CTR/BIT1/LORC | S32:     |
| CTR/BIT1/PULL | SIP5VX1: |
| CTR/BIT2/FF   | S112:    |
| CTR/BIT2/J    | S08:     |
| CTR/BIT2/K    | S08:     |
| CTR/BIT2/LDJ  | S00:     |
| CTR/BIT2/LDK  | S10:     |
| CTR/BIT2/LORC | S32:     |
| CTR/BIT2/PULL | SIP5VX1: |
| CTR/BIT3/FF   | S112:    |
| CTR/BIT3/J    | S08:     |
| CTR/BIT3/K    | S08:     |
| CTR/BIT3/LDJ  | S00:     |
| CTR/BIT3/LDK  | S10:     |
| CTR/BIT3/LORC | S32:     |
| CTR/BIT3/PULL | SIP5VX1: |
| CTR/CARRY     | S11:     |
| CTR/CK        | S04:     |
| CTR/CLR3      | S11:     |
| CTR/CNT       | S08:     |
| CTR/CNT1      | S11:     |
| CTR/CNT2      | S11:     |
| CTR/CNT3      | LS21:    |
| CTR/FLP3      | S32:     |
| CTR/LD        | S08:     |
| EQ/X0         | LS266:   |
| EQ/X1         | LS266:   |
| EQ/X2         | LS266:   |
| EQ/X3         | LS266:   |
| LD            | S02:     |
| MM            | S374:    |

## APPENDIX C

## BCDLoop - Location File.

BCDLoop.Loc after adding the location strings.

|               |          |        |
|---------------|----------|--------|
| CTR/BIT0/FF   | S112:    | U100-1 |
| CTR/BIT0/J    | S08:     | U11-1  |
| CTR/BIT0/K    | S08:     | U11-2  |
| CTR/BIT0/LDJ  | S00:     | U12-1  |
| CTR/BIT0/LDK  | S10:     | U13-1  |
| CTR/BIT0/LORC | S32:     | U14-1  |
| CTR/BIT0/PULL | SIP5VX1: | R1-1   |
| CTR/BIT1/FF   | S112:    | U100-2 |
| CTR/BIT1/J    | S08:     | U11-3  |
| CTR/BIT1/K    | S08:     | U11-4  |
| CTR/BIT1/LDJ  | S00:     | U12-2  |
| CTR/BIT1/LDK  | S10:     | U13-2  |
| CTR/BIT1/LORC | S32:     | U14-2  |
| CTR/BIT1/PULL | SIP5VX1: | R1-2   |
| CTR/BIT2/FF   | S112:    | U101-1 |
| CTR/BIT2/J    | S08:     | U16-1  |
| CTR/BIT2/K    | S08:     | U16-2  |
| CTR/BIT2/LDJ  | S00:     | U12-3  |
| CTR/BIT2/LDK  | S10:     | U13-3  |
| CTR/BIT2/LORC | S32:     | U14-3  |
| CTR/BIT2/PULL | SIP5VX1: | R1-3   |
| CTR/BIT3/FF   | S112:    | U101-2 |
| CTR/BIT3/J    | S08:     | U16-3  |
| CTR/BIT3/K    | S08:     | U16-4  |
| CTR/BIT3/LDJ  | S00:     | U12-4  |
| CTR/BIT3/LDK  | S10:     | U15-1  |
| CTR/BIT3/LORC | S32:     | U14-4  |
| CTR/BIT3/PULL | SIP5VX1: | R1-4   |
| CTR/CARRY     | S11:     | U17-1  |
| CTR/CK        | S04:     | U18-1  |
| CTR/CLR3      | S11:     | U17-2  |
| CTR/CNT       | S08:     | U19-1  |
| CTR/CNT1      | S11:     | U17-3  |
| CTR/CNT2      | S11:     | U20-1  |
| CTR/CNT3      | LS21:    | U21-1  |
| CTR/FLP3      | S32:     | U10-1  |
| CTR/LD        | S08:     | U19-2  |
| EQ/X0         | LS266:   | U22-1  |
| EQ/X1         | LS266:   | U22-2  |
| EQ/X2         | LS266:   | U22-3  |
| EQ/X3         | LS266:   | U22-4  |
| LD            | S02:     | U23-1  |
| MM            | S374:    | U106   |

## APPENDIX D

## BCDLoop - Wirelist.

An excerpt from ECDLoop.WL as produced by the DA Wirelist command.

; File "BCDLOOP.dp" as of 20 Jul 83 13:27:58

; PARTS LIST

U100 S112-1

U11 S08-1

U11 S08-2

U12 S00-1

U13 S10-1

U14 S32-1

RI SIP5VX1-1

U100 S112-2

U11 S08-3

U11 S08-4

U12 S00-2

U13 S10-2

U14 S32-2

RI SIP5VX1-2

U101 S112-1

U16 S08-1

U16 S08-2

U12 S00-3

U13 S10-3

U14 S32-3

RI SIP5VX1-3

U101 S112-2

U16 S08-3

U16 S08-4

U12 S00-4

U15 S10-1

U14 S32-4

RI SIP5VX1-4

U17 S11-1

U18 S04-1

U17 S11-2

U19 S08-1

U17 S11-3

U20 S11-1

U21 LS21-1

U10 S32-1

U19 S08-2

U22 LS266-1

U22 LS266-2  
U22 LS266-3  
U22 LS266-4  
U23 S02-1  
U106 S374

; RUNS LIST

CTR/BIT3/J/Y\ U16-8 U101-11  
CTR/BIT3/PULL/Z\ U101-10 U101-14 R1-5  
CTR/BIT3/QN\ U17-11 U101-7  
Q!P4\ U22-13 U17-13 U17-3 U101-9  
CTR/BIT3/LORC/Y\ U14-11 U16-12 U16-10  
CTR/BIT3/LDJ/Y\ U12-11 U16-9  
CTR/BIT3/LDK/Y\ U12-13 U15-12 U16-13  
MM/Q4\ U106-9 U15-1  
CTR/FLP3/Y\ U14-13 U10-3  
P\ U17-1 U19-1  
RC\ U17-12  
CK\ U18-1  
CTR/CLR3/Y\ U17-6 U10-1  
T\ U19-2  
CTR/CNT3/Y\ U21-6 U10-2  
LD/Y\ U23-1 U19-4  
EQLMAX\ U23-3 U22-3 U22-4 U22-10 U22-11  
MM/Q5\ U106-12 U22-1  
MM/Q6\ U106-15 U22-5  
MM/Q7\ U106-16 U22-8  
MM/Q8\ U106-19 U22-12  
RESET\ U23-2

## APPENDIX E

## BCDLoop - Wirewrap Wirelist.

An excerpt from BCDLoop.WW.Wrap as produced by DWL.WW.

This Run Was made using the following files:

BCDLoop.WW.PART

BCDLoop.WL

----- CTR/BIT3/J/Y  
U101-15 TO U16-14 L1  
----- CTR/BIT3/K/Y  
U101-16 TO U16-17 L1  
----- CTR/BIT3/LDJ/Y  
U16-15 TO U12-17 L1  
----- CTR/BIT3/LDK/Y  
U12-19 TO U15-18 L1  
U15-18 TO U16-19 L2  
----- CTR/BIT3/LORC/Y  
U14-17 TO U16-18 L1  
U16-18 TO U16-16 L2  
----- CTR/BIT3/PULL/Z  
U101-18 TO U101-14 L1  
U101-14 TO R5 L2  
----- CTR/BIT3/QN  
U101-7 TO U17-17 L1  
----- CTR/CK/Y  
U101-17 TO U101-1 L1  
U18-2 TO U100-17 L1  
U101-1 TO U18-2 L2  
U100-17 TO U100-1 L2  
----- CTR/CLR3/Y  
U10-1 TO U17-6 L1  
----- CTR/CNT/Y  
U21-1 TO U14-2 L1  
U17-4 TO U17-15 L1  
U19-3 TO U20-1 L1  
U14-2 TO U17-4 L2  
U17-15 TO U19-3 L2  
----- CTR/CNT1/Y  
U17-14 TO U14-5 L1  
----- CTR/CNT2/Y  
U20-18 TO U14-16 L1  
----- CTR/CNT3/Y  
U10-2 TO U21-6 L1

## APPENDIX F

## BCDLoop - Reformatted Wirelist.

An excerpt from BCDLoop.WW.WList as produced by DWL.WW.

This Run Was made using the following files:

BCDLoop.WW.PART  
BCDLoop.WL

Number Of Nets = 82  
Begin Wirelist

|  |                  |
|--|------------------|
| 41: U101-11 U16-8                          | .CTR/BIT3/J/Y    |
| 42: U101-12 U16-11                         | .CTR/BIT3/K/Y    |
| 43: U16-9 U12-11                           | .CTR/BIT3/LDJ/Y  |
| 44: U16-13 U15-12 U12-13                   | .CTR/BIT3/LDK/Y  |
| 45: U16-10 U16-12 U14-11                   | .CTR/BIT3/LORC/Y |
| 46: R1-5 U101-14 U101-10                   | .CTR/BIT3/PULL/Z |
| 47: U101-7 U17-11                          | .CTR/BIT3/QN     |
| 48: U101-13 U101-1 U100-1 U100-13 U18-2    |                  |
| 48:  | .CTR/CK/Y        |
| 49: U10-1 U17-6                            | .CTR/CLR3/Y      |
| 50: U19-3 U14-2 U17-9 U21-1 U20-1 U17-4    |                  |
| 50:  | .CTR/CNT/Y       |
| 51: U17-8 U14-5                            | .CTR/CNT1/Y      |
| 52: U20-12 U14-10                          | .CTR/CNT2/Y      |
| 53: U10-2 U21-6                            | .CTR/CNT3/Y      |
| 54: U10-3 U14-13                           | .CTR/FLP3/Y      |
| 55: U12-12 U14-12 U15-2 U12-9 U14-9 U13-10 |                  |
| 55: U12-1 U14-1 U13-2 U12-4 U14-4 U13-4    |                  |
| 55: U19-6                                  | .CTR/LD/Y        |

## APPENDIX G

## ECDLoop - Part List.

BCDLoop.WW.Part as produced by DWL.WW.

R1 SIP5VX1  
U10 S32  
U100 S112  
U101 S112  
U106 S374  
U11 S08  
U12 S00  
U13 S10  
U14 S32  
U15 S10  
U16 S08  
U17 S11  
U18 S04  
U19 S08  
U20 S11  
U21 LS21  
U22 LS266  
U23 S02

## APPENDIX H

## BCDLoop - Chip, Pin, Signal List.

An excerpt from BCDLoop.WW.Chip as produced by DWL.WW.

This Run Was made using the following files:

BCDLoop.WW.PART  
BCDLoop.WL

Number Of Chips = 18

## U10 S32

|     |            |        |     |
|-----|------------|--------|-----|
| 1:  | CTR/CLR3/Y | U10-1  | TI  |
| 2:  | CTR/CNT3/Y | U10-2  | TI  |
| 3:  | CTR/FLP3/Y | U10-3  | TO  |
| 4:  | *NC        | U10-4  | TI  |
| 5:  | *NC        | U10-5  | TI  |
| 6:  | *NC        | U10-6  | TO  |
| 7:  |            | U10-7  | GND |
| 8:  | *NC        | U10-14 | TO  |
| 9:  | *NC        | U10-15 | TI  |
| 10: | *NC        | U10-16 | TI  |
| 11: | *NC        | U10-17 | TO  |
| 12: | *NC        | U10-18 | TI  |
| 13: | *NC        | U10-19 | TI  |
| 14: |            | U10-20 | VCC |

\*\* Chip has unused inputs \*\*

## U101 S112

|     |                 |         |     |
|-----|-----------------|---------|-----|
| 1:  | CTR/CK/Y        | U101-1  | TI  |
| 2:  | CTR/BIT2/K/Y    | U101-2  | TI  |
| 3:  | CTR/BIT2/J/Y    | U101-3  | TI  |
| 4:  | CTR/BIT2/PULL/Z | U101-4  | TI  |
| 5:  | Q!P3            | U101-5  | TO  |
| 6:  | CTR/BIT2/QN     | U101-6  | TO  |
| 7:  | CTR/BIT3/QN     | U101-7  | TO  |
| 8:  |                 | U101-8  | GND |
| 9:  | Q!P4            | U101-13 | TO  |
| 10: | CTR/BIT3/PULL/Z | U101-14 | TI  |
| 11: | CTR/BIT3/J/Y    | U101-15 | TI  |
| 12: | CTR/BIT3/K/Y    | U101-16 | TI  |
| 13: | CTR/CK/Y        | U101-17 | TI  |
| 14: | CTR/BIT3/PULL/Z | U101-18 | TI  |
| 15: | CTR/BIT2/PULL/Z | U101-19 | TI  |
| 16: |                 | U101-20 | VCC |

APPENDIX I

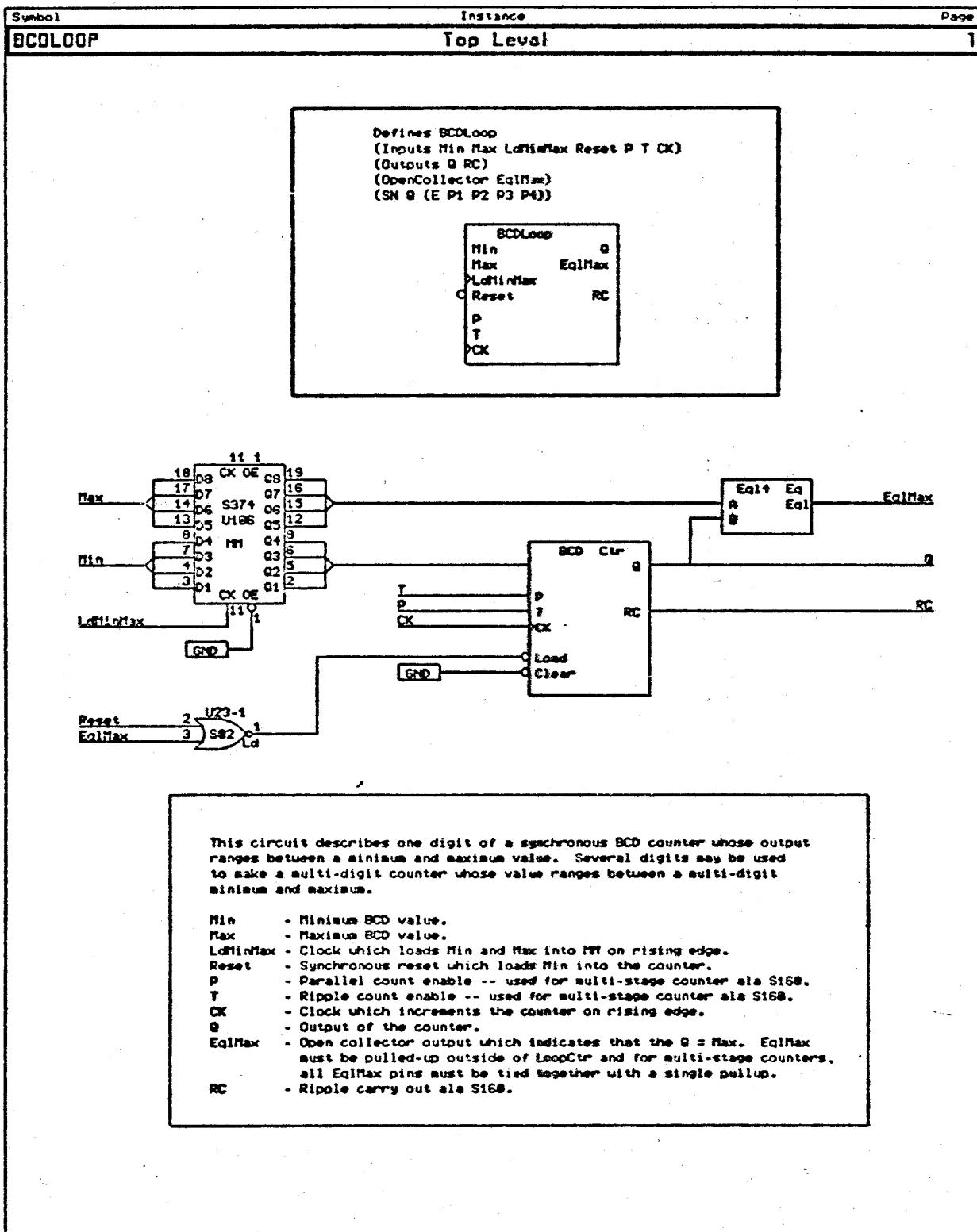
BCDLoop - Printset.

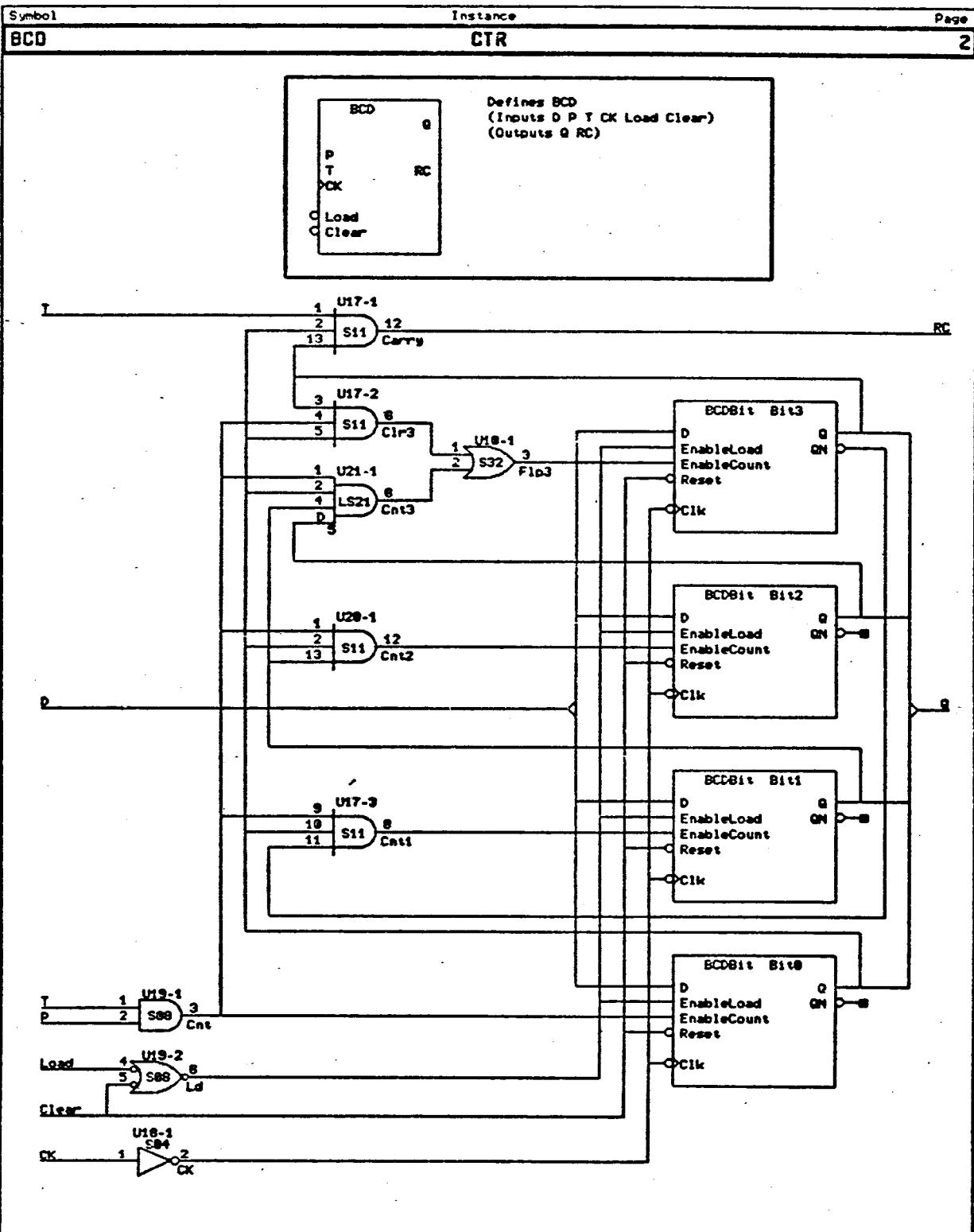
| Symbol         | Instance                  | Page |
|----------------|---------------------------|------|
| <b>BCDLOOP</b> | <b>Table of Contents</b>  | i    |
|                |                           |      |
| i              | Parts List by Location    |      |
| ii             | Parts List by Part Type   |      |
| iv             | Parts List by Icon Name   |      |
| 1              | Top Level                 |      |
| 2              | CTR                       |      |
| 3              | CTR/BIT0                  |      |
| 4              | CTR/BIT1                  |      |
| 5              | CTR/BIT2                  |      |
| 6              | CTR/BIT3                  |      |
| 7              | EQ                        |      |
| 8              | Signal Names to Net Names |      |
| 10             | Nets Sorted by Net Name   |      |

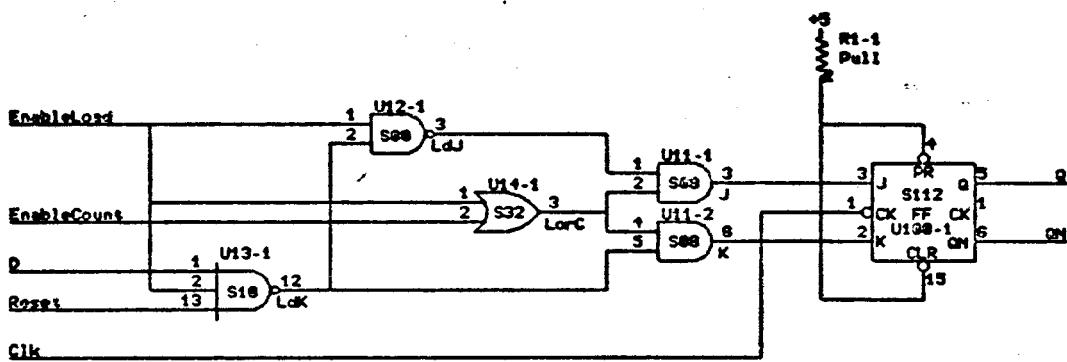
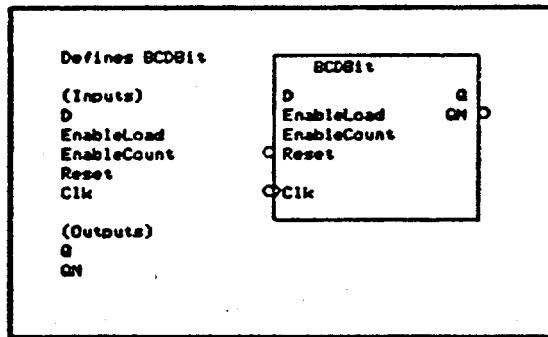
| Symbol   | Instance  | Page         |
|----------|-----------|--------------|
| BCDLOOP  |           | 11           |
| Location | Part Type | Page Numbers |
| R1       | S1PSV     | 3, 4, 5, 6   |
| U10      | S32       | 2            |
| U100     | S112      | 5, 4         |
| U101     | S112      | 5, 6         |
| U106     | S374      | 1            |
| U11      | S08       | 3, 4, 4      |
| U12      | S00       | 3, 4, 5, 6   |
| U13      | S10       | 3, 4, 5      |
| U14      | S32       | 3, 4, 5, 6   |
| U15      | S10       | 6            |
| U16      | S08       | 5, 6, 6      |
| U17      | S11       | 2, 2         |
| U18      | S04       | 2            |
| U19      | S08       | 2            |
| U20      | S11       | 2            |
| U21      | LS21      | 2            |
| U22      | LS266     | 7, 7, 7, 7   |
| U23      | S02       | 1            |

| Symbol    | Instance | Page                    |
|-----------|----------|-------------------------|
| BCDLOOP   |          | Parts List by Part Type |
| Part Type | Location | Page Numbers            |
| LS21      | U21      | 2                       |
| LS266     | U22      | 2, 7, 7, 7              |
| S00       | U12      | 3, 4, 5, 6              |
| S02       | U23      | 1                       |
| S04       | U18      | 2                       |
| S08       | U11      | 3, 4, 4                 |
| S08       | U16      | 5, 6, 6                 |
| S08       | U19      | 2                       |
| S10       | U13      | 3, 4, 5                 |
| S10       | U15      | 6                       |
| S11       | U17      | 2, 2                    |
| S11       | U20      | 2                       |
| S112      | U100     | 3, 4                    |
| S112      | U101     | 5, 6                    |
| S32       | U10      | 2                       |
| S32       | U14      | 3, 4, 5, 6              |
| S374      | U106     | 1                       |
| SIPSV     | R1       | 3, 4, 5, 6              |

| Symbol                  | Instance                                | Page |
|-------------------------|---|------|
| Parts List by Icon Name |   | iv   |
| BCD                     | Defined 2<br>Called 1                   |      |
| BCDBIT                  | Defined 3, 4, 5, 6<br>Called 2, 2, 2, 2 |      |
| BCDLOOP                 | Defined 1<br>Called 0                   |      |
| EQL4                    | Defined 7<br>Called 1                   |      |
| GND                     | Called 1, 1                             |      |
| LS21                    | Called 2                                |      |
| LS266                   | Called 7, 7, 7, 7                       |      |
| S00                     | Called 3, 4, 5, 6                       |      |
| S02                     | Called 1                                |      |
| S04                     | Called 2                                |      |
| S08                     | Called 2, 2, 3, 3, 4, 4, 5, 5, 6, 6     |      |
| S10                     | Called 3, 4, 5, 6                       |      |
| S11                     | Called 2, 2, 2, 2                       |      |
| S11Z                    | Called 3, 4, 5, 6                       |      |
| S32                     | Called 2, 3, 4, 5, 6                    |      |
| S374                    | Called 1                                |      |
| SIPSVX1                 | Called 3, 4, 5, 6                       |      |







| Symbol | Instance | Page |
|--------|----------|------|
| BCDBIT | CTR/BIT1 | 4    |

Defines BCDBit

(Inputs)

- D
- EnableLoad
- EnableCount
- Reset
- Clk

(Outputs)

- Q
- ON

U12-2

U13-2

U14-2

U11-3

S112

R1-2 Pull

EnableLoad

EnableCount

D

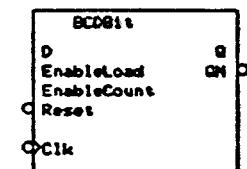
Reset

Clk

Perfume BCOB13

(Inouts)  
D  
EnableLoad  
EnableCount  
Reset  
C'tr

## (Outputs)



Faybl.com

### EnableCount

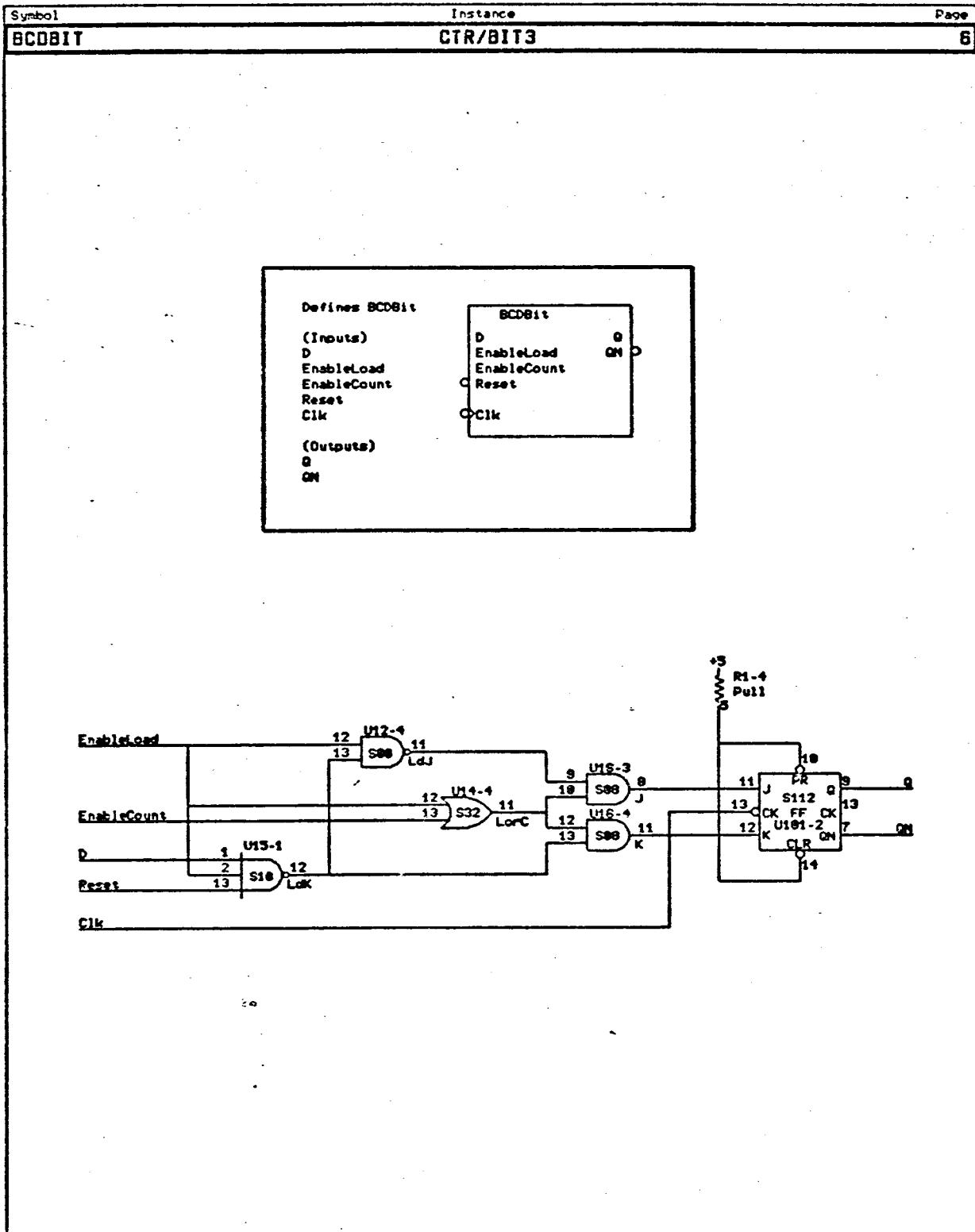
2

5k

9 U12-3

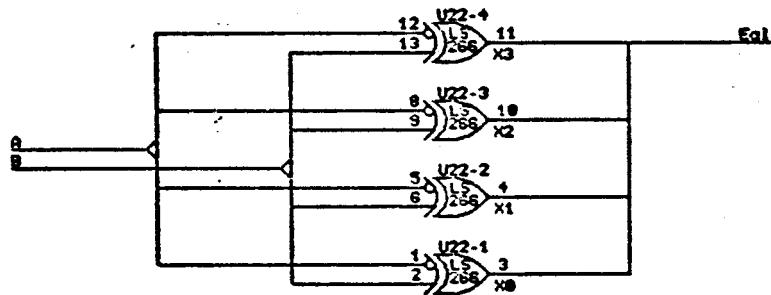
— 1 —

• 100 •



Eq14  
A Eq1  
B

Defines Eq14  
(Inouts A B)  
(OpenCollector Eq1)



| Symbol                            | Instance  | Page            |
|-----------------------------------|-----------|-----------------|
| BCDLOOP Signal Names to Net Names |           |                 |
| <b>1.</b>                         |           |                 |
| Loc-Pin                           | Inst/Name | Net Name        |
| U23-1                             | LD/Y      | LD/Y            |
| U23-2                             | LD/A      | RESET           |
| U23-3                             | LD/B      | ERLMAX          |
| U106-1                            | MM/GND    | 1\$/GND         |
| U106-2                            | MM/Q1     | MM/Q1           |
| U106-3                            | MM/P1     | MINI/P1         |
| U106-4                            | MM/P2     | MINI/P2         |
| U106-5                            | MM/32     | MM/02           |
| U106-6                            | MM/33     | MM/03           |
| U106-7                            | MM/33     | MINI/P3         |
| U106-8                            | MM/34     | MINI/P4         |
| U106-9                            | MM/34     | MM/04           |
| U106-11                           | MM/5K     | LDIN/IMAX       |
| U106-12                           | MM/05     | MM/05           |
| U106-13                           | MM/05     | MAXI/P1         |
| U106-14                           | MM/06     | MAXI/P2         |
| U106-15                           | MM/06     | MM/06           |
| U106-16                           | MM/07     | MM/07           |
| U106-17                           | MM/07     | MAXI/P3         |
| U106-18                           | MM/08     | MAXI/P4         |
| U106-19                           | MM/08     | MM/08           |
| <b>4. CTR/BIT1</b>                |           |                 |
| Loc-Pin                           | Inst/Name | Net Name        |
|                                   | R1-3      | PULL/Z          |
|                                   | U11-8     | J/Y             |
|                                   | U11-9     | J/A             |
|                                   | U11-10    | J/B             |
|                                   | U11-11    | K/Y             |
|                                   | U11-12    | K/A             |
|                                   | U11-13    | K/B             |
|                                   | U12-4     | LDJ/A           |
|                                   | U12-5     | LDJ/B           |
|                                   | U12-6     | LDJ/Y           |
|                                   | U13-3     | LDK/A           |
|                                   | U13-4     | LDK/B           |
|                                   | U13-5     | LDK/C           |
|                                   | U13-6     | LDK/Y           |
|                                   | U14-4     | LORC/A          |
|                                   | U14-5     | LORC/B          |
|                                   | U14-6     | LORC/Y          |
|                                   | U160-7    | FF/DN           |
|                                   | U160-9    | FF/0            |
|                                   | U160-10   | FF/PR           |
|                                   | U160-11   | FF/J            |
|                                   | U160-12   | FF/K            |
|                                   | U160-13   | FF/CK           |
|                                   | U160-14   | FF/CLR          |
| <b>2. CTR</b>                     |           |                 |
| Loc-Pin                           | Inst/Name | Net Name        |
| U10-1                             | FLP3/A    | CTR/CLR3/Y      |
| U10-2                             | FLP3/B    | CTR/CNT3/Y      |
| U10-3                             | FLP3/Y    | CTR/FLP3/Y      |
| U17-1                             | CARRY/A   | P               |
| U17-2                             | CARRY/B   | Q!P1            |
| U17-3                             | CLR3/A    | Q!P4            |
| U17-4                             | CLR3/B    | CTR/CNT/Y       |
| U17-5                             | CLR3/C    | Q!P1            |
| U17-6                             | CLR3/Y    | CTR/CLR3/Y      |
| U17-8                             | CNT1/Y    | CTR/CNT1/Y      |
| U17-9                             | CNT1/A    | CTR/CNT/Y       |
| U17-10                            | CNT1/B    | Q!P1            |
| U17-11                            | CNT1/C    | CTR/BIT3/DN     |
| U17-12                            | CARRY/Y   | RC              |
| U17-13                            | CARRY/C   | Q!P4            |
| U18-1                             | CK/A      | CK              |
| U18-2                             | CK/Y      | CTR/CK/Y        |
| U19-1                             | CNT/A     | P               |
| U19-2                             | CNT/B     | T               |
| U19-3                             | CNT/Y     | CTR/CNT/Y       |
| U19-4                             | LD/Y      | LD/Y            |
| U19-5                             | LD/B      | 2\$/GND         |
| U19-6                             | LD/Y      | CTR/LD/Y        |
| U20-1                             | CNT2/A    | CTR/CNT/Y       |
| U20-2                             | CNT2/B    | Q!P1            |
| U20-12                            | CNT2/Y    | CTR/CNT2/Y      |
| U20-13                            | CNT2/C    | Q!P2            |
| U21-1                             | CNT3/A    | CTR/CNT/Y       |
| U21-2                             | CNT3/B    | Q!P1            |
| U21-4                             | CNT3/C    | Q!P2            |
| U21-5                             | CNT3/D    | Q!P3            |
| U21-6                             | CNT3/Y    | CTR/CNT3/Y      |
| <b>3. CTR/BIT2</b>                |           |                 |
| Loc-Pin                           | Inst/Name | Net Name        |
| R1-2                              | PULL/Z    | CTR/BIT2/PULL/Z |
| U11-2                             | J/A       | CTR/BIT2/LDJ/Y  |
| U11-3                             | J/B       | CTR/LD/Y        |
| U11-4                             | J/Y       | CTR/BIT2/LDK/Y  |
| U11-5                             | K/A       | CTR/BIT2/LORC/Y |
| U11-6                             | K/B       | CTR/BIT2/LDK/Y  |
| U11-7                             | K/Y       | CTR/BIT2/K/Y    |
| U12-1                             | LDJ/A     | CTR/LD/Y        |
| U12-2                             | LDJ/B     | CTR/BIT2/LDK/Y  |
| U12-3                             | LDJ/Y     | CTR/BIT2/LDJ/Y  |
| U13-1                             | LDK/A     | MM/Q1           |
| U13-2                             | LDK/B     | CTR/LD/Y        |
| U13-3                             | LDK/Y     | CTR/BIT2/LDK/Y  |
| U13-12                            | LDK/C     | 2\$/GND         |
| U14-1                             | LORC/A    | CTR/LD/Y        |
| U14-2                             | LORC/B    | CTR/CNT/Y       |
| U14-3                             | LORC/Y    | CTR/BIT2/LORC/Y |
| U160-1                            | FF/CK     | CTR/CK/Y        |
| U160-2                            | FF/K      | CTR/BIT2/K/Y    |
| U160-3                            | FF/J      | CTR/BIT2/J/Y    |
| U160-4                            | FF/PR     | CTR/BIT2/PULL/Z |
| U160-5                            | FF/0      | CTR/BIT2/ON     |
| U160-6                            | FF/DN     | CTR/BIT2/PULL/Z |
| U160-15                           | FF/CLR    | CTR/BIT2/PULL/Z |
| <b>4. CTR/BIT1</b>                |           |                 |
| Loc-Pin                           | Inst/Name | Net Name        |
|                                   | R1-3      | PULL/Z          |
|                                   | U11-8     | J/Y             |
|                                   | U11-9     | J/A             |
|                                   | U11-10    | J/B             |
|                                   | U11-11    | K/Y             |
|                                   | U11-12    | K/A             |
|                                   | U11-13    | K/B             |
|                                   | U12-4     | LDJ/A           |
|                                   | U12-5     | LDJ/B           |
|                                   | U12-6     | LDJ/Y           |
|                                   | U13-3     | LDK/Y           |
|                                   | U13-4     | LDK/C           |
|                                   | U13-5     | LDK/Y           |
|                                   | U14-4     | LORC/A          |
|                                   | U14-5     | LORC/B          |
|                                   | U14-6     | LORC/Y          |
|                                   | U160-7    | FF/DN           |
|                                   | U160-9    | FF/0            |
|                                   | U160-10   | FF/PR           |
|                                   | U160-11   | FF/J            |
|                                   | U160-12   | FF/K            |
|                                   | U160-13   | FF/CK           |
|                                   | U160-14   | FF/CLR          |
| <b>5. CTR/BIT2</b>                |           |                 |
| Loc-Pin                           | Inst/Name | Net Name        |
|                                   | R1-4      | PULL/Z          |
|                                   | U12-8     | LDJ/Y           |
|                                   | U12-9     | LDJ/A           |
|                                   | U12-10    | LDJ/B           |
|                                   | U13-8     | LDK/Y           |
|                                   | U13-9     | LDK/A           |
|                                   | U13-10    | LDK/B           |
|                                   | U13-11    | LDK/C           |
|                                   | U14-8     | LORC/Y          |
|                                   | U14-9     | LORC/A          |
|                                   | U14-10    | LORC/B          |
|                                   | U16-1     | J/A             |
|                                   | U16-2     | J/B             |
|                                   | U16-3     | J/Y             |
|                                   | U16-4     | K/A             |
|                                   | U16-5     | K/B             |
|                                   | U16-6     | K/Y             |
|                                   | U160-1    | FF/CK           |
|                                   | U160-2    | FF/J            |
|                                   | U160-3    | FF/K            |
|                                   | U160-4    | FF/PR           |
|                                   | U160-5    | FF/DN           |
|                                   | U160-6    | CTR/BIT2/ON     |
|                                   | U160-15   | CTR/BIT2/PULL/Z |
| <b>6. CTR/BIT3</b>                |           |                 |
| Loc-Pin                           | Inst/Name | Net Name        |
|                                   | R1-5      | PULL/Z          |
|                                   | U12-11    | LDJ/Y           |
|                                   | U12-12    | LDJ/A           |
|                                   | U12-13    | LDJ/B           |
|                                   | U14-11    | LORC/Y          |
|                                   | U14-12    | LORC/A          |
|                                   | U14-13    | LORC/B          |
|                                   | U15-1     | LDK/Y           |
|                                   | U15-2     | LDK/A           |
|                                   | U15-3     | LDK/Y           |
|                                   | U15-4     | LDK/C           |
|                                   | U16-8     | J/Y             |
|                                   | U16-9     | J/A             |
|                                   | U16-10    | J/B             |
|                                   | U16-11    | K/Y             |
|                                   | U16-12    | K/A             |
|                                   | U16-13    | K/B             |
|                                   | U160-7    | FF/DN           |
|                                   | U160-9    | FF/0            |
|                                   | U160-10   | FF/PR           |
|                                   | U160-11   | FF/J            |
|                                   | U160-12   | FF/K            |
|                                   | U160-13   | FF/CK           |
|                                   | U160-14   | FF/CLR          |

| Symbol                    | Instance  | Page     |
|---------------------------|-----------|----------|
| BCDLOOP                   |           | 9        |
| Signal Names to Net Names |           |          |
| <b>7. EQ</b>              |           |          |
| Loc-Pin                   | Inst/Name | Net Name |
| U22-1                     | X9/A      | MM/G3    |
| U22-2                     | X8/B      | Q1P1     |
| U22-3                     | X0/Y      | EOLMAX   |
| U22-4                     | X1/Y      | EQLMAX   |
| U22-5                     | X1/A      | MM/G6    |
| U22-6                     | X1/B      | Q1P2     |
| U22-8                     | X2/A      | MM/G7    |
| U22-9                     | X2/B      | Q1P3     |
| U22-10                    | X2/Y      | EQLMAX   |
| U22-11                    | X3/Y      | EOLMAX   |
| U22-12                    | X3/A      | MM/G8    |
| U22-13                    | X3/B      | Q1P4     |

| Symbol                  | Instance | Page            |
|-------------------------|----------|-----------------|
| BCDLOOP                 |          | 10              |
| Nets Sorted by Net Name |          |                 |
| Net Name                |          |                 |
| Page                    | Loc-Pin  | Full Instance   |
| ---                     | -----    | -----           |
| 1\$/GND                 |          |                 |
| P.1                     | U106-1   | 1\$/GND         |
| P.1                     |          | IN/DE           |
| 2\$/GND                 |          |                 |
| P.6                     | U15-13   | CTR/BIT3/LDK/C  |
| P.5                     | U13-11   | CTR/BIT2/LDK/C  |
| P.3                     | U13-13   | CTR/BIT0/LDK/C  |
| P.4                     | U13-5    | CTR/BIT1/LDK/C  |
| P.2                     | U19-5    | CTR/LD/B        |
| P.1                     |          | 2\$/GND         |
| CX                      |          |                 |
| P.2                     | U18-1    | CTR/CX/A        |
| CTR/BIT0/J/Y            |          |                 |
| P.3                     | U109-3   | CTR/BIT0/FF/J   |
| P.3                     | U14-3    | CTR/BIT0/J/Y    |
| CTR/BIT0/K/Y            |          |                 |
| P.3                     | U100-2   | CTR/BIT0/FF/K   |
| P.3                     | U11-6    | CTR/BIT0/K/Y    |
| CTR/BIT0/LDJ/Y          |          |                 |
| P.3                     | U11-1    | CTR/BIT0/J/A    |
| P.3                     | U12-3    | CTR/BIT0/LDJ/Y  |
| CTR/BIT0/LDK/Y          |          |                 |
| P.3                     | U11-5    | CTR/BIT0/K/B    |
| P.3                     | U13-12   | CTR/BIT0/LDK/Y  |
| P.3                     | U12-2    | CTR/BIT0/LDJ/B  |
| CTR/BIT0/LRC/Y          |          |                 |
| P.3                     | U11-2    | CTR/BIT0/J/B    |
| P.3                     | U11-4    | CTR/BIT0/K/A    |
| P.3                     | U14-3    | CTR/BIT0/LRC/Y  |
| CTR/BIT0/PULL/Z         |          |                 |
| P.3                     | R1-2     | CTR/BIT0/PULL/Z |
| P.3                     | U100-15  | CTR/BIT0/FF/CLR |
| P.3                     | U100-4   | CTR/BIT0/FF/PR  |
| CTR/BIT0/ON             |          |                 |
| P.3                     | U100-6   | CTR/BIT0/FF/ON  |
| CTR/BIT1/J/Y            |          |                 |
| P.4                     | U100-11  | CTR/BIT1/FF/J   |
| P.4                     | U11-8    | CTR/BIT1/J/Y    |
| CTR/BIT1/K/Y            |          |                 |
| P.4                     | U100-12  | CTR/BIT1/FF/K   |
| P.4                     | U11-11   | CTR/BIT1/K/Y    |
| CTR/BIT1/LDJ/Y          |          |                 |
| P.4                     | U11-3    | CTR/BIT1/J/A    |
| P.4                     | U12-6    | CTR/BIT1/LDJ/Y  |
| CTR/BIT1/LDK/Y          |          |                 |
| P.4                     | U11-13   | CTR/BIT1/K/B    |
| P.4                     | U13-6    | CTR/BIT1/LDK/Y  |
| P.4                     | U12-3    | CTR/BIT1/LDJ/B  |
| CTR/BIT1/LRC/Y          |          |                 |
| P.4                     | U11-10   | CTR/BIT1/J/B    |
| P.4                     | U11-12   | CTR/BIT1/K/A    |
| P.4                     | U14-6    | CTR/BIT1/LRC/Y  |
| CTR/BIT1/PULL/Z         |          |                 |
| P.4                     | R1-3     | CTR/BIT1/PULL/Z |
| P.4                     | U100-14  | CTR/BIT1/FF/CLR |
| P.4                     | U100-18  | CTR/BIT1/FF/PR  |
| Net Name                |          |                 |
| Page                    | Loc-Pin  | Full Instance   |
| ---                     | -----    | -----           |
| CTR/BIT1/ON             |          |                 |
| P.4                     | U100-7   | CTR/BIT1/FF/ON  |
| CTR/BIT2/J/Y            |          |                 |
| P.5                     | U101-3   | CTR/BIT2/FF/J   |
| P.3                     | U16-3    | CTR/BIT2/J/Y    |
| CTR/BIT2/K/Y            |          |                 |
| P.5                     | U101-2   | CTR/BIT2/FF/K   |
| P.3                     | U16-6    | CTR/BIT2/K/Y    |
| CTR/BIT2/LDJ/Y          |          |                 |
| P.3                     | U16-1    | CTR/BIT2/J/A    |
| P.3                     | U12-8    | CTR/BIT2/LDJ/Y  |
| CTR/BIT2/LDK/Y          |          |                 |
| P.5                     | U16-5    | CTR/BIT2/K/B    |
| P.5                     | U13-8    | CTR/BIT2/LDK/Y  |
| P.3                     | U12-10   | CTR/BIT2/LDJ/B  |
| CTR/BIT2/LRC/Y          |          |                 |
| P.5                     | U16-2    | CTR/BIT2/J/B    |
| P.3                     | U16-4    | CTR/BIT2/K/A    |
| P.5                     | U14-9    | CTR/BIT2/LRC/Y  |
| CTR/BIT2/PULL/Z         |          |                 |
| P.3                     | R1-4     | CTR/BIT2/PULL/Z |
| P.3                     | U101-13  | CTR/BIT2/FF/CLR |
| P.3                     | U101-4   | CTR/BIT2/FF/PR  |
| CTR/BIT2/ON             |          |                 |
| P.5                     | U101-6   | CTR/BIT2/FF/ON  |
| CTR/BIT3/J/Y            |          |                 |
| P.6                     | U101-11  | CTR/BIT3/FF/J   |
| P.6                     | U16-8    | CTR/BIT3/J/Y    |
| CTR/BIT3/K/Y            |          |                 |
| P.6                     | U101-12  | CTR/BIT3/FF/K   |
| P.6                     | U16-11   | CTR/BIT3/K/Y    |
| CTR/BIT3/LDJ/Y          |          |                 |
| P.6                     | U16-9    | CTR/BIT3/J/A    |
| P.6                     | U12-11   | CTR/BIT3/LDJ/Y  |
| CTR/BIT3/LDK/Y          |          |                 |
| P.6                     | U16-13   | CTR/BIT3/K/B    |
| P.6                     | U15-12   | CTR/BIT3/LDK/Y  |
| P.6                     | U12-13   | CTR/BIT3/LDJ/B  |
| CTR/BIT3/LRC/Y          |          |                 |
| P.6                     | U16-10   | CTR/BIT3/J/B    |
| P.6                     | U16-12   | CTR/BIT3/K/A    |
| P.6                     | U14-11   | CTR/BIT3/LRC/Y  |
| CTR/BIT3/PULL/Z         |          |                 |
| P.6                     | R1-5     | CTR/BIT3/PULL/Z |
| P.6                     | U101-14  | CTR/BIT3/FF/CLR |
| P.6                     | U101-18  | CTR/BIT3/FF/PR  |
| CTR/BIT3/ON             |          |                 |
| P.6                     | U101-7   | CTR/BIT3/FF/ON  |
| P.2                     | U17-11   | CTR/CNT1/C      |
| CTR/CX/Y                |          |                 |
| P.6                     | U101-13  | CTR/BIT3/FF/CX  |
| P.5                     | U101-1   | CTR/BIT2/FF/CX  |
| P.3                     | U100-1   | CTR/BIT0/FF/CX  |
| P.4                     | U100-13  | CTR/BIT1/FF/CX  |
| P.2                     | U18-2    | CTR/CX/Y        |

| Symbol       | Instance                |         |                  | Page |
|--------------|-------------------------|---------|------------------|------|
| BCDLOOP      | Nets Sorted by Net Name |         |                  | 11   |
| Net Name     | Page                    | Loc-Pin | Full Instance    |      |
| CTR/CLR3/Y   |                         |         |                  |      |
| P.2          | U18-1                   |         | CTR/FLP3/A       |      |
| P.2          | U17-6                   |         | CTR/CLR3/Y       |      |
| CTR/CNT/Y    |                         |         |                  |      |
| P.2          | U19-3                   |         | CTR/CNT/Y        |      |
| P.3          | U14-2                   |         | CTR/BIT9/LDK/C/B |      |
| P.2          | U17-9                   |         | CTR/CNT1/A       |      |
| P.2          | U21-1                   |         | CTR/CNT3/A       |      |
| P.2          | U20-1                   |         | CTR/CNT2/A       |      |
| P.2          | U17-4                   |         | CTR/CLR3/B       |      |
| CTR/CNT1/Y   |                         |         |                  |      |
| P.2          | U17-8                   |         | CTR/CNT1/Y       |      |
| P.4          | U14-3                   |         | CTR/BIT1/LDK/C/B |      |
| CTR/CNT2/Y   |                         |         |                  |      |
| P.2          | U20-12                  |         | CTR/CNT2/Y       |      |
| P.3          | U14-10                  |         | CTR/BIT2/LDK/C/B |      |
| CTR/CNT3/Y   |                         |         |                  |      |
| P.2          | U18-2                   |         | CTR/FLP3/B       |      |
| P.2          | U21-6                   |         | CTR/CNT3/Y       |      |
| CTR/FLP3/Y   |                         |         |                  |      |
| P.2          | U18-3                   |         | CTR/FLP3/Y       |      |
| P.6          | U14-13                  |         | CTR/BIT3/LDK/C/B |      |
| CTR/LD/Y     |                         |         |                  |      |
| P.6          | U12-12                  |         | CTR/BIT3/LDJ/A   |      |
| P.6          | U14-12                  |         | CTR/BIT3/LDK/C/A |      |
| P.6          | U15-2                   |         | CTR/BIT3/LDK/B   |      |
| P.3          | U12-9                   |         | CTR/BIT2/LDJ/A   |      |
| P.3          | U14-9                   |         | CTR/BIT2/LDK/C/A |      |
| P.3          | U13-10                  |         | CTR/BIT2/LDK/B   |      |
| P.3          | U12-1                   |         | CTR/BIT0/LDJ/A   |      |
| P.3          | U14-1                   |         | CTR/BIT8/LDK/C/A |      |
| P.3          | U13-2                   |         | CTR/BIT8/LDK/B   |      |
| P.4          | U12-4                   |         | CTR/BIT1/LDJ/A   |      |
| P.4          | U14-4                   |         | CTR/BIT1/LDK/C/A |      |
| P.4          | U13-4                   |         | CTR/BIT1/LDK/B   |      |
| P.2          | U19-6                   |         | CTR/LD/Y         |      |
| E0/XMAX      |                         |         |                  |      |
| P.7          | U22-11                  |         | E0/X3/Y          |      |
| P.7          | U22-10                  |         | E0/X2/Y          |      |
| P.7          | U22-4                   |         | E0/X1/Y          |      |
| P.7          | U22-3                   |         | E0/X3/Y          |      |
| P.1          | U23-3                   |         | LD/B             |      |
| LD/Y         |                         |         |                  |      |
| P.2          | U19-4                   |         | CTR/LD/A         |      |
| P.1          | U23-1                   |         | LD/Y             |      |
| LD/MINMAX    |                         |         |                  |      |
| P.1          | U106-11                 |         | MIN/CK           |      |
| MAX1P1       |                         |         |                  |      |
| P.1          | U106-13                 |         | MIN/D5           |      |
| MAX1P2       |                         |         |                  |      |
| P.1          | U106-14                 |         | MIN/D6           |      |
| MAX1P3       |                         |         |                  |      |
| P.1          | U106-17                 |         | MIN/D7           |      |
| MAX1P4       |                         |         |                  |      |
| P.1          | U106-18                 |         | MIN/D8           |      |
| MIN1P1       |                         |         |                  |      |
| P.1          | U106-3                  |         | MIN/D1           |      |
| MIN1P2       |                         |         |                  |      |
| P.1          | U106-4                  |         | MIN/D2           |      |
| MIN1P3       |                         |         |                  |      |
| P.1          | U106-5                  |         | MIN/D3           |      |
| MIN1P4       |                         |         |                  |      |
| P.1          | U106-6                  |         | MIN/D4           |      |
| MIN1P5       |                         |         |                  |      |
| P.1          | U106-7                  |         | MIN/D5           |      |
| MIN1P6       |                         |         |                  |      |
| P.1          | U106-8                  |         | MIN/D6           |      |
| MIN1P7       |                         |         |                  |      |
| P.1          | U106-9                  |         | MIN/D7           |      |
| MIN1P8       |                         |         |                  |      |
| P.1          | U106-10                 |         | MIN/D8           |      |
| MIN1P9       |                         |         |                  |      |
| P.1          | U106-11                 |         | MIN/D9           |      |
| MIN1P10      |                         |         |                  |      |
| P.1          | U106-12                 |         | MIN/D10          |      |
| MIN1P11      |                         |         |                  |      |
| P.1          | U106-13                 |         | MIN/D11          |      |
| MIN1P12      |                         |         |                  |      |
| P.1          | U106-14                 |         | MIN/D12          |      |
| MIN1P13      |                         |         |                  |      |
| P.1          | U106-15                 |         | MIN/D13          |      |
| MIN1P14      |                         |         |                  |      |
| P.1          | U106-16                 |         | MIN/D14          |      |
| MIN1P15      |                         |         |                  |      |
| P.1          | U106-17                 |         | MIN/D15          |      |
| MIN1P16      |                         |         |                  |      |
| P.1          | U106-18                 |         | MIN/D16          |      |
| MIN1P17      |                         |         |                  |      |
| P.1          | U106-19                 |         | MIN/D17          |      |
| MIN1P18      |                         |         |                  |      |
| P.1          | U106-20                 |         | MIN/D18          |      |
| MIN1P19      |                         |         |                  |      |
| P.1          | U106-21                 |         | MIN/D19          |      |
| MIN1P20      |                         |         |                  |      |
| P.1          | U106-22                 |         | MIN/D20          |      |
| MIN1P21      |                         |         |                  |      |
| P.1          | U106-23                 |         | MIN/D21          |      |
| MIN1P22      |                         |         |                  |      |
| P.1          | U106-24                 |         | MIN/D22          |      |
| MIN1P23      |                         |         |                  |      |
| P.1          | U106-25                 |         | MIN/D23          |      |
| MIN1P24      |                         |         |                  |      |
| P.1          | U106-26                 |         | MIN/D24          |      |
| MIN1P25      |                         |         |                  |      |
| P.1          | U106-27                 |         | MIN/D25          |      |
| MIN1P26      |                         |         |                  |      |
| P.1          | U106-28                 |         | MIN/D26          |      |
| MIN1P27      |                         |         |                  |      |
| P.1          | U106-29                 |         | MIN/D27          |      |
| MIN1P28      |                         |         |                  |      |
| P.1          | U106-30                 |         | MIN/D28          |      |
| MIN1P29      |                         |         |                  |      |
| P.1          | U106-31                 |         | MIN/D29          |      |
| MIN1P30      |                         |         |                  |      |
| P.1          | U106-32                 |         | MIN/D30          |      |
| MIN1P31      |                         |         |                  |      |
| P.1          | U106-33                 |         | MIN/D31          |      |
| MIN1P32      |                         |         |                  |      |
| P.1          | U106-34                 |         | MIN/D32          |      |
| MIN1P33      |                         |         |                  |      |
| P.1          | U106-35                 |         | MIN/D33          |      |
| MIN1P34      |                         |         |                  |      |
| P.1          | U106-36                 |         | MIN/D34          |      |
| MIN1P35      |                         |         |                  |      |
| P.1          | U106-37                 |         | MIN/D35          |      |
| MIN1P36      |                         |         |                  |      |
| P.1          | U106-38                 |         | MIN/D36          |      |
| MIN1P37      |                         |         |                  |      |
| P.1          | U106-39                 |         | MIN/D37          |      |
| MIN1P38      |                         |         |                  |      |
| P.1          | U106-40                 |         | MIN/D38          |      |
| MIN1P39      |                         |         |                  |      |
| P.1          | U106-41                 |         | MIN/D39          |      |
| MIN1P40      |                         |         |                  |      |
| P.1          | U106-42                 |         | MIN/D40          |      |
| MIN1P41      |                         |         |                  |      |
| P.1          | U106-43                 |         | MIN/D41          |      |
| MIN1P42      |                         |         |                  |      |
| P.1          | U106-44                 |         | MIN/D42          |      |
| MIN1P43      |                         |         |                  |      |
| P.1          | U106-45                 |         | MIN/D43          |      |
| MIN1P44      |                         |         |                  |      |
| P.1          | U106-46                 |         | MIN/D44          |      |
| MIN1P45      |                         |         |                  |      |
| P.1          | U106-47                 |         | MIN/D45          |      |
| MIN1P46      |                         |         |                  |      |
| P.1          | U106-48                 |         | MIN/D46          |      |
| MIN1P47      |                         |         |                  |      |
| P.1          | U106-49                 |         | MIN/D47          |      |
| MIN1P48      |                         |         |                  |      |
| P.1          | U106-50                 |         | MIN/D48          |      |
| MIN1P49      |                         |         |                  |      |
| P.1          | U106-51                 |         | MIN/D49          |      |
| MIN1P50      |                         |         |                  |      |
| P.1          | U106-52                 |         | MIN/D50          |      |
| MIN1P51      |                         |         |                  |      |
| P.1          | U106-53                 |         | MIN/D51          |      |
| MIN1P52      |                         |         |                  |      |
| P.1          | U106-54                 |         | MIN/D52          |      |
| MIN1P53      |                         |         |                  |      |
| P.1          | U106-55                 |         | MIN/D53          |      |
| MIN1P54      |                         |         |                  |      |
| P.1          | U106-56                 |         | MIN/D54          |      |
| MIN1P55      |                         |         |                  |      |
| P.1          | U106-57                 |         | MIN/D55          |      |
| MIN1P56      |                         |         |                  |      |
| P.1          | U106-58                 |         | MIN/D56          |      |
| MIN1P57      |                         |         |                  |      |
| P.1          | U106-59                 |         | MIN/D57          |      |
| MIN1P58      |                         |         |                  |      |
| P.1          | U106-60                 |         | MIN/D58          |      |
| MIN1P59      |                         |         |                  |      |
| P.1          | U106-61                 |         | MIN/D59          |      |
| MIN1P60      |                         |         |                  |      |
| P.1          | U106-62                 |         | MIN/D60          |      |
| MIN1P61      |                         |         |                  |      |
| P.1          | U106-63                 |         | MIN/D61          |      |
| MIN1P62      |                         |         |                  |      |
| P.1          | U106-64                 |         | MIN/D62          |      |
| MIN1P63      |                         |         |                  |      |
| P.1          | U106-65                 |         | MIN/D63          |      |
| MIN1P64      |                         |         |                  |      |
| P.1          | U106-66                 |         | MIN/D64          |      |
| MIN1P65      |                         |         |                  |      |
| P.1          | U106-67                 |         | MIN/D65          |      |
| MIN1P66      |                         |         |                  |      |
| P.1          | U106-68                 |         | MIN/D66          |      |
| MIN1P67      |                         |         |                  |      |
| P.1          | U106-69                 |         | MIN/D67          |      |
| MIN1P68      |                         |         |                  |      |
| P.1          | U106-70                 |         | MIN/D68          |      |
| MIN1P69      |                         |         |                  |      |
| P.1          | U106-71                 |         | MIN/D69          |      |
| MIN1P70      |                         |         |                  |      |
| P.1          | U106-72                 |         | MIN/D70          |      |
| MIN1P71      |                         |         |                  |      |
| P.1          | U106-73                 |         | MIN/D71          |      |
| MIN1P72      |                         |         |                  |      |
| P.1          | U106-74                 |         | MIN/D72          |      |
| MIN1P73      |                         |         |                  |      |
| P.1          | U106-75                 |         | MIN/D73          |      |
| MIN1P74      |                         |         |                  |      |
| P.1          | U106-76                 |         | MIN/D74          |      |
| MIN1P75      |                         |         |                  |      |
| P.1          | U106-77                 |         | MIN/D75          |      |
| MIN1P76      |                         |         |                  |      |
| P.1          | U106-78                 |         | MIN/D76          |      |
| MIN1P77      |                         |         |                  |      |
| P.1          | U106-79                 |         | MIN/D77          |      |
| MIN1P78      |                         |         |                  |      |
| P.1          | U106-80                 |         | MIN/D78          |      |
| MIN1P79      |                         |         |                  |      |
| P.1          | U106-81                 |         | MIN/D79          |      |
| MIN1P80      |                         |         |                  |      |
| P.1          | U106-82                 |         | MIN/D80          |      |
| MIN1P81      |                         |         |                  |      |
| P.1          | U106-83                 |         | MIN/D81          |      |
| MIN1P82      |                         |         |                  |      |
| P.1          | U106-84                 |         | MIN/D82          |      |
| MIN1P83      |                         |         |                  |      |
| P.1          | U106-85                 |         | MIN/D83          |      |
| MIN1P84      |                         |         |                  |      |
| P.1          | U106-86                 |         | MIN/D84          |      |
| MIN1P85      |                         |         |                  |      |
| P.1          | U106-87                 |         | MIN/D85          |      |
| MIN1P86      |                         |         |                  |      |
| P.1          | U106-88                 |         | MIN/D86          |      |
| MIN1P87      |                         |         |                  |      |
| P.1          | U106-89                 |         | MIN/D87          |      |
| MIN1P88      |                         |         |                  |      |
| P.1          | U106-90                 |         | MIN/D88          |      |
| MIN1P89      |                         |         |                  |      |
| P.1          | U106-91                 |         | MIN/D89          |      |
| MIN1P90      |                         |         |                  |      |
| P.1          | U106-92                 |         | MIN/D90          |      |
| MIN1P91      |                         |         |                  |      |
| P.1          | U106-93                 |         | MIN/D91          |      |
| MIN1P92      |                         |         |                  |      |
| P.1          | U106-94                 |         | MIN/D92          |      |
| MIN1P93      |                         |         |                  |      |
| P.1          | U106-95                 |         | MIN/D93          |      |
| MIN1P94      |                         |         |                  |      |
| P.1          | U106-96                 |         | MIN/D94          |      |
| MIN1P95      |                         |         |                  |      |
| P.1          | U106-97                 |         | MIN/D95          |      |
| MIN1P96      |                         |         |                  |      |
| P.1          | U106-98                 |         | MIN/D96          |      |
| MIN1P97      |                         |         |                  |      |
| P.1          | U106-99                 |         | MIN/D97          |      |
| MIN1P98      |                         |         |                  |      |
| P.1          | U106-100                |         | MIN/D98          |      |
| MIN1P99      |                         |         |                  |      |
| P.1          | U106-101                |         | MIN/D99          |      |
| MIN1P100     |                         |         |                  |      |
| P.1          | U106-102                |         | MIN/D100         |      |
| MIN1P101     |                         |         |                  |      |
| P.1          | U106-103                |         | MIN/D101         |      |
| MIN1P102     |                         |         |                  |      |
| P.1          | U106-104                |         | MIN/D102         |      |
| MIN1P103     |                         |         |                  |      |
| P.1          | U106-105                |         | MIN/D103         |      |
| MIN1P104     |                         |         |                  |      |
| P.1          | U106-106                |         | MIN/D104         |      |
| MIN1P105     |                         |         |                  |      |
| P.1          | U106-107                |         | MIN/D105         |      |
| MIN1P106     |                         |         |                  |      |
| P.1          | U106-108                |         | MIN/D106         |      |
| MIN1P107     |                         |         |                  |      |
| P.1          | U106-109                |         | MIN/D107         |      |
| MIN1P108     |                         |         |                  |      |
| P.1          | U106-110                |         | MIN/D108         |      |
| MIN1P109     |                         |         |                  |      |
| P.1          | U106-111                |         | MIN/D109         |      |
| MIN1P110     |                         |         |                  |      |
| P.1          | U106-112                |         | MIN/D110         |      |
| MIN1P111     |                         |         |                  |      |
| P.1          | U106-113                |         | MIN/D111         |      |
| MIN1P112     |                         |         |                  |      |
| P.1          | U106-114                |         | MIN/D112         |      |
| MIN1P113     |                         |         |                  |      |
| P.1          | U106-115                |         | MIN/D113         |      |
| MIN1P114     |                         |         |                  |      |
| P.1          | U106-116                |         | MIN/D114         |      |
| MIN1P115     |                         |         |                  |      |
| P.1          | U106-117                |         | MIN/D115         |      |
| MIN1P116     |                         |         |                  |      |
| P.1          | U106-118                |         | MIN/D116         |      |
| MIN1P117     |                         |         |                  |      |
| P.1          | U106-119                |         | MIN/D117         |      |
| MIN1P118     |                         |         |                  |      |
| P.1          | U106-120                |         | MIN/D118         |      |
| MIN1P119     |                         |         |                  |      |
| P.1          | U106-121                |         | MIN/D119         |      |
| MIN1P120     |                         |         |                  |      |
| P.1          | U106-122                |         | MIN/D120         |      |
| MIN1P121     |                         |         |                  |      |
| P.1          | U106-123                |         | MIN/D121         |      |
| MIN1P122     |                         |         |                  |      |
| P.1          | U106-124                |         | MIN/D122         |      |
| MIN1P123     |                         |         |                  |      |
| P.1          | U106-125                |         | MIN/D123         |      |
| MIN1P124     |                         |         |                  |      |
| P.1          | U106-126                |         | MIN/D124         |      |
| MIN1P125     |                         |         |                  |      |
| P.1          | U106-127                |         | MIN/D125         |      |
| MIN1P126     |                         |         |                  |      |
| P.1          | U106-128                |         | MIN/D126         |      |
| MIN1P127</td |                         |         |                  |      |

| Symbol                  |      | Instance | Page          |
|-------------------------|------|----------|---------------|
| Nets Sorted by Net Name |      |          | 12            |
| Net Name                | Page | Loc-Pin  | Full Instance |
| RC                      | P.2  | U17-12   | CTR/CARRY/Y   |
| RESET                   | P.1  | U23-2    | LD/A          |
| T                       | P.2  | U19-2    | CTR/CNT/B     |

**APPENDIX J**

**Special DP Files for Printset.**

| Symbol     | Instance     | Page      |
|------------|--------------|-----------|
| [*Symbols] | [*Instances] | [*Pages*] |
|            |              |           |

Page.DP

| Symbol      | Instance     | Page     |
|-------------|--------------|----------|
| [#Symbols]  | [#Instances] | [#Pages] |
| [*Column1*] |              |          |
| [*Column1*] |              |          |
| [*Column1*] |              |          |

Contents.Dp

SignalToNet.Dp

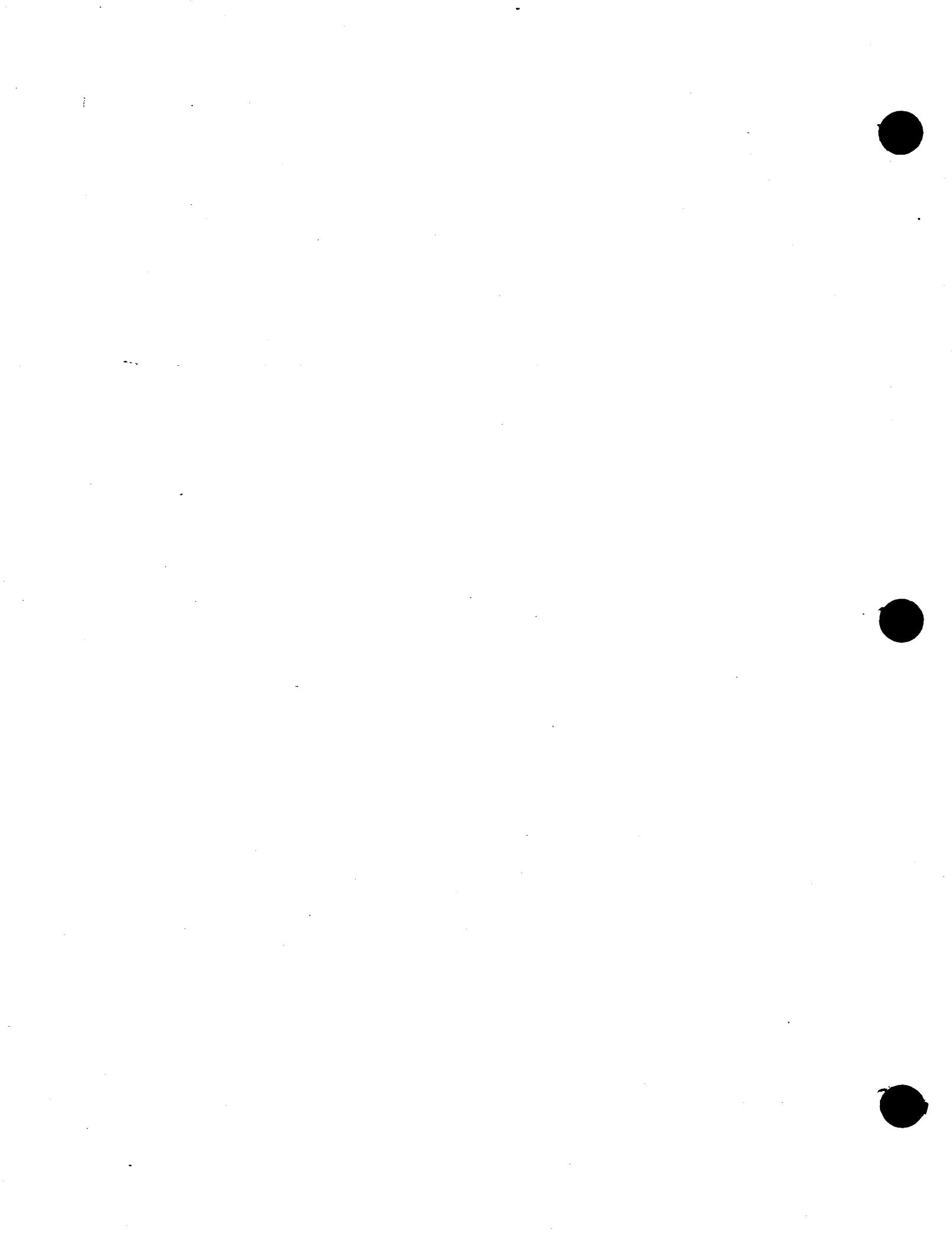
**CHAPTER 12**  
**SYSTEM D/L LIBRARIES**



## CHAPTER 12

## SYSTEM D/L LIBRARIES

|  |    |
|--|----|
| GATE ARRAY LIBRARIES                                 | 2  |
| COMPONENT LIBRARY                                    | 3  |
| The Component Phrase                                 | 3  |
| The Parts of the Component Phrase                    | 4  |
| The Package Library                                  | 5  |
| The Parts of the Package Phrase                      | 6  |
| The TIL Library Used by Dwl                          | 6  |
| APPENDIX A - Syntax of the Component Phrase          | 8  |
| APPENDIX B - DP Drawings of S240, S240x4, and S240x1 | 9  |
| APPENDIX C - Syntax of the Package Phrase            | 11 |
| APPENDIX D - Excerpt from Package.Lib                | 12 |
| APPENDIX E - Excerpt from the TIL Library            | 15 |



A System D/L part library is a collection of primitive parts which are available for designing circuits. The library consists of the DP and SL files for the primitive parts. The component design library includes some additional DP drawings and text files that are not needed for gate array design. This chapter describes the contents of the System D/L libraries, but assumes a general familiarity with System D/L including component design.

It is probably best to assign a single person the duty of librarian. This engineer or technician should be responsible for creating the initial library and making additions and corrections. This chapter provides a reference manual for the librarian and explains the differences between the various types of libraries.

System D/L uses two kinds of libraries: gate array libraries and component libraries. These libraries differ only in the way that physical information is represented. Gate array libraries need only describe the logical characteristics of the primitive parts. These parts are specified by the manufacturer of the gate arrays themselves. Component libraries describe physical components. These libraries describe the logical and physical characteristics of the primitive parts.

A library is created in several steps.

1. Use DP to draw the primitive parts.
2. Use SL to create the SL files that correspond to the DP files.
3. Create any additional DP drawings and text files that are needed.

A convenient way to organize libraries on a PERQ is to place all library files in a single directory. This directory should contain the DP files, the SL files, and any additional files needed. Adding this directory to the searchlist with the POS SetSearch command makes the contents of the library available to DP and DA.

GATE ARRAY LIBRARIES

The definition of a primitive part in a gate array library is similar to the definition of a non-primitive part. The definition rectangle of a primitive part contains the icons for the part. All icons for a particular part must have the same number of pins and the same pin names. The characteristics of each pin (input pin, output pin, etc.) must be described. See the chapter SL - SCHEMATIC WIRELISTER for a detailed description of how to define a part.

Unlike non-primitives, a primitive part has no implementation drawing; it is not defined in terms of other parts. SL, however, requires the drawing to have both a definition rectangle and an implementation drawing. When SL is applied to a primitive part, it complains that there is no implementation drawing. In spite of the error message, SL creates a correct output file.

If the part will be used in a Tegas simulation, its definition rectangle must contain a text string of the form

( TDL <fileName> )

containing the name of its Tegas Design Language file. If the part will be used in a Fujitsu gate array design, its definition rectangle must contain a text string of the form

( FLDL <fileName> )

containing the name of its Fujitsu Logic Design Language file.

COMPONENT LIBRARY

A component library consists of

- o The DP and SL files for the primitive parts.
- o The package library.
- o The TIL library used by Dwl.
- o Special DP files used by the DA PrintSet command. These files are described in the chapter COMPONENT DESIGN.

A DP file in a component library contains the same information as its gate-array counterpart and in addition contains a description of the physical component. The physical description is called the component phrase and is included within the part's definition rectangle.

The Component Phrase

The DP file for a primitive part in a component library differs from a part in a gate array library in that it must describe the physical and electrical characteristics of the component. This information is included inside the definition rectangle for the part and describes

- o The part name.
- o An English description of the part.
- o The name of the package shape.
- o The part's power requirements.
- o The name of each type of section.
- o The conversion from icon pin names to actual pin numbers.

This information is contained in the component phrase in the definition rectangle. The component phrase must conform to a strict syntax which makes liberal use of nested, parenthesized phrases. Appendix A shows the syntax of the component phrase. Appendix B shows the three DP files used to define the S240.

Note that the component phrase contained in S240.DP describes all ways of using the S240 even though they appear in different DP files. Text symbols are usually used for component phrases because they are generally very long. See the chapter SL - SCHEMATIC WIRELISTER for a description of Text symbols.

When a part appears in several DP files (because there are different ways to section it) the component phrase should appear in the DP file describing the whole component. The other DP files contain the phrase

(See <ParentDpName>)

instead. For example, S240x4 and S240x1 both contain the phrase

(See S240)

#### The Parts of the Component Phrase

- o The word Component inside the outermost parentheses identifies this as a component phrase.
- o The Label phrase contains the part name. This must be the name of the whole component.
- o The Description phrase contains an English description of the part.
- o The Package phrase contains the name of the physical packaging of the part. This name identifies one of the packages in the package library. See the next section for a description of the package library. Package names are generally DIP for Dual Inline Package or SIP for Single Inline Package followed by the number of pins on the package. Note, though, that there may be several packages with the same number of pins but different shapes. For example, Dip24 is a 0.6 inch wide 24-pin part and Slim24 is a 0.3 inch wide 24-pin part.
- o The Power phrase describes the power and ground characteristics of the part.
  - \* The Gnd phrase identifies the ground pin of the part.
  - \* Each Voltage phrase describes a single power pin of the part. Some devices may require more than one voltage level.
    - . The VCC phrase identifies the power pin.
    - . The Volts phrase specifies the voltage value for this pin.

- The Current phrase contains the minimum, typical, and maximum currents drawn at this voltage. This phrase is optional.
- o The Section phrases describe the different ways that this part may be divided into sections. For example, the S240 may be used as a single section (the whole component), two 4-bit sections, or eight 1-bit sections.
  - \* The word Section identifies this as a section phrase.
  - \* The Label phrase contains the icon name of this type of section. Using the S240 as an example, there are three section phrases--S240, S240x4, and S240x4.
  - \* Each Element phrase describes an actual section of the part. There is an Element phrase for each possible section. Thus for a whole component there is a single component phrase, but for a component divided into N sections there are N element phrases.
    - The Label phrase contains the section number of this section element. This corresponds to the section number which the designer assigns in the location file. The Label phrase is omitted for elements that describe the whole component.
    - The Pins phrase describes all the pins of a particular element by providing the association between pin name and pin number.

### The Package Library

The package library describes all of the package styles which are used by a particular component library. The package library is a text file named Package.Lib which may be edited with any text editor. This file contains the names and descriptions of all packages named in the component phrases of the component library. Each entry in the package library contains

- o The name of the package.
- o An English description of the package.
- o The width of the package in library-specific units. The unit is generally the smallest possible pin spacing. For TIL design, the unit is 0.1 inch.

- o The length of the package in units.
- o A list of the pins on the package and their X,Y offsets with respect to the lower-left corner (as the part is viewed from the top). The lower-left corner is usually pin 1.

Each entry in the package library is called a package phrase. Each package phrase must conform to a strict syntax which makes liberal use of nested, parenthesized phrases. Appendix C shows the syntax of the package phrase. Appendix D shows an excerpt from the default package library.

#### The Parts of the Package Phrase

- o The word Package inside the outermost parentheses identifies this as a package phrase.
- o The Label phrase contains the package name. This is the name used in Package phrases in the component library.
- o The Description phrase contains an English description of the package.
- o Width is the width if the part in units. For a DIP this is the pin-spacing between the rows of pins.
- o Height is the height (length) of the part in units. For a SIP or a DIP this is one larger than the number of pins on a side.
- o The Posts phrase describes the pins on the part. The word Posts is followed by the pin number, its X coordinate (width direction), and its Y coordinate (height direction). The X and Y coordinates are specified as offsets from pin 1 when the component is viewed from its labeled side and pin 1 is at the lower-left corner.

#### The TIL Library Used by DwI

The TIL library is a text file which is read by DwI from a file named TIL.Lib. TIL.Lib may be edited with any text editor. The TIL library describes the physical and electrical characteristics of components. The astute reader will note that this duplicates the information in the component phrases of the component library itself. This duplication is undesirable, but allows DA and DWL to read the information in the form that is most appropriate to them. Appendix E shows an excerpt from the default TIL library.

Comments may appear at the beginning of the TIL library and are indicated by a semicolon in column 1. Following these comments are part entries. Each part entry describes one or more parts. Parts which have the same pin-outs may be described by a single entry.

Each part entry starts with a blank line. Each subsequent line that has an asterisk in column 1 defines a single part. The part name immediately follows the asterisk. The part name is followed by the number of pins on the package and the width in tenths of inches. An English description of the part may be added to the right of the width.

The pins of the package are described on the lines that follow the part names. For a part with a single section (e.g. LS161), there is a single line for each pin. This line starts with a description of the pin as a ground pin (GND), power pin (VCC), input pin (TI), output pin (TO), tri-state output pin (TOT), open-collector pin (TOC), bi-directional pin (ANA), or analog pin (ANA). The pin number is to the right of the pin type.

When a part has multiple sections, each pin-description line has a pin number for each section. The pin number in the first section appears as the first number on the line, and so on. Thus an entire section is described by a column of pin numbers. For example, the S00 has four sections: pins 1, 2, and 3; pins 4, 5, and 6; pins 9, 10, and 8; and pins 12, 13, and 11. When a pin is shared among sections, its pin number appears once for each section. For example, pin 1 is shared among sections 1, 2, 3, and 4 of the S240x1 and pin 19 is shared among sections 5, 6, 7, and 8. Ground and power pins are described by a single pin number even if the part has multiple sections.

## APPENDIX A

## Syntax of the Component Phrase

- ' ... ' - Delimits literal text.
- " ... " - Delimits literal text.
- [ ... ] - Delimits optional portions.
- { ... } - Delimits portions which may be repeated any number of times (including zero).
- < ... > - Delimits a descriptive phrase.
- | - Separates alternatives.

**ComponentPhrase** = '( ' 'Component'  
 ' ' Label' <Component Name> ')'  
 ' ' Description' <Quoted String> ')'  
 ' ' Package' <Package Name> ')'  
**PowerPhrase**  
 { SectionPhrase }  
 ')'

**PowerPhrase** = '( ' 'Power'  
**GroundPhrase**  
 { VoltagePhrase }  
 ')'

**GroundPhrase** = '( ' 'Gnd' <Pin Number> ')'

**VoltagePhrase** = '( ' 'Voltage'  
 ' ' VCC' <Pin Number> ')'  
 ' ' Volts' <Voltage Number> ')'  
 [ '( ' 'Current' <Min> <Typ> <Max> ')']  
 ')'

**SectionPhrase** = '( ' 'Section'  
 ' ' Label' <Section Name> ')'  
 { ElementPhrase }  
 ')'

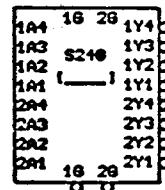
**ElementPhrase** = '( ' 'Element'  
 [ '( ' 'Label' <Section Number> ')']  
 { PinsPhrase }  
 ')'

**PinsPhrase** = '( ' 'Pins'  
 { '( ' <Pin Name> <Pin Number> ')'}  
 ')'

## APPENDIX B

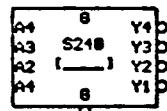
## DP Drawings of S240, S240x4, and S240x1

```
DEFINES S240
  (INPUTS 1A1 1A3 1A2 1A1 2A4 2A3 2A2 2A1 16 20)
  (TRISTATE 1Y4 1Y3 1Y2 1Y1 2Y4 2Y3 2Y2 2Y1)
  (TDL S240)
```



```
(Component
  (Label S240)
  (Description 'octal inverting buffer w/tristate output')
  (Package DIP20)
(Power
  (Gnd 18)
  (Voltage
    (Volts 5)
    (Current
      (100 125 150))
    (Vcc 20)))
(Section
  (Label S240)
  (element
    (pins
      (1A1 2)
      (1A2 4)
      (1A3 6)
      (1A4 8)
      (2A1 11)
      (2A2 13)
      (2A3 15)
      (2A4 17)
      (1Y1 18)
      (1Y2 16)
      (1Y3 14)
      (1Y4 12)
      (2Y1 9)
      (2Y2 7)
      (2Y3 5)
      (2Y4 3)
      (18 1)
      (20 19))))
(Section
  (Label S240X4)
  (element
    (Label -1)
    (pins
      (A1 2)
      (A2 4)
      (A3 6)
      (A4 8)
      (Y1 18)
      (Y2 16)
      (Y3 14)
      (Y4 12)
      (B 1)))
    (element
      (Label -2)
      (pins
        (A1 11)
        (Y 18)
        (B 1)))
    (element
      (Label -3)
      (pins
        (A 8)
        (Y 14)
        (B 1)))
    (element
      (Label -4)
      (pins
        (A 8)
        (Y 12)
        (B 1)))
    (element
      (Label -5)
      (pins
        (A 11)
        (Y 9)
        (B 19)))
    (element
      (Label -6)
      (pins
        (A 13)
        (Y 7)
        (B 19)))
    (element
      (Label -7)
      (pins
        (A 15)
        (Y 5)
        (B 19)))
    (element
      (Label -8)
      (pins
        (A 17)
        (Y 3)
        (B 19))))))
```

DEFINES S240X4  
(INPUTS A1 A2 A3 A4 8)  
(OUTPUTS Y1 Y3 Y2 Y1)  
(TDL S240X1)  
(SEE S240)



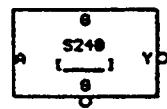
PERQ SYSTEM D/L

COMPONENT LIBRARY

DATE CREATED:  
83 Jan 83

© 1983 Three Rivers Computer Corporation, 720 Gross Street Pittsburgh, PA. 15224. All rights reserved.

DEFINES S240X1  
(INPUTS A 8)  
(OUTPUTS Y)  
(TDL S240X1)  
(SEE S240)



PERQ SYSTEM D/L

COMPONENT LIBRARY

DATE CREATED:  
83 Jan 83

© 1983 Three Rivers Computer Corporation, 720 Gross Street Pittsburgh, PA. 15224. All rights reserved.

## APPENDIX C

## Syntax of the Package Phrase

- ' ... ' - Delimits literal text.
- " ... " - Delimits literal text.
- [ ... ] - Delimits optional portions.
- { ... } - Delimits portions which may be repeated any number of times (including zero).
- < ... > - Delimits a descriptive phrase.
- | - Separates alternatives.

```
PackagePhrase = '( ' 'Package'
                  ' (' 'Label' <Package Name> ')'
                  ' (' 'Description' <Quoted String> ')'
                  ' (' 'Width' <Number> ')
                  ' (' 'Height' <Number> ')
                  PostPhrase
                ')'

PostPhrase = '( ' 'Posts'
                 ' (' <PinNumber> <XOffset> <YOffset> ')
               )'
```

(13 6 11)  
(14 6 10)  
(15 6 9)  
(16 6 8)  
(17 6 7)  
(18 6 6)  
(19 6 5)  
(20 6 4)  
(21 6 3)  
(22 6 2)  
(23 6 1)  
(24 6 0)))

## APPENDIX E

## Excerpt from the TIL Library

;for use with dp ver3.37 and up

|        |           |                                     |
|--------|-----------|-------------------------------------|
| *00    | 14 3      | quad 2-inp pos nand                 |
| *s00   | 14 3      |                                     |
| *ls00  | 14 3      |                                     |
| *03    | 14 3      | quad 2-inp pos nand w/oc out        |
| *s03   | 14 3      |                                     |
| *ls03  | 14 3      |                                     |
| *s08   | 14 3      | quad 2-inp pos and                  |
| *ls08  | 14 3      |                                     |
| *s09   | 14 3      | quad 2-inp pos and                  |
| *ls09  | 14 3      |                                     |
| *ls26  | 14 3      | quad 2-inp high voltage pos nand    |
| *s32   | 14 3      | quad 2 inp pos or                   |
| *ls32  | 14 3      |                                     |
| *s37   | 14 3      | quad 2-inp pos nand buffer          |
| *ls37  | 14 3      |                                     |
| *38    | 14 3      | quad 2-inp pos nand buffer w/oc out |
| *s38   | 14 3      |                                     |
| *ls38  | 14 3      |                                     |
| *f86   | 14 3      | quad 2-inp xor                      |
| *s86   | 14 3      |                                     |
| *ls86  | 14 3      |                                     |
| *ls136 | 14 3      | quad xor                            |
| ti     | 1 4 9 12  |                                     |
| ti     | 2 5 10 13 |                                     |
| to     | 3 6 8 11  |                                     |
| gnd    | 7         |                                     |
| vcc    | 14        |                                     |

\*ls161 16 3 sync 4-bit binary counter w/direct clr  
\*s163 16 3 binary counter w/sync clr  
\*ls163 16 3  
\*s169 16 3 sync u/d counter  
\*ls169 16 3  
\*s195 16 3 parallel-access shift reg  
\*ls195 16 3

ti 1  
ti 2  
ti 3  
ti 4  
ti 5  
ti 6  
ti 7  
gnd 8  
ti 9  
ti 10  
to 11  
to 12  
to 13  
to 14  
to 15  
vcc 16

\*s240 20 3 octal inverting buffer w/3-state out  
\*ls240 20 3  
\*s241 20 3 octal buffer w/3-state out - 4 inverting, 4 non  
\*ls241 20 3  
\*s244 20 3 octal buffer w/3-state out  
\*ls244 20 3

ti 1  
ti 2  
tot 3  
ti 4  
tot 5  
ti 6  
tot 7  
ti 8  
tot 9  
gnd 10  
ti 11  
tot 12  
ti 13  
tot 14  
ti 15  
tot 16  
ti 17  
tot 18  
ti 19  
vcc 20

## SYSTEM DAL LIBRARIES

20 Jul 83

\*s240xi 20 3 octal inverting buffer  
\*ls240x1 20 3  
\*s241x1 20 3 octal inverting/non invert buffer  
\*ls241x1 20 3  
\*s244x1 20 3 octal buffer w/3-state out  
\*ls244x1 20 3  
ti 1 1 1 1 19 19 19 19  
ti 2 4 6 8 11 13 15 17  
tot 18 16 14 12 9 7 5 3  
gnd 10  
vcc 20

\*s244x4 20 3

ti 2 11  
ti 4 13  
ti 6 15  
ti 8 17  
ti 1 19  
tot 18 9  
tot 16 7  
tot 14 5  
tot 12 3  
gnd 10  
vcc 20

