DIGITAL SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES
DEPARTMENT OF ELECTRICAL ENGINEERING
STANFORD UNIVERSITY · STANFORD, CA 94305

SPRINT - AN INTERACTIVE SYSTEM FOR PRINTED CIRCUIT BOARD DESIGN USER'S GUIDE

W.M. vanCleemput
T.C. Bennett
J.A. Hupp
K.R. Stevens

Technical Report No. 143

June 1977

This work was supported by the U. S. Energy Research and Development Administration under contract E4-76-C-03-0515 and by the Joint Services Electronics Program under contract N-00014-75-C-0601.

Also published by Stanford Linear Accelerator Center, Stanford, California, as Computation Research Group Technical Memorandum CGTM-187.

SPRINT - An Interactive System for Printed Circuit Board Design

User's Guide

W.M. vanCleemput
T.C. Bennett
J.A. Hupp
K.R. Stevens

Technical Report No. 143

June 1977

Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

This work was supported by the U. S. Energy Research and Development Administration under contract E4-76-C-03-0515 and by the Joint Services Electronics Program under contract N-00014-75-C-0601.

Also published by Stanford Linear Accelerator Center, Stanford, California, as Computation Research Group Technical Memorandum CGTM-187.

SPRINT - An Interactive System for Printed Circuit Board Design

User's Guide

W.M. vanCleemput
T.C. Bennett
J.A. Hupp
K.R. Stevens

Technical Report No. 143

June 1977

Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

ABSTRACT

The SPRINT system for the design of printed circuit boards is a collection of programs that allows designers to interactively design two-sided boards using a Tektronix 4013 graphics terminal. The major parts of the system are: a compiler for SDL, the Structural Design Language, an interactive component placement program, an interactive manual conductor routing program, an automatic batch router, a via elimination program and a set of artwork generation programs.

INDEX TERMS: printed circuit board design, computer-aided design, design automation

This work was supported by the U. S. Energy Research and Development Administration under contract E4-76-C-03-0515 and by the Joint Services Electronics Program under contract N-00014-75-C-0601.

Also published by Stanford Linear Accelerator Center, Stanford, California, as Computation Research Group Technical Memorandum CGTM-187.

SPRINT - An Interactive System for Printed Circuit Board Design

User's Guide

0. Introduction 1

1. System Overview 3

2. The Description of Circuits in SDL 5

2.1 SDL Language 5

2.2 An Example 13

2.3 How to Run SDLCOMP 16

2.4 Error Messages 17

2.5 SDL Object Language Format 21

2.6 Sample Output of the SDL Compiler 25

3. Logical and Physical Libraries 35

3.1 Logical Library 35

3.2 Physical Library 39

3.3 How to Add a Component to a Logical Library 42

3.4 Board Description 43

4. Design File Generation 45

4.1 Introduction 45

4.2 How to Run SDLPCGEN 48

4.3 How to Run LOADFILE 50

4.4 Error Messages 51

5. Interactive Placement Program (PLACER) 56

5.1 The Placement Process 56

5.2 Using PLACER 58

5.3 Error Messages 70

6. Manual Prewiring of Critical Connections (MWIRE) 76

6.1 Introduction 76

6.2 Using MWIRE 76

6.3 Error Messages 81

7. Batch Routing Program (HIWIRE) 85
7.1 Introduction 85
7.2 Using HIWIRE 87
7.3 Error Messages 96
8. Via Elimination 99
8.1 Introduction 99
8.2 Running VIAELIM 100
8.3 Error Messages 102
9. Output Plot Generation 104
9.1 PLOT 104
9.2 ARTWORK 106
10. Utilities 122
10.1 DUMP and RESTORE 122
10.2 PRINT 123
10.3 Running DUMP and PRINT 123
10.4 Running RESTORE 124
11. Future Enhancements 127

0. INTRODUCTION

The objective of this system is to allow interactive computer-assisted design of printed circuit boards. The SPRINT system allows for the manual placement of critical components and for automatic placement of certain other components such as 14 and 16-pin dual-in-line packages. The interconnection routing module allows for manual routing of critical connections and for automatic routing of non-critical connections. The current system is limited to two signal layers, but a future expansion to multilayer boards has been planned.

The input to the system is in the form of an input language called the Structural Description Language (SDL). In the future the SDL description will also be used as the input to a logic simulator, to a fault test generation/simulation system and to an automatic logic diagram generation capability.

The current system is implemented in MORTRAN and FORTRAN IV on the IBM 370 system at SLAC and makes use of a Tektronix 4013 terminal. The SDL compiler is implemented in SPITBOL, a SNOBOL dialect. Currently, the output of the system is in the form of a Calcomp plot, from which the artwork has to be generated manually.

Future enhancements will be discussed in section 11. In order to illustrate the working of the system a small example will be used consistently throughout this manual. (Figure 0.1).

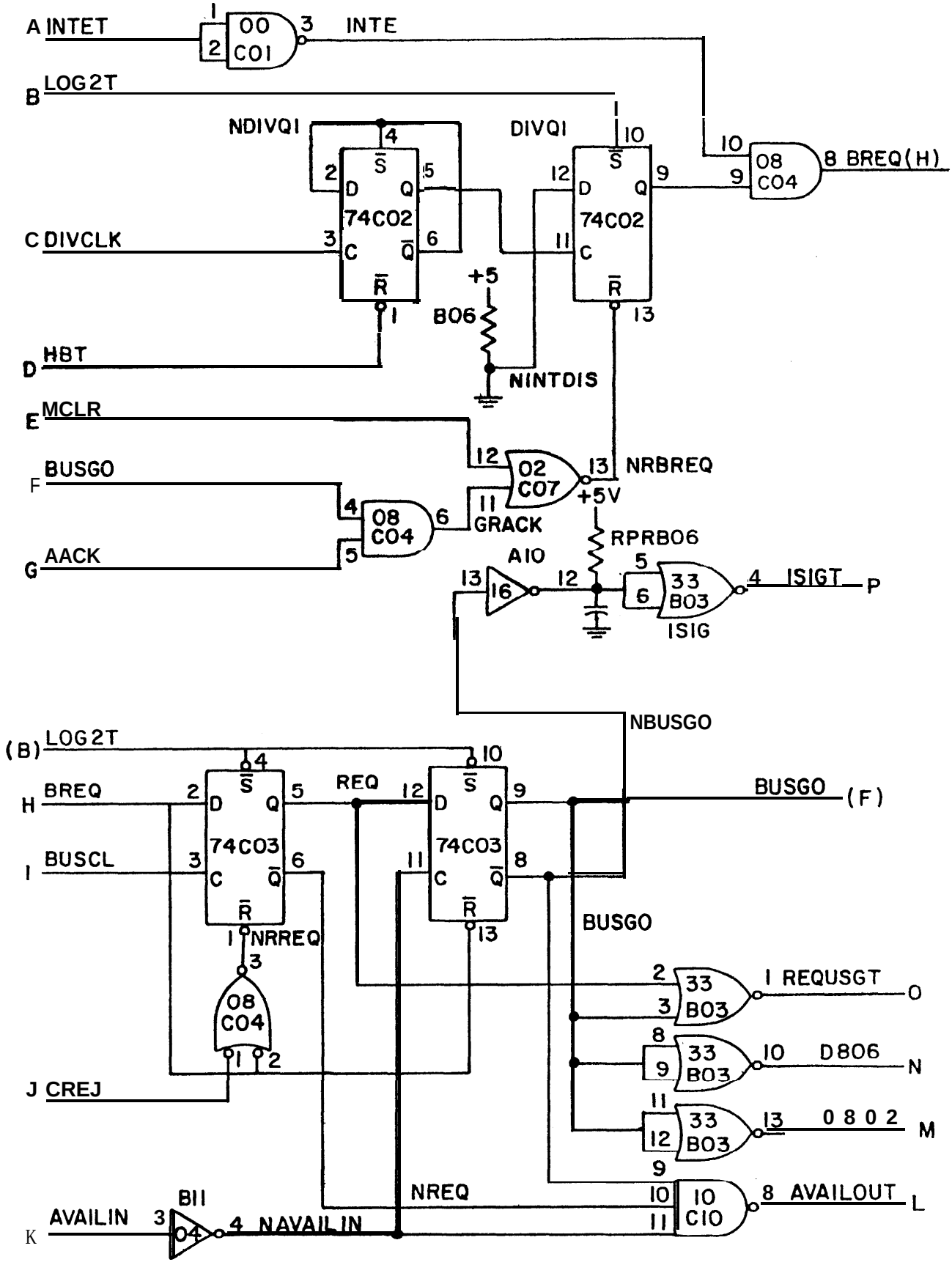

A INTET
00
C01
INTE
B LOG2T
NDIVQI
DIVQI
74C02
08
C04
8 BREQ(H)
C DIVCLK
+5
B06
NINTDIS
D HBT
E MCLR
F BUSGO
02
C07
NRBREQ
+5V
GRACK
G AACK
RPRB06
A10
33
B03
4 ISIGT
P
ISIG
NBUSGO
(B) LOG2T
H BREQ
REQ
BUSGO
(F)
74C03
I BUSCL
NRREQ
BUSGO
I REQUSGT
O
D806
N
0802
M
J CREJ
NREQ
B11
AVAILIN
K
NAVAIL IN
10
C10
8 AVAILOUT
L

Figure 0.1

1. SYSTEM OVERVIEW

The current release of the system consists of the following major programs:

1) SDLCOMP: The SDL compiler translates the circuit description provided by the user into an internal format. It makes use of a logical library. This logical library contains information for each component type used. For the purpose of printed circuit board design only the pin names for every component type are retrieved from this library.

2) SDLPCGEN: This program takes the sequential output file, generated by the compiler, and combines it with information retrieved from the physical library and the board description file. The output is a sequential file containing all the information necessary to initialize the design file.

3) LOADFILE: This program initializes the direct access design file using the output generated by SDLPCGEN.

4) PLACER: This is an interactive subsystem that allows the user to place components on the board.

5) MWIRE: Interactive subsystem for routing critical and multiple width wires manually.

6) HIWIRE: Batch router for automatic routing of non-critical connections.

7) VIAELIM: Batch program for elimination of unnecessary via holes.

The overall structure of the system is illustrated in Figure 1.1.

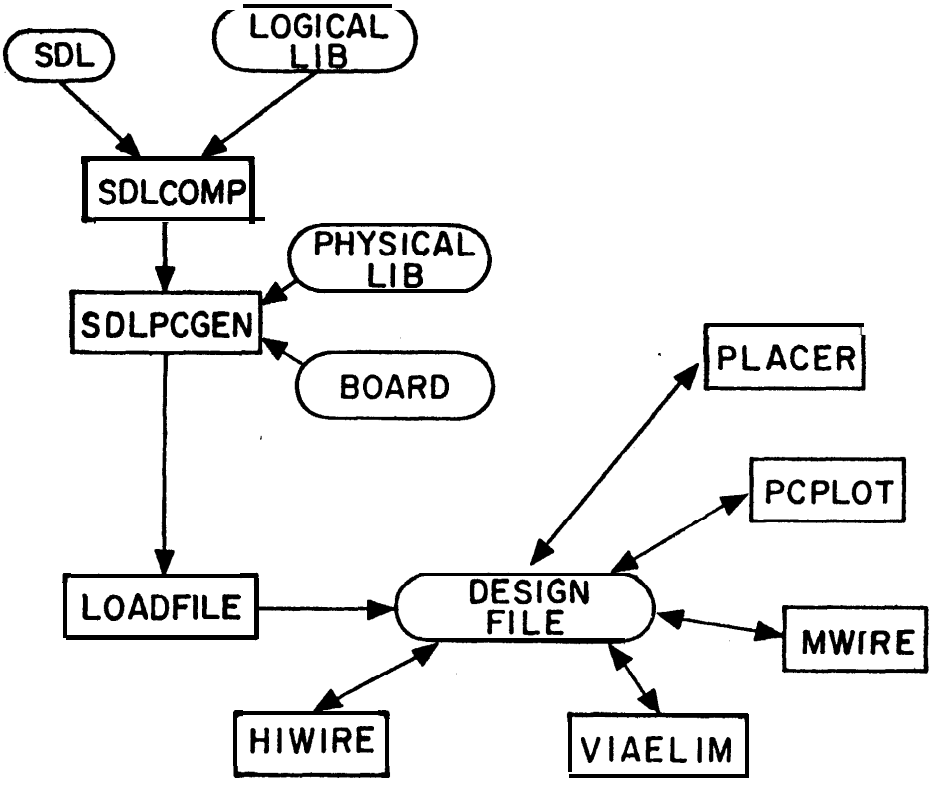


Figure 1.1

2. THE DESCRIPTION OF CIRCUITS IN SDL

2.1 Language Definition

The SDL language allows for format-free entry of a circuit. It performs checking of the input wherever possible. The SDL language removes all blanks from statements before processing. Therefore embedded blanks have no meaning, although they may be inserted in the source text in order to enhance readability.

The sequence of groups of statements has to be adhered to. The following description defines this sequence. Some statements are mandatory while others are optional. For each statement it will be indicated whether or not it is optional.

In general, names (identifiers) in the system may consist of the letters A-Z, the digits 0-9, the slash (/), the underscore (_) and angular brackets (<and >).

1) User Identification

USER: <name>;

where name is a character string of at most 20 characters.

Example: USER: TBENNETT;

This declaration is mandatory.

2) Project Identification

NAME: <projectname>;

where <projectname> is a character string of at most 20 characters.

Example: NAME: CAMACEXAMPLE;
This declaration is mandatory and should follow the USER identification.

A preferred naming convention in the system is of the form:

<Purposename>. <level name>. <Project name>

For PC layout

<Purposename> = PCBGEN

and <level name> = BOARD

The reason for this naming convention is the possibility of logic macro expansion which will be available in a future release of the system. However, if the description will be used for PC layout only you may use any name. If you do so a warning message will be issued.

3) Purpose Definition

PURPOSE: <purposename>;

where <purposename> is a string of at most 20 characters.
In general, the purpose of the description should be stated here.
For printed circuit board design this purpose is always defined as: PCBGEN.
This declaration is mandatory.

4) Level Definition

LEVEL: <levelname>;

where <levelname> is a valid identifier.

In general, the level of a description has to be specified before doing a macro expansion.
For printed circuit board design two levels are of interest:
1) COMP (the component level) which should be used to describe the board in terms of components and connections.
2) END (the lowest level) which should be used to describe component types to be entered in the logical library. The basic purpose of such a description is to associate names with each of the pins of a given component type.

5) Declaration Of All Logical Types Used

TYPES: <namelist>;

where <namelist> consists of a single logical type name or of several names separated by commas.

It is necessary to declare explicitly all logical component types used in the circuit being described.

Example: TYPES: AND, OR, 54180, 8080, SN7474, DFLIPFLOP;

The name of a type is restricted to 20 significant characters.

NOTE: The type EXT is implicitly defined and reserved for specifying the external connectors. Therefore, EXT is a reserved type name and should not be redefined.

6) Definition of External Connectors

a) EXT: <cname> : <namelist>;

This declaration is necessary for every external connector.

A <cname> is the name of that connector while <namelist> is a list of all the terminals of that connector in the order in which they appear. All terminals that are connected should have distinct names. When a terminal is unconnected and unnamed, this may be indicated by using the reserved name DUMMY.

Example: EXT: CONNECTOR1: A, B, C, DUMMY, D, E, F;

This declares a connector called CONNECTOR1 with 7

terminals with terminals 1-3 called A, B, C and 5-7 called D, E, F.

Terminal 4 is unconnected and without an actual name.

Again, the name of a connector and of terminals is limited to 20 characters without embedded blanks.

Every external connector implicitly defines a logical type and a component with the same name (i.e. a <cname>).

b)

INPUTS: <tnamelist>;

OUTPUTS: <tnamelist>;

A terminal of a connector may be defined as either an input or an output terminal or both or it may be left undefined.

The order in which these terminals appear in the INPUTS or OUTPUTS declaration is not important.

A <tnamelist> is either a single <tname> or several <tname>'s separated by commas.

A <tname> consists of a connector name followed by a period (.) followed by a terminal name. Both names are limited to 20 characters without embedded blanks.

Example:
```
INPUTS: CONNECTOR1.A,
          CONNECTQR1.B;
        CONNECTOR2.A, CONNECTOR1.E;
OUTPUTS: CONNECTOR1.D, CONNECTOR1.E;
```

In this example, terminals A, B of CONNECTOR1 were defined as inputs only, terminal D as output only, terminal E is both input and output, while terminals

C and F were left undefined.

7) Comoonent Declarations

<typename> : <cnamelist>;

where <typename> is a previously defined logical type and <cnamelist> is a list of component names.

Component name may be up to 20 characters long

Example: AND: C1, C5, TPX, 8080, P1;

Note that external connectors should not be declared as components here since this has been implied by the EXT statement(s).

8) Net Declarations

Each net is specified by the following declaration:

<netname> = <tnamelist>;

where <tname> is of the form: component name, followed by a period (.), followed by a terminal name.

e.g. :
```
NET1 = C1,IN1, CONNECTOR1.A;
NET2 = C1.IN2, CONNECTOR1.B;
NET3 = C1.OUT, C2. IN1, C3.IN2;
NET4 = C2.OUT,C3.OUT, CONNECTOR1.C;
```

The net descriptions are followed by an END; statement.

Some users may prefer to encode a signal in more than one statement. This is, e.g., the case when a logic diagram that is being encoded is spread out over several sheets.

Since allowing this removes some of the error-checking capability of the system, a warning message will be issued for multiply-defined nets.

9) Crosschecking of the Net Declarations

It is frequently desirable to input the net declarations in their dual form, i.e., for every component, one can enter the net each pin is connected to. The option here is:

NO CROSSCHECK; or

CROSSCHECK;

In the second case, the following input is expected:

<cname> : <nnamelist>;

where <nname> consists of the netname followed by a period (.) followed by the pinname.

If crosscheck is specified, a complete netname list has to be provided for every component. The order in which these statements appear is not critical. The description is terminated by an END; statement.

e.g. :
```
CROSSCHECK;
C1:  NET1.INL, NET2.IN2, NET3.OUT;
c2:  NET4.OUT, NET3.IN1;
C3:  NET3.IN2, NET4.OUT;
CONNECTOR1:  NET1.A, NET4.C, NET2.B;
```

Although the use of crosscheck will require twice the encoding work, it allows to detect otherwise difficult-to-find mistakes.

It is highly recommended to use the crosscheck option. Although it requires the logic diagram to be encoded twice, it ensures a more correct input and may eliminate some unnecessary layout runs.

10) CEND Statement

The circuit description is terminated by a statement of the form: CEND;

11) Definition of Obstructions

An obstruction is a physical component that completely occupies one or more layers of a board. The normal

components that were defined before may, under certain conditions, be obstructions. Since this is to be considered a physical property, this information will be provided in the physical description library. It will sometimes be necessary to declare physical obstructions that do not have any external connections. An example of this is a hole in the board.

It will be necessary to declare this hole as a component and to associate a logical and physical type with it. This will then require the appropriate entries in the logical and physical libraries.

12) In some cases, a designer may have special components that are not general enough to be placed in the normal component library. When this is the case, he can specify a user library as following:

USERLIB : <name>;

This statement has to follow the TYPES declaration. The library used has to be referred to in the JCL (this is taken care of by the #RUNCOMP exec file).

The name of the library is checked against the name in the leader record of the library. The generator of a library is discussed in Section 3.3.

13) In some cases, e.g., the generation of a user library, it may be desirable to compile several circuits in a single run. This is permitted in the current version of the compiler. Note that the CEND; statement determines the

end of the input deck. The USER statement should only appear once, i.e., is the first card. A typical batch may look as follows:

```
USER : WMVC;
NAME : CIRCUIT1 ;
NOCROSSCHECK;
NAME : CIRCUIT2 ;
CROSSCHECK;
END; (end of crosscheck list)
NAME : CIRCUIT3;
NOCROSSCHECK;
CEND;
```

2.2 An Example

The following simple example illustrates the use of SDL for describing a circuit. The logic diagram for the circuit in this example is given in Figure 0.1.

Note that every signal has to be given a distinct name. It is possible to describe a signal net using more than one net declaration, e.g., signals LOGIT, BUSGO, BREQ and ISIG in the example.

Note that pins in such a declaration may be specified only once. The compiler will merge all net declarations with the same name. In order to prevent accidental errors, a warning message will be issued for every duplicate net name encountered.

```
USER: WMVC;
NAME:EXA"PLE1;
PURPOSE:PCBGEN;
LEVEL:COMP;
TYPES:7400,7402,7408,7433,7410,7474,7404,7416,RES,CAP;
EXT:CONN20A:A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T;
7400:C01;
7474:C02,C03;
7408:C04;
7402:C07;
7416:A10;
7433:B03;
7410:C11;
7404:B11;
RES:R1,R2;
CAP:C1;
INTET=CONN20A.A,C01.1,C01.2;
INTE=C01.3,C04.10;
LOG2T=CONN20A.B,C02.10;
DIVCLK=CONN20A.C,C02.3;
HBT=CONN20A.D,C02.1;
MCLR=CONN20A.E,C07.12;
BUSGO=CONN20A.F,C04.4;
AACK=CONN20A.G,C04.5;
LOG2T=C03.4,C03.10;
BREQ=CONN20A.H,C03.2,C03.13;
BUSCL=CONN20A.I,C03.3;
CREJ=CONN20A.J,C04.1;
AVAILIN=CONN20A.K,B11.3;
C02Q=C02.9,C04.9;
C02QB=C02.6,C02.2,C02.4;
DIVQ1=C02.5,C02.11;
NINTDIS=C07.13,C02.13;
GRACK=C04.6,C07.11;
REQ=C03.5,C03.12,B03.2;
BUSGO=C03.9,B03.3,B03.8,B03.9,B03.11,B03.12;
NX=C03.8,C11.9,A10.13;
NREQ=C03.6,C11.10;
NAVAILIN=B11.4,C03.11,C11.11;
BREQ=C04.8;
ISIGT=B03.4,CONN20A.P;
ISIG=A10.12,R2.2,B03.5,B03.6;
ISIG=C1.1;
REQUSGT=B03.1,CONN20A.O;
D806=B03.10,CONN20A.N;
```

```
D802=B03.13,CONN20A.M;
AVAILOUT=C11.8,CONN20A.L;
PLUS5=R1.1,R2.1;
GND=C1.2,R1.2,C02.12;
END;
CROSSCHECK;
C01:INTET.1,INTET.2,INTE.3;
C02:HBT.1,NINTDIS.13,C02QB.2,DIVCLK.3,C02QB.4,DIVQ1.5,C02Q.9,
    C02QB.6,LOG2T.10,GND.12,DIVQ1.11;
CONN20A:INTET.A,LOG2T.B,DIVCLK.C,HBT.D,MCLR.E,BUSGO.F,AACK.G,
        BREQ.H,BUSCL.I,CREJ.J,AVAILIN.K,AVAILOUT.L,D802.M,
        D806.N,REQUSGT.O,ISIGT.P;
C03:BREQ.2,BUSCL.3,LOG2T.4,REQ.5,NREQ.6,NX.8,BUSGO.9,LOG2T.10,
    NAVAILIN.11,REQ.12,BREQ.13;
C04:BUSGO.4,CREJ.1,AACK.5,GRACK.6,BREQ.8,C02Q.9,INTE.10;
C07:GRACK.11,MCLR.12,NINTDIS.13;
A10:ISIG.12,NX.13;
B03:REQUSGT.1,REQ.2,BUSGO.3,ISIGT.4,ISIG.5,ISIG.6,BUSGO.8,
    BUSGO.9,D806.10,BUSGO.11,BUSGO.12,D802.13;
C11:AVAILOUT.8,NX.9,NREQ.10,NAVAILIN.11;
B11:NAVAILIN.4,AVAILIN.3;
R1:PLUS5.1,GND.2;
R2:PLUS5.1,ISIG.2;
C1:ISIG.1,GND.2;
END;
CEND;
```

2.3 How to Run SDLCOMP

The SDL compiler reads and checks an SDL source file for syntax errors and logical inconsistencies. If the compilation is successful SDL object language will be output. The logical library will contain circuit descriptions in the form of SDL object. All application programs (PC generation, logic diagram generation, IC layout etc.) will use the SDL object file of a circuit together with an application-oriented library to create input for the appropriate subsystem.

In order to simplify the use of this compiler, a Wylbur exec file (LIB#RUNCOMP under account number PCB$CG) has been created.

It can be used as follows:

```
exec from #runcomp user pcb group cg
THIS FILE RUNS THE SDL COMPILER.
<CR> CLEAR ACTIVE?
```

<CR> clears your active file otherwise use break to save it.

```
'ACCOUNT?= CG.PCB?'
ENTER? cg.wmv
```

Enter the account number that will be used as a prefix for all files e.g. cg.wmv

```
'SDL SOURCE FILE = SDLINPUT?'
ENTER? exl
```

Enter the name of the Wylbur format file containing your SDL source input. Two forms are possible:

a) for sequential files use the file name

b) for library members use the form libname (membername)

'SDL OBJECT OUTPUT FILE = SDLOBJ?'
ENTER? objl

Enter the name of the Wylbur format file into which the object language output will be written. <CR> will give you the default name SDLOBJ.

'VOLUME = SCFEV9?
ENTER?

Enter the volume on which the SDL object file is to be placed. <CR> will give you a scratch volume (i.e. SCFEV9).

'USER LIBRARY?<CR>=NONE'
ENTER? xlib

Enter the name of the user library if you have one, otherwise reply with a <CR>.

THE JCL IS NOW IN YOUR ACTIVE FILE READY TO RUN
-> EXE F PAU

Issue a run command to submit the job.

2.4 SDLCOMP Error Messages

The following list explains all error messages that are currently generated by the SDL compiler. A severity less than 4 is a warning; severity 4-7 are serious errors which need correction; severity 8 is a fatal error.

Depending on the nature of error the following actions are recommended:

1) correct the error and run the program again

2) keep your output and a copy of the input file and contact the person responsible for maintaining the compiler

3) contact the person responsible for maintaining the component library.

error number	severity	action	message
1	8	1	missing circuit name
2	8	1	missing or incorrect EXT declaration
3	8	1	missing or incorrect TYPE declaration
4'	4	1	undefined type
5	8	1	missing type in component declaration
6	4	1	maximum name length exceeded
7	4	1	expecting net definition or END but other statement found
8	8	3	missing header in logical library
9	8	3	syntax error in logical library
10	8	3	not all types in logical library
11	4	1	invalid pin name
12	4	1	previously defined type
13	4	1	undefined component
14	8	1	missing crosscheck statement
15	8	3	missing or incorrect PINS record in library
16	8	3	user library name conflict
17	4	1	undefined component
18	4	1	undefined netname

error number	severity	action	message
19	4	1 or 3	undefined pinname
21	4	1	previously defined component
22	4	1	previously defined pinname
23	8	1	missing user name
24	8	1	syntax error in EXT declaration
26	4	1	undefined component type
27	8	1	invalid netlist
28	4	1	invalid terminal name
29	4	1	syntax error
30	8	1	incomplete crosscheck list
31	8	1	missing END statement
37	8	3	missing or incorrect NTYP record in logical library
38	8	3	missing or incorrect NCOM record in logical library
39	8	3	missing or incorrect TNPN record in logical library
40	8	3	missing or incorrect NNET record in logical library
41	8	3	missing or incorrect TYPE record in logical library
42	8	3	missing or incorrect COMP record in logical library
43	4	1	error detected during processing of crosscheck information
44	8	3	invalid name in logical library
45	8	3	missing or conflicting purpose in logical library

error number	severity	action	message
46	8	3	missing or conflicting level in logical library
47	2	-	previously defined **netname**
48	8	2	one of the systems limit was exceeded
49	8	3	error in alias directory of logical library
50	4	1	pin connected to two different nets
57	2	-	unconnected pin

2.5 SDL Object Language Format

As indicated in section 2.3, the SDL compiler generates an intermediate input in the form defined below. This format is used

1) by each subsystem (printed circuit design, integrated circuit design, logic simulation etc...) together with a special library to create a specific design file (the printed circuit design file generation is discussed in detail in section 4).
2) in the logical library
3) by the macro-expansion facilities that will be available in the future.

The user need not be aware of the rest of this section. Nevertheless, it is included for the sake of completeness.

The SDL object language consists of records of length 80 with columns 1-5 reserved for a record type indicator. The following description gives these records in the order in which they will occur.

All records are fixed-format to make them easy to read from Fortran.

1) USER:

   1 record

   username: col. 6-25

2) NAME:

   1 record

   project name: col. 6-25

3) PURP:

1 record

purpose name: col. 6-25

4) LEVL:

1 record

level name: col. 6-25

5) NTYP:

1 record

# of logical types: col. 6-10

6) NCOM:

1 record

# of components: col. 6-10

7) TNPN:

1 record

total # of pins: col. 6-10

8) NXPN:

1 record

# of external pins: col. 6-10

9) NNET:

1 record

# of nets: col. 6-10

10) XPIN:

1 record per external pin

pinnumber: col. 6-10

pinname: col. 11-30

11) TYPE:

1 record per logical type

typenumber: col. 6-10

typename: col. 11-30

# of pins: col. 31-35

external flag: col. 36-40

12) COMP:

1 record per component

component #: col. 6-10

componentname: col. 11-30

# of pins: col. 31-35

logical type: col. 36-40

external flag: col. 41-45

13) PINS:

directly follows COMP: record

1 record for every pin of the component

pin #: col. 6-10

pinname: col. 11-30

14) HNET:

1 record for every net

net #: col. 6-10

netname: col. 11-30

# of pins: col. 31-35

15) PNET:

directly follows HNET: record

1 record for every pin in the net

component #: col. 6-10

pin #: col. 11-15

16) ENDC:

terminates the SDL object file

```
USER: WMVC;
NAME:EXAMPLEl;
PURPOSE:PCBGEN;
LEVEL:COMP;
TYPES:7400,7402,7408,7433,7410,7474,7404,7416,RES,CAP;
EXT:CONN20A:A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T;
7400:C0l;

 LIBRARY NAME = MYLIB
*** END OF LIBRARY FETCH ***

7474:C02,C03;
7408:C04;
7402:C07;
7416:A10;
7433:B03;
7410:Cll;
7404:Bll;
RES:Rl,R2;
CAP:Cl;
INTET=CONN20A.A,C01.1,C01.2;
INTE=C01.3,C04.10;
LOG2T=CONN20A.B,C02.10;
DIVCLK=CONN20A.C,C02.3;
HBT=CONN20A.D,C02.1;
MCLR=CONN20A.E,C07.12;
BUSGO=CONN20A.F,C04.4;
AACK=CONN20A.G,C04.5;
LOG2T=C03.4,C03.10;
*** SDLCOMP * WARNING 47 *** PREVIOUSLY DEFINED NETNAME LOG2T
BREQ=CONN20A.H,C03.2,C03.13;
BUSCL=CONN20A.I,C03.3;
CREJ=CONN20A.J,C04.1;
AVAILIN=CONN20A.K,B11.3;
C02Q=C02.9,C04.9;
C02QB=C02.6,C02.2,C02.4;
DIVQ1=C02.5,C02.11;
```

```
NINTDIS=C07.13,C02.13;
GRACK=C04.6,C07.11;
REQ=C03.5,C03.12,B03.2;
BUSGO=C03.9,B03.3,B03.8,B03.9,B03.11,B03.12;
*** SDLCOMP * WARNING 47 *** PREVIOUSLY DEFINED NETNAME BUSGO
NX=C03.8,C11.9,A10.13;
NREQ=C03.6,C11.10;
NAVAILIN=B11.4,C03.11,C11.11;
BREQ=C04.8;
*** SDLCOMP * WARNING 47 *** PREVIOUSLY DEFINED NETNAME BREQ
ISIGT=B03.4,CONN20A.P;
ISIG=A10.12,R2.2,B03.5,B03.6;
ISIG=C1.1;
*** SDLCOMP * WARNING 47 *** PREVIOUSLY DEFINED NETNAME ISIG
REQUSGT=B03.1,CONN20A.O;
D806=B03.10,CONN20A.N;
D802=B03.13,CONN20A.M;
AVAILOUT=C11.8,CONN20A.L;
PLUS5=R1.1,R2.1;
GND=C1.2,R1.2,C02.12;
END;
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: CONN20A.Q
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: CONN20A.R
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: CONN20A.S
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: CONN20A.T
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C01.4
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C01.5
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C01.6
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C01.7
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C01.8
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C01.9
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C01.10
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C01.11
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C01.12
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C01.13
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C01.14
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C02.7
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C02.8
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C02.14
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C03.1
```

```
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C03.7
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C03.14
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C04.2
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C04.3
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C04.7
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C04.11
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C04.12
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C04.13
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C04.14
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C07.1
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C07.2
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C07.3
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C07.4
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C07.5
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C07.6
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C07.7
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C07.8
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C07.9
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C07.10
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C07.14
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  A10.1
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  A10.2
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  A10.3
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  A10.4
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  A10.5
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  A10.6
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  A10.7
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  A10.8
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  A10.9
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  A10.10
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  A10.11
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  A10.14
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  B03.7
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  B03.14
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C11.1
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C11.2
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  c11.3
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  c11.4
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  c11.5
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN:  C11.6
```

```
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C11.7
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C11.12
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C11.13
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: C11.14
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: B1l.l
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: B11.2
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: B11.5
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: B11.6
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: B11.7
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: Bll.8
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: Bll.9
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: B1l.10
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: Bll.11
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: B11.12
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: B11.13
*** SDLCOMP * WARNING 57 *** UNCONNECTED PIN: B11.14
CROSSCHECK;
C01:INTET.1,INTET.2,INTE.3;
C02:HBT.1,NINTDIS.13,C02QB.2,DIVCLK.3,C02QB.4,DIVQ1.5,C02Q.9,
    C02QB.6,LOG2T.10,GND.12,DIVQ1.11;
CONN20A:INTET.A,LOG2T.B,DIVCLK.C,HBT.D,MCLR.E,BUSGO.F,AACK.G,
        BREQ.H,BUSCL.I,CREJ.J,AVAILIN.K,AVAILOUT.L,D802.M,
        D806.N,REQUSGT.O,ISIGT.P;
C03:BREQ.2,BUSCL.3,LOG2T.4,REQ.5,NREQ.6,NX.8,BUSGO.9,LOG2T.10,
    NAVAILIN.11,REQ.12,BREQ.13;
C04:BUSGO.4,CREJ.1,AACK.5,GRACK.6,BREQ.8,C02Q.9,INTE.10;
C07:GRACK.11,MCLR.12,NINTDIS.13;
A10:ISIG.12,NX.13;
B03:REQUSGT.1,REQ.2,BUSGO.3,ISIGT.4,ISIG.5,ISIG.6,BUSGO.8,
    BUSGO.9,D806.10,BUSGO.11,BUSGO.12,D802.13;
C11:AVAILOUT.8,NX.9,NREQ.10,NAVAILIN.11;
B11:NAVAILIN.4,AVAILIN.3;
R1:PLUS5.1,GND.2;
R2:PLUS5.1,ISIG.2;
C1:ISIG.1,GND.2;
END;

# OF TYPES = 12
# OF COMPONENTS = 13
# OF NETS = 29
```

OF PINS = 152
OF EXTERNAL PINS = 20

NTYPES = 12

THE COMPONENTS TYPES USED ARE:

1	7400	14	
2	7402	14	
3	7408	14	
4	7433	14	
5	7410	14	
6	7474	14	
7	7404	14	
8	7416	14	
9	RES	2	
10	CAP	2	
11	CONN20A	20	1
12	DIP14	14	

NCOMPS = 13

TOTAL # OF PINS = 152

THE COMPONENTS ARE:

1	CONN20A	11	1
2	C01	1	
3	c02	6	
4	C03	6	
5	C04	3	
6	C07	2	
7	A10	8	
8	B03	4	
9	C11	5	
10	B11	7	
11	R1	9	
12	R2	9	
13	C1	10	

NNETS = 29

THE NETS ARE:

1	INTET	3	1.1 2.1 2.2
2	INTE	2	2.3 5.10
3	LOG2T	4	1.2 3.10 4.4 4.10
4	DIVCLK	2	1.3 3.3
5	HBT	2	1.4 3.1
6	MCLR	2	1.5 6.12
7	BUSGO	8	1.6 5.4 4.9 8.3 8.8 8.9 8.11 8.12
8	AACK	2	1.7 5.5
9	BREQ	4	1.8 4.2 4.13 5.8
10	BUSCL	2	1.9 4.3
11	CREJ	2	1.10 5.1
12	AVAILIN	2	1.11 10.3
13	CØ2Q	2	3.9 5.9
14	CØ2QB	3	3.6 3.2 3.4
15	DIVQ1	2	3.5 3.11
16	NINTDIS	2	6.13 3.13
17	GRACK	2	5.6 6.11
18	REQ	3	4.5 4.12 8.2
19	NX	3	4.8 9.9 7.13
20	NREQ	2	4.6 9.10
21	NAVAILIN	3	10.4 4.11 9.11
22	ISIGT	2	8.4 1.16
23	ISIG	5	7.12 12.2 8.5 8.6 13.1
24	REQUSGT	2	8.1 1.15
25	D806	2	8.10 1.14
26	D802	2	8.13 1.13
27	AVAILOUT	2	9.8 1.12
28	PLUS5	2	11.1 12.1
29	GND	3	13.2 11.2 3.12

UNCONNECTED PINS:
1.17 1.18 1.19 1.20 2.4 2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13 2.14 3.7
3.8 3.14 4.1 4.7 4.14 5.2 5.3 5.7 5.11 5.12 5.13 5.14 6.1 6.2 6.3 6.4
6.5 6.6 6.7 6.8 6.9 6.10 6.14 7.1 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9 7.10
7.11 7.14 8.7 8.14 9.1 9.2 9.3 9.4 9.5 9.6 9.7 9.1 2 9.13 9.14 10.1 10.2
10.5 10.6 10.7 10.8 10.9 10.10 10.11 10.12 10.13 10.14 2.4 2.5 2.6 2.7
2.8 2.9 2.10 2.11 2.12 2.13 2.14 3.7 3. 8 3.14 1.17 1.18 1.19 1.20 4.1
4.7 4.14 5.2 5.3 5.7 5.11 5.12 5.13 5.14 6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8
6.9 6.10 6.14 7.1 7.2 7.3 7. 4 7.5 7.6 7.7 7.8 7.9 7.10 7.11 7.14 8.7
8.14 9.1 9.2 9.3 9.4 9.5 9.6 9.7 9.12 9.13 9.14 10.1 10.2 10.5 10.6 10.7
10.8 10.9 10.10 10 .11 10.12 10.13 10.14

MODULE-NET CROSS REFERENCE LIST

COMPONENT # 1 NAME = CONN20A TYPE = 11 TAG = 1
1 1 INTET
2 3 LOG2T
3 4 DIVCLK
4 5 HBT
5 6 MCLR
6 7 BUSGO
7 8 AACK
8 9 BREQ
9 10 BUSCL
10 11 CREJ
11 12 AVAILIN
12 27 AVAILOUT
13 26 D802
14 25 D806
15 24 REQUSGT
16 22 ISIGT

COMPONENT # 2 NAME = C01 TYPE = 1 TAG =
1 1 INTET
2 1 INTET
3 2 INTE

COMPONENT # 3 NAME = C02 TYPE = 6 TAG =
1 5 HBT
2 14 C02QB
3 4 DIVCLK
4 14 C02QB

```
5 15 DIVQ1
6 14 C02QB
9 13 C02Q
10 3 LOG2T
11 15 DIVQ1
12 29 GND
13 16 NINTDIS

COMPONENT #      4 NAME = C03 TYPE = 6 TAG =
2 9 BREQ
3 10 BUSCL
4 3 LOG2T
5 18 REQ
6 20 NREQ
8 19 NX
9 7 BUSGO
10 3 LOG2T
11 21 NAVAILIN
12 18 REQ
13 9 BREQ

COMPONENT #      5 NAME = C04 TYPE = 3 TAG =
1 11 CREJ
4 7 BUSGO
5 8 AACK
6 17 GRACK
8 9 BREQ
9 13 C02Q
10 2 INTE

COMPONENT #      6 NAME = C07 TYPE = 2 TAG =
11 17 GRACK
12 6 MCLR
13 16 NINTDIS

COMPONENT #      7 NAME = A10 TYPE = 8 TAG =
12 23 ISIG
13 19 NX
```

```
COMPONENT #      8 NAME = B03 TYPE = 4 TAG =
1  24 REQUSGT
2  18 REQ
3  7 BUSGO
4  22 ISIGT
5  23 ISIG
6  23 ISIG
8  7 BUSGO
9  7 BUSGO
10  25 D806
11  7 BUSGO
12  7 BUSGO
13  26 D802

COMPONENT #      9 NAME = C11 TYPE = 5 TAG =
8  27 AVAILOUT
9  19 NX
10  20 NREQ
11  21 NAVAILIN

COMPONENT #     10 NAME = B11 TYPE = 7 TAG =
3  12 AVAILIN
4  21 NAVAILIN

COMPONENT #     11 NAME = R1 TYPE = 9 TAG =
1  28 PLUS5
2  29 GND

COMPONENT #     12 NAME = R2 TYPE = 9 TAG =
1  28 PLUS5
2  23 ISIG

COMPONENT #     13 NAME = C1 TYPE = 10 TAG =
1  23 ISIG
2  29 GND
```

```
*** END OF CIRCUIT *** EXAMPLE1

CEND;
*** NORMAL TERMINATION OF SDLCOMP ***
```

3. LOGICAL AND PHYSICAL LIBRARIES

3.1 Logical Library

3.1.1 Logical Library

The library name definition record is specified as follows:

LLIB:<library name>;

3.1.2 Alias Directory

The alias directory consists of a number of alias definitions followed by: END;

An alias definition has the form

<typename1> = <typename2>;

where <typename1> is declared to have the same attributes as <typename2>. This requires the second type to be fully defined in the body of the library.

3.1.3 Library Body

This part consists of a number of SDL object format descriptions of various components. The format is identical to that given in section 2.5 except for the lack of a USER record.

3.1.4 End of Library

The end of the library is signalled by a record containing 'LEND' in col. 1-4.

```
LLIB:                    MYLIB
74LS00=DIP14;
74LS01=DIP14;
74LS02=DIP14;
74LS83=DIP16;
END;
NAME:     PCBGEN.COMP.DIP14
PURP:                 PCBGEN
LEVL:                    END
NTYP:      1
NCOM:      1
TNPN:     14
NXPN:     14
NNET:      0
XPIN:      1                      1
XPIN:      2                      2
XPIN:      3                      3
XPIN:      4                      4
XPIN:      5                      5
XPIN:      6                      6
XPIN:      7                      7
XPIN:      8                      8
XPIN:      9                      9
XPIN:     10                     10
XPIN:     11                     11
XPIN:     12                     12
XPIN:     13                     13
XPIN:     14                     14
TYPE:      1                           14
COMP:      1                            1 EXT
PINS:      1                      1
PINS:      2                      2
PINS:      3                      3
PINS:      4                      4
PINS:      5                      5
PINS:      6                      6
```

```
PINS:      7                       7
PINS:      8                       8
PINS:      9                       9
PINS:     10                      10
PINS:     11                      11
PINS:     12                      12
PINS:     13                      13
PINS:     14                      14
ENDC
NAME:     PCBGEN.COMP.DIP16
PURP:                    PCBGEN
LEVL:                       END
NTYP:      1
NCOM:      1
TNPN:     16
NXPN:     16
NNET:      0
XPIN:      1                       1
XPIN:      2                       2
XPIN:      3                       3
XPIN:      4                       4
XPIN:      5                       5
XPIN:      6                       6
XPIN:      7                       7
XPIN:      8                       8
XPIN:      9                       9
XPIN:     10                      10
XPIN:     11                      11
XPIN:     12                      12
XPIN:     13                      13
XPIN:     14                      14
XPIN:     15                      15
XPIN:     16                      16
TYPE:      1                             16
COMP:      1                              1 EXT
PINS:      1                       1
PINS:      2                       2
```

```
PINS:     3                3
PINS:     4                4
PINS:     5                5
PINS:     6                6
PINS:     7                7
PINS:     8                8
PINS:     9                9
PINS:    10               10
PINS:    11               11
PINS:    12               12
PINS:    13               13
PINS:    14               14
PINS:    15               15
PINS:    16               16
ENDC
LEND
```

3.2 Physical Library

The physical component library is stored as a sequential file with the following format:

3.2.1 Header Record:

PHYSICAL LIB: <name>;

3.2.2 Logical Type/Physical Type Associations

a) for every association there will be a statement of the form:

<logicaltypename> = <physicaltypename>;

b) END;

3.2.3 Physical Type Description:

a) NAME: <physicaltypename>;

b) NPINS: <number>;

Defines the number of pins for this type.

c) OFFSET: <offsetlist>;

Defines for every pin its offset w.r.t pin 1, the reference pin.

An <offsetlist> consists of a single <offset> or of several <offset>'s, separated by comma's. An <offset> is specified as follows: xoffset.yoffset. Both x and y offset are integer numbers expressing the offset in mils.

d) OBSTRUCTION: <n>;

Where <n> = NONE if no obstruction.

= COMPONENT if obstruction on component side.

= SOLDER if obstruction on solder side.

= BOTH if obstruction on both sides.

e) SIZE: <xsize>.<ysize>;

This indicates the physical size of the component in mils.

f) END;

3.2.4 End of Library Indicator

PLEND;

3.2.5 Example

```
PHYSICAL LIB:EXAMPLE;
DIP14=DIP14;
ICBOX=ICBOX8;
NAME:ICBOX8;
NPINS:8;
OFFSET:100.0,200.0,300.0,400.0,500.0,600.0,700.0,800.0;
OBSTRUCTION:BOTH;
SIZE:900.1000;
END;
NAME:DIP16;
NPINS:16;
OFFSET:0.0,100.0,200.0,300.0,400.0,500.0,600.0,700.0,
  700.300,600.300,500.300,400.300,300.300,200.300,100.300,0.300;
OBSTRUCTION:NONE;
SIZE:700.300;
END;
NAME:DIP14;
NPINS:14;
OFFSET:0.0,100.0,200.0,300.0,400.0,500.0,600.0,
  600.300,500.300,400.300,300.300,200.300,100.300,0.300;
OBSTRUCTION:NONE;
SIZE:600.300;
END;
PLEND;
```

3.3 How to Add a Component to a Logical Library

Currently no programs are available for updating the logical library. However, since its format is rather simple, one can use Wylbur to perform this task. Two major types of addition are possible:

1) If you want to add a component for which there is a similar description already in the library, just insert a line in the Alias Directory. (Try to keep this directory in alphabetical order.)

2) If no similar description exists, then encode the circuit in SDL and compile it. Remove from the object file the USER and CEND records and merge this file into the body of the library.

3.4 Board Description

The board description has to be stored in a file to be used by the SDLPCGEN program. It consists of the following information:

1) Board Name

NAME:<id>;

2) Physical Outline

PHYSICAL:<point list>;

where every point is specified as **<x>.<y>.** All dimensions are expressed in mils. All coordinates are relative to the lower **lefthand** corner of the physical board outline (see figure 3.4.1).

e.g. **PHYSICAL:0.0,0.10000,6000.10000,6000,0;**

3) Logical Size and Offset

The logical board is the space allowed for placing components and for routing connections. The logical board is always a rectangle and is specified by a size (LOGSIZE) and by the location of its lower **lefthand** corner (LOGOFFSET). This is illustrated in figure 3.4.1.

LOGSIZE:<xsize>.<ysize>;

LOGOFFSET:<xoffset>.<yoffset>;

4) External Connectors

NCONNECTORS:<n>;

for every connector:

NAME:<id>;

NPINS:<n>;

OFFSET:<point list>;

OR1 ENTATION:**<n>;**

LOCATION:<x>.<y>;

END;

Example

```
NAME:CAMAC1;
PHYSICAL:0.0,11990.0,11990.4390,10890.4390,10890.5340,
       11990.5340,11990.7230,0.7230,0.0;
LOGSIZE:10400.6300;
LOGOFFSET:500.500;
NCONNECTORS:1;
NAME:CONN1;
NPINS:42;
OFFSET:0.000,0.100,0.200,0.300,0.400,0.500,
       0.600,0.700,0.800,0.900,0.1000,0.1100,
       0.1200,0.1300,0.1400,0.1500,0.1600,0.1700,
       0.1800,0.1900,0.2000,0.2100,0.2200,0.2300,
       0.2400,0.2500,0.2600,0.2700,0.2800,0.2900,
       0.3000,0.3100,0.3200,0.3300,0.3400,0.3500,
        0.3600,0.3700,0.3800,0.3900,0.4000,0.4100;
ORIENTATION:0;
LOCATION:10900.600;
END;
```

4. DESIGN FILE GENERATION

4.1 Introduction

The SDLPCGEN program combines the output of SDLCOMP (the SDL compiler, as described in section 2.5), the physical library (as described in section 3.2) and the board description (as described in section 3.4) and generates a file which contains all the information necessary to start the PC layout process.

The output of the program is a new sequential file which consists of the original SDL object description plus the following information appended to it.

1) PHT:

1 record

# of physical types: col. 6-10

2) PNTP:

1 record for every physical type

type #: col. 6-10

type name: col. 11-30

# of pins: col. 31-35

obstruction type: col. 36-40

3) PSIZ:

1 record for every physical type

xsize in mils: col. 6-10

ysize in mils: col. 11-15

4) POFF:

1 record for every pin of this type

Xoffset in mils: col. 6-10

Yoffset in mils: col 11-15

5) GPH:

1 record for every logical type

logical type #: col. 6-10

corresponding physical type #: col. 11-15

6) ENDP:

1 record ending physical information

7) NAM:

1 record

board type name: col. 6-25

8) BPAR:

1 record

# of points in physical outline: col. 6-10

# of connectors: col. 11-15

logical Xsize in mils: col. 16-20

logical Ysize in mils: col. 21-25

logical Xoffset in mils: col 26-30

logical Yoffset in mils: col. 31-35

9) BPHO:

one record for every point of the physical outline

X in mils: col. 6-10

Y in mils: col. 11-15

10) CONN:

one record for every connector

corresponding physical type #: col. 6-10

orientation: col. 11-15

X location: col. 16-20

Y location: col. 21-25

The sequential file is then transformed by the LOADFILE program into a direct access design file which will then be used by the rest of the system.

4.2 How to Run SDLPCGEN

An exec file is available for creating the JCL for running the program. It can be used as follows:

```
exec from #runpcgen user pcb gro cg cle
MIS FILE FUNS THE SDL PC GENERATION PROGRAM
<CR> CLEAR ACTIVE?
```

A <CR> will clear your active file. Use a break to save if necessary.

```
'ACCOUNT = CG. PCB? '
```

```
ENTER? cg.wmv
```

Enter your account number in the format shown. This number will be used as a prefix for your files.

```
SDL OBJECT FILE = SDLOBJ?'
ENTER? obj 1
```

Enter the name of the SDL object file you desire.

A <CR> wi 11 give you the default name SDLOBJ

```
'SDL BOARD DESCRIPTION FILE = BOARD?'
ENTER? board1 ib(camac)
```

Enter the name of the file containing the board description

```
'OUTPUT FILE = PCOUT1?'
ENTER?
```

Enter the name of the output file. A <CR> will give you the default name PCOUTJ . The system will overwrite a file with this name or it will create a new file if none exists.

```
'VOLUME = SCFEV9?'
ENTER?
```

Enter the volume on which the output file should be generated.

<CR> will give you a scratch volume (i.e. SCFEV9).

```
THE JCL IS NOW IN YOUR ACTIVE FILE READY TO RUN
→EXE F PAU
```

Issue a run command to submit the job.

```
**************************************************************
*                                                            *
*          PCB INPUT FILE GENERATION PROGRAM.                *
*           VERSION 1 LEVEL 1 OF MARCH 30 1977               *
*           CONTACT W.M. VANCLEEMPUT (415) 497 1270          *
*                                                            *
**************************************************************
*
*
```

```
COMPILATION TIME     0.167 SECONDS
MEMORY USAGE (DECIMAL BYTES)
CODE:                 9976
STRINGS:              3056
VARIABLES:            7296
CONSTANTS:            1560
TOTAL:               21880
AVAILABLE:          381715
SUCCESSFUL  COMPILATION


 USER NAME = WMVC

PROJECT NAME = EXAMPLE1
*** SDLPCGEN * ERROR 24 *** INVALID NAME IN INPUT

PHYSICAL LIBRARY = EXAMPLE
*** SDLPCGEN *** NORMAL TERMINATION
```

```
1.000
2.000
3.000
3.100
3.200
4.000
5.000
6.000
7.000
```

4.3 How to Run LOADFILE

Once again an exec file is available for helping you to run this job. It can be used as follows:

```
exec from #runload user pcb gro cg cle
THIS FILE RUNS THE SPRINT DESIGN FILE INITIALIZATION
<CR> CLEAR ACTIVE?
```

A <CR> will clear your active file. Use a break to save it if necessary.

```
'ACCOUNT = CG.PCB?'
ENTER? cg.wmv
```

Enter your account number in the format shown. This number will be used as a prefix for your files.

```
'INPUT FILE = PCOUT1?'
ENTER?
```

Enter the name of the input file i.e., the output from the SDLPCGEN program. A <CR> will give you the default name PCOUT1.

```
'DESIGN FILE = DFILE?'
ENTER? mydfile
```

Enter the name you wish to give to the design file. A <CR> will give you the default name DFILE.

```
'VOLUME = SCFEV9?'
ENTER?
```

Enter the volume the design file is to be created on.

```
THE JCL IS NOW IN YOUR ACTIVE FILE READY TO RUN
→ EXE F PAU
```

Issue a run command to submit the job.

4.4 Error Messages

4.4.1. SDLPCGEN Messages

When the program terminates normally a message

***SDLPCGEN ***NORMAL TERMINATION

will be printed. This message is not to be confused with the message: NORMAL TERMINATION IN STATEMENT, which is generated by the SNOBOL compiler and which may occur even if the SDLPCGEN ends abnormally due to erroneous input data.

Some messages may be generated because of an erroneous input caused by a compiler run that did not terminate properly. Therefore, it is important to check successful completion of the SDLCOMP program before running SDLPCGEN.

error number	severity	action	Explanation
1	8	5	Missing circuit name specification
2	8	3	All physical types required not found in library
3	8	1	Missing or invalid board name
4	8	1	Missing or invalid board outline
5	8	1	Missing or invalid logical board size
6	8	1	Missing or invalid logical board offset
7	8	1	Missing or invalid number of connectors
8	8	1	Missing or invalid number of pins
9	8	1	Missing or invalid offset
10	8	1	Missing or invalid connector name
11	8	1	Missing or invalid orientation
12	8	1	Missing or invalid location
13	8	1	Missing end in board description
15	8	5	Missing or incorrect XPIN record
16	8	5	Missing ENDC record in lib
17	8	5	Missing user name
18	8	3	System limit exceeded
19	8	5	Missing or incorrect NTYP record
20	8	5	Missing or incorrect NCOM record
21	8	5	Missing or incorrect TNPN record
22	8	5	Missing or incorrect NNET record
23	8	5	Missing or incorrect TYPE record
24	4	4	Invalid name in input
25	4	4	Missing or conflicting purpose

error number	severity	action	Explanation
26	8	5	Missing or conflicting level
27	8	2	Missing physical library name
28	8	2	Missing NPINS in physical library
29	8	2	Missing offset in physical library
30	8	2	Missing obstruction in physical library
31	8	2	Missing size in physical library
32	8	2	Missing end in physical library
33	8	1	Invalid connector location
34	8	5	Missing or incorrect NXPN record
35	8	1	Incorrect connector in board file
36	4	1	Pins of connector not in straight line

Possible Actions

1. Correct the board description file
2. Contact the person responsible for maintaining the physical component library
3. System error; contact the person responsible for maintaining the system
4. Warning only
5. Check the compiler run; if it was correct take action 3

4.4.2. LOADFILE Messages

All error messages generated by the LOADFILE Program are fatal in nature. Most of these errors indicate a problem with the input file. Check the listing of SDLPCGEN that created this file and correct any errors. If SDLPCGEN finished with the message *** SDLPCGEN*** NORMAL TERMINATION, then consult the person responsible for maintaining the system.

If LOADFILE is successful, it will print out a message: DESIGN FILE GENERATION SUCCESSFUL and some statistics about the usage of records in the file.

A common error occuring when trying to run LOADFILE is that the new design file to be created already exists, especially when you use the default name DFILE. If this is the case, the program will abend with a JCL error. This is done in order to prevent accidental overwriting of a valuable existing design file.

5. INTERACTIVE PLACEMENT PROGRAM (PLACER)

5.1 THE PLACEMENT PROCESS

In order to allow maximal flexibility in placing components, this subsystem is highly interactive.

Several algorithms for optimizing the placement can be invoked by the designer. Depending on the nature of the design and on the inclination of the designer, placement may be done completely manually or completely automatically.

Usually the placement process consists of a combination of both manual placement and algorithmic placement improvement. The SPRINT system reflects this philosophy in the following four major steps:

1) placement of critical components. This step is manual and allows the designer to carefully place components for which he considers placement to be critical. The so-called "critical" components will not be moved by the automatic placement algorithms although the designer can do so at all times. The remaining components will be classified as "automatic" or "discrete" components. Discrete components, in the context of this system, are resistors and capacitors for which automatic placement is difficult. An example of this are decoupling capacitors between power and ground in TTL logic or termination resistors in ECL.

2) initial placement of the "automatic" components. In this phase the designer has the choice between three different initial

placement methods:

- manual initial placement: Useful if the designer has a good feeling of how the placement should look.
- constructive algorithm: This algorithm achieves a suboptimal initial placement.
- Random initial placement: Random initial placement randomly places the components. This can be useful for evaluating several runs of the placement improvement algorithms for different starting points.

3) placement improvement: Two different algorithms are available: a pairwise interchange algorithm and Steinberg's placement improvement algorithm. The designer can choose between these two for every iteration. He can also make manual changes if he deems this necessary.

4) placement of "discrete" components: In this last phase, the designer manually places all remaining components.

5.2 Using PLACER

5.2.1. Running the Placement Program

A) EXEC FROM #RUNPLCR USER PCB GROUP CG CLR

an explanation of the prompts:

1) Design file=DFILE?

a) <cr>:use the default file name

b) enter the name of the design file you wish to do the placement on

2) Volume=SCFEV9?

a) <cr>:use the default volume

b) enter the name of the volume which the design file is on

3) The job is now in the active file, type "exec" to run it

B) Signing on the placement program

1) If the user receives the message:

FROM OPERATOR: (uuuPLCRn) PLACER IS READY.

(where uuu=user id, and n is a single digit integer)

Type "uuuPLCRn" as a WYLBUR command when you want to do the placement. Then the prompt: WHAT'S THE MAGIC WORD? Respond: "punt". This will attach your 4013 to the placement program.

2) If the user receives the message:

FROM OPERATOR: (uuuPLCRn) DESIGN FILE HAS NOT BEEN INITIALIZED.

then the placement program has found an invalid design file (it has nothing in it). The user should check into how the design file was created.

5.2.2. The Different Stages

A) File identification and cross-checking

to the prompts:

USER NAME?

PROJECT NAME?

the user should identify the design he thinks he is going to work on. If the user doesn't give the correct names he will not be allowed to modify the current design. To stop the program if you have the wrong design, type "$wyl", this will let you out of the program.

B) Restarting previous placements

If the design has been wired you will receive the prompt:

DELETE ROUTING?

<cr> means no. If you respond "no" then the prompt:

LEAVE PLACER?

will appear. Type "yes" to terminate placer. <cr> will give the prompt:

DELETE ROUTING? again.

If the design has already been placed you will receive the prompt:

USE OLD PLACEMENT?

to which you may respond <cr> which means YES, or you

may respond "no" which will not allow you to use the previous placement. If you restart the placement you will not be asked for the gridsize or cellular definition. Instead the old placement will be reconstructed and the following message should appear:

SUCCESSFUL RESTORE

C) Grid size definition

Place the cross-hairs over the appropriate grid size and hit the space bar. This will give the desired gridsize.

D) Cellular definition

1) To the prompt:

NUMBER OF AUTOMATIC TYPES?

give the number of physically different types you wish to be defined as automatic. An automatic component (one whose physical type is defined as automatic) can be used specially in various parts of the program.

2) Defining automatic types

You will be given the prompt:

TYPE NO. n?

for each automatic type to be defined. Type the name of the physical types you wish to be automatic. By typing "?" all types available will be listed. Type "go back" to restart the cell definition stage.

3) Defining the cell dimensions

You will be prompted for the number cells per row and column.

4) Alignment

For automatic components a global direction (or orientation) is defined. To the prompt:

ALIGN VERTICALLY?

the user can respond <cr> which means YES or "no" which means align horizontally.

5) If the largest automatic type will not fit inside the defined cells than an error message will be issued and the user will have to respecify everything in the cellular definition stage.

E) Placement phase 1 (critical placement)

menu choice explanations

1) PLACE CRITICAL COMPONENTS

Critical placement allows the user to place any component exactly as desired.

To the prompt:

COMPONENT NAME?

you may respond

a) The name of the component you wish to place. After the carriage return, indicate the reference point (usually pin 1) with the cross-hairs, hit the space bar. The component has been placed.

b) RULER -- the ruler will be displayed

c) ? -- list the names of all the unplaced components. Components listed as xxx* must be given a board connector association.

d) ?xxx -- where xxx is a searching field. All components whose names start with xxx will be listed.

e) LABEL -- label the components

f) *xxx -- place logical connector xxx using a board connector definition. This gives the prompt:

CONNECTOR ASSOCIATION?

you may respond:

i) ?-- list names of board connectors

ii) the name of the associated board connector.

g) MENU -- allows the user to leave CRITICAL PLACEMENT and select something from the menu.

2) RESTART CRITICAL PLACEMENT

Gives the prompt:

DELETE ALL CRITICAL COMPS?

<cr> means don't delete anything, "yes" means remove all components placed either in CRITICAL or DISCRETE placement. (Does not affect automatic components.)

3) RESTART PLACEMENT

Gives the prompt:

DELETE ALL COMPONENTS?

a <cr> means don't delete anything, "yes" means delete all components

4) GO BACK TO CELLULAR DEFINITION

This allows the user to return to cellular definition and redefine those parameters. Components will remain placed, but all automatic components will become critical.

5) END OF CRITICAL PLACEMENT

Allows the user to proceed to placement phase 2.

6) See general utilities for other menu options.

F) Placement phase 2 (initial placement)

After all automatic components have been placed the placement program will go on to placement phase 3.

menu choice explanations:

1) CONSTRUCTIVE ALGORITHM -- not implemented

2) RANDOM INITIAL PLACEMENT

Place all unplaced automatic components into random cells.

3) MANUAL INITIAL PLACEMENT

Manual initial placement allows the user to place the automatic components into specific cells. Menu options:

a) The user will be prompted with an automatic component to be placed. Indicate the location by putting the cross-hairs in a cell and hit the space bar.

b) EXIT MANUAL INITIAL PLACEMENT

Returns the user to placement phase 2

c) See general utilities for other menu options.

4) RESTART INITIAL PLACEMENT

To the prompt:

DELETE ALL AUTOMATIC COMPS?

<cr> means NO, "YES", means delete all components placed in the initial placement phase.

5) GO BACK TO CRITICAL PLACEMENT

Allows the user to return to placement phase 1.

6) See general utilities for other menu options

G) Placement phase 3 (placement improvement)

menu choice explanations:

1) STEINBERG

Not yet implemented.

2) PAIRWISE INTERCHANGE

This algorithm tries all possible cell interchanges in an attempt to reduce the estimated wire length. Several important statistics are given:

a) Ratio of successful to tried exchanges

b) Estimated total wire length (ETWL)

c) Percentage improvement in ETWL

d) Estimated board utilization

After the pairwise interchange is done the prompt:

PAIRWISE INTERCHANGE AGAIN?

will appear. <cr> will cause the algorithm to be done again, "no" will let the user exit pair-wise interchange.

3) GO BACK TO INITIAL PLACEMENT

Allows the user to return to placement phase 2.

4) GO BACK TO CRITICAL PLACEMENT

Allows the user to return to placement phase 1.

5) END OF PLACEMENT IMPROVEMENT

Allows the user to proceed to placement phase 4.

6) RESTART INITIAL PLACEMENT

To the prompt:

DELETE ALL AUTOMATIC COMPS?

<cr> means NO, "YES" means delete all components placed in the initial placement phase.

7) See general utilities for other menu options..

H) Placement phase 4 (final manual changes)

menu choice explanations:

1) PLACE DISCRETE COMPONENTS

Discrete component placement prompts the user

with every unplaced component that is left. If the user doesn't want to place the current component, the menu response SKIP COMPONENT will give a new component to place.

2) GO BACK TO PLACEMENT IMPROVEMENT

Allows the user to return to placement phase 3.

3) SKIP COMPONENT

Valid only in DISCRETE placement.

4) END OF PLACEMENT

If all components have been placed the user is allowed to leave PLACEMENT and update the design.

5) See general utilities for other menu options.

5.2.3. General Utilities (an Explanation of Various Menu Options)

A) DISPLAY

Display allows the user to changethescaling for board viewing. Menu options:

1) CURRENT SCALE, MAGNIFY xx, REDUCE xx

Change the scaling by this factor. The user indicates the new center of the screen when the cross-hairs come up again.

2) See general utilities for other menu options.

B) RESET SCALE

Set the scale such that the entire board can be viewed.

C) LABEL

Write the names of each component over each component.

D) RULER

Display the ruler.

E) RESUME

Return to wherever you came from.

F) HELP

A future feature.

G) MANUAL CHANGE

Manual change allows the user to move and delete components from the board. Menu options:

1) EXCHANGE

Use the cross-hairs to indicate the two components to be interchanged. All locations are exchanged.

2) DELETE

The component indicated by the cross-hairs will be removed.

3) MOVE

a) Critical components are moved to the exact location indicated by the cross-hairs.

b) Automatic components are moved to the cell indicated by the cross-hairs.

4) CRITICAL MOVE

The component is moved to the exact location indicated by the cross-hairs.

5) CLEAN

Regenerate the picture on the screen.

6) See general utilities for other menu options.

H) SAVE/RESTORE

To the prompt:

SAVE OR RESTORE?

1) SAVE -- This will update the design file with this version of the placement. Routing cannot be done on a placement saved this way. To the prompt:

LEAVE PLACER?

<cr> means "no". A response of "yes" will terminate the PLACER.

2) RESTORE -- This will reconstruct the placement that is in the design file. (If no placement has been put in the design file then that is what will be reconstructed.)

5.2.4. WAIT INSTRUCTION

Whenever the prompt:

CONTINUE(CR)

appears this means that the user is to hit <cr> to continue with the program. (If the user responds "stop" the program will be exited and the design file will not be updated.)

5.2.5. SPECIAL PROMPT RESPONSES

A) "$wyl" -- Allows user to exit from the PLACER (go back to WYLBUR). To return to PLACER type "uuuPLCRn"

like was done initially.

B) "comment" -- This allows the user to type directly into the PLACER log. This is particularly useful if a user encounters an error he wishes to show to a systems programmer. After typing "comment" the prompt:

EXPLAIN:

will appear. Tell your problem. To get back to the original question make the character before the <cr> a PERIOD(.); this will allow you to stop commenting. Don't forget to take the output (PLACER log) to someone in charge of the SPRINT system.

5.2.6. PLACEMENT LOG

The placement program gives the user a complete log of his actions during the execution of the program. If you should encounter a problem:

A) Release the output.

B) Contact the person responsible for the system.

5.3 Error Messages

error number	severity	routine	description
1	1	GETCMD	Menu line selected is not an allowable choice. (Message was written to user.) Fixup: Reprompt
2		GETINT	A non-integer has been entered. (e.g. 7.1 or 7YES) Fixup: Reprompt
3		GETREL	A non-real has been entered. (e.g. 7YES) Fixup: Reprompt
4		GETREL	Multiple decimal points have been entered instead of a real. (e.g. 7.12.8) Fixup: Reprompt
5		GETGRD	Invalid gridsize entered. Fixup: Reprompt
6		INITPR	Incorrect project name. Fixup: Reprompt
		INITPR	Incorrect user name. Fixup: Reprompt
8	2	GETCLL	Cells defined by user are too small for type(s) specified. Fixup: Reprompt
9	5	SETOBJ	Object space range is out of order--PROGRAMMER error. No fixup. Job terminated.
10	5	SETSUB	Subject space range is out of order--PROGRAMMER error. No fixup. Job terminated.
11		GETCLL	Zero components per column is clearly not valid. Fixup: Reprompt

error number	severity	routine	description
12	1	GETCLL	Zero components per row is not valid. Fixup: Reprompt
13	1	GETCLL	Type name specified for automatic placement is not on catalogue. Fixup: Reprompt
14	1	GETCHA	Too many characters typed in. Fixup: Reprompt
15	1	IPLCMT	Instruction during initial placement cannot be identified. Fixup: Ignore and continue
16	1	I PLCMT	Assigning more than one automatic component per cell is not valid. Fixup: Reprompt
18	1	GETCLL	Too many components per column in the cell definition. Fixup: Reprompt
19	1	GETCLL	Too many components per row in the cell definition. Fixup: Reprompt
20	1	GETCMD	Invalid menu choice.
21	1	MANCHG	Component must be indicated when exchanging components. Fixup: Reprompt Invalid delete command.
22	1	CPLCMT	The component name indicated could not be located either because it had already been placed or the name was invalid. Fixup: Ignore
24	1	CPLCMT	Placement causes overlap. Fixup: Ignore

error number	severity	routine	description
26	2	MANCHG	Invalid exchange command.
27	5	DISPSG	Too many segments to display. PROGRAMMER error. No fixup. Job terminated.
30	2	DISPLY	Invalid menu choice. No message sent to user. Fixup: Reprompt
31	2	MANCHG	Invalid delete command.
32	2	REMOVE	The user has not indicated a component with the cross-hairs, and no component was removed. Fixup: Ignore
33	1	GETCLL	The user has indicated more types for automatic placement than are available. Fixup: Get new value (NAUTO)
34	5	GETCMP	No more records in comp. list. Probable USER error. No fixup. Job terminated.
35	5	GETCEL	No more records in cellst. Probable USER error. No fixup. Job terminated.
36	1	GETCLL	The user has indicated a type as an automatic type more than once. Fixup: Reprompt
37	2	PLACE	The same as error #30.
38	2	PLACE	The same as error #30.
39	2	PLACE	The same as error #30.
40	2	PLACE	The same as error #30.
41	2	IPLCMT	The same as error #30.
42	2	MANCHG	The same as error #30.

error number	severity	routine	description
43	6	NEWPIC	No more space in the graphic element. (Recompile with a larger element.) No fixup. Job terminated.
44	1	MANCHG	Component was not moved to new location because of overlap. Fixup: Replace component and reprompt for a new component to move
45	2	MANCHG	The same as error #21. Invalid move command.
46	6	GETCIR	Component and net cross listing dynamic lists have been overflowed. Probable USER error. No fixup. Job terminated.
47	6	GETCIR	Pin type in net list has an invalid value. Probable USER error. No fixup. Job terminated.
49	1	DPLCMT	User asked for discrete component placement while doing discrete placement. Fixup: Ignore
50	1	DPLCMT	The same as error #1.
51	1	DPLCMT	The same as error #24.
53	1	PA1RP	Component table overflow; Too many dummy components. Fixup: Stop placing dummy components.
54	4	IPRAND	Not all automatic types will fit on the board. No more will be placed Fixup: Continue

error number	severity	routine	description
55	6	PAIRP	Net list overflow; too many nets involved in component pair interchange. (Recompile with a larger array for common nets in subroutine PAIRP.) No fixup. Job terminated.
56	6	GETCIR	Too many types, nets, or components defined. (Recompile with more space for these data fields.) No fixup. Job terminated.
57	1	CPLCMT	The same as error #1
58	8	RESTOR	RUN error. PROGRAMMER error. One or more components have replaced on the board incorrectly. No fixup. Job terminated.
59	1	PLACE	The menu instruction "skip" is only valid while in Discrete Placement. Fixup: Ignore
60	6	GETCIR	Pointer to the board records is equal to zero. (Invalid design file--bad data.) USER error. No fixup. Job terminated.
61	6	GETCIR	Too many type pins defined by design file. (Recompile with more space for type pin definitions.) No fixup. Job terminated.
62	6	GETCIR	Component names have overflowed the name list. (Recompile with more room for names.) No fixup. Job terminated.
63	6	GETCIR	Typed names have overflowed the name list. (Recompile with more room for names.) No fixup. Job terminated.

error number	severity	routine	description
64	5	DISPNB	The integer to be displayed is out of range (too big). PROGRAMMER error. No fixup. Job terminated.
65	4	PROMPT	The prompt to be assigned is too long. Fixup: Truncate to prompt to the maximum size.
66		CPLCMT	Logical connector has fewer pins than associated board connector. Fixup: Continue
67		CPLCMT	Logical connector has more pins than associated board connector. Fixup: Reprompt
68	3	FDUMP	NAUTO is greater than 19. Fixup: NAUTO set to 19.
69		PA1 RP	No automatic components have been placed. Fixup: Exit pairwise interchange
71		CPLCMT	Cannot assign non-external components to board connectors. Fixup: Reprompt
72	8	CPLCMT	A non-external component has no physical type. No fixup. Job terminated.
73	8	GETCIR	A net has no pins. No fixup. Job terminated.
74	2	PLACE	Must associate all logical connectors with board connectors. Fixup: Go back to Critical Placement
99	8	GETREC	No more free records available. Probable USER error. No fixup. Job terminated

6. MANUAL PREWIRING OF CRITICAL CONNECTIONS (MWIRE)

6.1 Introduction

The MWIRE program allows a user to route critical connections manually, using an interactive graphics terminal. It can also be used for routing multi-width wires (e.g. for power and ground connections).

6.2 Using MWIRE

6.2.1 Exec File

An exec file has been provided for using MWIRE. It can be used as follows

```
exec from #runmwire user pcb group cg cle
THIS EXEC FILE RUNS THE MANUAL WIRER.
Type "?" for help.
```

A reply of ? to any prompt will explain the question.

```
<cr>=CLR ACT or SAV filename? clr act
```

A response clr act clears your active file while SAV x will save it in a file x under your account number.

```
DIRECT ACCESS FILE=DFILE? dfi le21
```

Enter the name of the design file containing the problem (in this case the name is DFILE21). The default name is DFILE.

```
FILE DFILE21 ON VOL=SCFEV9?
```

Enter the volume on which the design file is stored. A <CR> will give you the default volume SCFEV9.

GROUP.USER = CG.PCB? cg.wmv

Enter your group and user name in the format shown.

6.2.2 Explanation of Menu Commands:

A) SET NET -- SET NET sets the current net. Wires may be placed and removed only from the current net. You may set current net for any net. Even nets you have completely wired. This allows you to wire net A, wire net B, and then unwire and modify net A.

B) GET PIN -- GET PIN is used to release the pin which you have pointed to with the cursor. This is useful when you have pointed to the wrong pin when starting a run.

C) DISPLAY -- DISPLAY allows you to change the board area on the screen. The options include:

1) RES - reset picture to full board

2) RED - reduce magnification

3) CUR - use current magnification

4) MAG - increase magnification

Options 2, 3 and 4 require a second 'hit' with the cursor to indicate the center of the new screen image.

D) SET SIDE -- SET SIDE allows you to explicitly define the side on which the wires are placed. Options include:

1) H - put wires on horizontal side (auto router)

2) V - put wires on vertical side (auto router)

3) C - put wires on component side

4) S - put wires on solder side

5) BOTH - put wires on horizontal and vertical to match the auto router.

Note that H and V will be paired with C and S thus showing the auto router's preferred alignment. Wires placed on the wrong side (with respect to the auto router) are obstructions to the auto routing process and should therefore be used cautiously. The BOTH option will automatically choose the correct side for each wire placed and should therefore be used whenever possible. The current side set can be seen with the show status command.

E) SHOW STATUS -- the SHOW STATUS command displays the following useful information:

1) current wire width in grids

2) current side mode (see SET SIDE)

3) current net (see SET NET)

4) number of unwired pins in current net (if set)

This command should be used whenever such information would be helpful.

F) REMOVE SEGMENTS-- this command allows you to 'pop' segments off the net stack. You will be asked for the number of segments to remove. A response >= the current number of segments for the net will remove all segments.

Note that multi-width segments are considered as single segments.

G) Set wire width -- this command causes the prompt

wire width in grids?

to be issued. An answer of 1 will allow all wires to be placed single width from pin to pin or pin to segment. If a width of > 1 is given then you will be placed in multi-width mode. In multi-width mode you will not be allowed to intersect with pins and therefore wire from point to point or point to existing wire (point being a place on the board free from wires and pins).

** NOTE ** When in multi-width mode you are required to give three cursor 'hits' for each segment. They are:

1) starting point (not at pin or wire)
2) finishing point (to or through intersecting wire is allowed and wire will be trimmed)
3) direction of width (high or low from base line)

In single width mode wires require:

1) a hit at origin pin of run (only for first segment of run)
2) a 'hit' at every corner of run (to or through wire or segment;intersections will be trimmed).

It is generally recommended that all multi-width wires be placed early. After all multi-width wires are placed it

is very simple to reset width to 1 and connect all the multi-width wires to pins with single width wires.

H) SHOW UNWIRED PINS -- this command will flash the unwired pins on the screen. Note that this command is only valid if the current net is set.

I) LABEL COMPONENTS -- LABEL COMPONENTS simply labels each component on the current screen image.

J) END MANUAL WIRING -- this command allows you to exit the manual routing program. If you have placed any segments you will be asked to ok the exit. This is just a safety in the case that you don't want to exit. By responding with anything but 'yes' you will be returned to the wiring mode. Typing 'yes' will end the interactive session and update the design file with all new segments and vias.

6.2.3 Starting Sequence

When the interactive subsystem is active you will be asked for your name and the project name. This is to be sure you are using the correct design file. The prompts will be repeated until they are answered correctly. If you are not sure of the project name then type '$WYL' and cancel the job.

** NOTE ** If the design file has any wires in it (either from previous manual wiring or auto routing) then the prompt: 'board wired, type "yes" to start over'

will be issued. Should you not wish to remove all the old wires then answer '$WYL' and cancel the job. If you want to start over then answer'yes'as instructed.

** NOTE ** A response of 'yes' to 'board wires, type "yes" to start over' will not remove the old wires until the end of the job. If you respond yes and then later cancel the job abnormally then all wires will be left as before the job started.

It is possible to simply remove all the wires from a board by responding yes to the start over question and then not placing any manual wires. This will clean the board for the auto router.

6.3 Error Messages

error number	severity	procedure	cause	action
101	1	widen t	wrong user name	retry
102	1	widen t	wrong project name	retry
103	6	widen t	board not placed	abend
104				
105	7	wloadc	rec #1 comp ptr <=0	abend
106	7	wloadc	rec #1 board ptr <=0	abend
107	7	wloadc	rec #1 net ptr <=0	abend
108	2	wire	invalid cursor command	ignored

error number	severity	procedure	cause	action
109	2	wire	"show unwired pins" before set net	ignored
110	2	wire	"remove segments" before set net	ignored
111	2	wire	"start net" with an open run	ignored
112	2	wire	invalid net name given	reprompt
113	2	wire	"get pin" before start net	ignored
114	2	wire	"get pin" with an open run	ignored
115	2	wire	invalid pin pointed to by cursor	retry
116	2	wire	cursor off board	retry
117	2	wire	pin intersection on multi-width	retry
118	2	wire	invalid via this segment	wire not placed
119	2	wire	pin intersect with different net	wire not placed
120	2	wire	pin intersect causes loop	wire not placed
121	2	wire	subnet formed on pin intersection	wire not placed
122	2	wire	wire intersect with different net	wire not placed
123	2	wire	wire intersection causes loop	wire not placed

error number	severity	procedure	cause	action
124	2	wire	"end manual wiring" with open run	ignored
125	5	putseg	no more room for segments	abend
126	8	getrec	no free records in design file	abend
127	2	wire	"set wire width" with an open run	ignored

WARNING Special care must be taken when routing multi-width wires to avoid later errors in HIWIRE. In particular, if multi-width segments are used in a net they must form a single connected network branching out from one or more pins in that net. This is a special case of a general requirement: all prerouted segments in a net must be electrically common with each other and with a pin in the net. This requirement is enforced when wiring single-width segments but unfortunately has not been for multi-width segments. Failure to connect to at least one pin (with a single-width segment) will cause an error halt in HIWIRE; failure to keep all segments connected will result in disjoint paths for the net in the final routed board.

7.1 BATCH ROUTING PROGRAM (HIWIRE)

7.1 Introduction

7.1.1 Purpose

HIWIRE is an automatic batch router using a modified version of Hightower's algorithm to route two-sided PC boards. The algorithm provides a relatively high connection completion rate using relatively small amounts of computer time. It is, however, only a heuristic, and will not necessary find the optimal path between two pins or even find a path at all when one might exist. A fair amount of optimization is done by the program, though, to maximize the chance of finding a good path.

7.1.2 General Use

HIWIRE uses the design file for input and output. PLACER must have been run on the design file to provide pin locations for the networks, and MWIRE may have been run to route multi-width busses or critical wires, and to insert wiring obstructions. After routing is completed (as far as possible), the design file is updated by inserting information about the segments and vias. This information can then be used by VIAELIM to minimize the number of vias or be output on the plotter. In addition to the design file output, **HIWIRE** produces a listing of all connections it has made for each net, a list of pins it has been unable to connect and the paths it

tried, and general information about the design file and the job. This information may be printed, if desired.

7.1.3 Applicability

As stated above, HIWIRE can only handle boards with two signal planes at the present time. Routing is done with all horizontal segments on one plane and all vertical segments on the other, though VIAELIM will try to bring as much of each path on one side of the board as possible. The board size, number of pins, number of nets, etc., which can be handled is fairly arbitrary; most parameters can be set at run time instead of being fixed in the program. Another factor in applicability is the quality of the routing obtained for a given circuit. All that can be said here is that HIWIRE will do the best it can with the placement it is given. Care must be taken to avoid producing overly congested areas on the board (e.g., near an external connector) or manually routing too many (or too long) 'wrong' side wires (which act as obstructions to HIWIRE). With experience, you will probably develop your own estimation of what will and will not be routable at an acceptable level of completion. The ease with which a placement can be modified, combined with the relatively fast execution of HIWIRE provide a system for adaptively improving the router's

performance on a given board. Hopefully, all connections can be completed or few enough will be left that only minor editing by hand will be necessary.

7.2 Using HIWIRE

A) An exec file is provided to help you run the router. To use it type:

exe fro #runroute use pcb gro cg clr

After an introduction, the exec file will prompt you for information needed to run the job. The prompts are:

1. <CR>=CLR ACT or SAV filename?

   Typing a carriage return will clear the active file. If there is something there you want to keep, type a save command to put it into a file.

2. FILE ACCOUNT = CG.PCB?

   Type the account under which the design file is stored. The format is group.user. Type a carriage return if under the default account.

3. DIRECT ACCESS FILE = DFILE?

   Type the design file name or <CR> for default filename.

4. (FACC).(DAFNAME) ON VOLUME = SCFEV9?

   Type the name of the volume on which the previously specified design file is to be found (<CR> for default).

5. TIME = 4?

Specify the length of time you wish to let the job run. The default time of four minutes should be adequate for most boards. If the board is small, you may be able to get by with as little as half that much, and by specifying a smaller time limit you can increase the job's priority and shorten your wait time. Some very large boards (or fairly large boards with a bad placement) may take more than 4 minutes, but if you must specify a larger time limit, be forewarned that the job's priority will go down drastically and you may have a long wait ahead of you. Still, it is always best to be sure you have given the job enough time to run; a partially run job is of no use and in the worst case you may leave the design file in a partially altered state. The answer to this prompt should be <CR> for the default time, a single digit for a time limit in minutes, or something of the form '(m,ss)' for a time limit in minutes and seconds.

6. OUTPUT LEVEL = 1?

Not all of the output HIWIRE is capable of producing may be useful every time the program is run. This prompt allows you to select the amount

of output generated. Available levels are:

0 - Overall statistics and error messages only

1 - Level 0 plus an indication of unconnected pins for each net.

2 - Level 1 plus a list of what pin-to-pin connections were attempted and whether or not they were successful.

3 - Level 2 plus lists of endpoints for each successful path.

4 - Level 2 plus detailed information (discussed below) about the unsuccessful connection attempts.

5 - All possible output (Levels 3 and 4 combined).

Type the number of the desired output level or a **<CR>** to use the default.

7. **MODIFY CONSTANTS (<CR>=NO)?**

If you wish to run the standard version of **HIWIRE** (the array sizes are set to handle a fairly large and complex board), simply hit a **<CR>.**

Otherwise,you may change certain important constants in **HIWIRE** for this particular run. There are at least three possible reasons for

doing this:

A) You have a small board which does not require the large array limits provided in the standard version. Decreasing the memory requirements has the same effect as decreasing the time limit - your priority and turnaround time are improved.

B) You have a very large circuit and an error message says you have overflowed some table limit. You can increase the size of that table and try again (you may want to increase the size of several of the tables, since they are fairly well matched to a given maximum size circuit). The default memory size should be adequate for almost any increase in table sizes.

C) Some of the constants control how hard the router looks for a path and how desirable the final path will be. You may wish to change these values, although the default values are considered to be close to the optimum.

8. The constant prompts:

You will not get the following prompts if you typed <CR> to prompt #7. If you wish to use the default value for a constant, type a <CR>.

A) NETTL=300?

Maximum of 300 signal nets.

B) PINTL=1500?

Maximum of 1500 pins.

C) SEGTL=4000?

Somewhat difficult to define; a net will use one segment for each X or Y coordinate along which any of its segments lie. Some work space for the next connection is also required.

D) XTNTTL=6000?

Closer to the actual number of segments, but includes 2 elements for each unconnected pin (including those not belonging to any net) and some workspace for figuring the next connection.

E) BRDXSIZE=250?

BRDYS IZE=200?

Maximum X(Y) coordinate (in grid units) of any point on the logical board.

F) EPTL=41?

Approximately the number of segments HIWIRE

will draw before giving up on a given connection. Increasing this may increase the number of connections found, but the number of segments and vias in the added paths may be unacceptable.

G) MAXNETPASSES=2?

Maximum number of passes HIWIRE will make through the unconnected pins in a net, trying to connect them to the connected group of pins and segments (HIWIRE maintains a single connected group of pins; no separately connected subgroups of pins are allowed). HIWIRE will continue trying to connect unconnected pins to the connected group until this number is exceeded or no connections are made on a given pass.

H) MAXMISSES=10?

In the path refinement process where extraneous backtracking is eliminated from the original paths coming from the basic routing algorithm, this constant governs how far HIWIRE will look for a chance to improve the path. An increased value will increase run time but may improve on some paths found.

I) **VIATL=200?**

The via table must be able to hold all of the prerouted vias (from MWIRE or previous **HIWIRE** run) plus the maximum number of vias produced in any of the nets.

J) **GOREGION=512?**

Specifies the program's memory requirements in multiples of 1024 bytes. Default value should be able to **accomodate** almost any increase in table sizes; with enough decreases in table sizes for a small board, this can be reduced to 320 (possibly 256 in extreme cases). Decreasing memory requirements will improve priority and turnaround time.

B) Unless something goes terribly wrong (bad answers to some of the exec file prompts could be one cause), extensive printed output about the routing should be available. This includes:

1) Information about the input design file:

A) User name

B) Project name

C) **Gridsize**

D) Major component alignment (0 for horiz., 1 for vert.)

E) Total estimated wire length in grids

2) Nets are listed in the order in which they were attempted. Nets are routed in order of increasing size of the box that would enclose all pins in the net. Routing for one net is taken as far as possible before going to the next net and no backup is allowed.

3) Within a net, pairs of pins are chosen for each attempted path and printed in the form:

COMP #1 PIN #1 XLOC YLOC => COMP #2 PIN #2 XLOC YLOC

Where #1 is the source and #2 is the destination (in many cases pin #2 ends up only giving direction to the path since the actual connection will often be made to some existing segment of the net instead of to the pin itself).

4) A pin pair will either route successfully or not. If a path is completed, the coordinates of each segment endpoint are listed, starting with the source pin and ending either the destination pin or the point at which a connection was made with a previous segment. If no path is found, a message is printed to that effect. If HIWIRE thinks the failure may have been due to too small a value of 'EPTL' (see exec file prompts), it will add the message: 'path too long'. Finally, a table is printed to indicate how HIWIRE was attempting to complete the path. Columns in this table are:

A) Line number

B) X location of 'corner'

C) Y location of 'corner'

D) Minimum coordinate of test line through 'corner' in specified orientation

E) Maximum coordinate of test line (with 'D)' indicates where that line was looking for an intersection)

F) Orientation (1=horizontal, 2=vertical)

Entries coming down from the top of the table are from the source pin, those coming up from the bottom are from the destination pin (test lines are sent out from both pins to increase the chances of connection). This list of the automatically attempted path may aid the manual insertion of unroutable wires. It can also indicate areas of blockage on the board; if one or both of test line lists stop considerably short of the center of the table, it probably means a blockage exists.

5) The output for each net ends either with a statement that all pins were connected, or with a count and list of the unconnected pins.

6) After all possible routing is completed, a summary of the routing is given, including:

A) Number of pins on board

B) Number of unused pins (not in any net)

C) Number of unrouted pins (in net but not connected)

D) Number of segments generated

E) Number of segment elements used (LSEG) - compare to Exec file constant 'SEGTL' (doesn't include work area)

F) Number of extent elements used (LXTNT) - compare to Exec file constant 'XTNTTL' (doesn't include work area)

G) Actual number of segments put into design file (straight lines broken into segments by vias are output as a single segment to save space in the design file) and their total actual length

H) Number of vias put into design file

I) Index of next available record in design file

7.3 Error Messages

The classes of errors recognized by **HIWIRE** are:

A) WIRING ALREADY COMPLETE IN DA FILE

Safeguard to prevent overwrite of wrong design file.

B) INPUT OUT OF RANGE

Bad input is specified; check to see that it really is valid and change appropriate limit using exec file if necessary.

C) INCOMPLETE PLACEMENT IN DA FILE

Flag indicating completed placement is not set; placement has not been run or completed.

D) CHAINING/COUNT ERROR IN NET REALIZATION (VIA LIST) RECORDS

Number of segments(vias) specified does not equal that

found. Compress the design file and save it for later analysis along with the router output.

E) TOO MANY SEGMENT GROUPS (UNIQUE SEGMENTS) (VIAS)

Out of space in one of the dynamically allocated tables. Change the indicated limit using the exec file.

F) DESIGN FILE OVERFLOW

A fatal error. Not enough room to add all of the segments and vias produced by HIWIRE but some probably have been added. A good reason to keep a compressed backup copy of whatever work has been completed betweeen runs of any two SPRINT programs.

G) PIN TO BE ROUTED NOT FOUND AMONG OBSTACLE POINTS

All pins are considered to be obstacles, even to wires from their own net, until it is their turn to be routed. Here, something has gone wrong with that convention. Save the compressed design file and router output for later analysis.

H) PIN OFF LOGICAL BOARD

Probable program error in either PLACER or HIWIRE. Could also result from using a design file that has somehow become partially mangled. Please save the compressed design file and router output for later analysis.

I) PREROUTED WIRES BUT NO CONNECTED PINS IN NET #n

The segments for any net must be a connected whole which also includes at least one pin. MWIRE, unfortunately, does not check this condition when multi-width segments are involved. Manual and automatic routing must be redone.

8. VIA ELIMINATION PROGRAM (VIAELIM)

8.1 Introduction

8.1.1 Purpose

VIAELIM is a batch program designed to eliminate unnecessary via holes from the automatic routing produced by HIWIRE. This is a fairly important function, since HIWIRE routes all paths with horizontal and vertical segments on opposite sides of a board, resulting in a large number of vias. If carried through to the final board, these vias would increase the board's cost and decrease its reliability.

Another property of VIAELIM is that, in changing the locations of segments from one side of the board to the other to eliminate vias, it may clear previously blocked paths and allow further routing to be done by another pass through HIWIRE. Segments now on the 'wrong' side will act as new obstructions to automatic routing, so one should not expect large numbers of previously unroutable pins to be routed by this process. Some improvement in the completion ratio, however, could occur.

8.1.2 Implementation

VIAELIM is based, in part, on an algorithm presented by Stevens and Hashimoto at the 1971 Design Automation Workshop. Basically, it models the routing as a graph in which the nodes represent disjoint sets of mutually intersecting segments and the edges represent vias where segments from different groups connect. Each node is assigned one of two colors, indicating that segments in the set either stay where they are or switch sides, attempting to minimize the number of edges between two nodes of the same color. Edges between nodes of different colors represent vias that can be eliminated. This procedure is equivalent to trying to find the maximal bipartite subgraph of the modeling graph. Routing information is obtained from the design file and is updated when via elimination is complete.

8.2 Running VIAELIM

8.2.1 Exec file

An exec file is provided to help you run the via elimination program. To use it, type:

```
exe fro #runvelim use pcb gro cg clr
```

After an introduction, the exec file will prompt you for information needed to run the job. Typing a '?' to any of the major prompts will print a message reminding you how to answer. The prompts are:

1. <CR>=CLR ACT or SAV filename?

   Typing a carriage return will clear the active file. If there is something there you want to keep, type a save command to put it into a file.

2. FILE ACCOUNT = CG.PCB

   Type the account under which the design file is stored. The format is (group.user). Type a carriage return if under the default account.

3. DIRECT ACCESS FILE = DFILE?

   Type the design file name or <CR> for default filename.

4. (FACC). (DAFNAME) ON VOLUME = SCFEV9?

   Type the name of the volume on which the previously specified design file is to be found (<CR> for default).

5. MODIFY CONSTANTS (<CR>=NO)?

   If you wish to run the standard version of VIAELIM, type a carriage return. Otherwise, you may change certain constants (primarily array sizes) for this particular run. Increasing allowed execution time or the size of certain arrays may be necessary for very large or complex circuits, perhaps in response to an error on a previous run. If you type 'yes' to this prompt, you will be prompted for values for each of the constants listed below.

If the default value given is sufficient, hit <CR>; otherwise type a new value. Responding with a '?' is not valid for these prompts.

a. TIME=(m,ss)?

Minutes ('m') and seconds ('ss') of execution time allowed. New value, if desired, should be typed using the same format or you may type a single digit to specify minutes only. The larger this value is, the lower your job's priority will be.

b. NETTL=value?

Should be greater than the number of signal nets in the circuit.

c. SEGTL=value?

Should be somewhat greater than the number of segments routed in MWIRE and HIWIRE.

d VIATL-val ue?

Should be greater than the number of vias produced in MWIRE and HIWIRE.

e STKLN=val ue?

Should be greater than $LOG_2(2*SEGTL)$, i.e. if you double SEGTL, add 2 to STKLEN.

f GROTL=value?

Maximum number of sets of mutually intersecting segments. This varies greatly from one board to the next, and possibly between two routings of the same circuit. The output of VIAELIM contains a line 'NGRO=val' where 'val' is actual number of groups in that run (GROTL should be somewhat greater than this value).

g. CONNTL=value?

Two elements of the connection table (of which CONNTL is the length) are used for each pair of groups having any segments connecting at a common via. Also difficult to estimate; the line 'LCONN=val' in VIAELIM's output gives the number of elements actually used in that run.

h. GOREGION=value?

Specifies memory requirements in multiples of 1024 bytes. If you have substantially increased any of the table sizes, particularly NETTL, SEGTL, or VIATL, it may be necessary to increase this value.

6. After the exec file has set up the necessary JCL in your active file, it will type:

TYPE 'EXE' TO RUN THE VIA ELIMINATION JOB.

→EXE PAU

If you now type 'exe', the exec file will submit your VIAELIM job to the system to be run.

8.2.2 Output

Most of the activity in VIAELIM has to do with the design file. A small amount of printed output, however, is generated. This includes:

1. Information from the original design file, including the user and project identifications and a count of the number of segments and vias in the file.
2. Values for NGRO and LCONN mentioned above.
3. An indication of successful completion or an error message (see below).
4. If completed successfully, a count of the number of segments and vias in the updated design file. Note that the number of segments may change due to the splitting of a segment into two segments connected by a via if inserting that via eliminates others elsewhere.

8.3 Error Messages

Classes of errors recognized by VIAELIM include:

A. Non-applicability

1. No placement in design file
2. No routing in design file.
3. No non-critical (and therefore movable) wires in design file.

B. Input errors

1. Conflict between record chaining and stored count for net realization (segment) records.
2. Chaining/count conflict in via records.

C. Table overflow for each of the table sizes that can be set in the exec file.

D. Program error in the color optimization routine. If this occurs, please save your output and the design file.

E. Design file overflow
Should never happen since fewer records should always be needed after via elimination. Please save the output and design file.

F. Via in design file with 0 or 1 segments attached. This is a warning rather than a fatal error. The location of the bad via and the net it is in are printed, the via deleted and the program continues. This problem would generally come from incorrect connection of multi-width wires in MIWIRE. If you have not used any multi-width wires or are puzzled by the message's appearance, please save the output and design file.

9. OUTPUT PLOT GENERATION

Two separate programs, PLOT and ARTWORK, are provided for plotting a design produced with the SPRINT system.

9.1 PLOT

PLOT is a basic program for outputing a routed board on the Versatec plotter. It produces a simplified plot consisting of three pictures; one of each side of the board and a composite. The drawingsare produced at 1:1 scale and the picture of the solder side routing is a mirror image. A diagonal line is drawn through an intersection that represents a connection, regardless of whether or not a via exists at that point. While the plots produced by this program will probably not be acceptable as the final output from a design, they may be useful at some intermediate point.

To run this simplified plotting program, type:

exe fro #runplot use pcb gro cg clr

After an introduction you will receive the following prompts:

A) <cr>=CLR ACT or SAV filename

Type a <CR> to clear the active file for use as a work area. Otherwise type a SAVE command to store the active file under some filename.

B) FILE ACCOUNT = gg.uuu?

Give the account the design file is stored under. The default, chosen by hitting <CR>, is the logged

on user's account. Format is gg.uuu (group.user).

C) DIRECT ACCESS FILE=DFILE?

Give the name of the file containing the design you want to plot.

D) gg.uuu.filename ON VOLUME=SCFEV9?

The volume the specified design file is on. The default is the scratch disk 'SCFEV9'.

E) STANDARD ORIENTATION (<CR>=yes)?

Because this program always produces a 1:1 plot rather than scaling the picture to fit the page, a board may be defined which would not fit on the paper in the normal orientation (x-axis horizontal) but would fit if rotated 90°. To get this rotated plot, type 'no'; otherwise type a <CR>.

F) TYPE 'EXE' TO RUN THE PLOTTING JOB

The appropriate JCL has now been collected in the active file. Type 'exe' to run it.

When the program is done, its output will be in the print hold queue. If the plot was successful, there should be a data set labeled 'GO.UGDEVICE' containing several hundred lines. Release this data set to obtain the plot.

The only error message produced by PLOT concerns not being able to fit the drawing on the paper. If switching to the opposite orientation does not eliminate the problem, PLOT cannot be used for that board.

9.2 ARTWORK

ARTWORK produces a more comprehensive set of drawings of a design. The output can currently be sent to any of three devices:

A) Tektronix 4013 terminal - This option gives you a quick look at the design on the terminal's storage tube screen.

B) Versatec plotter - A reasonably fast way to get hard-copy drawings of the design, generally as an aid in making modifications during the design process.

C) Calcomp plotter (33" paper) - High-quality, 3-color plots at 2:1 scale from which the final taping of the board can be done.

9.2.1 Output

A set of nine pictures is produced by ARTWORK. These include:

A) Header and trailer pictures indicating the name and status of the design.

B) A top view of the board showing component placement.

C) A bottom view 'airline route' plot of the signal net chaining. This point-to-point representation of the nets can give a clear indication of how good the placement is and show which components should be moved closer together to improve an unsatisfactory

placement.

D) A bottom view composite drawing of the routing on both sides of the board. On the Calcomp plotter, routing from each side can be plotted in a different color.

E) A bottom view of the component side routing (as it would be seen looking through the board).

F) A bottom view of the foil side routing.

G) A top view of the component side routing.

H) A bottom view of the pin and via holes (pin holes are also drawn in pictures C through G, as square boxes, via holes are shown as X's in pictures D through G).

All pictures are labeled and are scaled as close to 2:1 as possible in the available space. The scale used is printed on the header and trailer sheets.

92.2 Running ARTWORK

To obtain ARTWORK plots on any of the devices, type:

exe fro #runart use pcb gro cg clr

After a short introduction, you will be given the following prompts:

A) <cr>=CLR ACT or SAV filename

Type a <cr> to clear the active file. If you want it saved under some filename, type a SAV command.

B) DIRECT ACCESS FILE=DFILE?

Type the name of the design file you want to produce art work for.

C) FILE filename ON VOL=SCFEV9?

Give the volume the specified design file is on.

D) GROUP.USER=gg.uuu?

Give the account the design file is stored under. The default is the logged on user's account.

E) Output device=VERSATEC?

This is where the program becomes specialized for a particular output device. Type a <CR> for the default device (the Versatec plotter in this case), 'ver' for the Versatec plotter, 'tek' for the Tektronix 4013 terminal, or 'cal' for the Calcomp plotter (33" paper).

F) TYPE 'EXE' TO RUN THE ARTWORK JOB

The specialized JCL has been assembled in your active file and will be run if you type 'exe'. The job's name will be of the form 'uuudddnn', where uuu is the user's name, ddd is the output device's name (e.g. 'TEK'), and nn is the last two digits of the current job number.

9.2.3 Getting the output

The extra steps necessary to obtain the output depend on the device used:

A) Tektronix 4013 - This job runs as an interactive

subsystem, much the same as PLACER and MWIRE. When the program is ready, it will send the message to the terminal:

FROM OPERATOR: (uuudddnn) THE PLOTTER IS READY

where 'uuudddnn' is the job name described above. To connect the program to your terminal, type the jobname. To "WHAT'S THE MAGIC WORD?" type, "punt". Each picture will be drawn on the screen (there may be some delay, particularly when the machine is heavily loaded), then the crosshairs will appear. When you want to generate the next picture, just hit the space bar. After the last picture, the screen is cleared and you are returned to Wylbur. Any printed output from the program can be purged.

NOTE: Due to limitations on the amount of memory available to CLASS I (interactive) jobs, some of the routing segments may not be output on large complex boards, particularly if VIAELIM has not been run. This is not a problem with the Versatec or Calcomp plots.

B) Versatec plotter - If the program completes successfully, the output in the print hold queue should include a data set labeled "GO.UGDEVICE" containing several hundred lines of output.

Releasing that data set will send the output to the plotter. The remaining printed output can be purged when the plotting is completed.

C) Calcomp plotter - The program should produce a plot tape which will be placed in your bin. You must then fill out a plot request card (available at the dispatch desk), attach it to the tape, and place the tape in the "TO CAMPUS' bin. A sample card, marked for the recommended configuration, is shown in Fig. 9.1. Some things to note:

1) 'CAMPUS ACCT' refers to an account valid for plotting at SCIP on campus, not a SLAC account. See your group secretary for an account to use or make arrangements with SCIP accounting.

2) Fill in 'REEL NUMBER' from the tape you receive.

3) The program scales everything for plotting on 33" paper. This will handle 2:1 plots of boards up to 16.5" long by 13.75" wide. For 2:1 plots of boards up to 5" long by 4.3125" wide or less than 2:1 plots of larger boards on 10" paper, type:

SCIP AT SLAC

1130 DRUM PLOTTER REQUEST

NAME Kim Stevens BIN 239

JOBNAME KRSCAL01 DATE 6/1/77

CAMPUS ACCT GG$UUU REEL NUMBER PLOT08

CIRCLE THE NEEDED ITEMS BELOW
* STARRED ITEMS ARE DEFAULT

PAPER SIZE = *10INCH (33INCH)

PEN# 1		PAPER STYLE
*PB	(*BLK)	(*BLANK)
L03	RED	GRAPH 10 DIV/IN
(L04)	BLU	
L05	.	
L06		
L08		

PEN# 2		PEN# 3	
PB	BLK	PB	BLK
L03	RED	L03	(RED)
(L04)	(BLU)	(L04)	BLU
L05		L05	
L06		L06	
L08		L08	

ABBREVIATIONS USED. PB=PRESSURIZED BALLPOINT

.NN=LIQUID INK, NN IS THE RAPIDOGRAPH PEN#

Figure 9.1'

exe fro #artcl0 use pcb gro cg clr

Everything in this exec file is identical to #RUNART except the Calcomp plots are scaled for 10" paper.

4) The 'L04' liquid ink pens should produce good plots. Larger pens may be used to produce greater line width, but 'L03' pens should never be used as they tend to dry out in the middle of complex pictures.

5) The color choice shown is recommended, but may be changed to fit personal requirements. Pen assignments are:

-Pen #1- used for lettering, board outline, component outlines, component side routing.

-Pen #2- used for 'airline route' picture and foil side routing.

-Pen #3- used for pin and via holes.

The plot should be completed within a day or so and will be returned, along with the tape to the 'FROM CAMPUS' bin. If the plot is satisfactory, you may send the tape back for another plot or return it to SLAC Dispatch.

9.2.4 Errors

Other than attempting to plot a non-existent or garbage

design file, there are no expected errors in ARTWORK. If the program should fail, please release and save the printed output and save a compressed version of the design file for later analysis.

The following pages contain the plots generated on the Versatec plotter for the example of Fig. 0.1.

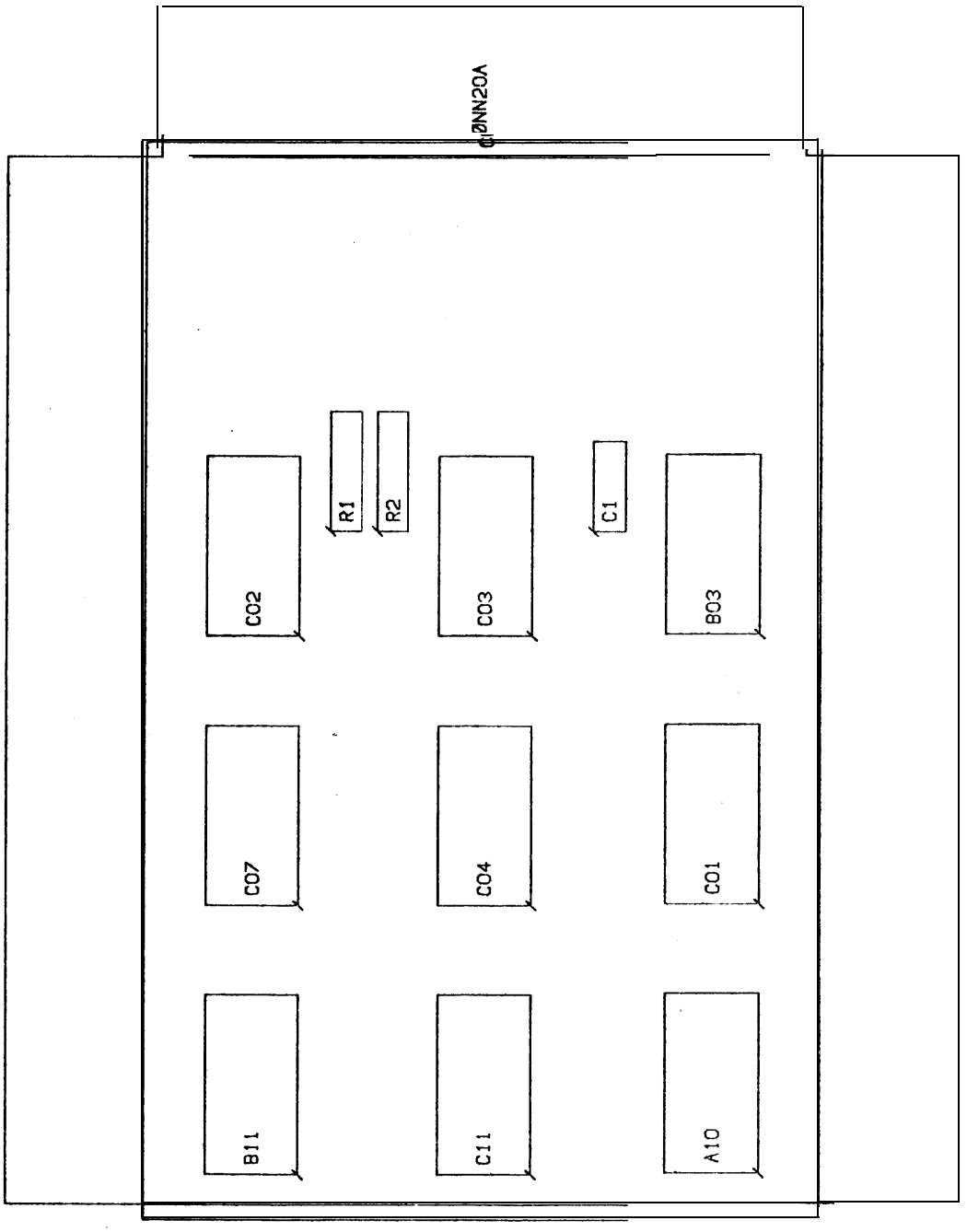
WMVC
EXAMPLE1

30 NETS
227 PINS
13 CØMPØNENTS
12 LØGICAL TYPES
5 PHYSICAL TYPES
0.0500" GRID SIZE

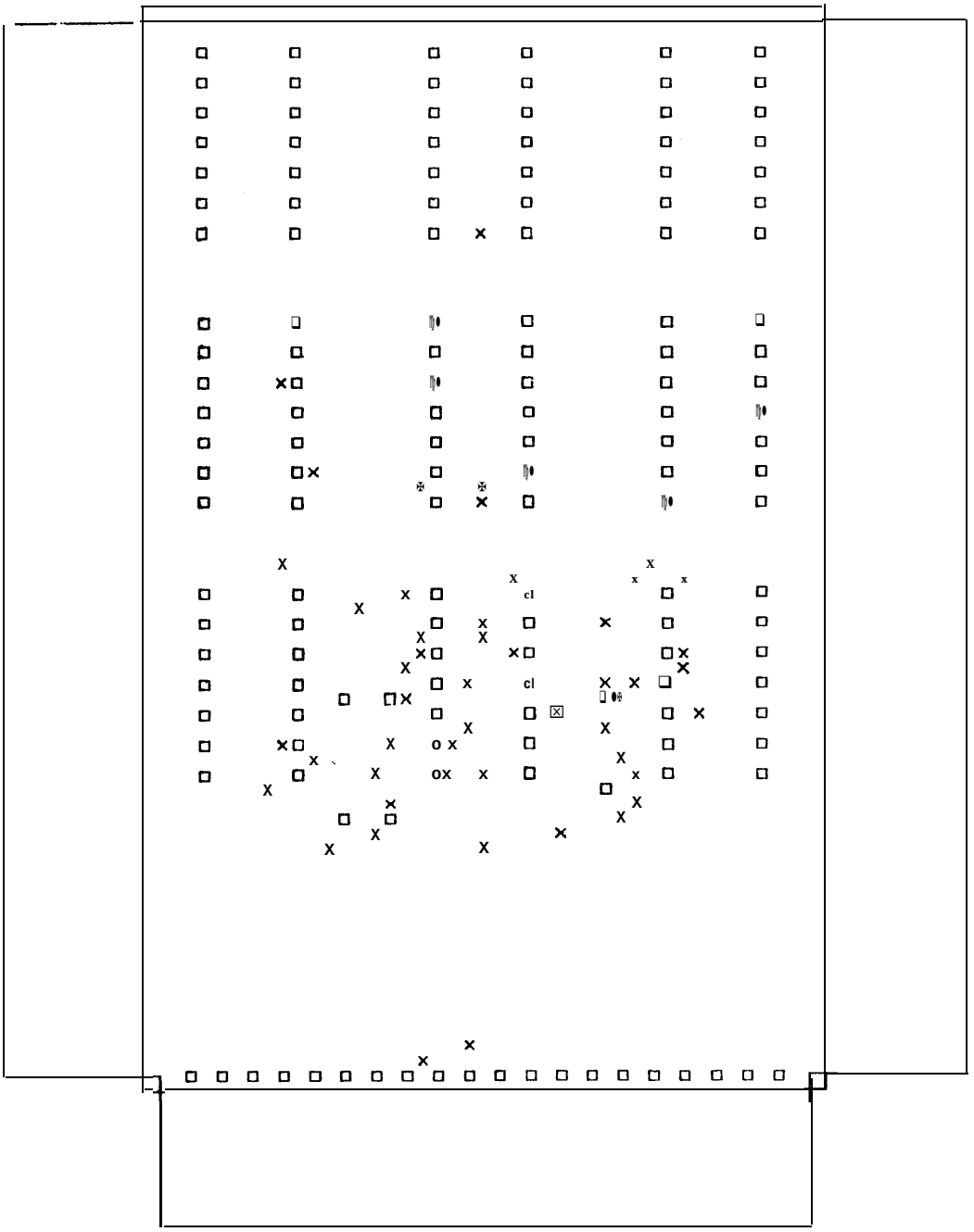
DESIGN STATUS 06/07/77
PLACEMENT CØMPLETED
WIRING STARTED
AUTØMATIC RØUTING CØMPLETED

VIA ELIMINATIØN CØMPLETED
2.0000 TØ 1 SCALE

COMPONENT PLACEMENT
COMPLETE PLACEMENT

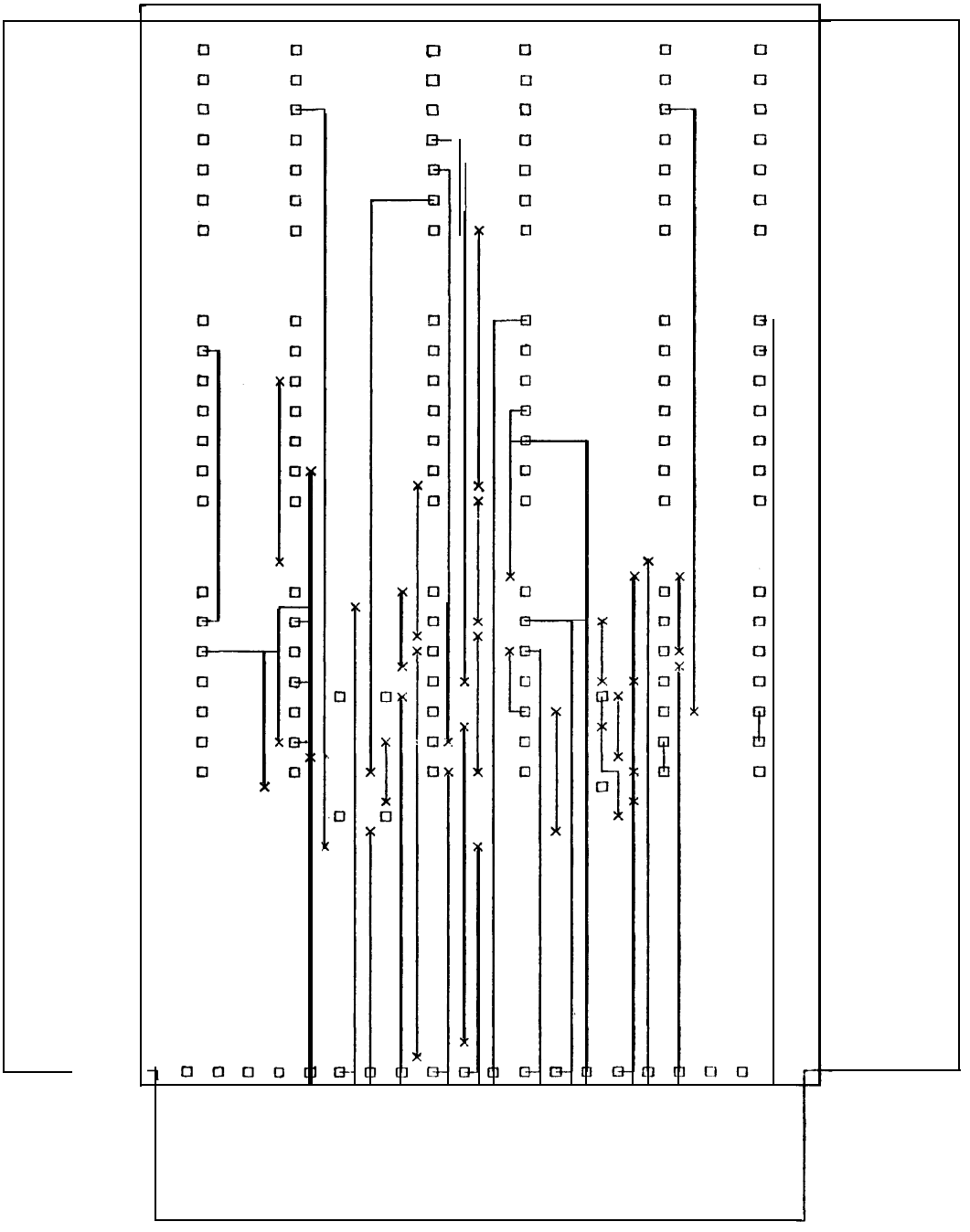


VIAS AND DRILL HØLES (BØTTØM)

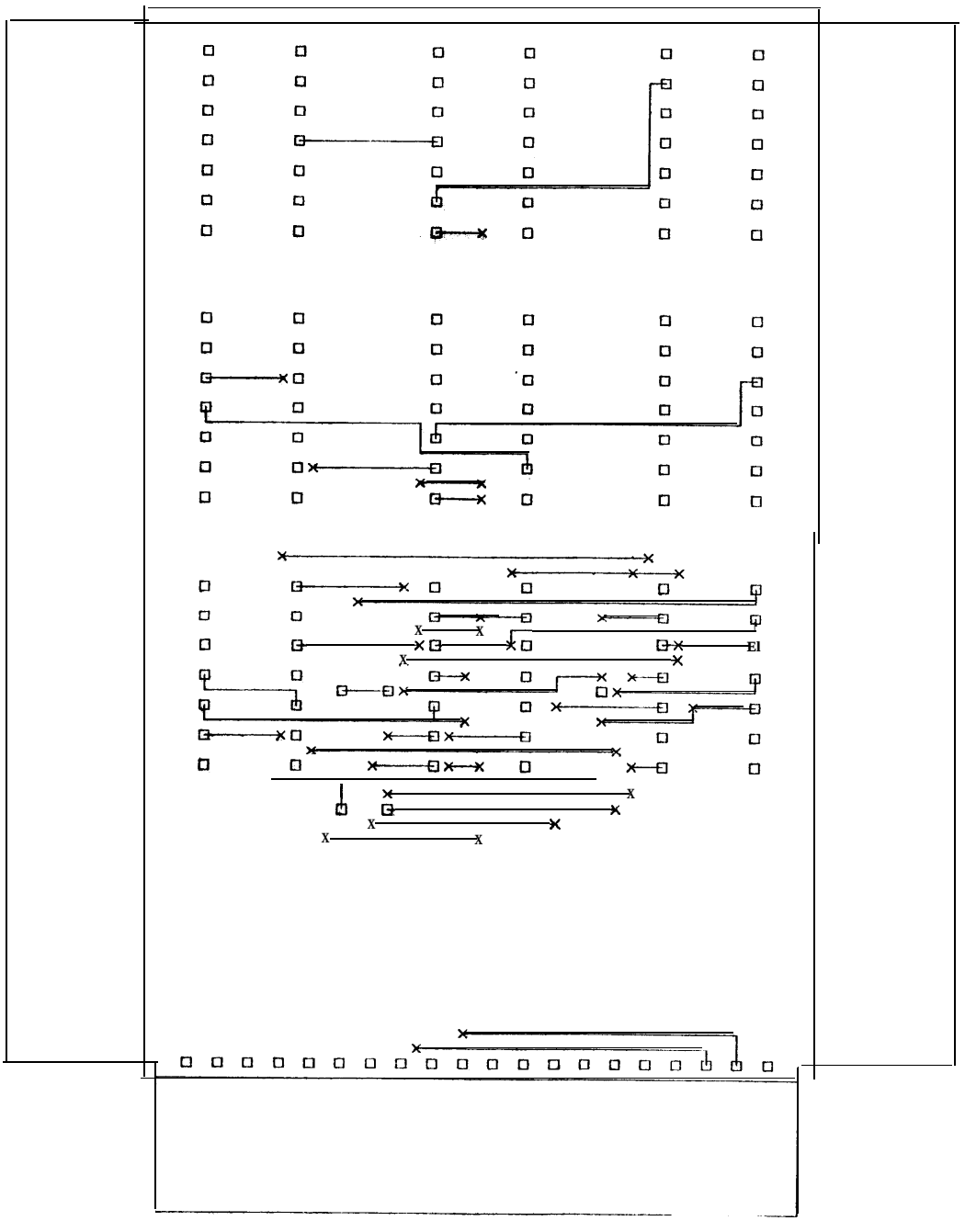


CØvnØNT SIDW RØUTING (TØP n

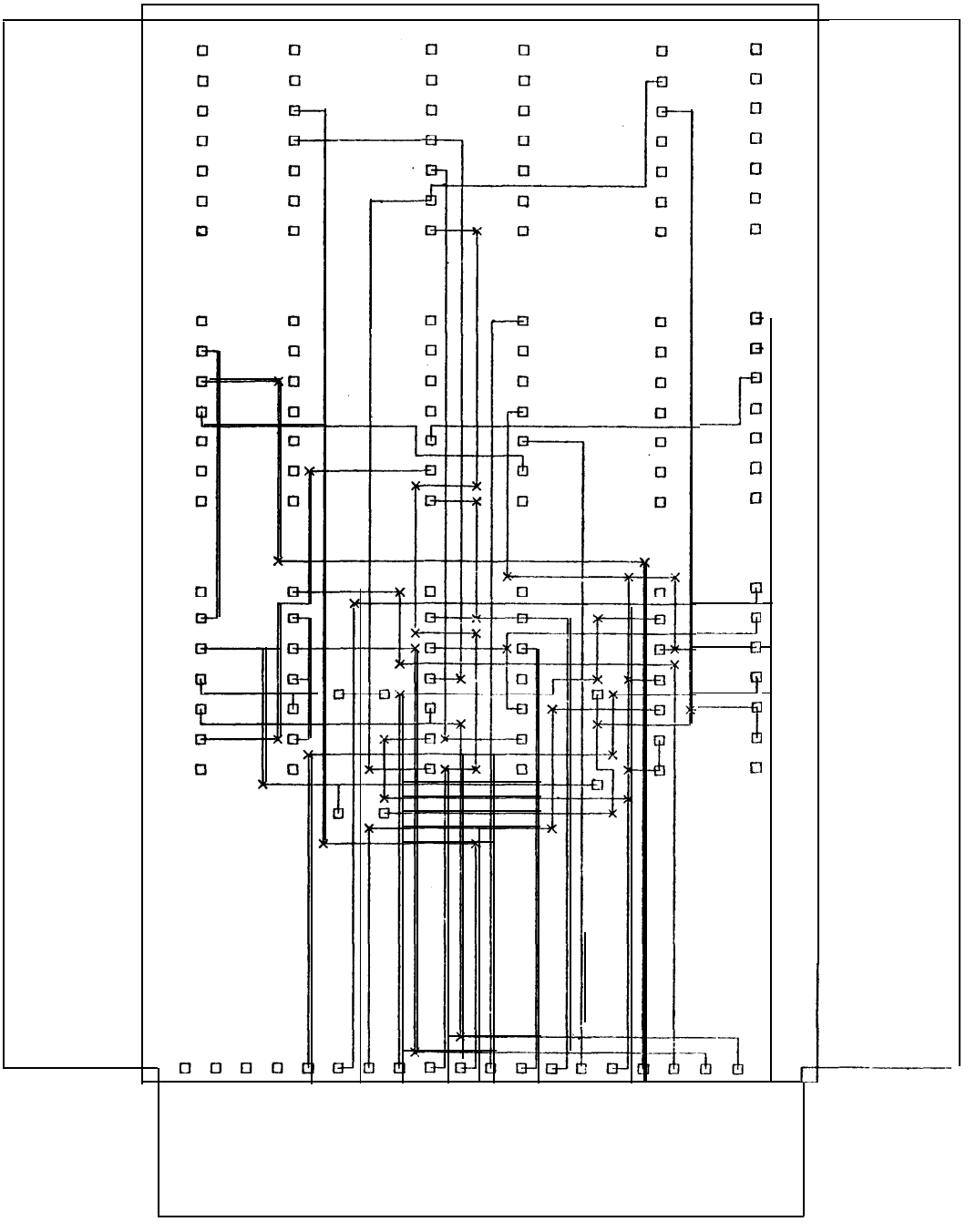
FØIL SIDE RØUTING (BØTTØM)



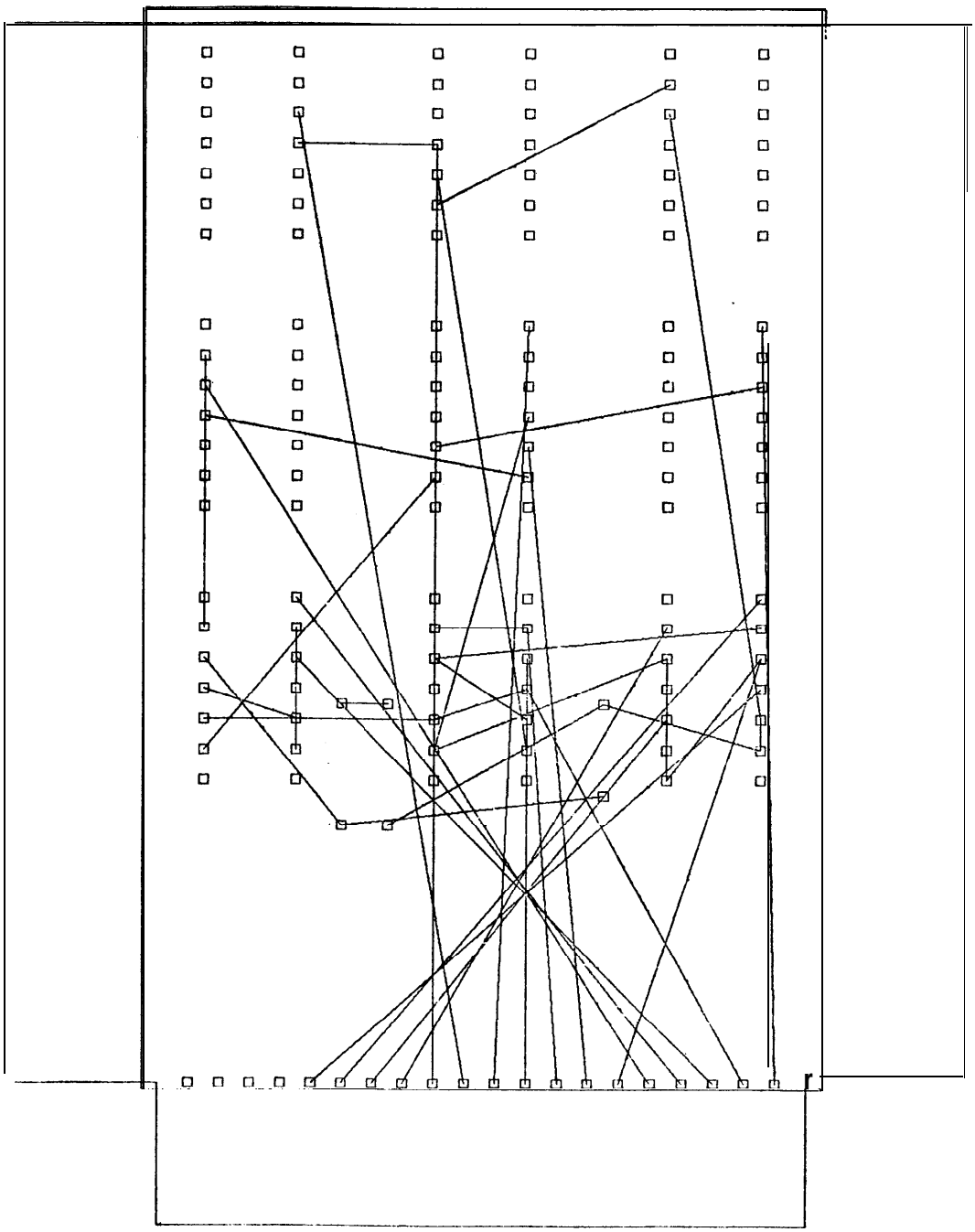
COMPONENT SIDE (BOTTOM)



COMPOSITE ROUTING (BOTTOM)



NET CHAINING (BOTTOM)



10. UTILITIES

This chapter describes some utility programs which, while not a necessary part of the design process, may be of some use.

10.1 DUMP and RESTORE

DUMP and RESTORE are a pair of programs providing the capability to compress a direct access design file into a sequential Wylbur edit format file and vice versa. There are two reasons one might wish to do this. First, a direct access file cannot be copied directly. If the designer wishes to try several alternate designs for a given board, he can easily produce several identical copies of the design file by dumping it once, then restoring it under several different names. The second, and probably more prevalent use of DUMP and RESTORE involves efficient use of disk space. A design file will generally be over 100 tracks long and be stored on a scratch disk while in use. It is a fixed size no matter what the size of the circuit. Length of the compressed file is proportional to the size of the circuit and compression factors between 5 and 10 can be obtained. This makes the file small enough to be saved on a tape or permanent disk, a necessary step if the design is not completed in one day, and something which is probably desirable in any case. Also, if there are problems with any of the SPRINT system programs, a dumped file containing the design is needed to discover the cause of the problem.

Both DUMP and RESTORE make use of a temporary file called 'TEMPPY' on volume SCFEV9. The user should not have a file by this name for any other purpose.

DUMP prints each record of the design file exactly as it was read. This output can be released to the printer if desired.

10.2 PRINT

PRINT produces a formatted printout of the design file. In other words, it knows the significance of the fields in the various records and can produce an easily readable account of the components, nets, segments, vias, etc., defined in a particular file. This kind of documentation may prove to be a useful supplement to the drawings produced by ARTWORK.

10.3 Running DUMP and PRINT

Due to the common nature of their functions, the exec files for DUMP and PRINT are almost identical. To run DUMP, type:

```
exe fro #rundump use pcb gro cg clr
```

For PRINT, type:

```
exe fro #runprint use pcb gro cg clr
```

After an introduction, each exec file will prompt for the following information:

A) <cr>=CLR ACT or SAV filename?

Type a SAVE command to save the active file under some filename or hit a <cr> to clear the active file immediately for use as a scratch area.

B) DIRECT ACCESS FILE=DFILE?

Type the name of the file you want dumped or printed.

C) FILE filename ON VOL=SCFEV9?

Give the volume the specified file is on.

D) GROUP.USER=gg.uuu?

Give the account the design file is saved under. The default is the logged on user's account. Note that even if you are dumping a file under a different account, the compressed file will be saved in 'TEMPPY' on SCFEV9 under the logged on user's account.

The DUMP or PRINT job is now in the active file. Type 'exe' to run it. The output will be held in the print hold queue when the job is completed. You can fetch it from your terminal or release it to the printer.

10.4 Running RESTORE

To run the exec file for running RESTORE, type:

exe fro #runrest use pcb gro cg clr

Here the prompts are:

A) <cr>=CLR ACT or SAV filename?

Same as DUMP

B) Source file for restore (<cr>=ACTIVE)?

Here you have three alternatives:

1) Typing a <cr> will save the active file in 'TEMPPY' to be restored.

2) Typing a Wylbur filename will cause that file to be loaded then saved in 'TEMPPY'.

3) Typing an '@' followed by an Orvyl filename will cause Orvyl to be set and that file to be loaded and stored in 'TEMPPY'.

No facilities are available for direct loading from tapes.

C) DIRECT ACCESS FILE=DFILE?

Same as DUMP

D) FILE filename ON VOL=SCFEV9?

Same as DUMP

E) GROUP.USER=gg.uuu?

Same as DUMP

F) These files are on volume volname (gg.uuu). The files on the given volume under the given account are listed. This allows the user to check for a duplicate design file name. RESTORE will abend if a duplicate filename exists.

The JCL for RESTORE is now in the active file and is run by typing 'exe'. The printed output from RESTORE is only useful if the job fails due to an invalid filename or other operating system

problem (RESTORE generates no error messages), and should be purged.

11. Future Enhancements

The following programs are in the design/implementation phase:

1) Macro expansion of the SDL output

2) Component Assignment

3) Wire wrap output

The following programs are in the planning phase:

1) Graphic input for the SDL compiler in the form of logic diagrams, drawn on a Tektronix 4013

2) Interactive editing of the completed board

3) Constructive initial placement of components

4) Gate and pin assignment (after the placement, before the routing)

5) Multilayer capability (4 signal layers initially)

6) Interface to TESTAID and TEGAS logic simulators

7) Interface to SPICE and MSINC circuit analysis programs

8) Automatic logic diagram generation subsystem