UNIVERSITY OF SOUTHERN CALIFORNIA

INFORMATION SCIENCES INSTITUTE
4676 Admiralty Way Marina del Rey, CA 90292

ISI Technical Manual
ISI/TM-93-389
November 1993

AUTO: Automatic Script Generation System

John Granacki, Ivan Hom and Tauseef Kazi

ISI/TM-93-389

November 1993

University of Southern California
Information Science Institute
4676 Admiralty Way, Marina del Rey, CA 90292

Unclassified/Unlimited

REPORT DOCUMENTATION PAGE

FORM APPROVED
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	11-17-93	Technical Manual

4. TITLE AND SUBTITLE
AUTO: Automatic Script Generation System

5. FUNDING NUMBERS
J-FBI-91-282

6. AUTHOR(S)
John Granacki
Ivan Hom
Tauseef Kazi

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
USC INFORMATION SCIENCES INSTITUTE
4676 ADMIRALTY WAY
MARINA DEL REY, CA 90292-6695

8. PERFORMING ORGANIZATON REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12A. DISTRIBUTION/AVAILABILITY STATEMENT
UNCLASSIFIED/UNLIMITED

12B. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This technical manual describes an automatic script generation system (Auto) for guiding the physical design of a printed circuit board. Auto accepts a printed circuit board design as specified in a netlist and partslist and returns a script to automatically provide all the necessary commands and file specifications required by Harris EDA's Finesse CAD system for placing and routing the printed circuit board. Auto insulates the designer from learning the details of commercial CAD systems, allows designers to modify the script for customized design entry, and performs format and completeness checking of the design files. This technical manual contains a complete tutorial/design example describing how to use the Auto system and also contains appendices describing the format of files required by the Finesse CAD system.

14. SUBJECT TERMS
Printed Circuit Board
Physical Design
Script Generation
Design Entry

15. NUMBER OF PAGES
28

16. PRICE CODE

17. SECURITY CLASSIFICTION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNLIMITED

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reoprts. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to stay within the lines to meet optical scanning requirements.

Block 1. Agency Use Only (Leave blank).

Block 2. Report Date. Full publication date including day, month,a nd year, if available (e.g. 1 jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element numbers(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract	PR - Project
G - Grant	TA - Task
PE - Program Element	WU - Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the repor.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory

Block 10. Sponsoring/Monitoring Agency Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ...; To be published in... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE - See authorities.
NASA - See Handbook NHB 2200.2.
NTIS - Leave blank.

Block 12b. Distribution Code.

DOD - Leave blank.
DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA - Leave blank.
NTIS - Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17.-19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contins classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

Table of Contents:

1.0 Overview

AUTO is a system that accepts a printed circuit design specified as a netlist and parts list and returns a script to automatically provide all the necessary commands and file specifications required by the Finesse CAD system for placing and routing the printed circuit board. The AUTO system:

- Insulates designers from learning the details of commercial software tools
- Provides design input format checking
- Provides design completeness checking
- Allows designers to enter options to modify the script for customized design entry
- Allows designers to enter options to aid in routing

Printed circuit board designers are faced with learning how to use commercial CAD tools having complex user interfaces and commands sequences to perform tasks. The learning curve for CAD tool is often quite long and steep. The AUTO program insulates the designer from the idiosyncracies of a CAD tool by allowing the designer to specify the design in formats which are simple and intuitive (Appendix B.1, B.2, and B.3 contain some examples of simple design file formats).

The AUTO system checks that designs submitted to the EZFAB service conform to the required specifications in the EZFAB user guide. The AUTO program also checks that the correct and sufficient files are available to completely specify a design. If the design is not in the correct format or not complete, the AUTO program will notify the user of the detected errors and will not produce the output script. The AUTO system also creates a log file to report the errors encountered.

Command line options can be invoked to customize the generated script for a particular design. AUTO offers options to change the default project and circuit names, alter the directory structure for the design information, and enter routing options. AUTO allows the user to specify the routing parameters either by command line options or in a default file (file named auto.ini). The AUTO system takes the parameter values from three prioritized sources: 1.) the command line, 2.) the auto.ini file, 3.) the hard coded defaults in the AUTO system.

2.0 Introduction

AUTO was developed to ease the routine tasks involved in the physical design of PCBs, and to check design formats and consistency. Interactively entering the design is a tedious, time consuming, and error prone task. A script is used to automatically load the design into the Finesse CAD tool for routing, this is especially useful when managing different design experiments. The script also provides a record of what design steps were performed for each design.

While trying to produce a design, an updated version of the Finesse software was installed with a different user interface. Since the updated version was backwards compatible, the script was used to decrease the learning curve for the new version of the Finesse software, (although some small changes had to be made to the AUTO system). In summary, the advantages of using a script include:

- Facilitates design entry and initiates routing
- Serves as an example of how to enter a design and what needs to be specified
- Makes it possible to replicate the design entry steps for placement and routing experiments
- Provides skeleton for customization by advanced users

Note: This document is intended to be a reference guide for experienced users of the Auto system. Novice users should follow the tutorial in Appendix A before reading the following sections. Experienced users can proceed to Section 3.0.

Before running AUTO the PCB design files have to be created and organized in a directory structure as described ins ection 3.0. Given the design files, AUTO goes through three major steps:

1. First it check the design files for errors (section 4.0).
2. Next it processes these files to create a script for the PCB layout tool (finesse) as described ins ection 5.0.
3. FInally it generates the script as well as output files (section 6.0) with which the designer can automatically execute the PCB layout tool.

Section 7 provides instruction on how to run AUTO. Appendix A is a tutorial that provides a walk-through of a session with AUTO. Appendix B gives the file formats and Appendix C describes the on-line help available in AUTO.

3.0 Directory Structure

The AUTO system expects the design to be organized in a directory structure for the logical and efficient processing of the input files and the creation of the output files. Otherwise, if the directory structure is not present, the AUTO system expects to find all of the design information in the current directory. The <projectname> is the unique identifier for the design. The directory <projectname/ckt/phy> is the directory in the AUTO directory structure under which all design information is stored, such as the netlist, partlist, parts-to-packages assignment, board definition file, auto initialization file, and the placement file (.plc or .trc). Refer to Figure 1 for top level view of the directory structure.

Table 1: AUTO System Directories

Directory Name (relative to directory <projectname>)	Description of Directory
prt	for all user defined parts in the Finesse system (i.e. *.prt files)
pkg	for all undefined packages in the Finesse system (i.e. *.pkg files)
ezfab	the output directory for files created by AUTO system
ckt/out	the directory where it looks for *.trc if not found in ckt/phy/, this is the directory where the automatic placement tool NAP [a]places its placement results
lib	if OMNI environment variable is not set, the file standard parts and packages are expected to be here
pad_info	contains the padstack definition scripts and supporting files for the generation of padstacks (manuf-tech, hole-table, new-padstacks, list-new-pad-scripts, auto-<new pad>, and finesse-pkg file)
ckt/phy	contains the netlist (.ntl), partlist (.prt), board description file (.bmo), part to package file (.ppt), placement information (.plc or .trc file), and auto.ini file

a. Granacki, John; Kazi, Tauseef; NAP Non-hierarchial Automatic Placement, working paper.

Directory structure with indicated files

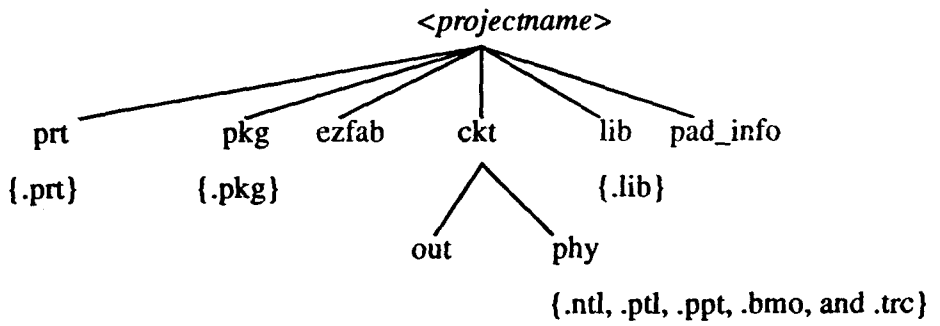

Figure 1.

3.1 Design files

The AUTO system requires the following files to operate. (Appendix B contains example format listings)

Table 2: Design Files for AUTO

Design File	Description of Design file
<project-name>.ntl	the netlist, lists nets and corresponding connections with components
<project-name>.ptl	the parts list, it relates the reference designator with the part numbers
<project-name>.ppt	assigns parts with their associated packages
<project-name>.pkg	definition of packages in Finesse format
<project-name>.prt:	definition of parts in the Finesse format

Table 2: Design Files for AUTO

Design File	Description of Design file
<project-name>.plc	information about the placement, it gives the physical location of every component, its part number and package number; so it replaces both .ptl and .ppt files
<project-name>.trc	same information as the .plc file but in the Finesse format
omni-std-parts.lib	a list of all the standard parts in the Finesse cad layout system
omni-std-pkgs.lib	a list of all the standard packages in the Finesse cad layout system
auto.ini	an initialization file for defaults for the auto program
<project-name>.bmo	a board file which describes the outline, special regions, and preplacement of connectors on the board

Other files which may be required depending on the options invoked:

Table 3: AUTO Files Needed with Options

Optional Design File (in pad_info directory)	Description of Optional Design File
manuf-tech	manufacturing design rules (e.g. trace width)
padstacks	dimensions of new user defined padstacks
list-new-pad-scripts	script names for new padstack definitions
auto-<new pad>	Finesse padstack definition script for newly defined padstacks

The following are minimal sets of input files for AUTO to work:

Table 4: Minimal Sets of Design Files

Sets of Input Files	File 1	File 2	File 3
No placement specified	<project-name>.ntl	<project-name>.ptl	<project-name>.ppt
	<project-name>.ntl	<project-name>.trc	
	<project-name>.ntl	<project-name>.plc	

Also required is an environment variable "CAD_LYOUT_SYSETM". This variable indicates where the omni-std-parts.lib and omni-std-pkgs.lib files are located. If this variable is not set, then the directory <projectname>/lib is searched to find the files. If it is not found an error message is displayed and the program exits. An example of setting the "CAD_LAYOUT_SYSTEM" UNIX environment variable is "**setenv "CAD_LAYOUT_SYSTEM=/nfs/asa2/tsacore/omni-projects/std-lib"**", where the Finesse projects are stored in the /nfs/asa2/tsacore/projects directory.

4.0 Design File Checking

1.) Before the design files are processed, the AUTO system checks:

a.) If the project name and pathname to the project directory are all lower case letters. If an error is detected it is reported and the program exits.

b.) That a minimal set of input files has been provided.

c.) That all lines in all files are less than or equal to 80 characters in length.

d.) The std-parts.lib and std-pkgs.lib files are in the lib directory.

2.) After performing the checks 1.a) through 1.d) successfully. The AUTO system reads the std-parts.lib and std-pkgs.lib files. This is done so that the AUTO system will know if it needs the part or package definitions. Then the design is loaded from the first available minimal set of input files.

3.) If the file <projectname>.ppt does not exist then it is created. This file contains a mapping from parts in the design to their corresponding packages. If the part definitions are not

available for all of the components, the AUTO system will generate a new netlist file with the packages substituted for the parts. When the packages are substituted for the parts, the ability to perform pin swapping is lost. This lose is more than offset by the decrease in effort required by the user since part definition files do not need to be created.

4.) After reading the standard libraries and design information, the AUTO system checks to see if all parts and packages used by the circuit are either standard in the Finesse CAD system or that their corresponding *.prt and *.pkg files exist. All parts and packages which are not standard to Finesse CAD system are displayed to the user in an ERROR message.

5.0 Processing Design Files

After the design files are checked:

1. If parts and packages are in the standard libraries, then the circuit is loaded into the layout system, and a script file is generated which does not have any commands to load any *.prt or *.pkg files.

2. If non-standard parts are detected and these parts have their corresponding *.prt files available, then a script will be produced that loads in the *.prt files before loading in the netlist.

3. If non-standard parts are used and their corresponding *.prt files are not present then the AUTO system checks for the following situation. If the packages are either standard in Finesse or the parts have their corresponding *.pkg files available, the user is asked if he/she wants to replace all instances of the non-standard parts with their corresponding packages. If the response is positive then all parts are renamed to there corresponding package names. If the user's response is negative the program exits and reports the listing of non-standard parts missing their corresponding *.prt files.

4. If there are non-standard parts with missing *.prt files and the packages are not standard to the Finesse CAD system nor are the corresponding *.pkg files present, the program exits.

5. All packages which are used to replace the part names (in step 3 above) must be either standard Finesse packages or they should have corresponding *.pkg files. In the later case, instructions are included in the script to load these *.pkg files in Finesse before loading the circuit.

6.0 Generated Output Files

The output files generated by the AUTO system (can be found in the ezfab directory):

Table 5: Output Files Generated by the AUTO System

Output Files	Description of Output Files
<projectname>.log	a log describing all of the events while running the AUTO system
<projectname>.script	the script to run the CAD layout system
<projectname>-ez.ntl	the net list after being processed by the auto program, (netlist with left justification for the net name)
<projectname>-ez.ptl	the parts list after being processed by the auto program, (parts list with left justification of the part name, and parts substituted with packages if required)
<projectname>-ez.plc	the placement information
<projectname>-ez.trc	the placement information in Finesse format
<projectname>-ez.ppt	file that relates parts to their associated packages
<projectname>-fs.do	the input parameters to the freestyle router
<projectname>-fs-input-line	the UNIX command line to invoke the freestyle router with command line options

7.0 Instructions to Use the Script

To use the script for the Finesse CAD Layout System, perform the following steps:

1. Go to the ezfab directory for the project.
2. Invoke Finesse at the UNIX prompt by typing "**finesse**"
3. Type "**fi *<project name>*.script**" at the Finesse system prompt, this command performs a file input for the script to read it into the Finesse system

Appendix: A.0 Tutorial and example of entering a design into the Finesse database

Note: user types bold face lettering

Appendix: A.1 Definition of terminology:

-projectname: the name of the design entered into Finesse. Ex. testbed is a project name.

-projectpathname: path from home directory through the directories ckt/phy appended with the projectname. Ex.~rajeev/projects/sacore/dflow/example/rajeev/ckt/phy/testbed.

1. Set CAD_LAYOUT_SYSTEM environment variable to path where the finesse std libraries indices are located.
2. If the design is created using Viewlogic tools: In the viewlogic directory (which contains the sch, sym and wir directories) The <projectname>.UPL file must be created by Powerview before starting, procedure for this varies depending on tools available on site. Run vb2omni on the .UPL file to create the files: <projectname>.{ont,opt,_spec.pkg } (netlist, partslist, and parts-package mapping files respectively) in the viewlogic directory. Ex. "vb2omni TESTBED.UPL".
3. Alternatively create the files: <projectname>.{ont,opt,_spec.pkg } manually.
4. Create the directory structure for AUTO by running "make-auto-dir".
5. Copy package files with the .pkg suffix from a library to the directory <projectname>/pkg.
6. Copy part files with the .prt suffix from a library to the directory <projectname>/prt.
7. Copy library files with the .lib suffix from the Finesse library to the directory <projectname>/lib.
8. Create and copy padstack script files to the directory <projectname/pad_info>.
9. Create and copy the placement file with .trc suffix to the directory <projectname>/ckt/phy.
10. Create and copy board outline file with .bmo suffix to the directory <projectname>/ckt/phy.
11. Copy viewlogic directory files <projectname>.{ont,ont,_spec.pkg files} to the directory <projectname>ckt/phy.
12. Change extensions from <projectname>.{ont->ntl, opt->ptl, and _spec.pkg->ppt}.
13. Change directory by cd ckt/phy, this is the directory where the netlist, parts list, and parts-package mapping files are copied into. Sample file formats are given in Appendix B.

14. Type "auto *<projectname>*", a sample run of the AUTO system is in Appendix A.0 and is reproduced here with explanations in **bold type.**

zelle:tsacore 26% auto snap2

Creating file "/nfs/asa2/tsacore/cad/examples/snap2/ezfab/snap2.log" ....

Reading "/nfs/asa2/tsacore/cad/examples/snap2/ckt/phy/auto.ini" ....

Sure that you don't have any custom defined padstacks [y/n]? **y, type y to specify no custom padstacks**

Reading Std. Prt. Lib. from file /nfs/asa2/tsacore/projects/std-lib/std-parts.lib ...

Reading Std. Pkgs. Lib. from file /nfs/asa2/tsacore/projects/std-lib/std-pkgs.lib ...

Reading parts from "/nfs/asa2/tsacore/cad/examples/snap2/ckt/phy/snap2.prt

Status of parts not found in STD-LIB

AUTO checking if every part used in the design has its corresponding .prt file and if the package is defined in the Finesse standard library

part	.prt file	status of .prt file	package	status of package
325PGA :	325pga.prt	not found,	325PGA	not in STD-LIB
60SIMM :	60simm.prt	not found,	60SIMM	not in STD-LIB
DIN96-M :	din96-m.prt	not found,	DIN96-M	not in STD-LIB
DIP14 :	dip14.prt	not found,	DIP14	is in STD-LIB
DIP16 :	dip16.prt	not found,	DIP16	is in STD-LIB
DIP20 :	dip20.prt	not found,	DIP20	is in STD-LIB
DIP24-3 :	dip24-3.prt	not found,	DIP24-3	not in STD-LIB
DIP28-3 :	dip28-3.prt	not found,	DIP28-3	not in STD-LIB
EISA :	eisa.prt	not found,	EISA	not in STD-LIB
HEADER14 :	header14.prt	not found,	HEADER14	not in STD-LIB

SIP10 : sip10.prt not found, SIP10 is in STD-LIB

Status of packages not found in STD-LIB

If a package doesn't exist in the omni-std-pkgs.lib file, it needs to have a corresponding .pkg file.

package file	**status of .pkg file in directory <projectname>/pkg**
325pga.pkg	exists
60simm.pkg	exists
din96-m.pkg	exists
dip24-3.pkg	exists
dip28-3.pkg	exists
eisa.pkg	exists
header14.pkg	exists

Following NON-STD parts are not defined (*.prt is missing) but *.pkg exist

If a part doesn't have a .prt file, it must have its .pkg file to be used as a substitute

PART	*PACKAGE*
325PGA	325PGA
60SIMM	60SIMM
DIN96-M	DIN96-M
DIP14	DIP14
DIP16	DIP16
DIP20	DIP20

DIP24-3 DIP24-3

DIP28-3 DIP28-3

EISA EISA

HEADER14 HEADER14

SIP10 SIP10

Replace all PARTS with PACKAGES and continue (y/n)? **y, type y to indicate that it is ok to substitute the packages for parts in the design**

(Note that if packages do not exist either the program will abort)

Replacing undefined parts with packages .....

Generating new design files (netlist, partslist, part-package mapping files) with the packages substituted in the place of the parts.

Generating "/nfs/asa2/tsacore/cad/examples/snap2/ezfab/snap2-ez.ntl" ....

Generating "/nfs/asa2/tsacore/cad/examples/snap2/ezfab/snap2-ez.ptl" ....

Generating "/nfs/asa2/tsacore/cad/examples/snap2/ezfab/snap2-ez.ppt" ....

Generating "/nfs/asa2/tsacore/cad/examples/snap2/ezfab/snap2.script" ...

Generating the Finesse script file

New Project Name= snap2

New Circuit Name= rev-1

Using board outline definition "/nfs/asa2/tsacore/cad/examples/snap2/ckt/phy/snap2.bmo"..

POWER NET = **VCC, type the name of the power net used in the design.**

Using 4 signal layers, with a VCC & GND layer **(by default)**

ROUTING GRID = 10

15. If errors ocur and auto aborts, check log file in the directory <projectname>/ezfab.
16. Type "cd ../../ezfab", this directory will contain the output files, such as the script.
17. Type "finesse" at the UNIX prompt to invoke the Finesse CAD Layout System.

18. To perform a file input to enter the script into the Finesse system, type "fi *<project name>*.script" at the Finesse prompt. This will automatically load the design files and initiate the router.
19. When Finesse has completely loaded the script, the layout system returns to the Finesse prompt. At the Finesse prompt, type "exit" to store the design and exit the Finesse software. The Finesse standard router will automatically route the design and store the results in the Finesse design database.

7.1 To view statistics of the routing job

At the UNIX prompt, follow the steps below:

1. Type "finesse st" to look at the status of the routing of the design.
2. Type "sp *<design id>*" to specify the correct project.- Type "sc rev-1" to specify which circuit to view, rev-1 is the default circuit name assigned by the AUTO system.
3. Type "lr" to list the routing statistics.
4. Type "ex" to exit the software and return to the UNIX prompt.

7.2 To view the finished routed board

At the UNIX prompt, follow the steps below:

1. After the design has been routed, to view the finished design, type "Finesse" to invoke the Finesse software.
2. Type "set proj def *<project name>*" at the Finesse prompt to set the project directory.
3. At the Finesse CKT prompt, type "get circuit rev-1" to retrieve the design.
4. At the Finesse PCB prompt, type "get pcb" to retrieve the routed board.
5. Type "esc" at the Finesse PCB prompt to save the design, but not modify the traces.
6. Type "esc" at the Finesse CKT prompt to save the design, but not modify any information.
7. Type "exit" at the Finesse prompt to leave the Finesse CAD software and return to the UNIX prompt.

Appendix: B.0 File Formats

Appendix: B.1 ".ntl" format

Note: All files contain only ASCII characters, a maximum of 80 characters per line, and more spaces may be added to the format for readability

The format for records in the ".ntl" file is:

<signal name><space><reference designator><space><pin number><space> ...

An example of the ".ntl" record format is:

```
BSYSOUTBEF      U219 2U219  4  U219  6  U219  8   U219   11
CCBOEINVP       U62  17  U192  14
CCDEN0INVREG    U54  11  U226  2   U227  11 U102  23  U69 4
CCDEN1INVREG    U54  13  U225  15  U227  6  U102  22  U69 3
CCDISWEINV      U60  16  U148  18
```

Appendix: B.2 ".ptl" format

Note: All files contain only ASCII characters, a maximum of 80 characters per line, and more spaces may be added to the format for readability.

The format for records in the ".ptl" file is:

<reference designator><space><part number>

note: An example of the ".ptl" record format is:

```
U236        PAL20L8
U195        PAL20L8
U71         PAL20L8
U58         74F374
U104        74F371
U192        74F374
U54         74FCT244
U219        74FCT244
U53         74FCT24
```

Appendix: B.3 ".ppt" format

Note: All files contain only ASCII characters, a maximum of 80 characters per line, and a semicolon is used as a line delimiter.

The format for records in the ".ppt" file is:

<part number>;<package name>;

An example of the ".ppt" record format is:

```
PAL20L8;DIP24-3;
74F374;DIP20;
74FCT244;DIP20;
74FCT24;DIP14;
```

Appendix: B.4 ".pkg" format

Note: All files contain only ASCII characters and a maximum of 80 characters per line.

An excerpt from the actual file "299grid.pkg" follows:

Finesse system standard keywords used in the .pkg file:

STRPRNTIN : used to allow automatic scrolling of the input file

$COMT and $END : used as delimiters for comment lines, any line in between needs to have a "*" asterisk character preceding it to indicate that it is part of the comment

SPN : used to indicate that a layer in the PCB is going to be defined

ALINEIN : define x and y coordinate endpoints for a line

ATXTIN : define text to appear on a PCB layer

APCPIN : used to define a pin number, x and y coordinates, and padstack type

STORE : end of .pkg file and store it into the Finesse package library

```
SFTPRNTIN GRID299
$COMT
*     DEFINE ASSEMBLY-LEVEL and SILKSCREEN-LEVEL OUTLINES
*
*        OUTLINES:
*        ASSEMBLY:    rectangle, size of TEXTOOL socket
```

```
*       SILKSCREEN: square, approximately size of PGA
*
$END
SPN LEVEL ASSEMBLY
SPN HEIGHT 150
ALINEIN   -525,   700   1925,   700;
ALINEIN   1925,   700   1925, -2100;
ALINEIN   1925,-2100   -525, -2100;
ALINEIN   -525, -2100   -525,   700;
ATXTIN REFERENCE-DES 750,  -750  C 0-DEG # V
$COMT
*    DEFINE THE SILKSCREEN OUTLINE
$END
SPN LEVEL SILKSCREEN
ALINEIN   1950,   725   1950, -2125;
ALINEIN   1950, -2125   -350, -2125;
ALINEIN   -350, -2125   -350,   725;
ALINEIN   -350,   725   -550,   725;
ALINEIN   -550,   725   -550, -1100;
ALINEIN   -550, -1100   -350, -1100;
ALINEIN   -275,  200   -200,   275;
ALINEIN   -200,   275   1775,   275;
ALINEIN   1775,   275   1775, -1775;
ALINEIN   1775, -1775   -275, -1775;
ALINEIN   -275, -1775   -275,   200;
ATXTIN SILKSCREEN-DES   0,   600  C 0-DEG # V
$COMT
*    DEFINE PAD LOCATIONS AND PADSTACKS
*
*
*    NOTE: All BAM Coordinates have been shifted to
*         place PGA pin C3 at 0,0. That is, all offsets have
been
*         adjusted so that A1, which would normally be 0, 0, is
*         located at (x, y) = (-200, 200), C3 at (0, 0), and E13
*         at (1000, -200).
$END
APCPIN    1       0,      0C1
APCPIN    2     200,   -200C8
APCPIN    3       0,    100C8
APCPIN    4     300,   -200C8
APCPIN    5     100,      0C8
APCPIN    6     300,   -100C8
APCPIN    7     200,   -100C8
...
STORE;
```

Appendix: B.5 ".prt" format

Note: All files contain only ASCII characters and a maximum of 80 characters per line.

An excerpt from the actual file "74fct244.prt" follows:

Finesse system standard keywords used in the .prt file:

START-PART-DEF : start of .prt file definition

ELEC-PIN : specify pin number and type

ELEC-PIN-PRO : specify pin number, specify signal, ground, or power pin, and drive

FUNCTION : specify which pins have the same functionality

END-PART-DEF : end of .prt file definition

```
START-PART-DEF
CORPORATE-NUMBER 'CORP.PART.74FCT244'
PART-NUMBER 74FCT244
PART-DESCRIPTION 'IC,TTL,OCT BUFFER 3-STATE OUTPUT'
PACKAGE-NAME DIP20
PIN-COUNT 20
HEADER-PROPERTY REF-DES-PREFIX,U
HEADER-PROPERTY PART-SYM-DEF,DRVR3
ELEC-PIN 1,IN
ELEC-PIN-PRO 1,PART-PIN-DRIVE,'LS-TTLIN'
...
ELEC-PIN 19,IN
ELEC-PIN-PRO 19,PART-PIN-DRIVE,'LS-TTLIN'
ELEC-PIN 20,POWER
ELEC-PIN-PRO 20,PART-POWER-NET,'POWER-NET'
FUNCTION DRVR3,5
FUNCTION-PIN DRVR3,5,9,Y
FUNCTION-PIN DRVR3,5,11,A
FUNCTION-PIN DRVR3,5,19,C
FUNCTION DRVR3,4
FUNCTION-PIN DRVR3,4,1,C
FUNCTION-PIN DRVR3,4,8,A
FUNCTION-PIN DRVR3,4,12,Y
...
END-PART-DEF
```

Appendix: B.6 ".plc" format

Note: All files contain only ASCII characters, a maximum of 80 characters per line, and the 2 header lines at the beginning of the file must be included.

The format for the ".plc" file is:

```
reference_name   part_name package_name        x      y     rt
-------------- ------------ ---------------- ------- ------ --
P1             DIN96-M      DIN96-M          000450  013800 01
P2             DIN96-M      DIN96-M          000450  008550 01
P3             DIN96-M      DIN96-M          000450  003300 01
U54            74FCT244     DIP20            002700  002800 02
U219           74FCT244     DIP20            008800  005200 02
```

In the above example: Where "reference_name" is the reference designator used in Viewdraw, "part_name" is the symbol name in Viewdraw and "package_name" is the package type, "x" is the location of the component along the south edge of the board in mils (see the conventions in Step I), "y" is the location of the component along the east edge of the board in mils, information for the side of the board the component is located on and "rt" is its rotation as defined in Table 1 ("bot" is an abbreviation for bottom.). All the data is left justified and delimited by spaces. The rotations are defined around the origin of the package footprint — when the part is oriented with its longitudinal axis aligned from north to south, the upper left corner is the origin. Rotations are counter clockwise about this point.

Table 6: Rotation Codes for ".plc" file

code	*side*	*rotation* ∠	*code*	*side*	*rotation* ∠
01	top	0 \deg	05	bot	0 \deg
02	top	90 \deg	06	bot	90 \deg
03	top	180 \deg	07	bot	180 \deg
04	top	270 \deg	07	bot	270 \deg

Appendix: B.7 ".trc" format

There is Finesse CAD tool header specific information, (which is indicated by the environment variable "CAD_SYSTEM" set to the appropriate CAD layout tool).

The essential format for the information in Finesse is:

<ref-des> <part-name> <package-name> <x-loc> & <y-loc> <rotation>

906 000001 MIL .001INCH MIL

001 000 018 ROUTER 3.3001 0.0000

003 ----------

003 REF #/NAME RECORD:

003 LEV rnum reference_name part_name package_name x & y rt

003 --- ---- ---------- ---------- ------ -- --

001 000 017 0001 U103 BAM GRID299 -09700 & 006400 02

001 000 017 0002 U179 SCB68154 DIP40 -02200 & 011500 02

001 000 017 0003 U188 EP910-30 DIP40 -02200 & 012300 02

001 000 017 0004 P1 DIN96-M DIN96-M 000450 & 013800 01

001 000 017 0005 P2 DIN96-M DIN96-M 000450 & 008550 01

001 000 017 0006 P3 DIN96-M DIN96-M 000450 & 003300 01

Appendix: B.8 std-parts.lib format

Listing of parts, Local library

PART NUMBER

1488

1489

1n4003

1n4728

1n4729

1n4730

Appendix: B.9 std-pkgs.lib format

Package listing from Local library

PACKAGE

3ag

cap20

cap21

cap22

cdip

cer1

cer15

Appendix: B.10 auto.ini format

ROUTING_GRID 42-16-42

POWER_NET VDD

Appendix: B.11 ".bmo" file format

EDIT BOARD

ADD OUTLINE -15090 -218 -15090 14218 660 14218 660 -218;

CHAMFER CORNER 100 14218 660;;

CHAMFER CORNER 100 -218 660;;

ADD CROPMARK 25 -15090 182 -14690 -218;;

ADD CROPMARK 25 -15090 13818 -14690 14218;;

ADD HOLE M3 550 5250 ;;

ADD HOLE T4 1000 -400 ;;

RESTRICT TRACE ALL RECTANGLE -15090 -218 -15030 14218 ;;

RESTRICT TRACE ALL RECTANGLE -15090 14218 660 14158 ;;

RESTRICT TRACE ALL CIRCLE -5850 7440 -5750 7440 ;;

RESTRICT TRACE ALL CIRCLE -5850 14000 -5750 14000 ;;

STORE;

Appendix: C.0 Online Help for Auto

Appendix: C. 1 Command line options

The command line for auto is as follows:

Format:

sample format: auto ~tsacore/cad/examples/bam_demo/ckt/phy/bam_demo [options]

auto <project name> [-b <Board spec. file>] [-c <Finesse circuit Name>]
[-d <project directory>] [-e] [-f] [-i] [-l] [-L] [-m] [-n]
[-o <ezfab dir absolute path>] [-P <Finesse Project Name>]
[-p <Power Net>] [-R <routing grid>] [-r]
[-s] [-t <Placement file>]

-b: Specify an Finesse board specification file.

-c: Specify a circuit name for Finesse, default= rev-1

-d: Specify a different project directory than the default.

-e: Use existing project in Finesse

-f: Use the Freestyle router, default is the standard finesse router

-i: Ignore the standard Finesse parts and packages.

-l: Calculate the number of signal layers, the default is 4 layers.

-L: Specify the number of signal layers, the default is 4 layers.

-m: Specify manufacturing technology file, else use default values.

-n: Do not generate script. Only check consistency.

-o: Specify the ezfab dir absolute path, otherwise use default path

-P: Specify a different project name for Finesse

-R: Routing grid. Overrides "auto.ini". Choices are:
10 20 25 8-9-8 50 13-12-12-13 17-16-17 8-17-17-8
100 33-17-17-33 33-34-33 38-12-12-38 40-10-10-40
40-20-40 42-16-42 42-8-8-42

Default (if not specified in auto.ini) is 42-8-8-42

-s: Specify that new padstacks are defined.

-t: Specify the Finesse Placement info. file. Default=<project>.trc

-p: Specify the name of the Power Net. This will override the one specified in the "auto.ini" file.

-r: Replace parts with packages if necessary.

Appendix: C.2 Explanation of the options defined in AUTO system

-b: Specify a new board specification to get the dimensions of the board.

-c: Specify a circuit name for the current script, default is rev-1.

-d: Specify a different project directory to put the new Finesse project.

-e: Use an existing project in the projects directory, rather than creating a new one.

-f: Use the freestyle router, generate both the <project>-fs.do and <project>-fs-inline files to run the freestyle router. The script has the command to create the <project>-write file as the design info. to the freestyle router.

-i: Ignore the standard Finesse parts and packages, this is good if you are using a different set of packages and parts.

-l: *Calculate the number of signal layers needed to route the board, the default is 4.

-L: Specify the number of signal layers needed to route the board, the default is 4.

-m: Specify there is a manuf-tech file (e.g. trace widths...), else use default values. place in the path ../../pad_info (with respect to the pathname)

-n: Do not generate a script. Only check for consistency.

-o: Specify the ezfab dir absolute path, the default location of the ezfab dir is the ezfab dir location given in the directory structure.

-P: Specify a different project name for the design, instead of the project given in the pathname.

-R: Choices for the routing grid (symmetrical for x and y direction), overrides the choice in "auto.ini" file.

-s: Specify that there are user defined padstacks. Put all of the package definitions in the ../../new-out-pkg directory, even if the package does have any user defined padstacks. Must also have the appropriate files for the padstacks definitions as indicated in Section 5.0, Basic Operation. Run the Finesse CAD system with a redirection of the script from the standard input. By indicating that new padstacks are defined, a new routing grid is calculated to specially suit the new padstack. This routing grid overrides any previously defined grid.

-t: Specify the Finesse Placement info. file. The default is <project>.trc.

-p: Specify the name of the power net. This will override the one specified in the "auto.ini" file, which is usually defined to be VDD.

-r: Replace the parts with packages if not all of the parts definitions are available. This is possible because the router is only concerned with the outline of the package and not the actual operation of the part.

note: * is currently being developed