Bartels AutoEngineer®
User Manual

Bartels AutoEngineer® User Manual
Published by: Bartels System GmbH, Munich
Last printing: November 2013

Preface

The Bartels AutoEngineer® - User Manual describes in detail how to use the **Bartels AutoEngineer** CAE/CAD/CAM design system. The following main topics are covered by this manual:

- Introduction: System Architecture, general Operating Instructions, Design Database
- Circuit Design (CAE), **Schematic Editor**
- Netlist Processing, Forward and Backward Annotation
- PCB Design and manufacturing data processing (CAD/CAM), **Layout Editor**, **Autoplacement**, Autorouting, **CAM Processor**, **CAM View**
- IC/ASIC Design, **Chipeditor** for interactive IC mask layout, **Cell Placer** and **Cell Router** for place & route, GDS and CIF import and export
- Neural Rule System
- Utility Programs

The reader of this documentation should be familiar with the use of his operating system and a text editor for generating ASCII files on his system.

Kindly note the Copyright before making use of the information provided with this documentation or applying the herein described products. The reader should also be familiar with the Notations and Conventions used throughout this documentation.

Organization of this Documentation

Chapter 1	describes the **Bartels AutoEngineer** system architecture, provides general operating instructions and introduces the design database.
Chapter 2	describes in detail how to use the **Schematic Editor** for creating SCM library symbols and designing circuits.
Chapter 3	describes how to use the **Packager** program module and the Backannotation function for performing forward and backward annotation of net list data.
Chapter 4	describes in detail how to use the **Layout Editor**, the **Autoplacement** functions and the **Autorouter** for creating layout library symbols and designing PCB layouts, and how to use the **CAM Processor** and **CAM View** modules for creating and processing CAM output and manufacturing data for the PCB production.
Chapter 5	describes how to use the **Chip Editor**, the **Cell Placer**, and the **Cell Router** for the interactive and/or automatic design of IC mask layouts.
Chapter 6	describes the **Bartels Neural Rule System**, i.e., how to define neural rules with the **Bartels Rule Specification Language**, how to compile rule specification source files using the **Bartels Rule System Compiler** and how to apply neural rules throughout the **Bartels AutoEngineer** design process.
Chapter 7	describes the utility programs of the **Bartels AutoEngineer** software.

Related Documentation

The Bartels AutoEngineer® - Installation Guide describes the **Bartels AutoEngineer** configurations and system requirements and provides detailed **Bartels AutoEngineer** installation instructions for all supported hardware and software platforms.

The Bartels AutoEngineer® - Symbol and Part Libraries documentation contains detailed information about the symbol and part libraries provided with the **Bartels AutoEngineer** CAE/CAD/CAM design system.

The Bartels User Language - Programmer's Guide describes how to use the **Bartels User Language** in **Bartels AutoEngineer**, i.e., how it is integrated to the **Bartels AutoEngineer** EDA system and how it can be applied. The following main topics are covered by this manual:

- basic concepts and description of the **Bartels User Language**
- the **Bartels User Language** programming system: **User Language Compiler** and **User Language Interpreter**
- **User Language** example source code listings, short information on the **User Language** programs supplied with the **Bartels AutoEngineer**
- special data types defined for accessing **Bartels AutoEngineer** design data
- **User Language** system function reference

Problems, Questions, Suggestions

We appreciate comments from the people who use our system. In particular we are thankful for suggestions on how to improve the **Bartels AutoEngineer** and/or the **Bartels User Language** by introducing new or improving existing functions. Please do not hesitate to contact Bartels Support if you have questions or problems related to the use of the **Bartels AutoEngineer**. Check the Bartels Website at http://www.bartels.de for our address.

Documentation Notations

The reader should be familiar with the following notations used throughout the **Bartels AutoEngineer** documentation:

Mouse	pointing device (mouse, trackball, etc.) to be used for moving the menu and graphic cursors as well as for selecting functions
Info Field	menu field on the right top of the screen for displaying system status messages
Main Menu	function menu permanently available in the upper right screen area used for selecting a subordinate function menu
Menu	function menu in the lower right screen area currently selected from main menu
Submenu	subordinate function menu in the lower right screen area activated intermediately whilst using another menu function
Graphic Workarea	workarea for graphic interaction in the upper left screen area
Status Line	lower left screen line used for displaying system status messages and for performing interactive user queries
Menu Cursor	rectangle-shaped cursor for selecting a menu function
Graphic Cursor	cross-shaped cursor in the graphic workarea (crosshairs)
Menu Prompt	user query in the status line
Popup Menu	menu optionally displayed on top of the graphic workarea for selecting function-specific objects or for activating menu-specific functions
Button	selectable popup menu entry for choosing a certain menu element or for activating a menu-specific function
Select Function	move menu cursor to a function of the currently active function menu
Activate	hitting the mouse button
Pick	select an object to be manipulated using the graphic cursor
Place	move an element to a certain position in the graphic workarea
Select	select an element to be manipulated or a function to be executed by pressing the mouse button
Confirm	confirm the execution of a function verified by user query

The following acronyms are used throughout the **Bartels AutoEngineer** documentation:

BAE	acronym for identifying the **Bartels AutoEngineer** EDA software
BAEICD	acronym for the **Bartels AutoEngineer** IC/ASIC design system optionally included with workstation-based BAE configurations
SCM	acronym for the **Schematic Editor** program module of the **Bartels AutoEngineer** circuit design system
GED	acronym for the graphical PCB **Layout Editor** program module of the **Bartels AutoEngineer** PCB design system
AP	acronym for the **Autoplacement** program module of the **Bartels AutoEngineer** PCB design system
AR	acronym for the **Autorouter** program module of the **Bartels AutoEngineer** PCB design system
NAR	acronym for the advanced **Neural Autorouter** program module of the **Bartels AutoEngineer** PCB design system
CAM	acronym for the **CAM Processor** program module of the **Bartels AutoEngineer** PCB design system
CV	acronym for the **CAM View** program module of the **Bartels AutoEngineer** PCB design system
CED	acronym for the **Chip Editor** program module of the **Bartels AutoEngineer** IC/ASIC design system
CP	acronym for the **Cell Placement** program module of the **Bartels AutoEngineer** IC/ASIC design system
CR	acronym for the **Cell Router** program module of the **Bartels AutoEngineer** IC/ASIC design system
UL	acronym for the **Bartels User Language** programming language
ULC	acronym for the **Bartels User Language Compiler**
ULI	acronym for the **Bartels User Language Interpreter**

Documentation Conventions

Unless otherwise mentioned, the following symbolic conventions are used throughout the **Bartels AutoEngineer** documentation:

`Lineprint`	Lineprint font represents text output generated by the system.
Boldface	Boldfaced words or characters in format or command descriptions represent topic definitions or syntactic terminals, i.e., commands or keywords to be inserted directly.
Emphasize	Emphasized text is used for optical accentuation.
" "	Double quotes denote names and/or path names or enclose characters and/or character sequences directly to be inserted.
[ ]	Square brackets enclose optional items in format or command descriptions.
{ }	Braces enclose a list of items in format or command description, from which one has to be chosen.
|	A vertical bar separates items in a list of choices.
< >	Angle brackets enclose the logical name of a key on the keyboard. In format or command descriptions, angle brackets enclose values to be supplied.
>	Boldfaced greater signs in lineprint font are used for denoting prompts on operating system level.
...	Horizontal ellipsis points indicate either optional repetition of the preceding element in format or command descriptions or absence of irrelevant parts of a figure or example.
:	Vertical ellipsis points indicate absence of irrelevant parts of a figure, an example or a format or command description.
■■■	Any Mouse Button (MB)
■□□	Left Mouse Button (LMB)
□■□	Middle Mouse Button (MMB)
□□■	Right Mouse Button (RMB)
↵	Keyboard (input) - Return/Enter key (CR)
a b ...	Keyboard (input) - standard key(s)
F1 F2 ...	Keyboard (input) - function key(s)
`filename`	File or directory path name.
`keyword`	Topic definitions or syntactic terminals, i.e., commands or keywords to be inserted directly.
`message`	BAE/system status or error message display.
Menu	**Bartels AutoEngineer** menu.
Menu Function	**Bartels AutoEngineer** menu function.
Menu Option	**Bartels AutoEngineer** menu option.
Button	**Bartels AutoEngineer** (popup) menu button.
ul.ulh	(Hypertext link to) **Bartels User Language** include file.
ULPROG	(Hypertext link to) **Bartels User Language** program description.
ul.ulc	(Hypertext link to) **Bartels User Language** program source file.
ul_function	(Hypertext link to) **Bartels User Language** system function description.
UL_INDEX	(Hypertext link to) **Bartels User Language** index type description.

UTILPROG	(Hypertext link to) **Bartels AutoEngineer** utility program description.
new feature	New features which are made available with regular (weekly) software updates/builds are highlighted in the online documentation.

The character sequences mentioned above may regain original meaning when used in programming languages, interpreter languages, specification languages, syntax description languages, etc.

Contents

Tables

Figures

Chapter 1 Introduction

This chapter describes the **Bartels AutoEngineer** system architecture, provides general operating instructions and introduces the design database.

Contents

Figures

1.1 Product Information

Bartels AutoEngineer (BAE) is a fully integrated EDA software system with powerful CAE/CAD/CAM program modules for circuit design, PCB (printed circuit board) layout and IC/ASIC design. The system is based on the famous **Bartels AutoEngineer** which has been incorporated in most leading PCB layout systems throughout the world, setting new industrial standards of autorouting success and dramatically reducing the requirement for manual routing.

1.1.1 BAE Software Configurations

The following **Bartels AutoEngineer** software configurations are available:

- **Bartels AutoEngineer Schematics**
- **Bartels AutoEngineer Light**
- **Bartels AutoEngineer Economy**
- **Bartels AutoEngineer Professional**
- **Bartels AutoEngineer HighEnd**
- **Bartels AutoEngineer IC Design**
- **Bartels AutoEngineer FabView**

All BAE software configurations are provided with the same user interfaces and support different languages (English, German, etc.). **Bartels AutoEngineer** features binary-compatible design data management on different host platforms, i.e., BAE design data can be transferred "as-is" between all supported hardware and/or operating system platforms (Windows, Linux/Unix, DOS, etc.).

Bartels AutoEngineer Professional

Bartels AutoEngineer Professional is the basic BAE software configuration described in this manual. **BAE Professional** is available for PCs with Windows, Linux or DOS operating systems. The following components are included with the **BAE Professional** software:

- **Schematic Editor** with hierarchical circuit design support
- Forward/Backward Annotation
- PCB Layout system including **Layout Editor**, **Autoplacement**, automatic copper fill, **Bartels AutoEngineer**, **Bartels Autorouter®**
- **CAM Processor** and **CAM View** with Gerber viewer
- integrated, object-oriented database system (DDB, Design DataBase)
- in-built **Neural Rule System**
- **Bartels User Language Compiler**, **User Language** programs with source code
- utility programs for database management, foreign net list and design data import, etc.
- extensive part libraries for SCM and PCB layout
- design data examples
- Documentation (Bartels AutoEngineer User Manual, Bartels User Language Programmer's Guide)

BAE Schematics, the **Schematic Editor** of **BAE Professional** is *freely available* and can be operated in stand-alone mode. Demo software configurations of **BAE Professional** (**BAE Demo**, fully-featured except for data output) are available free of charge for test and evaluation purposes.

Bartels AutoEngineer Light

Bartels AutoEngineer Light is available for PCs with Windows, Linux or DOS operating systems. **BAE Light** is a shareware price-level BAE configuration for educational purposes and/or semi-professional users. **BAE Light** provides full **BAE Professional** functionality, however, with the following restrictions:

- PCB layout size limited to 180mm*120mm
- max. 2 signal layers for manual routing and Autorouter
- no power layer support
- layouts from **BAE Professional**, **BAE Economy** and **BAE HighEnd** only loadable if **BAE Light** restrictions are not violated

Bartels AutoEngineer Economy

Bartels AutoEngineer Economy (formerly known as **Bartels AutoEngineer Educate/Entry**) is available for PCs with Windows, Linux or DOS operating systems. **BAE Economy** is a low-price BAE configuration for educational purposes and/or small business users. **BAE Economy** has full **BAE Professional** functionality, however, with the following limitations:

- PCB layout size limited to 350mm*200mm
- max. 4 signal layers simultaneously routable by **Autorouter** (however, like in **BAE Professional**, 100 signal layers for manual routing and support for 12 power layers in both manual routing and **Autorouter**)
- layouts from **BAE Professional** and **BAE HighEnd** only loadable if **BAE Economy** restrictions are not violated

Bartels AutoEngineer HighEnd

Bartels AutoEngineer HighEnd is available on workstations as well as on Windows and Linux PC platforms. **BAE HighEnd** utilizes special operating system characteristics (multi-tasking, multi-windowing, virtual memory management, etc.) to implement advanced features and functions such as:

- HighSpeed kernel
- in-built messaging system to support advanced communication between different BAE modules
- in-built multi-tasking to support multiple project views
- global net highlight, cross-probing
- place layout parts according to schematic plan
- selective layout short-circuit display
- extremely fast Mincon airline calculation
- DRC with multi-processor system support
- layer stack setup according to trace impedance
- layer-specific clearance paramaters for DRC and copper fill functions
- internal data structures optimized for high **Autorouter** performance
- advanced **Neural Rule System** features
- rule-driven **Neural Autorouter**
- nettype-specific routing area definitions
- net-specific maximum via count settings
- net-specific maximum connection length settings
- CAM output to suppress unconnected inner layer pads

BAE HighEnd is data-compatible to **BAE Professional** in both directions; requested data transformations are automatically applied during element load procedures.

Bartels AutoEngineer IC Design

BAE HighEnd can be upgraded to a fully featured ASIC design system. **Bartels AutoEngineer IC Design (BAEICD)** is a complete CAD/CAM system for the physical design of integrated circuits (gate arrays, standard cells, custom ICs and/or ASICs). **BAEICD** consists of a series of system components such as **IC Mask Editor**, **IC Autoplacement**, **IC Autorouter** and **IC DRC** (Design Rule Check). GDS-II and CIF standard interfaces are provided for converting foreign data and/or producing CAM output (mask data, bonding data, etc.). The **BAEICD** CAM tools include a module for displaying CIF data in order to perform visual CAM output checks. Net list data is usually transferred by the BAE **Packager** after defining the circuitry with **BAE Schematics**, which provides features for hierarchical circuit design. Alternative solutions for importing foreign/third-party netlist data/formats can be provided on request.

Bartels AutoEngineer FabView

Bartels AutoEngineer FabView is a low-cost PCB layout viewer with manufacturing data output functions. **BAE FabView** is intended for PCB manufacturing departments and service providers who only have to produce manufacturing data and print/plot outputs but don't have to edit layouts. **BAE FabView** can be used together with **BAE Professional** and/or **BAE HighEnd**. **BAE FabView** provides the same functionality, however, the functions for saving layout design changes to BAE project files are deactivated.

1.1.2 BAE System Components

Bartels AutoEngineer basically consists of a **Schematic Editor**, a **Layout Editor** with **Autoplacement** and **Autorouter** and a **CAM Processor** with a supplemental module for viewing and processing CAM data. A **Packager** program module and a Backannotation function are provided for automatic forward and backward annotation of net list data from the schematics to the layout and vice versa. BAE supports all phases of modern computer-aided PCB design including schematic capture, physical PCB layout featuring powerful **Autoplacement** and Autorouting facilities and, finally, generating all of the required manufacturing data and documentation. See figure 1-1 for a design flow diagram of the **Bartels AutoEngineer**.

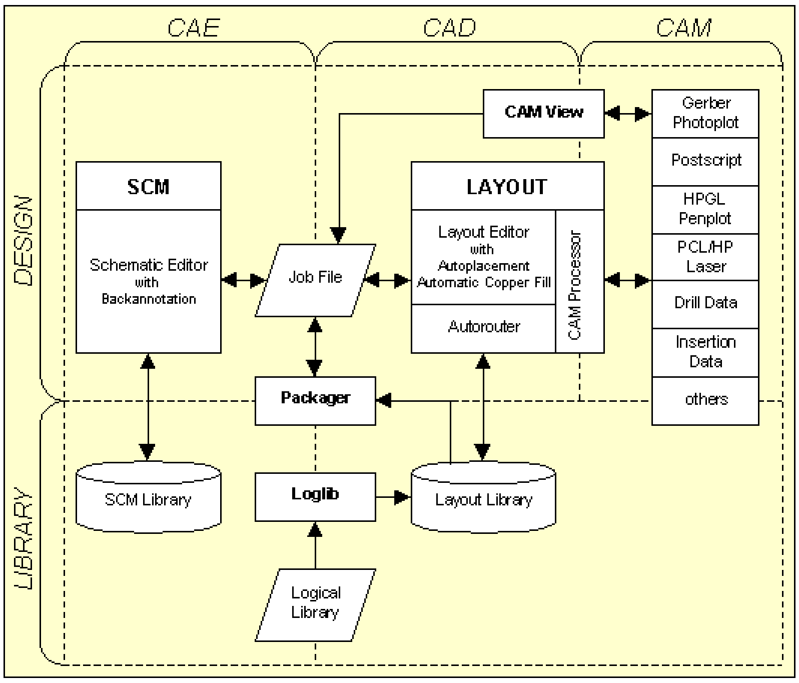


Figure 1-1: Bartels AutoEngineer System Flow Diagram

1.1.3 BAE Database Structure

CAD systems must process and/or manage huge amounts of design data. Therefore, the internal database structure of a CAx system is an important criteria for estimating the power of the system. The **Bartels AutoEngineer** is based on especially designed, homogeneous, object-oriented, hierarchical database structures featuring optimized B-tree search algorithms for fast database object access. Variable keyword lengths and generic data structures guarantee that neither redundancies nor system limits are imposed by the software. The system is only limited by the available amount of main memory. I.e., the software imposes no restrictions with regard to the number of symbols and/or sheets of a schematic circuit, number of parts placed on a layout board, number of pins defined on a part, number of traces routed on a layout, etc.

All BAE CAD data is organized and managed in a fully integrated database with a common binary data format for all supported hardware platforms. Each CAD object is dynamically constructed of its individual components. Library elements from lower database hierarchy levels (e.g., parts on a PCB) are copied (with all sub-elements) from the currently selected library to the currently processed design file when they are first requested, e.g., for placement on the upper hierarchy level. This results in a job-specific library in the design file. Subsequent requests for the same element will then refer to the element already copied to the current design file, i.e., the job-specific library elements are accessed with highest priority. This concept prevents any library and/or design file modification from unintentionally affecting another design and/or library file. **AutoEngineer** objects can be composed of sub-elements from different libraries. A BAE library file can be build up by accessing one or more other libraries. Even a particular design file can be used as library for another design file. Library definitions such as pin or part symbols can be modified at any stage. Any such modification is automatically reflected up the library hierarchy.

All data relating to one particular design is kept in a single file thus avoiding the confusion of multiple file structures. This so-called design database (DDB) file contains different element types relating to schematic, layout, net lists, part lists, etc. all of which go together to make a complete design. This concept supports highly efficient data management features for the creation, modification, backup, update and redesign of design files and library files alike.

1.1.4 BAE Data Types and Application Concepts

The BAE system is comfortable to use with simple instructions aided by convenient user control over menu and mouse. The menus and manuals are available in different languages (German, English, etc.). The standard BAE user interface is the same on all hardware platforms. With the BAE windows versions the user can optionally use the windows-like BAE user interface with pull-down menus. It is easy to learn how to work with the system since the functions for creating and modifying objects are quite similar for all object types of the different library hierarchy levels. The BAE software also provides most powerful tools for customizing the BAE user interface with menu assignments and online key binding for calling user-defined functions.

The floating point database used throughout the **Bartels AutoEngineer** has many advantages. Firstly it means that there are virtually no grid limits to designs. Any pad shape, component shape, track width, board size or shape, copper area size or shape, etc. can easily be generated. Secondly it means that online design rule checking (DRC) of copper and track clearances and void areas run at all time during layout providing instant indication of errors. Floating point calculation also means that all geometry is stored extremely accurately. The system can either accept metric or imperial numeric input which is automatically converted to floating point information.

A powerful Undo/Redo function is provided to ensure data security and to allow for comfortable evaluation of different design modifications. Real-time graphic interaction (re-display at zoom and pan, moving graphic elements, etc.) is self-evident, as well as graphical built-in features such as definable color tables or display, query and highlight of elements, connections and errors, etc.

A pool is used for managing data in main memory. This also controls basic features such as the display functions or Undo/Redo. Complex operations such as moving parts are extremely fast since all data is converted in real-time from internal hierarchical representation to vector/polygon-oriented display mode. Fast workspace and clipping window checks are performed on graphical elements to guarantee high-speed re-display. These functions are implemented with internal programming interfaces to support different graphic controllers. Large objects like wiring paths and areas are stored in compressed internal format to avoid redundant memory usage. An entire trace path located on a particular layer occupies just one pool element, as well as e.g., a 84 pin PGA on subsequent placement (only the connectivity is stored separately, the geometry refers to the symbol instantiated with the first placement request). The maximum number of pool elements on 32 bit computer systems is 2^31=2,147,483,648. I.e., the pool on non-virtual systems is limited rather by available memory and/or pointer address space than by the complexity of the design.

All coordinate and angle values are stored in internal 32 bit IEEE floating point format. Online operations requiring higher precision are performed with double (64 bit) precision. Each element can be placed with arbitrary coordinates at any rotation angle. Skillful linkage to the hierarchical database system avoids performance disadvantage even on slow floating point processors. References to named elements (such as pins or parts) are always stored as strings. Identical strings are linked together to one file entry. I.e., naming of e.g., plug or PGA pins such as **c32** is easily possible without creating redundancies.

1.1.5 Exchanging Data with other Systems

Bartels AutoEngineer provides tools for importing different ASCII net list formats, placement data and routing data (via Gerber format). Functions for generating insertion data, drill data, Gerber photoplots, Postscript output, HP-GL pen plots and HP-Laser (PCL) output are integrated to the **CAM Processor** of the **Bartels AutoEngineer**. A special programming language (**Bartels User Language**) can be applied for implementing user-specific programs for importing and/or exporting almost any data such as part lists, net lists, geometry data, drill data, insertion data, milling data, etc. in freely definable formats.

1.2 Operating the Bartels AutoEngineer

1.2.1 BAE Startup and BAE User Interface

BAE Startup

After successful installation with correct path setting the **Bartels AutoEngineer** can be started from any directory, but it is recommended to start BAE from the directory where the projects and/or the design files should be generated (this considerably simplifies job file access).

You can start the **Bartels AutoEngineer** from your projects directory by typing

```
> bae ↵
```

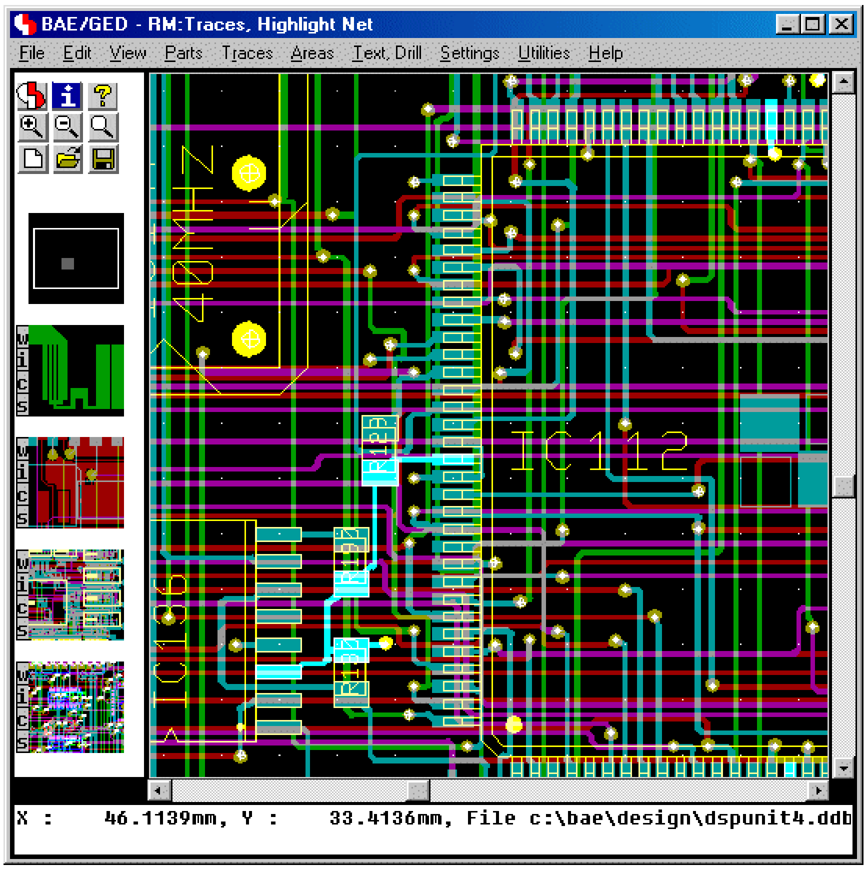
to the operating system prompt (for testing the software you can use the BAEJOBS directory created during the software installation).

Within Windows, the **Bartels AutoEngineer** can be started by selecting the **`bae.exe`** file using the Run function from the Program Manager Files menu. Windows- and X11/Motif-based operating systems also allow for application startup by clicking the icon of the corresponding program file. Startup icons also allow for the definition of the directory to be entered on default when starting the corresponding application, i.e., BAE can be configured to be started from the BAE jobs directory or from any user-specific BAE project directory. A reference of the BAE startup icon can be included with the operating system launchpad and/or Start menu to provide an even more convenient method of starting up the **Bartels AutoEngineer**. The default BAE setup links the **`.ddb`** file name extension to the BAE application in such a way that the **Layout Editor** automatically starts and loads the default project layout when double-clicking a **`.ddb`** file. Right-clicking the **`.ddb`** file activates a context menu which provides the choice of starting either the **Layout Editor** with the default layout or the **Schematic Editor** with the last modified SCM sheet. Please consult also your operating system manual for more details on how to configure applications for startup.

BAE User Interface with Pulldown Menus (Standard)

On default, the Windows and Motif versions of the **Bartels AutoEngineer** are configured with pull-down menus. This user interface is vertically organized and consists of the main menu bar on top, the graphic work area in the middle and an info line followed by the status and input line at the bottom. Function menus are activated in pull-down mode when selecting the corresponding main menu item. Under Windows, the Toolbar submenu from the View menu provides the Tree View Menu function for activating an Explorer-style tree view function menu which can be displayed either to the left or to the right of the graphic workarea.

After starting the BAE, the Bartels company logo and/or a copyright note is displayed in the graphic workarea and the status line displays the program version and/or the user names.



```
X :     46.1139mm, Y :     33.4136mm, File c:\bae\design\dspunit4.ddb
```

Figure 1-2: BAE Pulldown Menu User Interface

If the **`Please check your User Authorization!`** message appears, then check for correct software authorization (i.e., hardlock key and appropriate license file installation; see the Bartels AutoEngineer® Installation Guide for more details).

BAE Side Menu User Interface

The **Bartels AutoEngineer** can optionally be operated with a side menu user interface. The graphic display of the side menu user interface is divided into different areas as shown in the figure below. It consists of the graphic workarea, an input and/or status line below the graphic workarea and an info field and function menus on the right side.

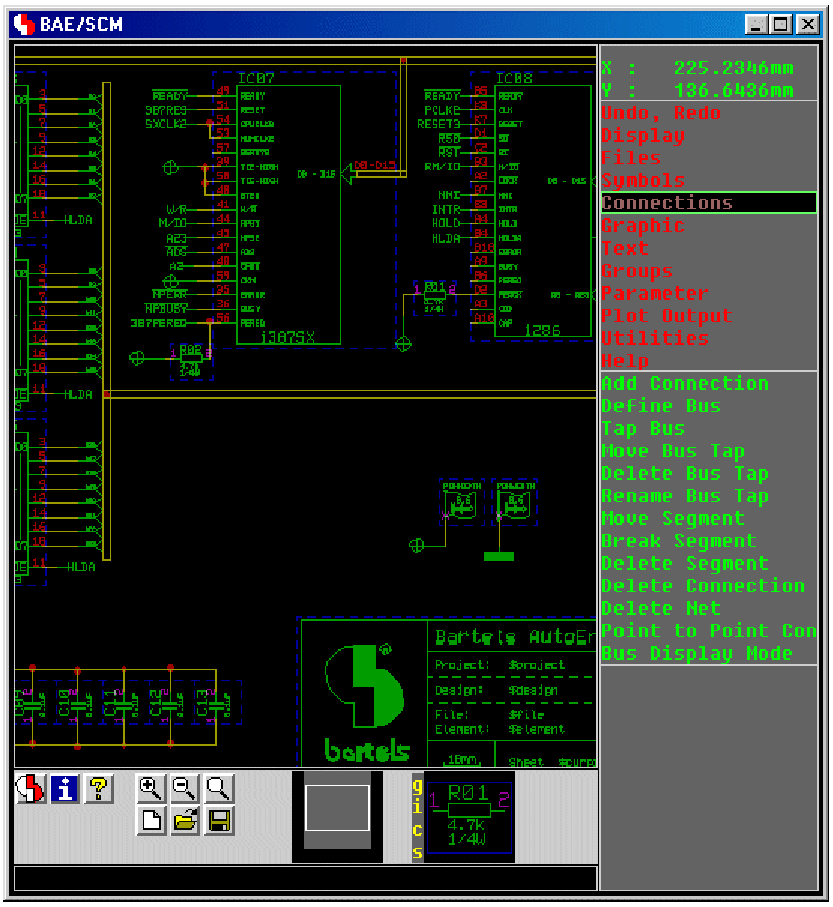


Figure 1-3: BAE Side Menu User Interface

Under Windows and Motif, the BAE user interface can be selected and/or configured through the Setup dialog from the BAE main menu. Alternatively, the **BSETUP** can be used to select the BAE user interface (see chapter 7.2 for more details).

If the BAE DOS version fails to load the graphic display, then check for correct BAE graphic device driver installation (see the Bartels AutoEngineer® Installation Guide for more details).

1.2.2 Function Selection

BAE Windows/Motif User Interface

The pull-down user interfaces of the BAE Windows/Motif versions provide a permanently displayed main menu bar at the top. Main menu selections usually activate a more particular pull-down function menu. A particular menu function is selected by moving the mouse to the corresponding menu item and pressing the left mouse button. The menu item strings usually contain underlined characters to denote hotkeys for fast activation of the corresponding menu and/or function. I.e., main menu items can also be activated and/or selected by pressing the denoted character together with the `Alt` key, and pull-down menu functions can be activated and/or selected by simply pressing the denoted hotkey. The middle mouse button provides instant access to the View menu, i.e., pressing the middle mouse button is possible at any time and from any other function and provides fast access to frequently required display functions such as zoom and pan, changing color setup, setting the input and/or display grid, etc. Pressing the middle mouse button will also cause a status line display of the current file and element name if no other menu function is currently active. The middle mouse button interaction can also be simulated by simultaneously holding down both the left and the right mouse button. I.e., it is possible to activate the online display menu even if only a two-button mouse is installed and/or configured. Any currently active menu function can be immediately aborted and/or canceled by pressing the escape key `ESC` (ASCII code 27; abort hotkey).

To call the **Layout Editor** function Add Part, activate the Parts function menu by selecting the main menu item Parts, then select the Add Part function from the Parts pull-down menu. The left mouse button provides fast access to the previously processed function, i.e., pressing the left mouse button re-activates the same function as was processed with the previous operation. The function currently assigned to the left mouse button is displayed in the BAE window title bar. Right-clicking the toolbar button `H` provides quick access to the 16 last called menu functions.

The Windows pull-down menus as well as the menu functions are context-sensitive, i.e., they are only selectable when currently applicable, otherwise they are faded-out ("ghost" menus). Menu separator lines are used throughout the Windows pull-down menu system to separate function groups.

The Windows and Motif versions provide context menus with element-specific functions which can be activated by right-clicking elements in the graphic workarea. The `p` (properties) key activates a dialog for displaying and/or modifying the properties of the element at the current mouse position.

For certain functions such as file name selection, the BAE Windows/Motif user interfaces provide Windows- and/or Motif-specific dialogs or popup menus instead of BAE standard popup menus. Scrollbars are displayed with BAE Windows and Motif workarea text popups to provide unrestricted access to the contents of lengthy listings and/or protocols.

The cursor/arrow keys can be used under Windows and Motif to scroll the BAE display by half of its dimension in the key-specific direction. The `Page Up` and `Page Down` keys scroll the display up or down by its full height. In combination with the `Shift` key, `Page Up` and `Page Down` scroll to the left and right, respectively. Scrolling is limited by the boundaries of the currently loaded element. The `Home` and `End` keys can be used to jump immediately to the upper or lower element boundary. In combination with the `Shift` key, `Home` and `End` scroll to the left and right element boundary, respectively.

Under Window and Motif, the mouse wheel can be used to move the current view port upwards or downwards by half its size. In combination with the `Shift` key, the mouse wheel moves the view port to the left or right, respectively. With the left mouse key pressed, the mouse wheel can be used to zoom in or out.

BAE Windows and Motif sessions can be finished using the standard Close function from the application window system menu or by clicking the Windows close button of the application window. To prevent from accidentally discarding design changes, these exit procedures might require user confirmation with an option for saving the currently processed element.

When ending a BAE Windows or Motif session, the dimensions and positions of the BAE application and dialog windows are automatically saved to a configuration file with the name **`baewin.dat`** or **`baexwin.dat`** in the BAE programs directory. The next BAE session automatically loads and restores the windows dimensions and positions from the configuration file. **BAE HighEnd** stores window positions not only with BAE module names but also with current session window numbers, thus allowing for multiple project windows of to be restored with the next BAE session. This is very useful when using **BAE HighEnd** with multi-monitor systems.

BAE Side Menu User Interface

The menu area on the right side of the BAE standard user interface is divided into a main menu and a standard function menu. Selections are allowed in both of these menus. The permanently displayed main menu is used for activating the more particular function menu displayed below the main menu. Each function of the same process can be reached with only one mouse interaction and each other function can be reached with a maximum of only two mouse interactions.

Within the menu fields of the BAE standard user interface the green menu cursor can be moved with the mouse. A particular menu function is selected by moving the mouse to that function and by pressing either the left or the right mouse button. To call the Add Part function the **Layout Editor**, simply activate the Parts function menu by selecting the Parts main menu item, then select the Add Part function from the Parts menu. Once the Add Part function is completed, it can be re-activated by simply pressing the left or right mouse button or you can select any other function from the still active Parts menu. If you e.g., want to change to the Add Trace function, just choose the Traces main menu item and select the Add Trace function from the Traces menu.

After selecting a particular function, either another menu is displayed or the user is prompted for keyboard input via the input line or graphic input via the graphic cursor is expected in the graphic workarea. Any messages displayed in the status line are displayed as long as they are relevant. Input prompts (e.g., for coordinate and/or length/width inputs) and error messages contain the name of the processed element if available. Graphic cursor inputs are accompanied by a status line message indicating the expected type of input. The color of the menu cursor changes to red to designate that the BAE system is waiting for some user input. The middle mouse button provides instant access to the View menu, i.e., pressing the middle mouse button is possible at any time and from any other function and provides fast access to frequently required display functions such as zoom and pan, changing color setups, setting the input and/or display grid, etc. Pressing the middle mouse button will also cause a status line display of the current file and element name if no other menu function is currently active. The middle mouse button interaction can also be simulated by simultaneously holding down both the left and the right mouse button. It is possible to activate the online display menu even if only a two-button mouse is installed and/or configured. Any currently active menu function can be immediately aborted and/or canceled by pressing the escape key **ESC** (ASCII code 27; abort hotkey).

A series of functions are implemented with popup menus, where the object to be processed can be selected by mouse-pick. These popup menus simplify the use of basic data and file management functions such as Load Element, Delete Element, File Contents, Load Colors, etc. With each popup menu, the input line is enabled for manual element name input via keyboard and special popup menu buttons such as Abort (for canceling the current function), Next (for scrolling down the popup menu selections list) or Back (for scrolling up the popup menu selection list) are usually provided.

The dialogs and/or popup menus for net list part and net name selections support **`?prefix`** input for scrolling the list display to the specified name prefix. A name prefix specification such as **`?r4`** scrolls to the first name starting with **`?r4`** or, if no such name exists, to the next name thereafter. Part and net list scrolling positions specified through name prefices are saved for subsequent name queries.

The menu color settings of the BAE standard user interface can be changed with the **BSETUP** utility program. Please note that suitable menu colors might achieve considerable ergonomic advantages such as better recognition of the currently active menu and/or function. See chapter 7.2 of this manual for a description of the **BSETUP** utility program.

Customizing the BAE User Interface

Bartels User Language provides system functions for performing key programming and defining menu assignments and/or toolbars. It is possible to define key bindings such as key r for activating **User Language** program **ROTATE**. New or existing menus and/or menu entries can be (re-)configured to support special **User Language** program calls. These features provide a most powerful tool for configuring the menus of the **AutoEngineer** modules. It is a good idea to utilize the **User Language** startup programs for performing automatic key binding and menu setup. Even dynamic changes to the **AutoEngineer** user interface can be supported with special **User Language** programs for performing online key and menu programming. Note that due to these features your currently configured **AutoEngineer** user interface might provide special user-specific add-on functions which are not described in this documentation. See the Bartels User Language Programmer's Guide for a detailed description of the **Bartels User Language** and its implicit program call features.

Facilities for cascading submenu definitions are implemented for the BAE Windows and Motif pulldown user interfaces. Submenus can be attached to menu items. The **UIFSETUP** **User Language** program is designed to configure cascading menus for the BAE Windows/Motif modules. This allows for easy submenu function location (and activation) without having to activate (and probably cancel) submenus. The function repeat facility provided through the right mouse button also supports cascading menus. This simplifies repeated submenu function calls significantly.

A series of Windows/Motif dialogs are implemented such as display and general parameter settings in all BAE modules, SCM plot parameters settings, **Autoplacement** and copper fill parameter settings, **Autorouter** routing batch setup and routing options, strategy and control parameter settings and **CAM Processor** control plot, Gerber photoplot and drilling data output parameter settings. These dialogs can be activated through the **bae_callmenu** **User Language** function. The **UIFSETUP** **User Language** program is designed to provide menu functions for activating the dialogs in the BAE Windows/Motif modules.

BAE HighEnd Message System

A message system is integrated to **BAE HighEnd** to enable advanced communication between different **AutoEngineer** program modules. The **BAE HighEnd** Shell (i.e., the **BAE HighEnd** main menu) is used as message exchange switchboard. Hence the BAE Shell knows only about its own descendants, i.e., descendant BAE tasks must be started using appropriate **BAE HighEnd** functions (New Task from the BAE Shell, New SCM Window from the Utilities menus of the **Schematic Editor** or the **Layout Editor** or switch between different program modules of the current BAE session). With **BAE HighEnd**, several views of a project, e.g., overview and zoomed details can be displayed at the same time. The use of multitasking and pipes also enables simultaneous processing of schematics and layout. I.e., **BAE HighEnd** supports advanced cross-module features such as simultaneous and/or global net highlight for schematic plans and layout of the same design (project specific multi-windowing/multitasking, cross-probing).

1.2.3 Basic System Functions

Display Functions

The middle mouse button is used to call the View menu whilst performing a graphical manipulation like component placement or routing. This allows for the change of display options such as zoom scale, grids, colors, etc. without canceling the current operation. After the display choice has been made you will automatically return to the graphical manipulation that was in progress before the middle mouse button was pressed. I.e., the global placement of a part can be performed with the complete layout displayed, whilst the final placement of that still picked part can be accomplished in a more detailed zoom window.

Colors

The colors for displaying the design objects can be changed with the Change Colors function from the View menu. At overlaps of different elements the resulting mixed color is displayed. The highlight color is also mixed with the color of the element to be marked, thus resulting in a brighter display of that element. The Save Colors function from the View menu is used to saved the current color table definition to a system file (whichever appropriate for the current program module). Once a color table has been saved, it can be reloaded at any time using the Load Colors function from either the View menu. Special color tables (e.g., for library edit, for finding unroutes, etc.) can be defined and reloaded on request. The default color table to be loaded after the startup of a particular BAE program module is the one named **`standard`**.

Changing some item-specific color is accomplished by selecting the desired display item using the left mouse button and then selecting the desired color button from the Change Colors function. In the layout system, Change Colors provides a feature for fast display item fade-out/fade-in. Activating and/or deactivating some item-specific display is accomplished by selecting the desired display item entry with the right mouse button which works as a toggle between fade-out and fade-in. The system won't loose information on currently defined colors of faded-out display items; strike-through color buttons are used for notifying currently faded-out display items.

Input Grid

The floating point database used throughout **Bartels AutoEngineer** allows to specify arbitrary placement coordinates. Input grids and angles can be released and/or locked at any time from throigh the View menu which can be activated with the middle mouse button. Nevertheless, the choice of a suitable input grid has fundamental meaning for design processes such as Autorouting or manufacturing data generation. Placing parts in 1/10" or 1/20" grids with just exceptional deviation (e.g., for plugs) will considerably facilitate both manual routing as well as Autorouting. Trace corners at 45 degree angle steps are recommended for better manufacturing results unless deviation is indispensable.

The cursor/arrow keys can be used together with the **`Shift`** key to move the mouse/graphic cursor to the next input grid point in direction of the arrow key. **`Shift`** together with **`Enter`** selects the current input grid coordinate (as if a corner was selected with the left mouse key) and a subsequent **`Enter`** key input terminates the definition of a point list as if Done was selected through the right mouse key. This allows for on-grid traces or polygons to be created through the keyboard only.

Coordinates

BAE uses a conventional coordinate system which always references the current position of the origin. It can be useful to reposition the origin to make absolute coordinate references easier - even for one command. A special submenu can be activated with the right mouse button when manipulating objects in the graphic workarea (placing/moving elements, creating polygons, etc.). This submenu provides functions for specifying absolute or relative coordinate values. Jump Absolute accepts an absolute coordinate value referring to the origin of the currently loaded element. Jump Relative accepts a coordinate value relative to the previous input coordinate value (e.g., when drawing polygons). Floating point coordinates can be specified, where fractional parts of corresponding numeric values must follow the decimal point (**.**). Coordinate values are interpreted either in mm units or in inch units, depending on whichever default input units are defined with the **USERUNITS** setup parameter (see also chapter 7.2 of this manual for a description on how to define **USERUNITS** with the **BSETUP** utility program). Non-default metric input is forced by attaching **mm** to the input value. Non-default imperial input is forced by attaching the double quote character **"** to the input value. Metric and imperial values can be mixed arbitrarily. Precision throughout the whole system is ensured with BAE's floating point database based on common internal system units. The system also support polar coordinate input through the **Polar Coordinates** button and subsequent radius and angle value prompts. Attaching an **R** to the angle value will force the system to interpret the value in Radians instead of (default) Degree units.

All numeric input fields of the BAE dialogs support simple arithmetic expressions with add, subtract, multiply and divide operators and round brackets. An equal sign at the end of the expression causes the system to calculate and display the result immediately in the input field. Otherwise, the expression is evaluated when the dialog box closes successfully. This allows for, e.g., a Jump relative to be carried out through a Jump absolute dialog with the relative coordinates added to the absolute coordinates using the **+** operator.

File Management

Most of the BAE functions require an element (e.g., SCM sheet, layout board, library symbol, etc.) to be loaded. A particular element is specified by element type, file name and element name. The element name is the unique name of the element in the selected design database (DDB) file. The process of loading an element is activated after specifying the element name. There are two methods for specifying file and element names. Either select the file and/or element name with popup menu and mouse-pick or perform direct keyboard input by typing in the name to the corresponding input line prompt. The file name of the currently loaded element is used if you select the Project popup menu button or if you type in an empty string (by pressing the return key ↵) to the file name prompt. DDB file name queries in the **Schematic Editor** and the layout system also accept **!** input for selecting the SCM and/or the layout standard library defined through the BAE setup (see also chapter 7.2).

Please do not forget to save the currently loaded element before exiting from the program or loading or creating another element. The system activates a popup menu with options for discarding and/or saving changes to prevent from unintentionally discarding design changes in cases where an unsaved element is about to be unloaded.

When loading a BAE DDB file element, the element's modification time is retrieved. This modification time is checked against the current time when saving the element. User confirmation request is issued if the DDB file element appears to have been changed, thus providing advanced support and security for BAE network installations where different users might simultaneously modify the same DDB file element.

With the BAE standard user interface, an intelligent popup menu for optionally selecting directories is integrated to the file name query functions. This feature can be activated by selecting the Dir. button from the currently active file selection popup menu. The **BSETUP** command **PROJROOTDIR** (see chapter 7.2) can be used to define the root of the directory tree to be displayed for directory name selection; on default the current directory (relative path name **.**) will be used. The background color for the directory selection popup menus can be defined using the **POPMFILL** option of the **FRAMECOLOR** **BSETUP** command (see also chapter 7.2). Directories with subdirectories are displayed in hierarchically arranged graphic frames. Directories without subdirectories are displayed with their name only. Directories including files with extension **.ddb** (DDB files) are marked by appending a plus sign (**+**) to the end of the directory name. The Next button of the directory selection menus is used to scroll in the directory selection menu. The ...Zoom button can be used to switch to a more detailed display of the selected directory; the Parent button can be used to switch back to the directory survey. Selecting the Abort button will cancel the file name query. An error message such as **No subdirectories found!** is issued if there are no subdirectories available in the directory defined by **PROJROOTDIR**. After selecting a valid directory, the file name query is reactivated with the file names of the selected directory provided for selection.

With the Windows and Motif user interfaces of the BAE software, Windows- and/or Motif-specific popup menus for file name selection with directory navigation and listboxes for element name queries are automatically provided.

The file and element name queries provide default names (current project file name, selected library, current element name, etc.) if at all possible and/or appropriate. The Delete Element functions from the File menus can only be applied to DDB file elements which are not referenced by any other element from the same DDB file.

Automatic Design Data Backup

A feature for optionally performing automatic design data saving is implemented with the **Schematic Editor**, the **Layout Editor** and the **Neural Router**. This feature is controlled with the Autosave function from the Settings menus. The Autosave function requires a positive integer input designating the autosave time interval in minutes. On zero or dash (`-`) input the automatic save facility will be deactivated. With Autosave activated, the system automatically saves the currently processed element to a backup file at the specified time intervals. However, to prevent Autosave from overwriting backup files in situations where an element is only loaded for viewing/checking purposes, the backup is only performed if the currently loaded element was modified during the autosave interval. The name of the backup file is automatically derived from the current job file name and has the extension `.bak`. Autosaved elements can be restored using features such as the Save Element As function from the File menu or the **COPYDDB** utility program.

Automatic Parameter Backup

Important design and operational parameters such as autosave time interval, name of the currently loaded color table, input and display grid, angle and grid lock, coordinate display mode, standard placement angle and mirror mode, standard text size, library access paths, plot file names, standard trace widths, Mincon function class, airline display mode, placement matrix, copper fill parameters, etc. are automatically saved with the currently processed layout board and/or SCM sheet or with the processed library hierarchy level (part, padstack, pad, SCM symbol, etc.). When loading an element, the corresponding parameter set is automatically loaded as well, thus providing a convenient way of setting up a default design environment suitable for processing the selected database and/or design element.

Element Boundary, Workspace

At the creation of a new element the system will prompt for the element boundary. The element boundary corresponds with the workspace on a paper for manual drawing, i.e., the element boundary defines the overall size of the data element and must not be confused with the board outline or any other artwork drawing. It is utilized for preventing the system from creating or placing objects somewhere outside in hidden infinite regions, i.e., nothing can be drawn or placed outside the workspace (which otherwise would be possible because of the floating point coordinates supported by the system). The element boundary can be enlarged or reduced with the Upper/Right Border and Lower/Left Border functions from the Settings menu. Note that workspaces larger than necessary are worsening performance at screen redraw, zoom, pan and certain other functions. It is a good idea to shrink each element's workspace; element workspaces can easily be enlarged at any time lateron.

Group Functions

The **Bartels AutoEngineer** group functions allow powerful manipulation in both the **Schematic Editor** and the **Layout Editor**. The group functions are featuring set principles. Groups can be defined either by selecting individual items or by defining the area around the items that you want to select. Choice of item types and the ability to de-select with the same technique makes it easy to define a group. Group-selected elements are marked by highlight, and these are the only objects to be affected by subsequent group functions. Groups can be moved, copied, deleted or saved and loaded. When saved they are stored as an element of the same type as that from which the group was selected and can be accessed by the appropriate editor. When saved a group origin must be defined which becomes the origin of the new element and is used as the reference point for group load commands. Group facilities can be used for a variety of tasks such as replicating circuitry and/or tracking, saving and loading standard SCM blocks or PCB templates, stealing from existing and proven designs, etc.

Undo, Redo

With the Undo and Redo functions from the Edit menu, the **Bartels AutoEngineer** can be used without fear of causing damage. Previously executed commands can be reversed or undone using Undo and then reprocessed with Redo. This ensures data security and provides a powerful feature for validating different design options.

On default, the system supports twenty Undo steps. The Setup dialog from the BAE main menu provides options for increasing the number of supported Undo steps for the **Schematic Editor** and/or the **Layout Editor** to up to one hundred.

1.2.4 Graphic Input

Input to the BAE graphic workarea is performed with the graphic cursor and the mouse. The most important graphical interactions are pick and place. Pick means selecting an item which is already placed in the graphic workarea. Place means placing a new item to the graphic workarea. A pick function is often followed immediately by a place function, e.g., when moving a part.

The left mouse button is used both for selecting an element (pick) for further manipulations and for selecting the current graphic cursor position for place operations such as placing a selected part or defining the next corner point of the trace or polygon to be currently created. The right mouse button is used either for canceling pick operations or for activating a submenu with special options. This options submenu provides choices appropriate for the current graphic operation. These include important functions such as setting rotation angles and mirroring during placement, specifying direct coordinate input (Jump Relative, Jump Absolute) during placement or polygon point definition, performing layer changes or setting trace widths during manual routing, generating arcs whilst drawing lines or defining areas, defining text sizes whilst moving texts, finishing a manually routed trace or a polygon definition, etc.

Each area and line (or trace) in the **Bartels AutoEngineer** is defined as a polygon. The corresponding menus for the creation and manipulation of such items are Graphic in the **Schematic Editor** and Areas (or Traces) in the **Layout Editor**. BAE supports different polygon types such as graphic area, graphic line and dotted line in schematics or passive copper area, active copper area, keepout area, documentary area, documentary line, copper fill workarea and trace in the layout system. Each polygon can consist of an arbitrary number of polygon points and arc segments. An arc segment is created by defining the arc segment start point, then choose the requested orientation with submenu function Arc Left or Arc Right and select the arc center point, and finally define the arc segment end point. When finishing the polygon definition with the Done submenu function, the system distinguishes between area and line definitions, and the last polygon point is connected to the first point when creating an area. No area can contain intercrossing segments because otherwise basic operations like checking polygon intersection for clearances and short circuits, polygon filling features, area size changes, etc. would not be possible. The polygons created by the **Bartels AutoEngineer** are not just simple line drawings but intelligent arbitrary shaped polygons. The system is fully capable of applying complex operations on these polygons such as move, copy, enlarge, shrink the polygon, move, delete, insert polygon corner points and/or polygon segments, perform automatic copper fill and/or hatching, perform design rule checking, etc.

Redundant polygon points should be avoided because they might cause trouble, e.g., when generating certain CAM data. In this context, the definition of full circles in BAE is worthwhile to be mentioned here especially. A full circle is created by defining a point on the circle, then choose either Arc Left or Arc Right from the submenu and select the circle center point and finish the definition with the Done submenu function. Alternatively, the c key can be pressed to set the circle center point, and then a second point can be selected to set the circle radius.

1.2.5 Special Remarks

The preceding paragraphs explained the basic concepts of operating the BAE design system. Now you can become more familiar with the BAE system functions by working through the examples provided with this manual. You should also have a close look at the **BSETUP** utility program and the **AutoEngineer** database concept before starting the design of real projects since the user-definable setup parameters and database conventions have most considerable inpact on the design process and the possibilities of manufacturing data output.

Waiting for the Completion of Complex Functions

Some of the more complex BAE functions such as loading or moving groups might require some CPU time. Within the Windows version of the BAE software the mouse cursor will change to a sand clock symbol to indicate that the system is waiting for the completion of some function. Within the BAE standard user interfaces the menu cursor changes its color (to red) to indicate that the system is currently busy. This is also true for operations where the system expects some user input in the graphic workarea or via keyboard input. After the completion of a particular function the menu bar color of the BAE standard user interface changes back (to green) to indicate that the user can activate another function. More time-consuming functions such as batch design rule check or connectivity generation will even report the percentage of completeness for better information. It is strongly recommended to wait for each BAE function to be successfully completed, because interrupting a function by e.g., resetting your computer might cause irreparable damage of design data.

Design Data Backup

Kindly note the importance of backing up your design data when working on real projects. Every now and then we are asked to restore damaged design data on faulty hard disks. A hard disk can suffer a genuine loss of data by headcrash or other hardware defects, and we are not able to help if not even a simple backup has been performed. A regular backup of your project (DDB) files is strongly recommended. You can also use the Autosave function from the Settings menu (see above) for activating the automatic design data backup feature.

Releasing Manufacturing Data

Before generating and releasing manufacturing data, you should always perform a complete design rule check using the Batch DRC function from the Utilities menu of the **Layout Editor**. Subsequently, the DRC result should be examined with the Report function. Never start the CAM data output if clearance violations or short circuits are indicated since otherwise you might produce useless PCBs in almost any case. Also ensure that the CAM process does not cause any troubles such as overdraw errors. Apply **CAM View** on generated Gerber data for visual checks. Furthermore it is recommended to perform extensive tests on prototypes before starting any mass production. You should be able to achieve good project and production results with the **Bartels AutoEngineer** if you follow these instructions.

1.3 BAE Design Database

1.3.1 Database Concept

The key feature of the **Bartels AutoEngineer** design system is its powerful design database. It is a genuine object-oriented database since this has been proven to be optimum for accessing and processing the complex data types of different size used throughout the **AutoEngineer**. Powerful search algorithms are implemented for fast access to the database objects, and a high-sophisticated library management program provides simultaneous access to different symbol libraries. The user is able to define both standard libraries and job-specific libraries for particular projects. These libraries can be stored with the job and completely and/or partially transferred to or from a central standard system library.

Object Classes, Hierarchy

Each database entry is assigned to a predefined database class and is identified by an element name which is unique in this database class. Each element contains the data defined on that element, i.e., graphic items, texts, pin positions and pin designators on SCM and/or layout symbols, etc. Higher level database entries can refer elements of subordinate database hierarchy levels. Such a reference consists of the element placement coordinates, a name referring the subordinate database symbol, and the name of the placed element if it is a named reference. E.g., a layout contains the named references of the part symbols placed on the layout, the parts contain the named references to the padstack symbols defined on the part, and the padstacks contain the unnamed references to the pad symbols used on these padstack. All references of a certain element are transparent throughout the corresponding DDB file. When loading or copying that element, all pertinent references are automatically be loaded and/or copied.

Homogeneity

All database classes are subject to general file and database management functions. I.e., the **Schematic Editor** is suitable for creating and manipulating SCM plans as well as for editing SCM library elements such as symbols, labels, and pins. The same is true for the **Layout Editor** concerning the creation and manipulation of layouts, parts, padstacks and pads. All functions are automatically adjusted to the current environment and database hierarchy level. Corresponding menus and functions from the **Schematic Editor** and the **Layout Editor** are quite similar. I.e., the Add Connection SCM function is equivalent to the **Layout Editor** Add Trace function, and the **Layout Editor** Add Part function is used for placing parts on layouts, for placing padstacks on parts or for placing pads on padstacks, whichever is appropriate for the currently loaded element. This concept creates far-reaching analogies for the processing of objects of different database classes and makes it fairly easy to learn how to use the BAE design system.

File Format

All program modules of the **Bartels AutoEngineer** are working with the same database and file format. This file format is called Design DataBase (DDB) format. The `.ddb` file name extension is used for indicating the DDB file type. All of the design data relating to one particular design is kept in a single DDB file, thus avoiding the confusion of multiple file structures. This DDB file contains different element types relating to schematics, PCB layout, net list data, library symbols, parameter settings, etc. all of which go together to make a complete design. This concept introduces most efficient data management features for creation, modification, backup, update and redesign of both design and library files. The BAE library files including SCM and layout symbols, logical library entries, etc. are also prepared in DDB format. **AutoEngineer** objects can be composed of sub-elements from different libraries. A library or design file can be build up by accessing one or more other libraries, and even a particular design file can be used as library for another design.

Data Consistency

Each element is constructed dynamically during load. E.g., when loading a layout element to the **Layout Editor**, first the data pertinent to the layout is read and then the referenced part elements are loaded from the same project file; with each part element the referenced padstack symbols are loaded, etc. This dynamic load process presupposes that all required elements are available in the project file. The system will automatically check on missing library symbols, and requested elements not available are copied and loaded from the currently selected library. With this process, consistent construction of job-specific libraries without the need of storing redundant library elements in a job file is always ensured. This concept introduces considerable advantages with regard to project archiving or independence from master library availability. Modifications in a certain DDB file are reflected up the library hierarchy levels of that file without affecting any other DDB file, and the functions for deleting library elements prevent from erasing DDB file elements which are referenced by other elements from the same DDB file. I.e., data consistency throughout any particular DDB file is automatically ensured. Nevertheless, BAE provides powerful features for transferring library data between different DDB files such as the Update Library function from the File menu which can be used for correlating job-specific libraries with master library contents.

Manufacturing Process Control

A further advantage of the BAE database concept is the possibility of adapting project-specific libraries to special manufacturing processes. For this purpose, so-called technology parts containing special pin and/or padstack definitions can be defined in technology-specific library files. With a technology part a complete set of technology-dependent pin definitions can be defined (e.g., SMD pad definitions for a special SMT soldering process or annular pad shapes for manual drilling). Functions such as Replace Element or Update Library can be utilized for copying a technology part to a certain project file in order to include pin and/or pad definitions requested for a special manufacturing process.

Creating Library Elements

The New function from the File menu is used in both the **Schematic Editor** and the **Layout Editor** for creating new BAE library (and design) elements. After specifying the new element library hierarchy level the user is asked for the name of the DDB file where the element is to be stored, the name of the element to be created and the element boundaries. Subsequently, the element is defined by placing elements from subordinate library hierarchy levels and by creating additional objects such as documentary graphic, text, drill holes, contact areas, keepout areas, etc. (whichever is permissible on the currently edited library hierarchy level).

SQL Functions, Relational Databases

Bartels User Language provides SQL (Structured Query Language) functions for maintaining relational databases, thus introducing powerful software tools for programming database management systems. These tools e.g., can be utilized for integrating a component database to the **Bartels AutoEngineer** to perform stock and cost expenditure analysis on different variants of a layout including facilities for choosing components with controlled case selection and part value assignment. This however is just one example from the wide range of possible database applications; utilizing database systems could be worthwhile also in the fields of project and version management, address list maintenance, production planning and inventory control, supplier and customer registers management, etc. See the Bartels User Language Programmer's Guide for a detailed description of **Bartels User Language** and its integrated SQL functions.

1.3.2 SCM Database Hierarchy

Figure 1-4 shows the structure of the database hierarchy supported throughout the **Bartels AutoEngineer** SCM system.

The SCM sheet level is the top hierarchy level of the BAE SCM design system database. On SCM sheet level, the circuit diagram of a particular design is defined by generating SCM sheets, placing symbols and creating connections and busses. Labels, bus taps and module ports can be placed for defining signal names and for connecting different sheets. Graphic and text can be created for documentation purposes such as creating SCM sheet frames, including commentary text, etc. Part and net attribute values can be assigned for setting variable component properties and/or for controlling subsequent design processes such as Autorouting or CAM data output.

On SCM symbol level the schematic part symbols are defined (and stored to SCM symbol libraries). A schematic symbol is usually defined by placing elements from the subordinate marker level, thus creating the logical pins of the corresponding part. Graphic and text can be created on symbol level for including symbol outlines, part name references, attribute definitions, commentary text, etc.

Special symbols for signal naming purposes are defined on SCM label level. These symbols can be utilized on SCM sheet level for assigning signal names or signal levels to connections and/or busses, for tapping busses, for connecting different SCM sheets, etc.

Pin symbols are created by defining a contact area on SCM marker level. Marker symbols can be placed on SCM symbol and/or label level to determine the positions of the corresponding part pins. The contact area is required for connecting the corresponding pin on SCM sheet level. A reference designator text can be defined on marker level for showing pin names on symbol level. Marker symbols with a normal graphic area instead of a contact area can be utilized on SCM sheet level for creating and displaying T-connections.

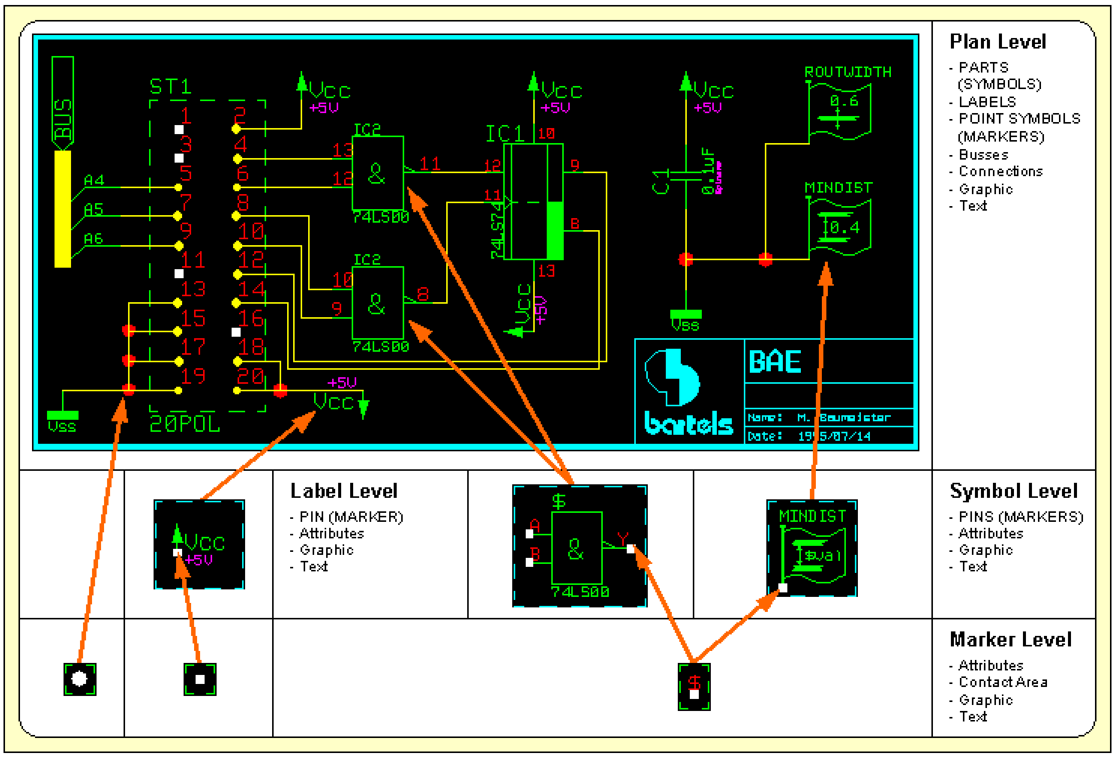


Figure 1-4: SCM Database Hierarchy

1.3.3 Layout Database Hierarchy

Figure 1-5 shows the structure of the database hierarchy supported throughout the **Bartels AutoEngineer** PCB design system.

The layout level is the top hierarchy level of the BAE PCB design system database. On layout level the PCB contour is defined, the parts (from the subordinate layout part level) are placed, keepout areas, power planes and copper areas are defined, the traces are routed, and, finally, the CAM output is generated. Drawing items and text can be created on layout level for such things as plot registration markers, measurement, project identification, etc.

On layout part level the layout part symbols (i.e., the part package types) are defined (and stored to a layout part library). A particular layout part symbol is usually defined by placing elements from the subordinate padstack level in order to define the types and positions of the physical pins of the corresponding part. Traces and vias (e.g., for printed inductors), keepout areas (for part clearance check, defining via keepout areas, etc.), copper areas, drawing information (component outline on insertion plan) and text (for part name reference, insertion data pick point, attribute value display, etc.) can be created optionally.

On layout padstack level the layout pin symbols and vias are defined by placing symbols from the subordinate pad level. Each pad can be assigned to a signal and/or documentary layer thus designating contact areas for the routing or defining pad shapes for solder resist, SMD masks, etc. A drill hole and drill plan info can be created optionally for the definition of vias or drilled pins. Keepout areas can be utilized for controlling the pin contact mode. Documentary lines or areas can serve as pin designators on the silk screen or insertion plan and reference texts can be used for displaying pin names on part and/or layout level.

On layout pad level the pad shapes (i.e., the pin contact areas) are defined by creating passive copper areas. Different pad symbols can be assigned to different layers on a single padstack symbol thus defining a particular layout pin type.

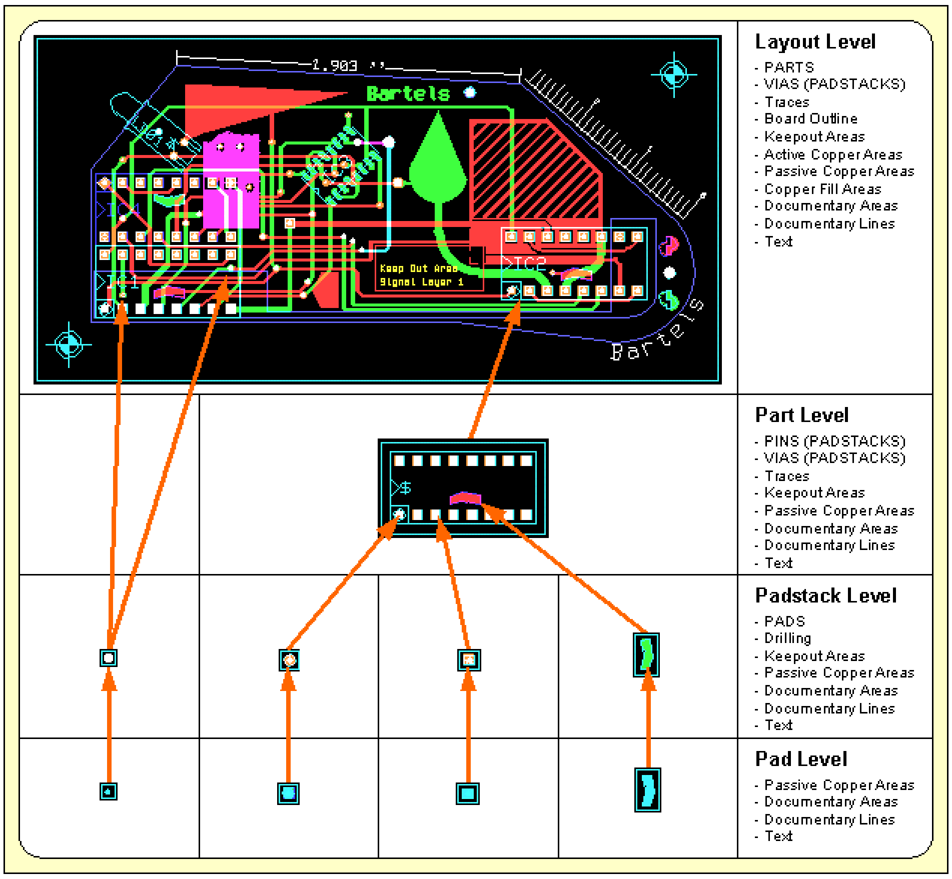
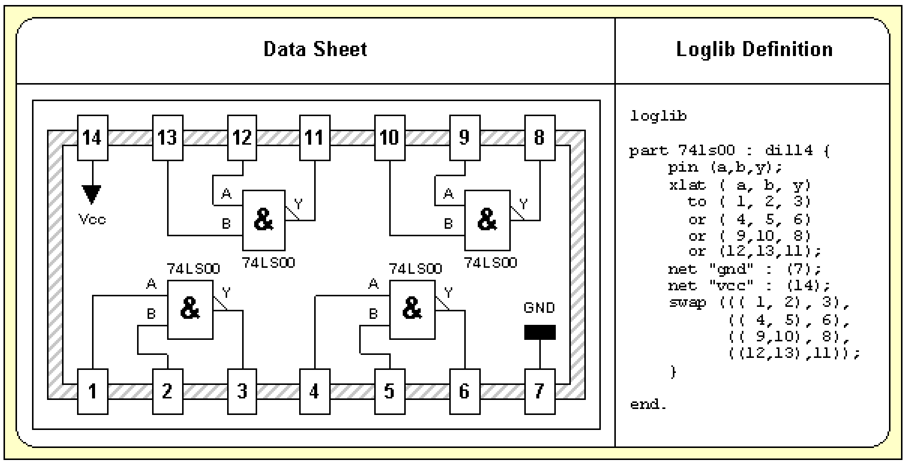


Figure 1-5: Layout Database Hierarchy

1.3.4 Logical Library

The **Bartels AutoEngineer** logical library provides the link between the SCM library and the layout library. The logical library contains information about the assignment of SCM symbols to layout packages including gate definitions and pin mapping, pin/gate swap rules, predefined power supply pins, fixed part attributes, etc. All these definitions can be entered to an ASCII file to be subsequently transferred to a BAE DDB file using the **LOGLIB** utility program (see chapter 7.11 of this manual for a description of the **LOGLIB** utility program). The logical library definitions are required by the **Packager** for compiling logical net list data created by the **Schematic Editor** to physical net list data which can be processed by the BAE layout system (see chapter 3.2 of this manual for a more detailed description of the BAE **Packager**). During a **Packager** run, the logical library entries are checked against the corresponding layout library symbols for ensuring correct pin mappings. Please note that the **Packager** can evaluate only one library file at the same time, i.e., both the required logical library data and the requested layout library data must be stored to the same DDB file.

Figure 1-6 shows a **LOGLIB** file example containing a logical library part definition according to the manufacturer's part data sheet.



```
loglib

part 74ls00 : dil14 {
    pin (a,b,y);
    xlat ( a, b, y)
      to ( 1, 2, 3)
      or ( 4, 5, 6)
      or ( 9,10, 8)
      or (12,13,11);
    net "gnd" : (7);
    net "vcc" : (14);
    swap ((( 1, 2), 3),
          (( 4, 5), 6),
          (( 9,10), 8),
          ((12,13),11));
    }

end.
```

Figure 1-6: Part Data Sheet with Loglib Definition

The following sequence of operations is recommended for the definition of a new part:

- create the SCM symbol and store this symbol to a SCM library
- create the layout part symbol and store this symbol to the central layout library (if not yet existing)
- create an ASCII loglib file containing the requested logical library part definition and
- transfer the loglib file to the central layout library using the loglib utility program

Once the above tasks are completed, the new SCM symbol can be used for circuit drawing, the **Packager** is able to assign this SCM symbol to the correct layout package, and the corresponding layout part is available for placement in the layout. For less-experienced users it is a good idea to use a test file for new definitions, and to perform a **Packager** test run before releasing new library parts for real projects. This should prevent from unintentionally introducing erroneous library definitions to BAE library and/or project files.

Chapter 2
Circuit Design / CAE

This chapter describes in detail how to use the **Schematic Editor** for creating SCM library symbols and designing circuits. The examples presented with this chapter introduce the concepts and advanced features of the BAE **Schematic Editor** in a logical sequence and will take the user through the design and modification of library symbols and a circuit drawing which will be subject to further processing in the subsequent chapters. The reader should work through this chapter without missing any sections to gain full understanding of the BAE **Schematic Editor**. Once a command has been used and/or explained, the operator is assumed to have understood its function and be able to perform it again. Subsequent instructions containing this command will be less verbose for easier reading and more speedy learning.

Contents

Tables

Figures

2.1 General

The schematic capture system of the **Bartels AutoEngineer** essentially consists of an interactive **Schematic Editor** with integrated SCM symbol editor and integrated Backannotation and the **Packager** program module for converting logical into physical netlists ("Forward Annotation"). The following sections of this manual describe in detail how to use the **Schematic Editor** for creating and editing SCM symbols and schematic circuitry.

2.1.1 Components and Features

Schematic Editor

When working with a **Schematic Editor**, the user expects the net list to agree exactly with the circuit drawing. The **Schematic Editor** of the **Bartels AutoEngineer** is not just a simple drawing program but is also capable of instantly controlling and correlating the net list data with the circuit diagram. Every single net list change results in an incremental update of the net list data. Reliability on correct net list data recognition is achieved with an internally managed list of changes which is also used for controlling the Undo/Redo facilities. BAE utilizes contact areas for indicating unconnected pins. Contact areas will automatically disappear with correct connections to the corresponding pins. This feature provides most useful graphical utility for controlling net list data whilst editing circuit diagrams. The system is also able to recognize T-shaped connections, and a junction point marker is automatically placed for the indication of T-connections.

With **Bartels AutoEngineer**, SCM symbols can be freely defined without restrictions referring to symbol size, symbol drawing information, number of pins on a part symbol, etc. The user can utilize but is not necessarily restricted to some standard. A series of extensive SCM symbol libraries is delivered with the **Bartels AutoEngineer**. The symbol placement functions provide features such as rotating the symbol at any angle, symbol mirroring, specifying arbitrary placement coordinates, etc.

Signal names can be freely defined. Inverted signal names are displayed with a line on top of the corresponding text. User-definable label symbols are automatically loaded when referencing the corresponding signal names, e.g., a ground label symbol can be defined for indicating ground connections.

Bartels AutoEngineer provides full capability of checking connectivity with busses, bus tapping, sub-bus definitions, bus naming, etc. Both numerical and arbitrary alphanumerical names can be used for bus signals. Incremental net list data update is included with all bus definition features. The SCM library even supports bus synthesis on part level, thus providing highest flexibility at the creation of symbols of any complexity.

The BAE system supports hierarchical circuit design. It is possible to define SCM block circuit diagrams which then can be referenced from other SCM plans using special block symbols. For correct net list data management a distinction between global and local signals is possible.

The Undo/Redo facilities allow to use the **Schematic Editor** without fear of causing damage. Up to twenty commands can be reversed or undone by applying the Undo function and then reprocessed with the Redo function. Undo/Redo ensures data security and provides a powerful feature for estimating design alternatives.

With the **Bartels User Language** integrated to the **Schematic Editor** the user is able to implement enhanced CAE functions and macros, user-specific post processors, report and test functions, etc. **User Language** programs can be called by applying the Run User Script function from the File menu or by pressing a key on the keyboard (hot key).

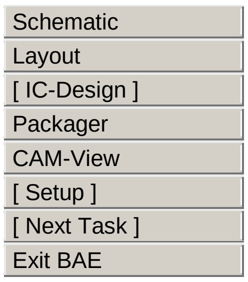
The **Schematic Editor** provides a series of enhanced features such as group functions, arbitrary attribute definitions and part and/or net attribute value settings, automatic part naming with definable part name patterns, virtual symbols for company logos or circuit legends, net highlight, automatic reconnection at symbol movement, etc. Net list modifications introduced to the layout such as pin/gate swaps or part name changes are reflected back to the SCM automatically by applying **Backannotation**.

2.1.2 Starting the Schematic Editor

It is recommended to start the **Bartels AutoEngineer** from the directory where the design files should be generated since this considerably simplifies job file access. If you intend to process the examples provided with this manual it is recommended to move to the BAE examples directory installed with the BAE software. The **Schematic Editor** can be called from the **Bartels AutoEngineer** main shell. Start the BAE shell by typing the following command to the operating system prompt:

```
> bae
```

The **AutoEngineer** comes up with the Bartels logo and the following menu (the Setup function is only available under Windows/Motif; the IC-Design and Next Task menu items are available only with special software configurations such as **BAE HighEnd** or **BAE IC Design**):



Move the menu cursor to the Schematic menu item and confirm this choice by pressing the left mouse button:

Now the **Schematic Editor** program module is loaded and the SCM menu will be activated. If this fails to happen then check your BAE software installation (see the Bartels AutoEngineer® Installation Guide for details on how to perform a correct installation).

2.1.3 Schematic Editor Main Menu

The **Schematic Editor** standard/sidemenu user interface provides a menu area on the right side, consisting of the main menu on top and the currently active menu below that main menu. After entering the **Schematic Editor**, the Files menu is active and the menu cursor points to the Load Element function.

The Windows and Motif versions of the **Schematic Editor** can optionally be operated with a pull-down menu user interface providing a horizontally arranged main menu bar on top. The **`WINMENUMODE`** command of the **BSETUP** utility program is used to switch between **`SIDEMENU`** and **`PULLDOWN`** Windows/Motif menu configurations (see chapter 7.2 for more details).

The following main menu is always available whilst processing SCM elements with the **Schematic Editor**:

- Undo, Redo
- Display
- Files
- Symbols
- Connections
- Graphic
- Text
- Groups
- Parameter
- Plot Output
- Utilities

Undo, Redo

The functions provided with the Undo, Redo menu allow you to use the **Schematic Editor** without fear of causing damage. Up to twenty commands can be reversed or undone using Undo and then reprocessed with the Redo. This is true even for complex processing such as group functions or **User Language** program execution. Undo, Redo ensures data security and provides a powerful feature for estimating design alternatives.

Display

The View or Display menu can either be activated by selecting the corresponding main menu item or by pressing the middle mouse button. Activation through the middle mouse button is even possible whilst performing a graphical manipulation such as placing or moving an object. The View or Display menu provides useful functions for changing display options such as zoom window, zoom scale, input and/or display grids, grid and/or angle lock, color settings, etc. The View or Display menu also contains advanced display functions such as Find Part and Highlight Net.

Files

The Files menu provides functions for creating, loading, saving, copying, replacing and deleting DDB elements. The Files menu also allows to load and/or store color tables or to call important database management functions such as listing DDB file contents and performing library update.

Symbols

On SCM sheet level, the functions from the Symbols menu are used for placing, moving and deleting symbols, labels, and module ports, for assigning attribute values and for displaying symbol logic definitions from the Logical Library. On symbol and label level, the functions from the Symbols menu are used for placing, moving and deleting pins (i.e., marker symbols), and for defining name patterns for the automatic part naming facility.

Connections

The Connections menu is used on SCM sheet level to create the logical net list by generating and/or manipulating connections and busses.

Graphic

The Graphic menu is used on any SCM hierarchy level for creating, moving, copying and deleting graphic items such as lines and areas.

Text

The Text menu is used on any SCM hierarchy level for defining, changing, moving and deleting text.

Groups

The Groups menu provides functions for selecting elements to group, for moving, copying, deleting, saving and loading groups, and for replacing symbols in a group.

Parameter

The Parameter menu provides functions for selecting the SCM library, setting the origin of the currently loaded element, defining the element boundaries, selecting the pin symbol and the T-connection marker, defining the SCM sheet hierarchy for hierarchical circuit designs, and activating the automatic design data backup feature.

Plot Output

The Plot Output menu provides the functions for generating plot output in HP-GL (Hewlett Packard Graphics Language), HP Laser (PCL, Printer Command Language) or Postscript format.

Utilities

The Utilities menu provides functions for exiting BAE, returning to the BAE main shell, producing a report for the currently loaded element and starting **User Language** programs.

2.1.4 Customized Schematic Editor User Interface

Menu Assignments and Key Bindings

The BAE software comes with **User Language** programs for activating a modified **Schematic Editor** user interface with many additional functions (startups, toolbars, menu assignments, key bindings, etc.). The **BAE_ST** **User Language** program is automatically started when entering the **Schematic Editor**. **BAE_ST** calls the **UIFSETUP** **User Language** program which activates predefined **Schematic Editor** menu assignments and key bindings. Menu assignments and key bindings can be changed by modifiying and re-compiling the **UIFSETUP** source code. The **HLPKEYS** **User Language** program is used to list the current key bindings. With the predefined menu assignments of **UIFSETUP** activated, **HLPKEYS** can be called from the Key Bindings function of the Help menu. Menu assignments and key bindings can be listed with the **UIFDUMP** **User Language** program. The **UIFRESET** **User Language** program can be used to reset all currently defined menu assignments and key bindings. **UIFSETUP**, **UIFDUMP** and **UIFRESET** can also be called from the menu of the **KEYPROG** **User Language** program which provides additional facilities for online key programming and **User Language** program help info management.

Context-sensitive Function Menus

Pressing the left mouse button in the graphic workarea activates a context-sensitive menu with specific functions for the object at the current mouse position if no other menu function is currently active. The Load Element and/or Create/New Element file management functions are provided if no element is currently loaded. This feature is implemented through an automated call to the **SCM_MS** **User Language** program.

Cascading Windows/Motif Pulldown Menus

The Windows and Motif pulldown menu user interfaces of the **Schematic Editor** provide facilities for cascading submenu definitions. I.e., submenus can be attached to other menu items. The **UIFSETUP** **User Language** program configures cascading submenus for the pulldown menu interfaces of the Windows/Motif **Schematic Editor** modules. This allows for easy submenu function location (and activation) without having to activate (and probably cancel) submenus. The function repeat facility provided through the right mouse button supports cascading menus to simplify repeated submenu function calls.

Windows/Motif Parameter Setup Dialogs

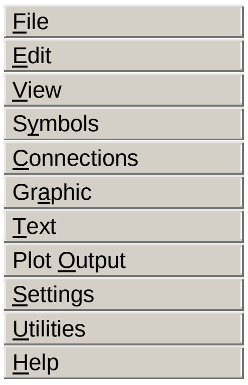
The following Windows/Motif parameter setup dialogs are implemented for the **Schematic Editor**:

- Settings - Settings: General **Schematic Editor** Parameters
- View - Settings: Display Parameters
- Plot Output - Settings: SCM Plot Parameters

The **UIFSETUP** **User Language** program replaces the parameter setup functions of the Windows and Motif pulldown menus with the above menu functions for activating the corresponding parameter setup dialogs.

Windows/Motif Pulldown Menu Konfiguration

When using pulldown menus under Windows and Motif, the **UIFSETUP** **User Language** program configures the following modified **Schematic Editor** main menu with a series of additional functions and features:



2.1.5 In-built Schematic Editor System Features

Automatic Parameter Backup

The **Schematic Editor** provides an in-built feature for automatically saving important design and operational parameters with the currently processed SCM sheet and/or SCM library hierarchy level. The following parameters are stored to the current design file when activating the Save Element function:

- Autosave Time Interval
- Name of the currently loaded SCM Color Table
- Input Grid
- Display Grid
- Grid/Angle Lock
- Coordinate Display Mode
- Symbol/Label Placement Default Rotation Angle
- Symbol/Label Placement Default Mirror Mode
- Symbol/Label Placement Signal Routing On/Off Mode
- Default Bus Tap Swap Flag
- Default Text Size
- Symbol Library File Name
- Logical Library File Name
- Plot File Name

Parameter sets are stored with special names according to the currently processed SCM database hierarchy level. Parameter set name **`[plan]`** is used for SCM sheet elements, **`[symbol]`** is used for SCM symbol elements, **`[label]`** is used for SCM label elements and **`[marker]`** is used for SCM marker elements. When loading an element, the corresponding parameter set is automatically loaded and/or activated as well, thus providing a convenient way of activating a default parameter set suitable for processing the selected design and/or library element type.

User Language

The **Bartels User Language Interpreter** is integrated to the **Schematic Editor**, i.e., **User Language** programs can be called from the **Schematic Editor**, and it is possible to implement any user-specific SCM function required such as status display, parameter setup, reports and test functions (fanout control, electronic rule check), special plot and/or documentation output functions, automatic or semi-automatic symbol edit routines, symbol library management utilities, customer-specific batch procedures, etc.

The **Schematic Editor** provides both explicit and implicit **User Language** program call facilities. **User Language** programs can be started with explicit program name specification using the Run User Script function from the File menu (empty string or question-mark (**?**) input to the program name query activates a **User Language** program selection menu).

User Language programs can also be called by simply pressing special keys of the keyboard. This method of implicit **User Language** program call is supported at any time unless another interactive keyboard input request is currently pending. The name of the **User Language** program to be called is automatically derived from the pressed key, i.e. pressing a standard and/or function key triggers the activation of a **User Language** program with a corresponding name such as **scm_1** for digit key **1**, **scm_r** for standard key **r**, **scm_#** for standard key **#**, **scm_f1** for function key **F1**, **scm_f2** for function key **F2**, etc.

The **Schematic Editor User Language Interpreter** environment also features event-driven **User Language** program calls, where **User Language** programs with predefined names are automatically started at certain events and/or operations such as **SCM_ST** at **Schematic Editor** module startup, **SCM_LOAD** after loading a design element, **SCM_SAVE** before saving a design element, **SCM_TOOL** when selecting a toolbar item and **SCM_ZOOM** when changing the zoom factor. The module startup **User Language** program call method is most useful for automatic system parameter setup as well as for key programming and menu assignments. The element save and load program call methods can be used to save and restore element-specific parameters such as the zoom area, color setup, etc. The toolbar selection event must be used to start **User Language** programs which are linked to toolbar elements. The zoom event can be used to apply an update request to a design view management feature.

Bartels User Language also provides system functions for performing key programming, changing menu assignments and defining toolbars. These powerful features can be applied for user interface modifications. Please note that a large number of additional functions included with the **Schematic Editor** menu are implemented through the **User Language** programs delivered with the BAE software.

See the Bartels User Language Programmer's Guide for a detailed description of the **Bartels User Language** (chapter 4.2 lists all **User Language** programs provided with the BAE software).

Neural Rule System

A series of advanced **Bartels AutoEngineer** features are implemented through the integrated **Neural Rule System**. See chapter 6.3.1 for the rule system applications provided with the **Schematic Editor**.

2.2 SCM Library Symbol Design

The **Bartels AutoEngineer** is shipped with a series of extensive SCM libraries. Nevertheless, you might require a certain symbol which has not yet been defined in these libraries. This section shows in detail how to create SCM library symbols. Some example symbols are created step-by-step starting with the lowest DDB hierarchy level. I.e., first of all a pin symbol (on marker level) is defined, subsequently an SCM part symbol (on symbol level) is defined and, finally, two label symbols (on label level) are defined. All these symbols will be stored to a DDB file named **`demo.ddb`**. Use the following commands to move to the BAE examples directory (e.g., **`c:\baejobs`**), and start the **Bartels AutoEngineer**:

```
>  C: ↵
>  cd c:\baejobs ↵
>  bae ↵
```

The BAE main menu is activated, and you can start the **Schematic Editor** with the following command:

Schematic

The **Schematic Editor** is actiavted, and you can create SCM library elements. Before generating your own symbols you should familiarize yourself with the conventions used for circuit symbol design. Company-specific conventions are frequently to be considered with regard to symbol size, pin naming and grouping (input, output, clock, reset, etc.), text size, pin placement grids, symbol origin, etc. Most important for the placement of a symbol on superior design levels are the symbol origin and the pin placement grid. It is advisable to choose a pin placement grid (e.g., 2mm) which allows for easy pin connections in the grid used on SCM sheet level (e.g., 1mm); the symbol origin should be set accordingly.

Figure 2-1 shows the SCM library symbols to be created in the following sections.

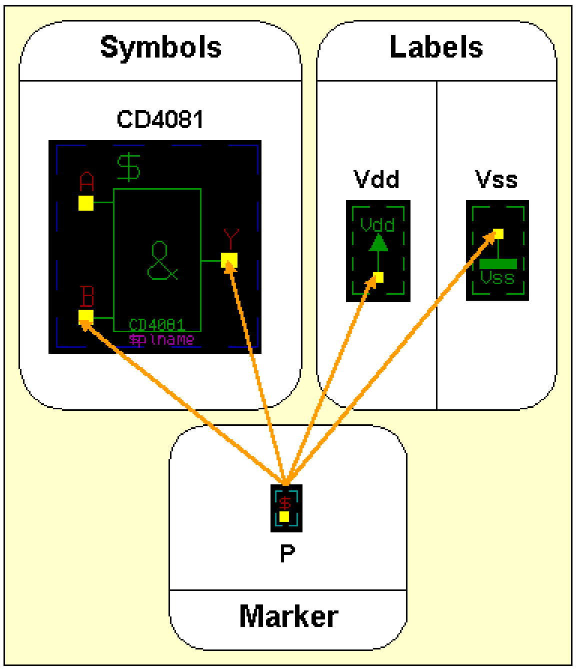


Figure 2-1: SCM Library Symbols

2.2.1 Creating SCM Markers

Creating a new Marker Symbol

Use the following commands to create a new marker element named **p** with a size of 10 by 10 mm in the **demo.ddb** DDB file:

- File ▮▯▯
  - New ▮▯▯
    - Marker ▮▯▯

File Name ?	demo ↵
Element Name ?	p ↵
Element Width (mm/") ?	10 ↵
Element Height (mm/") ?	10 ↵

The display now shows a square frame with a cross in the middle. The frame describes the element boundaries of the marker, and the cross marks the position of the element origin.

Defining the Contact Area

Contact areas on pin symbols are utilized for displaying net list changes on SCM sheet level. As soon as a pin is connected correctly, the contact area defined on the corresponding pin marker will disappear. Contact areas will not be plotted.

Use the following commands to define a square contact area with an edge length of 1mm on the currently loaded marker element:

- Graphic ▮▯▯
  - Add Contact Area ▮▯▯
    - ▯▯▮
      - Jump Absolute ▮▯▯
        - Absolute X Coordinate (mm/") ? 0.5 ↵
        - Absolute Y Coordinate (mm/") ? 0.5 ↵
    - ▯▯▮
      - Jump Relative ▮▯▯
        - Relative X Coordinate (mm/") ? -1 ↵
        - Relative Y Coordinate (mm/") ? 0 ↵
    - ▯▯▮
      - Jump Relative ▮▯▯
        - Relative X Coordinate (mm/") ? 0 ↵
        - Relative Y Coordinate (mm/") ? -1 ↵
    - ▯▯▮
      - Jump Relative ▮▯▯
        - Relative X Coordinate (mm/") ? 1 ↵
        - Relative Y Coordinate (mm/") ? 0 ↵
    - ▯▯▮
      - Done ▮▯▯

Defining the Reference Designator

The graphical design of the marker symbol is completed now. Nevertheless a pin reference designator should be defined on the marker. For this purpose the **$** text string is utilized as a variable for indicating the name of corresponding references on superior hierarchy levels. I.e., a **$** text string placed on marker level displays pertinent pin reference names on symbol level, a **$** text string placed on symbol level displays pertinent part reference names on SCM sheet level, etc.

Use the following commands to place the **$** text with a size of 2mm at coordinate [-0.5,0.5]:

```
Text                                  🖱
    Add Text                          🖱
                                      Text ? | $ ↵
        🖱
            Text Size                 🖱
                    Text Size ( 4.00mm) ? | 2 ↵
        🖱
            Jump Absolute             🖱
             Absolute X Coordinate (mm/") ? | -0.5 ↵
             Absolute Y Coordinate (mm/") ? | 0.5 ↵
```

Modifying the Element Boundaries

The marker symbol's element boundaries should be reduced in order to enclose the marker definition as densely as possible. This is accomplished with the following commands:

```
Settings                              🖱
    Upper/Right Border                🖱
        🖱
            Jump Absolute
             Absolute X Coordinate (mm/") ? | 0 ↵
             Absolute Y Coordinate (mm/") ? | 0 ↵
    Lower/Left Border                 🖱
        🖱
            Jump Absolute
             Absolute X Coordinate (mm/") ? | 0 ↵
             Absolute Y Coordinate (mm/") ? | 0 ↵
```

Saving the Marker Symbol

Use the following commands to save the marker symbol:

File
  Save Element

Now the new marker symbol named **`p`** is completely defined and stored to the DDB file **`demo.ddb`**. Use the following commands to list the marker(s) defined in **`demo.ddb`**:

File
  File Contents
    Marker
      File Name ?

An empty string to the file name prompt causes the system to use the file name of the currently loaded element (i.e., **`demo.ddb`** in our example). The system should produce the following listing of the markers contained in **`demo.ddb`**:

```
Type : Marker / File : demo.ddb

:p                   - End -
```

Hit the spacebar to continue.

2.2.2 Creating SCM Symbols

In this section we will create a new SCM library symbol named **`CD4081`**. This symbol can be created in a new or in an already existing library and/or job file. In our example we will store the new symbol to the currently processed DDB file named **`demo.ddb`**.

With the **`CD4081`**, it is advisable not to create an SCM symbol for the complete part, but to define the **`CD4081`** as single gate which then occurs four times in the corresponding package. The operations in the following section will produce the symbol definition shown in figure 2-2.

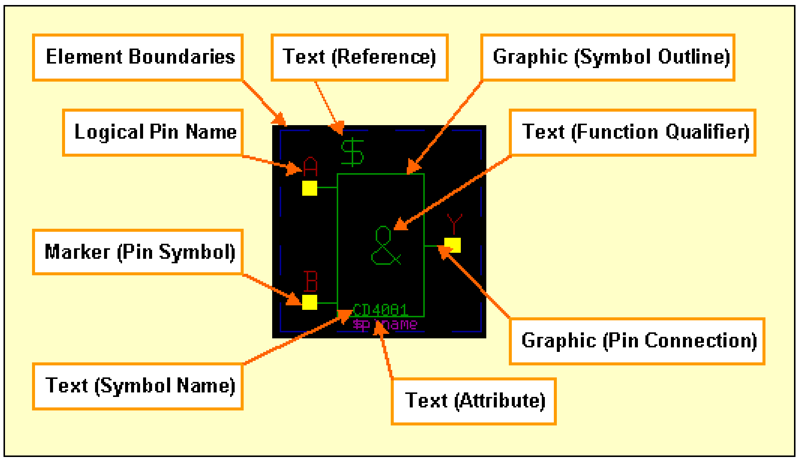


Figure 2-2: SCM Symbol CD4081

Creating a new SCM Symbol

Use the following commands to create a new symbol named **`CD4081`** in the DDB file **`demo.ddb`** (an empty string input is sufficient for the specification of the DDB file name if an element of file **`demo.ddb`** is still loaded):

File
New
Symbol

File Name ?	demo ↵
Element Name ?	cd4081 ↵
Sheet (C)omplex/(L)ogic/(G)ate/(D)iscrete/(M)an. ?	g ↵

The symbol size determines the element's size and boundary. The system does not allow for the placement of objects outside the element boundaries, however, the boundary can modified lateron. You can either enter the element width and element height manually, or simply select one of the following predefined symbol sizes:

Option	Element/Symbol Size
Complex	50 × 90 mm
Logic	30 × 30 mm
Gate	15 × 15 mm
Discrete	15 × 10 mm

Selecting the Pin Symbol

In the previous section, we defined a marker symbol with the name `p`. Use the following commands to select this marker as pin symbol for subsequent pin definitions on the currently loaded SCM symbol:

```
Settings
    Settings
        Select Pin Symbol
                        Pin Marker Name (pin) ?  p
```

Parameter settings as shown above can be utilized for assigning different pin symbols on a single SCM part symbol. On default, the system uses the marker named `pin` (make sure that this marker is available in the currently accessed SCM default symbol library; see also the **`SCMDEFLIBRARY`** command of the **BSETUP** utility program).

Placing the Pins

Now the pins of the symbol are to be defined. The **`CD4081`** part contains 4 equivalent AND gates with input pins **`A`** and **`B`** and output pin **`Y`**. These are the logical pin names (other logical pins could be named **`INPUT`**, **`OUT`**, **`CLK`**, etc.). Logical to physical pin assignment is applied with a corresponding logical library definition which can be translated with the **LOGLIB** utility program (see the **Packager** description in chapter 3.2 of this manual for more details). Equal names for logical and physical pins (1:1 pin assignments) are also allowed.

The marker level is the lowest SCM database hierarchy level. Each SCM part symbol consists of markers defining the part pins. I.e., the marker is the part symbol to be used on symbol level. This principle is applied on all database levels, thus enabling basic functions such as symbol placement to work in different database levels. Placing markers on symbol level will produce SCM part symbols, whilst placing SCM symbols on SCM sheet level produces circuitry.

Use the following commands to place the pins **`A`**, **`B`** and **`Y`** at coordinates [2,10], [2,2] and [12,6] on the currently loaded symbol:

```
Symbols
    Add Symbol
                        Part Name ?  a
                     Move to [2,10]
    Add Symbol
                        Part Name ?  b
                      Move to [2,2]
    Add Symbol
                        Part Name ?  y
                     Move to [12,6]
```

The markers placed with the commands above are representing the pins **`A`**, **`B`** and **`Y`** of the **`CD4081`** gate. Instant placement coordinates display is provided in the status line and/or info field whilst performing interactive placement operations.

A slash **`/`** preceding the part name causes a line to be displayed on top of the name and/or text. This feature can be used to name negated signals and/or pin names such as **`/y`**. This also works for normal text.

Graphic

The symbol now consists of all pin definitions required for the **CD4081** gate, but symbol geometry and/or graphical information for documentation purposes such as symbol outline, pin connection designators, function indicator, etc. is still missing.

Use the following commands to define a symbol outline for the **CD4081**:

```
Graphic
  Add Graphic Line
                                   Move to [4,1]
                                   Move to [4,11]
                                   Move to [10,11]
                                   Move to [10,1]
                                   Move to [4,1]

      Done
```

Use the following commands to draw a graphic line connecting pin **A** with the symbol outline:

```
Graphic
  Add Graphic Line
                          Move to Pin "A", [2,10]
                                   Move to [4,10]

      Done
```

With the commands above a horizontal line has been drawn from pin **A** to the symbol outline, thus indicating that pin **A** is defined on symbol **CD4081**. Use the following commands to connect also pin **B** and pin **Y** with the symbol outline:

```
Graphic
  Copy Graphic
                          Move to Pin "A", [2,10]
                           Move to Pin "B", [2,2]
  Copy Graphic
                          Move to Pin "A", [2,10]
                                    Move to [10,6]
```

Text and Attributes

The graphical design of the **`CD4081`** part symbol is completed now. Nevertheless, some texts should be defined on the symbol. First of all a reference designator should be placed on the symbol. For this purpose, the **`$`** text string is utilized as a variable for indicating the name of corresponding references on superior hierarchy levels. I.e., a **`$`** text string placed on symbol level will display pertinent part reference names (e.g., **`IC01`**, **`R20`**, **`V2`**) on the circuit drawing.

Use the following commands to place the **`$`** text string with a size of 3mm on the symbol:

```
Text
  Add Text
    Text ? $
    Text Size
      Text Size ( 2.00mm) ? 3
    Move to [4,11]
```

Use the following commands to place the **`&`** string (text size 4mm) for denoting the AND gate function of the symbol:

```
Text
  Add Text
    Text ? &
    Text Size
      Text Size ( 3.00mm) ? 4
    Move to [6,4]
```

Use the following commands to place the gate name **`CD4081`** with a text size of 1mm:

```
Text
  Add Text
    Text ? CD4081
    Text Size
      Text Size ( 4.00mm) ? 1
    Move to [5,1]
```

Use the following commands to define a commentary text referring the **`$plname`** attribute (text size 1mm is still activated):

```
Text
  Add Text
    Text ? $plname
    Commentary Text
    Move to [5,0]
```

The **Schematic Editor** distinguishes between standard and commentary text. Both standard text and commentary text are displayed on the screen, but only standard text will be plotted, thus featuring a mechanism of fading out certain text and/or attributes on the generation of plot data.

Texts can also be defined with frames. Such definitions are possible with the Frame 1, Frame 2 and Open Frames options from the submenu to be activated with the right mouse button whilst moving texts with the Add Text, Move Text and/or Copy Text function. Frame 1 creates a surrounding box at 1/8 text height distance. Frame 2 creates a surrounding box at 1/4 text height distance. Open Frames removes the vertical frame line at the text origin. Open Frames can be applied to define labels with dynamically adjusted text frames to be attached to connections. Using large pens for plot outputs can cause the negation lines of small texts being merged with text frames. The Standard Text option resets all text frame definitions.

The submenus of the Add Text, Move Text and Copy Text functions also provide the No Rotation and Center options for switching the selected text into norotate mode and/or center aligning the text at its placement position. Text in norotate mode is displayed and/or plotted without rotation on any database hierarchy level. The No Rotation and Center text modes can be reset by applying the Standard Text option from the same submenu.

Bartels AutoEngineer supports arbitrary part attributes on symbol level. A particular attribute is defined by a text consisting of the `$` character followed by the (lowercase) attribute name. Attributes are like variables, i.e., on SCM sheet level, values can be assigned to attributes. The attributes listed in table 2-1 have special meanings due to BAE-specific conventions.

Table 2-1: Special BAE Attributes

Attribute Name	Function/Meaning
`$`	Reference Name
`$$`	Logical Reference Name
`$llname`	Logical Library Name
`$plname`	Physical Library Name (and/or Alternate Layout Part Package Types)
`$ulname`	Used Library Name (Alternate Layout Part Package Type Assignment)
`$rpname`	Requested Part Name
`$rbname`	Requested Backannotated Part Name
`$gp`	Gate Pin
`$blkname`	Hierarchical Block Name
`$pageref`	SCM Label Sheet Reference Name List
`$pagecref`	SCM Label Sheet Reference Comment List
`$orgname`	Original/internal SCM symbol/part name(s) of layout part
`$pagename`	Schematic sheet name(s) of layout part
`$blkrname`	Schematic block symbol reference name(s) of layout part
`$rlname`	Requested Logical Library Name
`$rlext`	Requested Logical Library Name Extension
`$val`	Value
`$pow`	Power
`$type`	Component Type
`$comment`	Comment (English)
`$commentge`	Comment (German)
`$manufacturer`	Component Manufacturer
`$partside`	Part Side (`top` = unmirrored, `bottom` = mirrored)
`$pltbaeversion`	**Bartels AutoEngineer** Software Version Number (read-only)
`$pltbaebuild`	**Bartels AutoEngineer** Software Build Number (read-only)
`$pltfname`	Project File Path Name
`$Pltfname`	Project File Path Name (uppercase)
`$pltfsname`	Project File Name (without Directory Path)
`$Pltfsname`	Project File Name (uppercase, without Directory Path)
`$pltpagecnt`	Project SCM Sheet Count
`$pltename`	Element Name
`$Pltename`	Element Name (uppercase)
`$pltdatede`	Current Date (German Format)
`$pltdateus`	Current Date (US Format)
`$pltdate2de`	Current Date (German Format; two-digit year display)
`$pltdate2us`	Current Date (US Format; two-digit year display)
`$plttime`	Current Time

Attribute Name	Function/Meaning
`$pltsdatede`	SCM/Layout Save Date (German Format)
`$pltsdateus`	SCM/Layout Save Date (US Format)
`$pltsdate2de`	SCM/Layout Save Date (German Format; two-digit year display)
`$pltsdate2us`	SCM/Layout Save Date (US Format; two-digit year display)
`$pltstime`	SCM/Layout Save Time
`$pltpname`	Layout element name for last **Packager** run
`$Pltpname`	Layout element name for last **Packager** run (uppercase)
`$pltpdatede`	Date (German Format) of last **Packager** run
`$pltpdateus`	Date (US Format) of last **Packager** run
`$pltpdate2de`	Date (German Format; two-digit year display) of last **Packager** run
`$pltpdate2us`	Date (US Format; two-digit year display) of last **Packager** run
`$pltptime`	Time of last **Packager** run
`$pltcname`	Layout Element Name
`$pltcdatede`	Date of last name update (German Format)
`$pltcdateus`	Date of last name update (US Format)
`$pltcdate2de`	Date of last name update (German Format; two-digit year display)
`$pltcdate2us`	Date of last name update (US Format; two-digit year display)
`$pltctime`	Time of last name update
`$pltfbname`	File Base Name - Element-specific Project File Name without `.ddb` extension
`$Pltfbname`	File Base Name - Element-specific Project File Name without `.ddb` extension (uppercase)
`$pltfbsname`	File Base Short Name - Element-specific Project File Name without `.ddb` extension and directory path
`$Pltfbsname`	File Base Short Name - Element-specific Project File Name without `.ddb` extension and directory path (uppercase)
`$pltecomment`	DDB Element Comment - Element-specific comment text
`$drcblk`	Design Rule Check Block (Pin/Net Attribute)
`$net`	Pin Net Name
`$netname`	Net Name (Net Attribute)
`$nettype`	Net Type (Pin/Net Attribute)
`$powpin`	Power Pin Definition (Pin Attribute)
`$viastk`	Via Padstack Type (Net Attribute)
`$notest`	Exclusion from Automatic Test Point Generation (Net Attribut)
`$routdis`	Exclusion from Autorouting Process (Net Attribute)
`$layers`	Autorouting Layers (Net Attribute)
`$@`	Layout Part Name (Padstack Attribute)

The `$pltbaeversion` and `$pltbaebuild` display the **Bartels AutoEngineer** software version and build numbers. This allows for the inclusion of software version information on schematic plans as required for ISO certification.

The `$`, `$llname`, `$plname`, `$gp`, `$ulname` and `$blkname` attributes are automatically processed by the system. The reference designator attribute (`$`) is replaced with the name of the referenced symbol on the superior hierarchy level. On SCM sheet level, the part and/or pin names produced by the **Packager** are displayed through the `$` attribute. `$$` attribute definitions on SCM symbol level can be used to retain logical SCM part name display on SCM sheet level after **Packager** processing. `$$` attribute definitions on SCM marker level can be used to retain logical SCM symbol pin names display on SCM sheet level after **Packager** processing.

Symbol level **`$llname`** text definitions are substituted with symbol macro names on SCM plan level. The **Packager** automatically assigns the **`$llname`** attribute value, i.e., the **`$llname`** attribute is not available for interactive and/or explicit attribute value assigments. Symbol name display through **`$llname`** text definitions on SCM symbol label eliminate the need for redefining symbol name texts when creating new SCM symbols from existing SCM symbols.

The logical library name (**`$llname`**) can also be used for displaying SCM symbol names on the layout. The **Packager** automatically transfers SCM symbol pin names to the **`$llname`** pin attribute, thus allowing logical pin name display in the layout through **`$llname`** text definitions on padstack level.

The physical library name (**`$plname`**) is used for assigning non-default layout packages and/or alternative layout part package types. I.e., selectable SCM symbols can be assigned to layout packages different from the logical library default assignment (e.g., a **`so14`** SMD package could be used instead of the default **`dil14`** by assigning the value **`so14`** to the **`$plname`** attribute, or a choice of layout package types could be passed to the layout system through a **`[dil14,so14]`** **`$plname`** attribute value assignment; see the descriptions of the **LOGLIB** utility program and the **Packager** for more details).

The Backannotation transfers layout part package type assignments back onto **`$ulname`** (Used Library Name) SCM symbol attributes, thus allowing for alternate layout part package type assignments display and/or query on SCM plan level.

The requested part name (**`$rpname`**) is used for forcing the packaging of certain SCM symbols to certain layout parts (e.g., assignment to part **`IC28`** instead of the automatically generated assignment); this function is required for special part definitions consisting of different SCM symbols such as relays with strip and contact, opamps with variable power supply, etc.

The requested backannotated part name (**`$rbname`**) provides the same functionality like **`$rpname`**, but without prohibiting subsequent layout part name changes. I.e., the **Layout Editor** allows for **`$rbname`** part names to be changed and the Backannotation transfers changed layout part names back onto the **`$rbname`** attributes of the corresponding SCM symbols.

The gate pin (**`$gp`**) attribute can be used to assign SCM symbols to layout gate positions by setting the **`$gp`** value to the desired gate's first pin name from the **`xlat`** command of the corresponding logical library definition (see also **LOGLIB**). When using **`$gp`** assignments, the **`$gp`** value should be set for all gates of the layout part to avoid gate assignment conflicts. The **`$gp`** attribute allows for specific assignments of dual operation amplifier components, and it can even be used to define connector pin symbols for single pins of multi-pin connectors where each pin is defined as a gate and the **`$gp`** attribute specifies the name of the connector pin. Gate and pins assigned through the **`$gp`** attribute are excluded from layout pin/gate swaps.

The hierarchical block name (**`$blkname`**) is used for hierarchical circuit design; corresponding attribute values are automatically set by the **Packager** denoting the name of the hierarchical block where a certain part is referenced from.

The **`$pltpagecnt`** can be used in the schematics to display the total number of schematic sheets of the current project.

On SCM sheet level, label level **`$pageref`** text definitions are substituted with the list of all SCM sheets on which the corresponding net is used.

On SCM sheet level, label level **`$pagecref`** text definitions are substituted with the list of comments of all SCM sheets on which the corresponding net is used.

The **Packager** automatically assigns the **`$orgname`** (original/internal SCM symbol/part name(s) and/or SCM (sub-)net name(s)), **`$pagename`** (schematic sheet name(s)) of layout part') and **`$blkrname`** (schematic block symbol reference name(s)) attributes for schematic sub-block tracking information to layout parts.

`$rlname` (Requested Logical Library Name) symbol attribute value assignments cause the **Packager** to transfer symbols using non-default logical library definitions. This allows for the selection of part definitions with, e.g., manufacturer-specific library attribute settings or non-default package assignments. The non-default **LOGLIB** entry assigned through **`$rlname`** must refer to the same part class like the default **LOGLIB** definition (for **LOGLIB** part class definitions see also chapter 7.11). **`mainpart/subpart`** symbol sets must be assigned to corresponding parts. I.e., heterogeneous symbol definitions such as **`amain/asub`** and **`bmain/bsub`** can be assigned to **`amain/asub`** or **`bmain/bsub`**, but not to **`amain/bsub`** or **`bmain/asub`**.

$rlext (Requested Logical Library Name Extension) symbol attribute value assignments work similar to **$rlname** assignments. **$rlext** assignments cause the **Packager** to transfer symbols using non-default logical library definitions with specific name extensions (separated from the base name by an underscore _). This allows for the selection of part definitions with, e.g., standardized manufacturer-specific library attribute settings or non-default standardized package assignments.

The predefined attributes **$val** (value), **$pow** (power), **$type**, **$comment**, **$commentge** and **$manufacturer** are optionally used throughout the delivered BAE libraries. The user is free to apply these attribute definitions and/or to assign corresponding attribute values on demand.

The **$partside** part system attribute can be used to block (attribute value **top**) or force (attribute value **bottom**) part mirroring. These settings have priority over any other mirror modes during interactive and automatic part placement operations. Parts violating these placement preferences are marked by the DRC, and the Utilities / Report functions list such parts in a part side error list. The ROUTE library provides the tag_sym_partside tag symbol for assigning **$partside** attribute values.

The **$plttime** (current time), **$pltdatede** (current date, German Notation) and **$pltdateus** (current date, US notation) system attributes are substituted with the current time and/or the current date when displayed and/or plotted on SCM sheet or layout level. The **$pltstime** (layout save time), **$pltsdatede** (layout save date, German Notation) and **$pltsdateus** (layout save date, US notation) system attributes are substituted with the time and/or the date the currently loaded layout was last saved. Automatic time and date substitutions are applied throughout all database hierarchy levels (marker, symbol, label, sheet and/or pad, padstack, part, layout), unless other attribute values are explicitly set for these attributes.

The **$pltpname** (layout element name), **$pltpdatede** (date, German Notation), **$pltpdateus** (date, US notation) and **$pltptime** (time) system attributes are substituted with the layout element name and/or the date and/or the time of the last **Packager** run when displayed and/or plotted on SCM sheet or layout level.

The **$pltdate2de**, **$pltdate2us**, **$pltsdate2de**, **$pltsdate2us**, **$pltpdate2de** and **$pltpdate2us** attributes are two-digit year display versions of the **$pltdatede**, **$pltdateus**, **$pltsdatede**, **$pltsdateus**, **$pltpdatede** and **$pltpdateus** attributes.

The **$pltcname**, **$pltcdatede**, **$pltcdate2de**, **$pltcdateus**, **$pltcdate2us** and **$pltctime** attributes are substituted with the layout element name and the date and time of the last name update. These attributes are not only updated by the **Packager** but also by the Backannotation.

The **$pltfbname** (File Base Name) and **$pltfbsname** (File Base Short Name) system attributes can be used for displaying the current element's project file name. **$pltfbname** displays the project file name without the **.ddb** extension. **$pltfbsname** displays the project file name without the **.ddb** extension and without the directory path.

The **$pltecomment** system attribute can be used to assign and display DDB element comments. Element comments can be assigned through the File / Element Comment submenu or, for the currently loaded element, through the Settings / Settings dialog. The element selection dialogs are displaying element comments together with the element names. The SCM PDF output functions are displaying SCM sheet comments rather than SCM sheet element names when creating tables of contents for SCM plans.

The **Packager** supports net name assignments through the **$netname** net attribute. Such net names have priority over net name assignments through label connections. I.e., the **$netname** attribute can be used to control net name assignments for nets with different labels which would otherwise be named after the label which comes first in the alphabet.

$nettype pin attributes are automatically transferred to connected nets. The **$nettype** value **mixed** is assigned to nets with different **$nettype** attribute values. The **BAE HighEnd Packager** automatically transfers **$drcblk** pin attributes to connected nets. The **$drcblk** attribute value addresses a **BAE HighEnd** design rule check parameter block to be assigned to the corresponding net.

The **$powpin** pin attribute setting causes the system to set a symbol pin's connection width to the power pin connection width defined through the **net** command and power width net attribute. The tag_pin_powerpin tag symbol from the ROUTE symbol library can be used for **$powpin** pin attribute assignments. Alternatively, the system supports fixed **$powpin** assignments through the logical definition of the symbol:

```
newattr "$powpin" = "1" : (pinname);
```

The **`$viastk`** net attribute can be used to assign non-default via types to nets. Net-specific via padstack assignments are considered by the **Autorouter**. **`$notest`** net attribute assignments can be used to deactivate automatic test point generation by the **Packager** for specific nets. **`$routdis`** net attribute assignments can be used to exclude specific nets from the autorouting process. The **`$layers`** net attribute can be used to assign net-specific routing layers (comma-separated signal layer numbers) for the autorouting process in **BAE HighEnd**.

The **`$@`** attribute is padstack-specific. Text on padstack level is on layout level subsituted with the name of the part on which the padstack is placed. This feature can be useful if part names need to be displayed for pins which are placed outside the part body (e.g. during Move Pin operations).

BAE supports not only the attributes listed in table 2-1, but also arbitrary user-defined attributes such as **`$tolerance`**, **`$identno`**, **`$partnumber`**, **`$price`**, **`$supplier`**, **`$delay`**, **`$insertion_height`**, etc.

The Move Attribute function from the Text and/or Symbol menu can be used for moving and/or placing selectable symbol attributes. Attribute text offsets defined through Move Attribute override the global text offset set with Move Name. The attribute to be moved is selected through a mouse-click on the attribute text. Since symbol names are internally stored as attributes, the Move Attribute function can also be used to move symbol names without changing the placement of other symbol attributes.

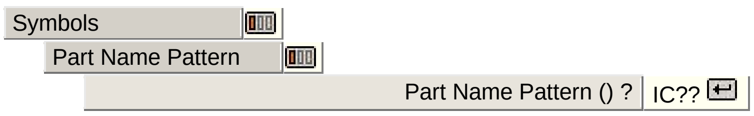
Attribute value setting is performed either in the **Schematic Editor** (using the Assign Value function from the Symbols menu) or by fixed attribute value assignment in the logical library.

A dialog box for simultaneously displaying up to 12 symbol attributes is provided with the Assign Value function under Windows and Motif. For symbols with more than 12 attribute definitions, Next and Previous buttons are provided for forward and/or backward scrolling. The dialog box contains a line for each attribute. This line consists of a No Value button, an attribute value input/edit field and the attribute name/label display. The No Value button can be used to clear the attribute value setting which is indicated through a **`!not_set!`** attribute value display. Clearing the attribute value is different from specifying an empty string. Empty string attribute values are assigned and transferred as such to the net list, whilst clearing an attribute value prevents the system from assigning and/or transferring any attribute value to the net list.

Attributes are part of the net list and are annotated by the **Packager**, thus providing access from the layout (e.g., by appropriate text definitions on selectable documentary layers). The **USERLIST** utility program and **Bartels User Language** can be utilized for performing automatic attribute value assignments and/or for evaluating attributes for cost analysis, production planning, manufacturing control, purchasing and ordering management, producing output data for third party systems, etc.

Part Name Pattern

BAE supports patterns for symbol names to follow if a name is not specified when the symbol is placed on the SCM sheet. Use the following commands to define a part name pattern for the currently loaded CD4081 symbol:


Symbols
Part Name Pattern
Part Name Pattern () ? IC??

With the commands above a part name pattern consisting of the string **`IC`** and two digits for the numbering is defined. The default pattern for symbols without a part name pattern definition is **`N????`**, i.e., the default pattern consists of **`N`** and four digits for the numbering. The start number for automatic part numbering emerges from 10 with pattern **`??`**, 100 with pattern **`???`**, 1000 with pattern **`????`**, etc. This helps to avoid conflicts when renaming layout parts afterwards. Parts are renamed on the layout e.g., for improving the legibility of the insertion plan (for manual insertion). This is usually more important than designating SCM parts according to some devised rules since adequate SCM documentation is achieved by displaying the part and/or circuit function with appropriate symbol definitions. We recommend to apply the automatic part naming features at the placement of SCM symbols and to perform renaming on the layout only. Functions for changing net list part names and/or for automatic part numbering are provided with both the **Layout Editor** and the **Autoplacement**.

Modifying the Element Boundaries

Use the following commands to set the upper right corner of the element boundaries to coordinate [14,14]:

- Settings
  - Upper/Right Border
    - Move to [14,14]

The symbol origin refers to the pick point in superior hierarchy levels. This is the point of the SCM symbol where it is picked for placement and/or movement on the SCM sheet. Usually, it makes sense to set the origin at a certain pin coordinate (e.g., left top or left bottom). Use the following commands to define the origin on pin **`A`** (coordinate [2,10]) of the currently loaded **`CD4081`** symbol:

Defining the Origin

- Settings
  - Set Origin
    - Move to Pin "A", [2,10]

Saving the Symbol

The definition of symbol **`CD4081`** is completed now. Do not forget to save this symbol with the following commands:

- File
  - Save Element

2.2.3 Creating SCM Labels

Label symbols are used on SCM sheet level to define signals and/or net names. Labels can be utilized to connect signals from different schematic sheets of the same job file. The Add Label function from the Symbols menu is used for placing labels onto the currently loaded SCM sheet. If a label symbol is available with the same name as specified for the signal name, then the corresponding label symbol is loaded (thus displaying special graphic e.g., for the ground signal). If no label symbol with the specified net name is available, then the label symbol named `standard` is used on default. The `standard` label symbol should always contain a reference text (`$`) for displaying the net name. BAE uses two more predefined label symbols named `bustap` and `port`, respectively. The `bustap` label symbol is used for tapping bus connections (use the Tap Bus function from the Connections menu). The `port` label symbol is used for defining module ports in hierarchical circuit designs (use the Add Module Port function from the Symbols menu). In this section we will create two label symbols named `vss` and `vdd` in DDB file `demo.ddb`. These symbols are intended especially for CMOS ground and power supply.

Creating a new Label Symbol

Use the following commands to produce a new label symbol (i.e. a new element of type label) named `vss` in the DDB file `demo.ddb` (if an element of `demo.ddb` is currently loaded, then the file name can be specified either by selecting the Project button from the popup menu or by specifying an empty string, i.e., pressing the return key ↵to the file name query):

File	🖱
New	🖱
Label	🖱
File Name ?	demo ↵
Element Name ?	vss ↵
Sheet (S)tandard/(M)anual ?	m ↵
Element Width (mm/") ?	6 ↵
Element Height (mm/") ?	10 ↵

Placing the Pin Symbol

Use the following commands to place the marker symbol `p` at coordinate [3,7]:

Symbols	🖱
Add Symbol	🖱
Library Element Name ?	p ↵
Move to [3,7]	🖱

Graphic

Use the following commands to draw a graphic line (starting at the marker and ending at coordinate [3,4]) and a rectangle-shaped graphic area:

```
Graphic
  Add Graphic Line
                         Move to Marker, [3,7]
                                 Move to [3,4]
    
      Done
  Add Graphic Area
                                 Move to [1,3]
                                 Move to [1,4]
                                 Move to [5,4]
                                 Move to [5,3]
    
      Done
```

Text

Use the following command to place the label name text string **Vss** (text size 2mm) at coordinate [1,1]:

```
Text
  Add Text
                                    Text ? Vss
    
      Text Size
                       Text Size ( 1.00mm) ? 2
                                 Move to [1,1]
```

Defining the Origin

Use the following commands to set the label origin to the current marker position (i.e., at coordinate [3,7]):

```
Settings
  Set Origin
                         Move to Marker, [3,7]
```

Saving the Label Symbol

The definition of the label symbol **vss** is completed now. Do not forget to save this symbol with the following commands:

```
File
  Save Element
```

Creating a new Label Symbol from an existing

In this section we will create the label symbol **vdd** from the previously defined label symbol **vss**. Use the following commands to copy the currently loaded label symbol **vss** to **vdd** and load the copied label symbol **vdd**:

```
File
    Save Element As
                                   File Name ?
                                Element Name ?  vdd
    Load Element
        Label
                                   File Name ?
                                Element Name ?  vdd
```

The display now shows the **vdd** label symbol, which still looks like the **vss** label symbol. We will now make some modifications to create the desired **vdd** symbol design.

Use the following commands to delete the text **Vss** and the rectangle-shaped graphic area:

```
Text
    Delete Text
                               Move to [-2,-6]
Graphic
    Delete Graphic
                               Move to [-2,-4]
```

Use the following commands to move down the vertical graphic line:

```
Graphic
    Move Graphic
                                Move to [0,-3]
                                Move to [0,-5]
```

Use the following commands to move down the marker symbol:

```
Symbols
    Move Symbol/Label
                                 Move to [0,0]
                                Move to [0,-5]
```

Use the following commands to create a V-shaped graphic area, i.e., an arrow-head pointing to top (the angle lock display option must be switched off to create this irregularly shaped triangle):

```
Graphic
    Add Graphic Area
                               Move to [-1,-2]
                                Move to [1,-2]

    Grids/Rotation
        Rotation Off
                                 Move to [0,0]

            Done
```

Use the following command to place the label name text string **Vdd** at coordinate [1,1] (text size 2mm is still activated):

```
Text
    Add Text
                                         Text ? Vdd
                                Move to [-2,0]
```

Use the following commands to set the origin to the current marker position (i.e., at coordinate [0,-5]):

```
Settings
    Set Origin
                        Move to Marker, [0,-5]
```

The definition of the **vdd** label symbol is now completed. Do not forget to save this symbol with the following commands:

```
File
    Save Element
```

2.3 Designing SCM Circuits

This section describes how to create SCM plans using the BAE **Schematic Editor**. Two example SCM sheets named **`sheet1`** and **`sheet2`** will be edited in the **`demo.ddb`** DDB file. **`sheet1`** will contain the circuit logic. On **`sheet2`**, the plug pin assignment and some net attributes for the **Autorouter** will be defined.

The following paragraphs will introduce the basic functions for creating SCM plans, placing symbols, assigning attribute values, creating connections, placing labels, defining busses, placing texts and drawing graphic. This section also shows how to use special features such as virtual symbol definitions, group functions, defining plug pin assignments, setting net attribute values, using templates, etc.

Use the following commands to move to the BAE examples directory (i.e., where the example job file **`demo.ddb`** resides, e.g., **`c:\baejobs`**), and start the **Bartels AutoEngineer**:

```
>  C: ↵
>  cd c:\baejobs ↵
>  bae ↵
```

Use the following command to start the **Schematic Editor**:

Schematic

The **Schematic Editor** is activated and you can start to edit SCM plans.

2.3.1 Creating and Editing SCM Plans

Creating a new SCM Sheet

Use the following commands to create a new schematic sheet named `sheet1` with paper size A4 in the DDB file `demo.ddb`:

```
File
    New
        Sheet
                              File Name ?  demo ↵
                           Element Name ?  sheet1 ↵
            A4 Landscape
```

A frame designating the element boundaries of the SCM sheet will appear on the screen. If the system issues an error message such as `Element does already exist!`, then an SCM sheet with the specified element name has already been generated. In this case the requested element cannot be created, but must be loaded (see below). The element name query is indispensable due to the fact that a schematic circuit drawing can consist of multiple sheets. You can specify arbitrary sheet names such as `sheet1`, `sheet2`, `plan_1`, `1`, `2`, `io`, `memory`, `plug_pinning`, etc. An options menu for selecting the sheet size is activated after specifying the file and element names. This allows for the selection of predefined A0, A1, A2, A3, A4, Letter (8.5*11 Inches) and Tabloid (11*17 Inches) formats in either portrait or landscape orientation. The Other Format option can be used to specify any other sheet size through subsequently activated element width and height prompts.

Use the following commands to store the new SCM sheet and return to the BAE main shell:

```
File
    Save Element
File
    Main Menu
```

Editing an existing SCM Sheet

Start the **Schematic Editor** and use the following commands to reload the previously created SCM sheet element (you can select the file and element name via popup menu and mouse-pick since you are now accessing an existing element):

```
File
    Load
        Sheet
                              File Name ?  demo ↵
                           Element Name ?  sheet1 ↵
```

The system now loads the previously created SCM sheet. An error message such as `File not found!` is issued if the requested DDB file is not available. An error message such as `Element not found!` is issued if the requested element is not available in the selected DDB file.

When entering the **Schematic Editor**, the system knows the name of the file previously processed in any other BAE module. On subsequent file name queries this global project name can alternatively be specified by selecting the Project button from the file name popup menu or by entering an empty string (i.e., by pressing the return key ↵) to the file name prompt.

2.3.2 Symbols

On SCM sheet level, the appropriate functions from the Symbols menu are used for placing, moving and deleting symbols, labels and module ports, for assigning attribute values and for querying symbol logic definitions. The symbol placement routines support access to different, selectable SCM symbol libraries. The functions provided with the Symbols menu are essentially applied for generating the part list and for defining signal names.

Input Grid

BAE supports arbitrary grids and also grid-free input. Nevertheless, it is recommended to use a reasonable grid for placing the symbols to enable easy pin connections. Use the following commands to set a 2mm input grid with grid lock (this is intended for subsequently creating pin connections in a 1mm grid):

```
View
    Grids/Rotation
        Set Input Grid
        2.0 mm
        Grid+Rotation On
```

Library Access

The Select Library function from the Settings menu is used for selecting the SCM library, from which the symbols should be loaded. Use the following commands to check on how the library path is currently set:

```
Settings
    Select Library
        Library Name ('c:/baelib/stdsym.ddb') ? ↵
```

The name of the currently selected library is indicated with the library name prompt. When calling the **Schematic Editor**, a default library path is automatically set according to the system setup (see also chapter 7.2 of this manual for a description of the **BSETUP** utility program).

An empty string input to the library name query does not change the current library path setting. A dash string input (`-`) resets the library path setting (no library selected). `!` and/or `.` input restores the default library path defined with the setup. Under Windows, the library file name is specified through file dialog box. Use the following commands to check the library path settings:

```
Settings
    Select Library
        Library Name ('c:/baelib/stdsym.ddb') ? - ↵
    Select Library
        Library Name ('') ? ! ↵
```

The **Bartels AutoEngineer** supports a most flexible database concept. This concept allows any DDB file to be used as symbol library. When loading a symbol the system will first check the job-specific library (i.e., the currently edited DDB file) for the requested library element. If the element is not contained in the project file, then the search is expanded onto the currently selected library. When loading a symbol from an external library file the **Schematic Editor** will automatically create a complete copy of the requested library symbol in the current project file, and subsequent requests for the same symbol will from then on refer to the job-specific library. Figure 2-3 illustrates the **Schematic Editor** library access concept.

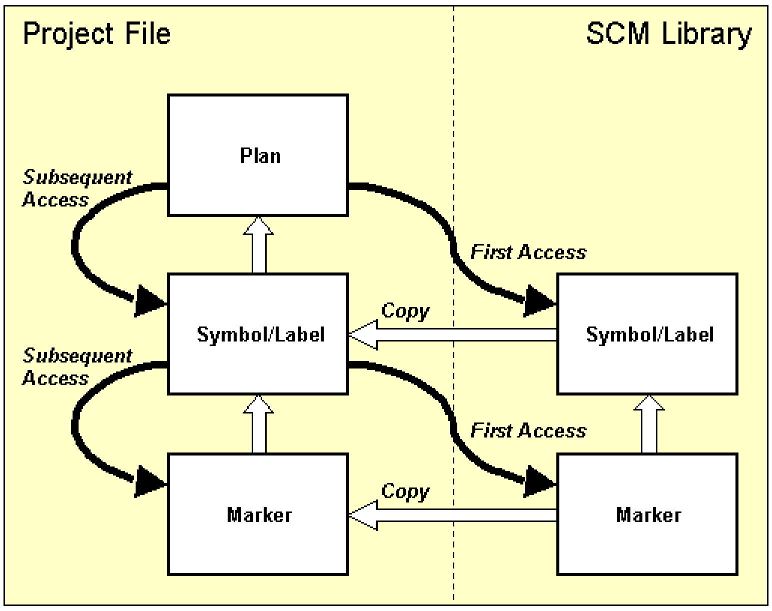


Figure 2-3: SCM Library Access

Changing the library path setting is meaningful only if a series of different symbols are to be loaded from a library file which is not accessible through the predefined library path. Use the following commands to set the library path to the `demolib.ddb` DDB file of the current directory:

Settings
Select Library
Library Name ('c:/baelib/stdsym.ddb') ? demolib

With the commands above the library path is set to the `demolib.ddb` DDB file of the working directory, i.e., requested symbols on subsequent load operations are copied from `demolib.ddb`.

Placing Symbols

Use the following commands to load the transistor symbol `tr_bc517` with part name `V1`, and place it at coordinate [240,130]:

- Symbols
  - Add Symbol
    - Part Name ? — v1 ⏎
    - Library Element Name ? — tr_bc517 ⏎
    - Move to [240,130]

The part name query is used for specifying the symbol reference (e.g., `IC10`, `R8`, `U004`, ...). If the user types in an empty string (by pressing the return key ⏎), then the system will automatically generate a part name according to the part name pattern defined with the corresponding symbol (see also chapter 2.2.2, Creating SCM Symbols). An empty string input to the part name query can also be activated by pressing a mouse button. The system will issue an error message when the part name space defined with the part name pattern is exhausted, i.e., if no new part name corresponding to the name pattern can be generated automatically any more.

The part names used throughout a schematic design must be unique to avoid conflicts in subsequent **Packager** and Backannotation processes. Part name uniqueness is implicitly ensured when using the automatic part naming facilities. With explicit part naming the system performs special part name uniqueness checks to avoid name conflicts. If the specified part name is already used on the currently loaded SCM sheet, then the user must confirm to use this part name. If the part name is used on the currently loaded sheet, then the corresponding part is deleted and/or replaced with the new part (with a different SCM symbol on request and without any attribute value settings). However, if the specified part name is used on a different sheet, then the corresponding part will not be deleted automatically, i.e., it is up to the user to delete this part later to avoid name conflicts on subsequent **Packager** processes.

The library element name query is used for selecting the library symbol (e.g., `74as00`, `c`, `led`, `pal_20r4`, `i80286`, etc.). A mouse-click or a question-mark input string (`?`) to the library element name query activates a popup menu with a list of all library files available with the predefined default library path. The Lib. button or `>` input to the library file name prompt selects the standard SCM library defined through the Select Library function from the Parameters menu. The Project button from this menu provides optional access to the job-specific library in the currently processed DDB file. After selecting the library file, another popup menu with a list of all symbols available in the selected library file is activated.

Use the following commands to place one capacitor symbol `c` and one resistor symbol `r`, both with automatically generated part name:

- Symbols
  - Add Symbol
    - Part Name ? — ⏎
    - Library Element Name ? — c ⏎
    - Move to [140,190]
  - Add Symbol
    - Part Name ? — ⏎
    - Library Element Name ? — r ⏎
    - Move to [60,130]

The Next Symbol function is used to place an automatically named part which refers the same symbol as with the preceding symbol load operation. Use the following commands to place three more resistor symbols on the circuit diagram:

- Symbols
  - Next Symbol/Label
    - Move to [100,130]
  - Next Symbol/Label
    - Move to [140,130]
  - Next Symbol/Label
    - Move to [180,130]
  - Next Symbol/Label
    - Move to [210,130]

Pressing the right mouse button during symbol placement will activate a submenu which provides functions such as specifying absolute placement coordinates, symbol rotation and symbol mirroring. Use the following commands to load another resistor symbol, rotate it and place it at coordinate [170,170]:

```
Symbols ▮▯▯
    Next Symbol/Label ▮▯▯
        ▯▯▮
            Rotate Left ▮▯▯
        Move to [170,170] ▮▯▯
```

The **Schematic Editor** provides a function for setting a default rotation angle for symbol placement operations. If you intend to place a series of symbols at a special rotation angle (and/or mirror mode) it is most useful to set the default rotation angle accordingly, thus avoiding the need of explicitly specifying the rotation angle for each symbol. Use the following commands to set the default symbol placement rotation angle to 90 degree:

```
Symbols ▮▯▯
    Set Default Angle ▮▯▯
        90 Degrees Left ▮▯▯
```

The standard rotation angle setting above will cause symbols to be automatically rotated by 90 degree on subsequent load operations. Use the following commands to place some more symbols at 90 degree rotation angles:

```
Symbols ▮▯▯
    Add Symbol ▮▯▯
        Part Name ? ↵
        Library Element Name ? c ↵
        Move to [30,80] ▮▯▯
    Add Symbol ▮▯▯
        Part Name ? ↵
        Library Element Name ? d ↵
        Move to [220,180] ▮▯▯
    Add Symbol ▮▯▯
        Part Name ? ↵
        Library Element Name ? s_1dil ↵
        Move to [50,140] ▮▯▯
    Next Symbol/Label ▮▯▯
        Move to [90,140] ▮▯▯
    Next Symbol/Label ▮▯▯
        Move to [130,140] ▮▯▯
    Next Symbol/Label ▮▯▯
        Move to [170,140] ▮▯▯
```

Use the following commands to reset the standard rotation angle to 0 degree, and place some more symbols:

```
Symbols ▮▯▯
    Set Default Angle ▮▯▯
        0 Degrees Left ▮▯▯
    Next Symbol/Label ▮▯▯
        Move to [80,70] ▮▯▯
    Next Symbol/Label ▮▯▯
        Move to [80,50] ▮▯▯
    Next Symbol/Label ▮▯▯
        Move to [80,30] ▮▯▯
```

All library symbols stored to the project file are directly available for repeated load operations. I.e., we can also place the symbol CD4081 (which was created in chapter 2.2.2 and stored to the project file **`demo.ddb`**) without accessing an external library file. Use the following commands to place some CD4081 symbols; note that mouse-click and popup menu features are utilized for specifying the part name and selecting the library element, i.e., no keyboard input is necessary for performing the following operations:

```
Symbols
  Add Symbol
                              Part Name ?
                   Library Element Name ?
                      Select Button Project
                    Select Element "cd4081"
                          Move to [60,110]
  Next Symbol/Label
                         Move to [100,110]
  Next Symbol/Label
                         Move to [140,110]
  Next Symbol/Label
                         Move to [180,110]
```

The popup menu for selecting the SCM library file provides access to all library files of the predefined library path as well as to the job-specific library selectable with the Project button. The subsequently activated popup menu for selecting the library symbol simultaneously provides access to the table of contents of the selected SCM symbol library (the symbol load function can be canceled by selecting the Abort button).

The system also provides features for directly accessing certain library files of the predefined library path. Use the following commands to access the symbol **`z80`** from the library file **`zilog.ddb`**:

```
Symbols
  Add Symbol
                              Part Name ?
                   Library Element Name ?  zilog/z80 ↵
```

Now the symbol **`z80`** is picked with the graphic cursor and can be placed. Use the following commands to abort this operation:

```
    Abort
```

Aborting the symbol load function will prevent the system from copying the symbol from the library file to the currently edited project file, thus avoiding redundant database entries in the job file.

The general syntax for specifying the library element name is:

```
<library_file_name>/<library_element_name>
```

If the user hits a mouse button or types a question-mark (**`?`**) to the library element name prompt, then the system will activate a popup menu for selecting the library file. If a question-mark is specified for the library element name, then the system will activate the popup menu for selecting symbols from the given library file (i.e., **`zilog/?`** activates the popup menu for selecting symbols from the **`zilog.ddb`** library file).

Bartels AutoEngineer provides features for assigning different schematic symbol types to the same layout package type (see the descriptions of the **Packager** and the **LOGLIB** utility program for more details). Use the following commands to place one coil symbol **`rels`** and two contact symbols **`relc`**:

Symbols	🖱
Add Symbol	🖱
Part Name ?	↵
Library Element Name ?	rels ↵
Move to [240,180]	🖱
Add Symbol	🖱
Part Name ?	↵
Library Element Name ?	relc ↵
Move to [210,80]	🖱
Next Symbol/Label	🖱
Move to [210,60]	🖱

The symbols placed with the commands above are assigned to a single relay part according to their logical library definition (see also the contents of the **`demolib.def`** logical library file). Note that the **Packager** will allocate a single layout package for these symbols.

Attribute Value Assignment

On SCM sheet level, attribute values can be assigned to all parts where the corresponding symbols contain attribute definitions (i.e., lowercase text starting with a dollar sign, such as **`$val`**, **`$plname`**, **`$pow`**, etc.). Use the following commands to assign values to the **`$val`** attribute defined on **`c100`**, **`r100`** and **`v1000`**:

Symbols	🖱
Assign Value	🖱
Move to "c100", [140,190]	🖱
$val	🖱
Attribute Value ?	10uF ↵
Return	🖱
Assign Value	🖱
Move to "r100", [60,130]	🖱
$val	🖱
Attribute Value ?	100k ↵
Return	🖱
Assign Value	🖱
Move to "v1000", [220,180]	🖱
$val	🖱
Attribute Value ?	1N4001 ↵
Return	🖱

Apply commands such as shown above to assign values to the **`$val`** attribute defined with the parts **`r101`**, **`r102`**, **`r103`** (set value **`100k`**, respectively), **`r104`** (value **`10k`**), **`r105`** (value **`1M`**) and **`c101`** (value **`100pF`**). Note that part and symbol name (e.g., **`Log. Name: c100, Macro: c`**) is displayed in the status line during attribute name queries, thus indicating the symbol currently selected for attribute value assignment.

The attributes submenu provides a list of the attributes defined on the selected symbol, respectively. I.e., this submenu can vary for different symbols. The Return option is only provided for symbols containing more than one attribute definition. Return is used to end the attribute value assignment function.

With the `rels` and `relc` symbols, only one attribute named `$rpname` is defined. This attribute is used to designate the requested part name, i.e. the `$rpname` value will cause the **Packager** to allocate a physical part with a user-defined part name for assigning the logical parts referring to that requested part name. This feature should be applied to force the **Packager** to assign different symbols to the same part whenever necessary (e.g., for op-amps with power supply, relays, etc.). Use the following commands to set the `$rpname` attribute of the symbol `k10` to `k1`, i.e. to assign the logical part named `k10` to the physical part named `k1`:

```
Symbols
    Assign Value
        Move to "k10", [240,180]
        $rpname
            Attribute Value ?  k1
```

Perform the same attribute value assignment as above (value `k1` for attribute `$rpname`) also for the parts `KK100` and `KK101` (relay contacts `relc`). This will cause the **Packager** to pack these two symbols together with the relay coil `K10` into a physical part named `K1`.

The `$plname` attribute is used for assigning non-default package types to logical parts. Setting the `$plname` attribute value for a certain logical part will cause the **Packager** to assign this logical part to the layout package type defined with the `$plname` value instead of the default package definition from the logical library. Use the following commands to set `$plname` attribute values in order to change the default (standard) package type assignments of the logical parts `c100`, `c101`, `r104` and `r105` to SMD package types:

```
Symbols
    Assign Value
        Move to "c100", [140,190]
        $plname
            Attribute Value ?  chip1210
        Return
    Assign Value
        Move to "c101", [30,80]
        $plname
            Attribute Value ?  chip1206
        Return
    Assign Value
        Move to "r104", [210,130]
        $plname
            Attribute Value ?  minimelf
        Return
    Assign Value
        Move to "r105", [170,170]
        $plname
            Attribute Value ?  chip1206
        Return
```

If you have executed all work steps of this paragraph correctly, then your circuit diagram should look like the one shown in figure 2-4.

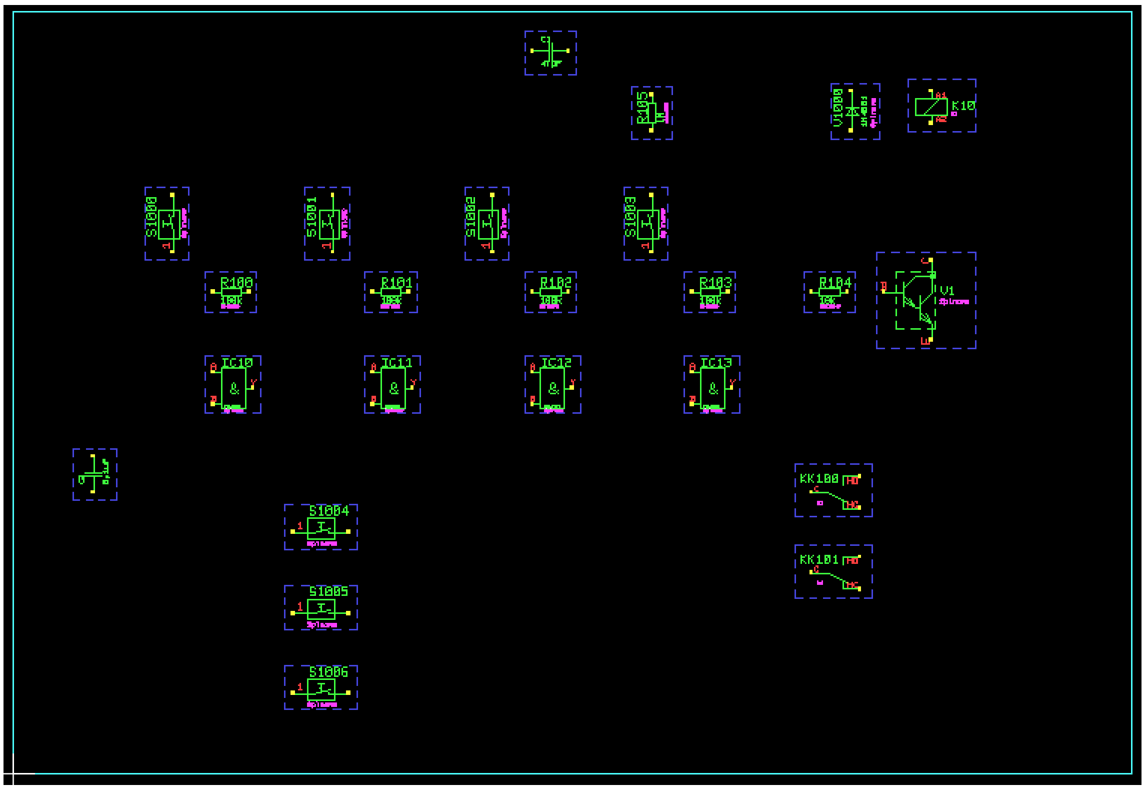


Figure 2-4: SCM Sheet with Symbols

Moving Symbols, Deleting Symbols, Changing Symbol Names

The Move Symbol/Label function is used to move symbols already placed on the sheet. The **Schematic Editor** applies an built-in signal router for automatically rerouting connections to the moved symbol (see below). Pressing the right mouse button during symbol movement will activate a submenu with special options for rotating and mirroring, absolute coordinates input, activate and/or deactivate the signal router, etc. The Del. Symbol/Label function is used to delete symbols. Part names and/or references can be changed with the Change Part Name function. You should test these functions on the currently loaded sheet and use the Undo and Redo functions for estimating design modifications and returning to the original design stage. Please note that the symbol pick function picks the symbol which is placed inside other symbols at the pick position to prevent from unintentionally selecting frame symbols.

2.3.3 Connections, Labels, Busses

The Connections menu provides functions for creating, modifying, and deleting connections and/or busses on SCM sheet level.

Selecting the Connection Point Marker

Use the following commands to select the marker symbol **tconnector** to be the connection point marker for indicating T-shaped connections:

Settings
  Set Junction Point

Please confirm (Y/N) :	y
Junction Point Name (junction) ?	tconnector

The connection point marker is a special marker symbol. The connection point marker symbol should contain a normal graphic area instead of a contact area as with the pin marker symbols. The connection point marker symbol **junction** is assigned on default (note that the system expects this marker symbol to be available in the SCM standard symbol library; see also the description of the **SCMDEFLIBRARY** command supported by the **BSETUP** utility program).

Graphic Control Function, Input Grid

Contact areas on pin symbols are utilized for displaying net list changes on SCM sheet level. As soon as a pin is connected correctly, the contact area defined on the corresponding pin marker will disappear, thus providing most useful graphical indication of net list changes. This feature is required, since BAE allows connections ending at any point (as to be utilized for bus definitions; see below). Note that *single*-segment pin connections will also cause the pin contact area to fade out, thus providing a convenient way of marking pins which should not be connected as already being processed. Contact areas attached to open connections consisting of *more than one* segment, however, are displayed with error highlight, and the Report function from the Utilities menu counts such an open connection as drawing error.

If you have problems connecting certain pins at the current input grid settings, then you should use the Display functions for selecting a smaller input grid (or even release the grid lock).

Use the following commands to select 1mm input grids for subsequent pin connection interactions:

View
  Grids/Rotation
    Set Input Grid
      1.0 mm
      Grid+Rotation On

View, Display

The middle mouse button can be used to activate the View menu at any time during symbol placement and/or the drawing of geometry and connections. Particularly the zoom functions are very helpful at the generation of connections. The following example shows how to zoom to a certain window:

  Zoom Window
    Move to Window Start Point
    Move to Window End Point

The following commands can be used to restore the full view display:

  Zoom All

Creating Connections

The first connection to be created should connect part `S1000` with part `IC10`. Use the following commands to zoom to the workspace around these two symbols:

```
Zoom Window
    Move to Window Start Point, [40,80]
    Move to Window End Point, [110,170]
```

Use the following commands to connect pin `1` of part `S1000` with pin `A` of part `IC10`:

```
Connections
  Add Connection
    Move to Pin "S1000.1", [50,140]
    Move to Connection Corner Point, [50,110]
    Move to Pin "IC10.A", [60,110]
```

Undo function and try again (if necessary use a finer input grid). If the connection is completed correctly, then use the following commands to connect pin `1` of the resistor `R100` to this connection:

Each connection corner point is set by pressing the left mouse button, and a connection is completed by pressing the right mouse button. After correct completion of the above-mentioned work step the pin markers should have become invisible. If this is not true, then step back using the

```
Connections
  Add Connection
    Move to Pin "R100.1",[60,130]
    Move to Connection,[50,130]
```

A T-shaped connection has been created. Note how the system places a junction point marker to indicate the connection cross point.

Now you should create further connections to get a circuit drawing as shown in figure 2-5. You can also experiment with the functions for moving, cutting and deleting segments and for deleting connection and/or nets. Apply the Undo and Redo functions for estimating design modifications and returning to the original design stage. Please consider the Point to Point Connection function for automatically connecting two selectable points on the currently loaded SCM sheet if at all possible with up to three connection segments.

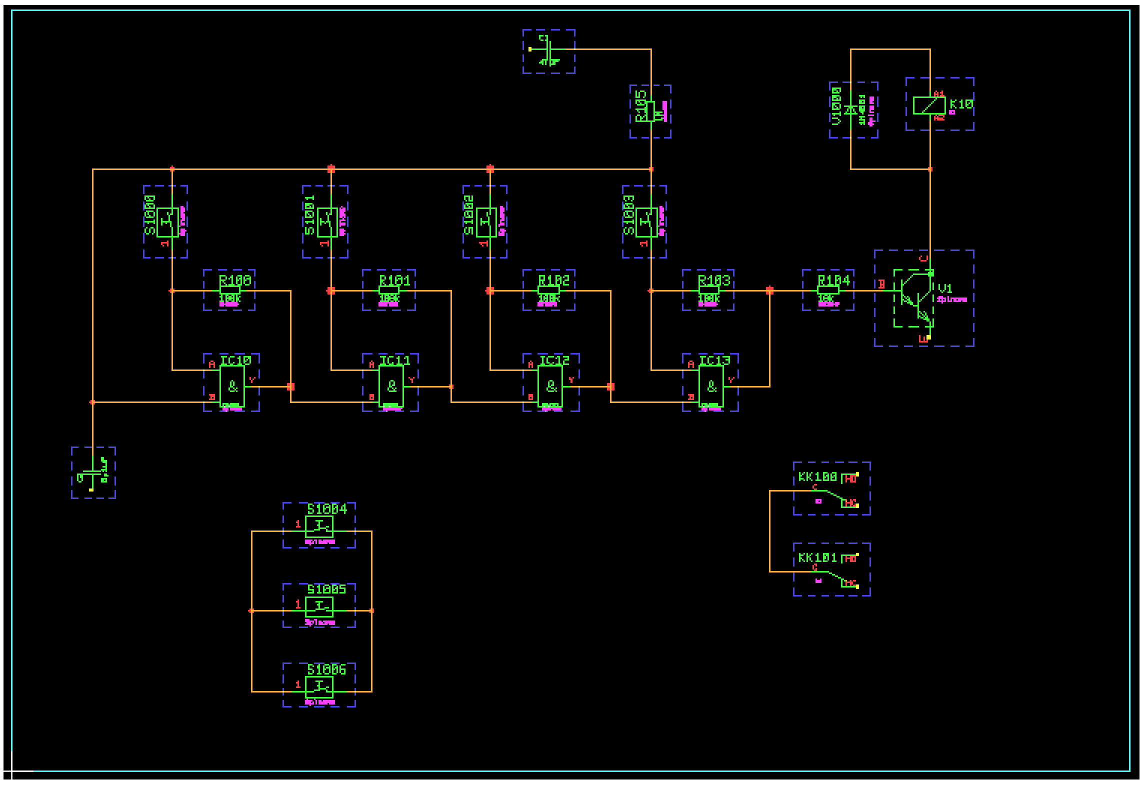


Figure 2-5: SCM Sheet with Symbols, Connections

Labels

Labels are special symbols to be utilized for defining net names. Labels can be applied for assigning certain signal names to connections. Label-defined signal names are known throughout all sheets of the schematics, i.e., labels can also be applied for connecting nets on different sheets.

In chapter 2.2.3 we created the two label symbols **`vss`** (0V CMOS supply) and **`vdd`** (positive CMOS supply) in our job file **`demo.ddb`**. I.e., these two symbols are available in the currently edited project file and can be loaded to the sheet using the Add Label function from the Symbols menu.

Use the following commands to define the signal **`Vss`** by placing the label symbol **`vss`** on the circuit drawing:

- View
  - Zoom All
  - Zoom Window
    - Move to Window Start Point, [120,160]
    - Move to Window End Point, [180,200]
- Symbols
  - Add Label
    - Net Name ? vss
    - Move to [130,190]

Use the following commands to connect pin **`1`** of capacitor **`C100`** with signal **`Vss`** by creating a connection between the capacitor pin and the previously placed label:

- Connections
  - Add Connection
    - Move to Label Pin,[130,190]
    - Move to Pin "C100.1",[140,190]

Take care that the contact area of the label pin will extinguish in order to ensure a correct connection between the capacitor and signal **`Vss`**.

It is not necessary to explicitly connect a label if the label is placed to a connection corner point. Use the following commands to load the label symbol **`Vdd`**, and connect this label to the connection between capacitor **`C100`** and resistor **`R105`**:

- Symbols
  - Add Label
    - Net Name ? vdd
    - Move to Connection Corner Point, [170,190]

Again take care that the contact area of the label pin will extinguish.

If no special label symbol for the requested net name is defined and/or available, then the system uses the default label symbol named **`standard`** (presuming that at least this label symbol is available in the project file or in the selected library). Use the following commands to assign signal name **`NET`** to the connection at pin **`1`** of switch **`S1004`**:

```
Zoom All
Zoom Window
    Move to Window Start Point, [10,40]
    Move to Window End Point, [100,100]
Symbols
    Add Label
        Net Name ? net
        Set Angle
            Angle (Deg./(R)ad.) ? 180
        Move to Connection Corner Point, [70,70]
```

On default, the **`standard`** label symbol is used when placing labels for net names without name-matching label symbol. This default label symbol can be changed through the Select Label Macro parameter setting of the Settings dialog from the Settings menu. This feature allows for the assignment of different label symbols to certain net names and/or net types without having to define a net-specific label symbol for each of these nets.

As shown in the example above the same placement functions apply for labels and symbols. I.e., you can rotate labels, set a standard placement angle, specify absolute placement coordinates, etc.

Now you should load and place further labels to get a circuit drawing as shown in figure 2-6 (connect label **`Vss`** to pins **`V1.E`**, **`C101.1`** and **`S1006.2`**; connect label **`Vdd`** to pins **`K10.A1`** and **`KK100.C`**; connect label **`NET`** to pin **`IC10.B`**). It is recommended to apply the Next Symbol/Label function for the repeated placement of the same label. You should also utilize the popup menu for selecting the label and/or signal name. This net name popup menu is activated when pressing a mouse button or typing the question-mark string (**`?`**) to the net name query. The net name popup offers a menu with all signal names of the currently loaded SCM sheet, i.e., the net name menu does contain all those labels which are already connected to pins of the currently defined net list. The Old button of the net name popup menu is used for activating the Next Label function.

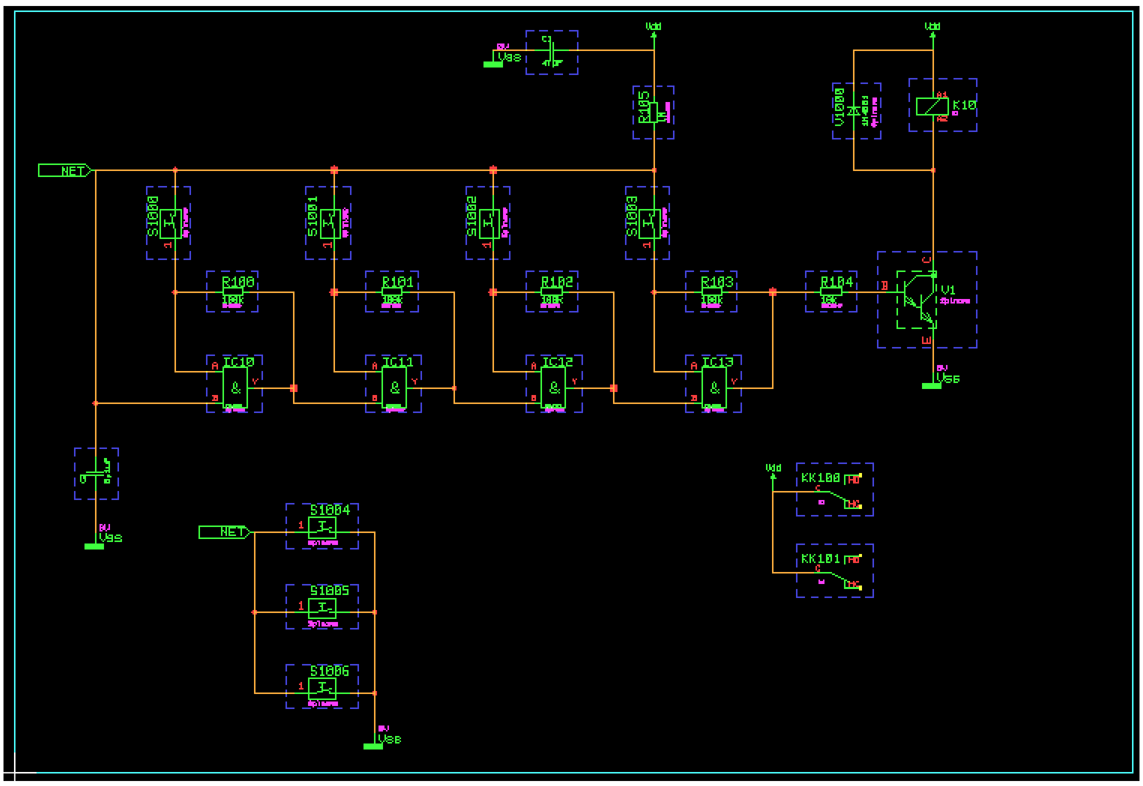


Figure 2-6: SCM Sheet with Symbols, Connections, Labels

The BAE design system instantly updates the net list with the circuit drawing. I.e., the **Schematic Editor** is capable of controlling and correlating the net list data with the circuit diagram at any stage of the design process. Use the following commands to test this most important feature:

Highlight Net
Move to Connection

The Highlight Net function from the View menu is used to mark all connections of the selected net with a special ("Highlight") color. The Highlight Net function works as a toggle; to reset the highlight of a certain net simply re-select that net with the Highlight Net function. In **BAE HighEnd**, the Highlight Net function causes a highlight and/or de-highlight of the selected nets in *all* currently loaded plans of the current project file on schematic sheet and layout board level (global net highlight, cross-probing).

Signal Router

The **Schematic Editor** provides an built-in signal router for performing automatic connection re-routing when moving symbols and/or labels or groups on SCM sheet level. Signal routing can be activated and/or deactivated through the Settings dialog from the Settings menu or with the Signal Router On and/or Signal Router Off options from the Move Symbol/Label function submenu to be activated by pressing the right mouse button whilst moving a symbol or label. The selected signal routing mode is saved with the currently loaded SCM sheet (see also chapter 2.1.5).

Warning

Please note that the signal router should be considered "as-is", i.e., it is straight-forward designed with the intention to simplify work rather than to provide an academically optimum SCM autorouter solution with a long time-to-application/user period. We are aware of the fact that the signal router could produce unpredictable results (e.g., elimination of connections) in cases where symbol movement might cause net list conflicts. It is recommended to move symbols in a step-by-step approach rather than to move long distances and to refrain from placing symbols onto each other. You can always use the Undo function and retry in cases where you are not pleased with the result.

Busses

The next operation is to connect the pins on the right side of the relay contacts **KK100** and **KK101** to a bus. First of all use the following commands to zoom to a suitable workspace:

```
🖱
    Zoom Window 🖱
                        Move to Window Start Point, [190,40] 🖱
                        Move to Window End Point, [290,100] 🖱
```

Use the following commands to define a bus:

```
Connections 🖱
    Add Connection 🖱
                        Move to Connection Start Point,[240,90] 🖱
                        Move to Connection Corner Point,[240,50] 🖱
                        Move to Connection End Point,[250,50] 🖱
        🖱
    Define Bus 🖱
                        Move to Connection 🖱
```

The definition of a bus is accomplished by defining a connection without pin contacts and then selecting that connection with the Define Bus function. The system uses beam-shaped outlines for displaying bus connections.

Use the following commands to define the bus tap **OUT0** and connect that bus tap with pin **N0** of part **KK100**:

```
Connections 🖱
    Tap Bus 🖱
                        Net Name (Range) ? out0 ↵
                        Move to Bus,[240,80] 🖱
        🖱
            Mirror 🖱
                        Move to Level of Pin "KK100.N0" 🖱
    Add Connection 🖱
                        Move to Bus Tap "OUT0" 🖱
                        Move to Pin "KK100.N0" 🖱
        🖱
```

After specifying the bus pin net name the system will load the **bustap** label symbol, which then can be placed on the bus; the submenu available during bus tap placement provides the Mirror option which can be used for mirroring the bus tap to the opposite side of the bus. The **bustap** symbol is a special label symbol, which is always used for displaying bus pins thus defining sub-signals on the corresponding bus (ensure that the **bustap** label symbol is available in the preset library)

The bus tap net name query also accepts net name range specifications. Use the following commands to place three more bus taps named **OUT1**, **OUT2** and **OUT3** (the previously selected bus tap mirror mode is still activated):

```
Connections 🖱
    Tap Bus 🖱
                        Net Name (Range) ? out(1-3) ↵
                        Move to Bus,[240,80] 🖱
                        Move to Level of Pin "KK100.NC" 🖱
                        Move to Level of Pin "KK101.N0" 🖱
                        Move to Level of Pin "KK101.NC" 🖱
```

You should now connect the bus taps **OUT1** (to pin **KK100.NC**), **OUT2** (to pin **KK101.N0**) and **OUT3** (to pin **KK101.NC**).

Please note also the special Connections menu functions for manipulating busses and/or bus taps (Move Bus Tap, Delete Bus Tap, Rename Bus Tap).

The **Schematic Editor** provides features for assigning bus signal names to support bus connections between different sheets of the schematics. Use the following commands to load a label named **`BUS`**, and connect that label to the previously defined bus (take care that the contact area of the label extinguishes to ensure a correct connection):

```
Symbols
    Add Label
        Net Name ?            bus
        Move to Bus,[250,50]
```

Figure 2-7 illustrates the **Bartels AutoEngineer** features for bus definitions. The two bus signals **`S1`** in figure 2-7a are not connected since they are tapped to different busses. In figure 2-7b the bus signals **`S1`** are tapped to the same bus, i.e., they are connected. Figure 2-7c illustrates how to define sub-busses (**`DATA`** and **`ADDR`**); the bus signals **`S1`** are not connected since they are tapped to different sub-busses. Figures 2-7d and 2-7e show how to use labels for the assignment of bus signal names. The signals **`S1`** in figure 2-7d are tapped to busses with the same label (**`BUS`**) and thus are connected, whilst in figure 2-7e the signals **`S1`** are tapped to busses with different labels (**`BUS1`** and **`BUS2`**) and thus are not connected.

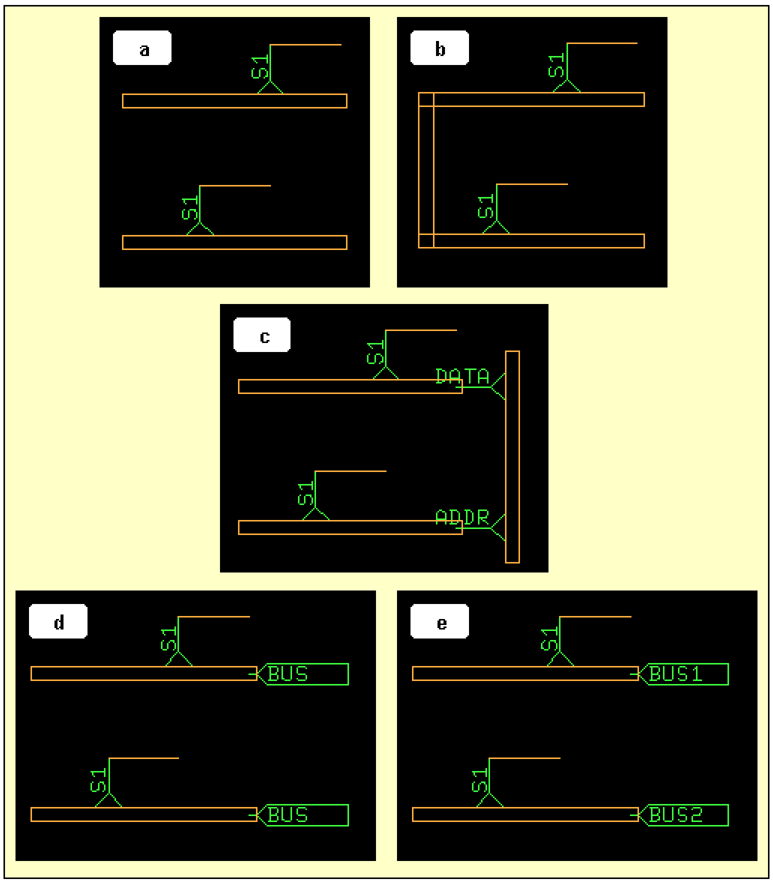


Figure 2-7: SCM Bus Connections

2.3.4 Text and Graphic

Text and graphic can be defined on SCM sheet level. Use the following commands to draw a graphic frame with a legend box around your circuit drawing:

```
Graphic
  Add Graphic Line
                                   Move to [10,10]
                                  Move to [290,10]
                                 Move to [290,200]
                                  Move to [10,200]
                                   Move to [10,10]

      Done
  Add Graphic Line
                                  Move to [290,40]
                                  Move to [220,40]
                                  Move to [220,10]

      Done
  Add Dot Line
                                  Move to [290,20]
                                  Move to [220,20]

      Done
```

Use the following commands to enter the text strings **DEMO** (with text size 10mm) and **sheet1** (with text size 4mm) to the previously defined legend box:

```
Text
  Add Text
                                            Text ?  DEMO ↵

      Text Size
                                Text Size ( 4.00mm) ?  10 ↵
                                  Move to [230,25]
  Add Text
                                            Text ?  sheet1 ↵

      Text Size
                                Text Size (10.00mm) ?  4 ↵
                                  Move to [230,13]
```

2.4 Special SCM Functions

2.4.1 Virtual Symbols

Special design items such as legend boxes, company logos, etc. can also be defined as virtual symbols, and then placed to the circuit drawing. Use the following commands to place the **`logo`** symbol for displaying the Bartels company logo:

Symbols	
Add Symbol	
Part Name ?	↵
Library Element Name ?	logo ↵
Move to [190,10]	

2.4.2 Groups

Group functions belong to the most powerful BAE features. These functions can be used for tasks such as move, copy, delete or replicate arbitrarily selectable parts of the design.

The group functions are featuring set principles. Elements can be added (selected) to or removed (deselected) from the currently defined group. Highlight display is used to indicate group-selected items. The Group Polygon function is used to select and/or deselect parts, traces, areas, texts or elements of any type by defining an area around the items to be selected. The Group Elements function is used to select and/or deselect single elements of a certain type (symbols/labels, connections, graphic or text) by picking the desired items. The Group Elements function will stay active with its parameter settings as long as valid pick elements are selected (thus omitting the need of reactivating the Group Elements function when selecting a series of objects of a certain type). The Reset Group function is used to deselect *all* items which are selected to the currently defined group. All group-selected elements are subject to subsequent group functions such as Save Group, Move Group, Copy Group and Delete Group.

The Save Group function is used to save the currently defined group. The Save Group function requires a group origin definition which becomes the origin of the new element and is used as the reference point for group load commands When saving groups they are stored as an element of the same type as that from which the group was selected. To prevent from unintentionally overwriting existing database elements, Save Group prompts for confirmation if the specified element already exists in the destination file.

The Load Group function can be used to reload previously saved groups (as well as SCM sheet and/or symbol/label elements) to different SCM plans and/or symbols/labels. I.e., the Save Group and Load Group functions can be used for a variety of tasks such as replicating circuitry, saving and loading SCM templates, stealing from existing and proven designs, etc.

Pressing the right mouse button during group movement operations such as Move Group, Copy Group or Load Group will activate a submenu which provides functions for placing the group to relative or absolute coordinates (Jump Relative, Jump Absolute), or rotating and/or mirroring the group (Rotate Left, Rotate Right, Set Angle, Mirror Off, Mirror On).

The Group Macro function is used to replace group-selected symbol macros (on SCM sheet level) or marker macros (on symbol level). This function is most useful for fast part or pin type replacement (technology change). Attribute assignments and text positions from the original elements are transferred to the new elements if possible.

The Load Group function first resets the currently defined group to deselect all currently group-selected elements before performing the load operation. After successfully loading a group, all loaded group elements are automatically group-selected. The advantage of this feature is that loaded group elements (and only these objects) can be subject to subsequent group functions without the need to perform any group selection and/or deselection.

Use the following commands to select a group containing the switches **`S1004`**, **`S1005`** and **`S1006`** and copy that group to the right:

```
Groups
  Group Polygon
    All
      Select
                              Move to [50,10]
                             Move to [110,10]
                             Move to [110,80]
                              Move to [50,80]
        
          Done
  Copy Group
                           Move to [70,70]
                          Move to [130,70]
  Reset Group
```

With the commands above three symbols and two labels have been placed, and a series of connections have been generated. Note that the system has performed automatic part naming on the copied symbols.

If you have executed all operations of this paragraph correctly, then **`sheet1`** of your circuit diagram should look like the one shown in figure 2-8.

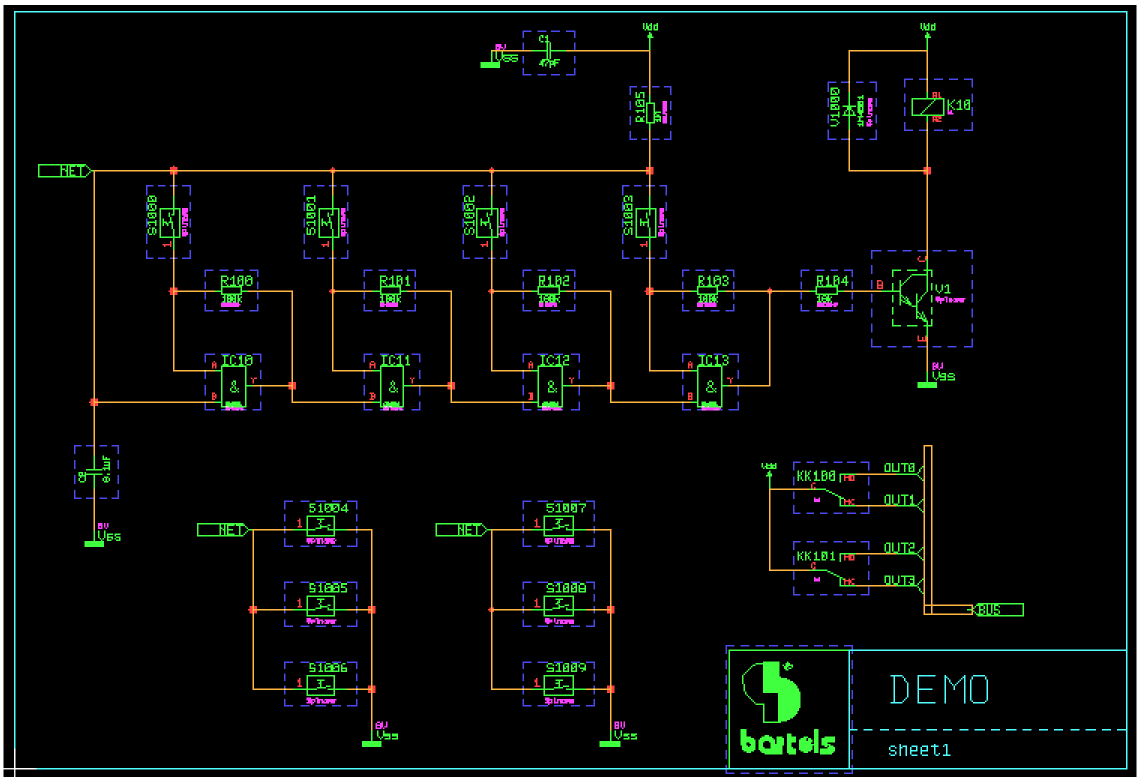


Figure 2-8: SCM Sheet Demo/Sheet1

Don't forget to *save the circuit drawing* with the following commands:

File
Save Element

The save operation can be activated at any time. It is recommended to save the design generally after a certain time of editing to minimize any possible loss of data caused by power failure, hardware crash, etc.

Group Symbol Numbering

The submenus to be activated during group placement with the Load Group, Move Group and Copy Group functions provide the New Names option. This option renames and/or renumbers all currently selected parts according to their symbol part name patterns with part numbering starting at 1, and collisions with existing part names being avoided. This feature can be used for creating consecutive part numbering for the currently selected symbol group.

2.4.3 Plug Pin Assignment

With the operations above we have defined the complete (logical) circuitry of our project on **`sheet1`** of the schematic drawing. What still is missing is the plug pin assignment, which we can define on a second schematic sheet of the project file.

Use the following commands to create a second sheet named **`sheet2`** in job file **`demo.ddb`** and place the text string **`Plug Pin Assignment:`**; also place the **`x_subd9b`** plug symbol and assign value **`xsubd9bl`** to the **`$plname`** attribute of that symbol:

File	▤
New	▤
Sheet	▤
File Name ?	demo ⏎
Element Name ?	sheet2 ⏎
A4 Landscape	▤
Text	▤
Add Text	▤
Text ?	Plug Pin Assignment: ⏎
Place	▤
Symbols	▤
Add Symbol	▤
Part Name ?	⏎
Library Element Name ?	x_subd9b ⏎
Place	▤
Assign Value	▤
Move to Symbol	▤
$plname	▤
Attribute Value ?	xsubd9bl ⏎

You now should connect the signals **`Vss`**, **`Vdd`** and the bus signals **`OUT(0-3)`** of **`BUS`** to the plug symbol (see figure 2-9 for pin assignments).

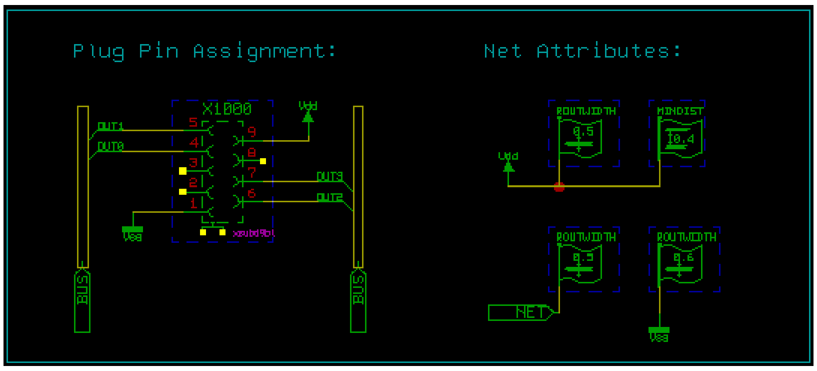


Figure 2-9: SCM Sheet Demo/Sheet2

2.4.4 Net Attributes

Bartels AutoEngineer supports net-specific attributes for controlling the **Autorouter** such as trace width, minimum distance, net routing priority, and power supply routing width.

Place both the label **NET** and the symbol **att_rw** (net attribute **ROUTWIDTH**) as shown in figure 2-9, connect the pins of the **NET** label and the **att_rw** symbol with each other and assign a value of "0.5" to the **$val** attribute of the **att_rw** symbol. This net attribute assignment will cause the **Autorouter** to route the net named **NET** with a trace width of 0.5mm.

Place the label **Vdd** as shown in figure 2-9 and connect this label to the **att_rw** symbol (with value "0.5" for the **$val** **ROUTWIDTH** attribute) and to the **att_md** symbol (with value "0.4" for the **$val** **MINDIST** attribute). These net attribute assignments will cause the **Autorouter** to route the net named **Vdd** with a trace width of 0.5mm and a minimum distance (i.e., spacing) of 0.4mm.

Use the same procedure as described above to define a trace width of 0.6mm for the net named **Vss** (place label **Vss** and connect it to symbol **att_rw** with a value of "0.6" for the **$val** attribute).

Bartels AutoEngineer also supports the **PRIORITY** and **POWWIDTH** net attributes for controlling the **Autorouter**. See the description of the **LOGLIB** utility program for more details on net attribute definitions.

Finally, place the **Net Attributes:** text string to the circuit diagram as shown in figure 2-9 and don't forget to save the circuit drawing.

2.4.5 Tag Symbols

Tag symbols can be used for assigning attributes and/or attribute sets to (groups of) parts, pins or nets. This feature can also be used to introduce more complex design information such as preferences for test procedures or logical relations between parts, pins and/or nets.

The Symbol Tagmode function from the Symbols menu can be used to change the currently loaded symbol to either a Netlist Tag or a Virtual Tag. Netlist Tag assignments are transferred to the physical net list. Virtual Tag assignments are only processed on SCM sheet level.

The SCM color setup provides the Tag Symbol and Tag Link entries for displaying tags and/or tag assignments. Usually, tags are not plotted.

Tag symbol pins of certain types are required for assigning tag symbol entries to nets, symbols or pins, respectively. Tag Pin Function from the submenu to be activated with the right mouse button during pin placement can be used to define the pin mode. The pin modes provided are Standard Pin (for standard symbols and labels), Symbol Tag (for assigning tags to symbols), Pin Tag (for assigning tags to pins) and Net Tag (for assigning tags to nets).

At least one attribute should be defined with each tag symbol by either placing the required attribute name on tag symbol level or by defining a fixed attribute value assignment with the logical library definition of the tag symbol. In the logical library definition of tags each tag symbol pin must be referred by a **LOGLIB** pin command. Pure attribute definition tags must be defined virtual.

On SCM sheet level, tags can be placed using the Add Symbol function from the Symbols menu. Once a tag is placed, the Assign Symboltags function can be used for assigning tags. The target objects to be selected with the Assign Symboltags function must correspond with the tag pin types of the selected tag symbols, i.e., either a symbol, a pin or a net named by label can be assigned to each tag pin. The attribute value assignments of net list tag symbols are automatically transferred to the assigned target objects by the **Packager**. Special processing of predefined attributes such as **$plname**, **$rpname**, **$routwidth**, **$powwidth**, etc. stays in effect. The **$routwidth** attribute can be tagged to pins to define pin-specific routing widths by assigning the required millimeter distance value.

2.4.6 Templates

Certain parts of the circuitry such as legends, plug pin assignments or router control definitions often are quite similar or even equal for different designs. The **Bartels AutoEngineer** data base concept supports multiple use of such circuit design templates, i.e., templates can be defined and referred as if they would be library elements. Templates can be saved and copied either by using group functions (Save Group, Load Group) or by applying the Save Element As function from the File menu and utilizing the **COPYDDB** utility program for copying requested SCM elements to desired project files.

2.4.7 Exiting the Schematic Editor

Don't forget to save the currently edited SCM element before exiting the **Schematic Editor**:

File
  Save Element

Returning to Main Menu

Use the following commands to return from the **Schematic Editor** module to the BAE Main Menu (BAE Shell):

File
  Main Menu

The currently loaded SCM element will automatically be saved when returning to the BAE Shell. Use the following commands to return from the BAE Shell to the operating system:

Exit BAE

Exiting BAE

Use the following commands if you wish to exit BAE from the **Schematic Editor**:

File
  Exit BAE

The Exit BAE function activates a confirmation prompt if the element currently loaded to the **Schematic Editor** has not yet been saved. In this case you should abort the Exit BAE function, save the current element, and then call Exit BAE again, as in:

File
  Exit BAE
    Please confirm (Y/N) : n
File
  Save Element
File
  Exit BAE

Subsequent Tasks

Use the following command from operating system level (e.g., under DOS) to list the project file(s) created in this chapter.

```
> dir demo.*
```

The job file **`demo.ddb`** contains the all SCM sheets of the schematic circuit as well as a logical, unpacked net list. The next task of the design cycle would be to apply the **Packager** for translating the logical net list into a physical, packed net list (see chapter 3.2). After successfully packaging the net list the printed circuit board design can be created using the BAE Layout modules (see chapter 4).

2.5 SCM Plot Output

The SCM functions for generating HP-GL, HP Laser, or Postscript output are provided with the Plot Output menu of the **Schematic Editor**. SCM plots can only be generated with an element loaded. I.e., you should first start the **Schematic Editor**, and load the SCM sheet **`sheet1`** from the **`demo.ddb`** file using the following commands:

File	
Load	
Sheet	
File Name ?	demo ↵
Element Name ?	sheet1 ↵

2.5.1 General Plot Parameters

Plot Device

The Plot Device function is used for specifying the plot data output device. The output can either be written to a file or directly to the plotter and/or printer. When plotting to a file, the desired output file name must be specified. When writing directly to the plotter, the name of the output port where the plotter is connected to (e.g., **`com2`**, **`lpt1`** under DOS) must be specified. Take care that enough disk space is available when writing to a file and that the corresponding interface is correctly initialized when directing output to a hardware device. The plot functions will abort with error messages such as

```
Error writing ASCII file!
```

if neither of these requirements are met.

The system will prompt for the output device name after activating the desired output function if no default plot device is specified with the Plot Device function. Popup menus for fast plot file selection are integrated to the Plot Device, HP-GL Output, Postscript Output and HP Laser Output functions of Plot Output menu. Files ending on **`.ass`**, **`.con`**, **`.ddb`**, **`.def`**, **`.exe`**, **`.fre`**, **`.ulc`** and **`.usf`** are faded out from the plot file menus for data security reasons. New plot file names can be typed in via keyboard.

Plot Scale

The Plot Scale function is used to specify a scale factor for the plot output. The default scale factor value is 1.0. Plot scale factor value specifications can range from 0.1 to 100.0.

Plot Rotate

The Rotate Plot function provides the options No Rotate and Rotate Left. The No Rotate option sets the 0 degree plot rotation (which is the default). The Rotate Left option will produce a 90 degree counter-clockwise rotated plot output. Negative output coordinates might be produced when rotating plots. Some printers or plotters refuse to process negative coordinates. Positive output coordinates can be enforced however by replacing the origin of the plot element. I.e., the Set Origin function from the Settings menu can be used to set the origin to the upper left element corner before producing a rotated plot.

2.5.2 HP-GL Pen Plot

The Plotter Pen Width function is used to define the pen width for generating the HP-GL output data. The default plotter pen width is 0.3mm. Pen width value specifications can range from 0.01mm to 10.0mm.

The Plotter Speed function is used to control the pen plotter speed, i.e. the speed at which the pen can be moved by the HP-GL plotter. The system accepts either **`S`** input for maximum speed or non-negative integer plotter speed values in cm/s units. The default plotter speed corresponds with **`S`** (maximum speed). The maximum speed value specification is 99cm/s.

The Fill Mode HP-GL function provides the options Fill Mode Off and Fill Mode On. Fill Mode Off causes the system to refrain from filling the plot structures, i.e., only outlines are drawn for filled areas. Switching off the fill mode is most useful for fast control plots. Fill Mode On is used to select the default fill mode.

The HP-GL Output function is used to produce pen plots in HP-GL (Hewlett-Packard Graphics Language). The user is prompted for the pen number (1..99) and for the name of the output file (if no default output device was specified with Plot Device). Use the following commands to switch off the fill mode, set the plotter pen width to 0.1mm, and then plot the currently loaded SCM sheet to an HP-GL plot file named **`demo_s1.plt`** using pen number 4:

```
Plot Output
  Plotter Pen Width
                        Plotter Pen Width ( 0.30mm) ?  0.1
  Fill Mode HP-GL
    Fill Mode Off
  HP-GL Output
                        Plotter Pen Number (1..99) ?   4
                                   Plot File Name ?    demo_s1.plt
```

The system will issue the following message after successfully generating the HP-GL plot:

```
HP-GL plot done!
```

A **`PG`** HPGL command is added to the end of HPGL output files to force a page eject and to prevent from subsequent plots being plotted onto the same sheet.

2.5.3 HP-Laser Output

The HP Laser Output function is used to produce HP Laser plots in PCL (Printer Command Language) format. HP Laser plots are automatically scaled to A4 paper size, i.e., neither plot scaling factor nor pen width settings have effect. Use the following commands to generate an HP Laser plot of the currently loaded SCM sheet and direct the output to **`lpt1`** (e.g., for interfacing an appropriate DOS-connected laser writer device):

Plot Output
HP Laser Output
Plot File Name ? lpt1

The system will issue the following message after successfully generating the PCL plot:

```
HP Laser output done (scale 1:...)!
```

The scale information in the status messages indicates any non-default scaling factor being used for automatic plot to sheet size scaling.

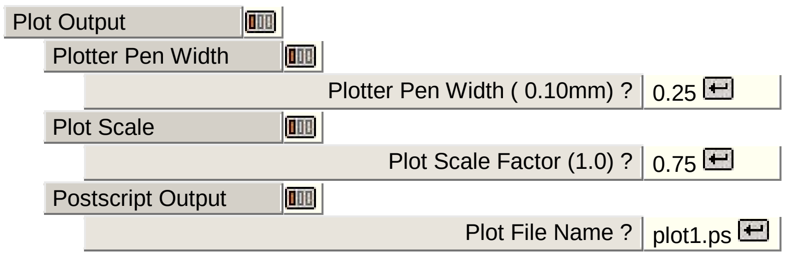
Binary copy mode is required for transferring PCL plot data to the desired output device. I.e., the **`/b`** option must be used when sending PCL files from hard disk to laser printer with the DOS COPY command as in

```
> copy pclplot lpt1 /b
```

with **`pclplot`** being the PCL plot file name and **`lpt1`** designating the output device.

2.5.4 Postscript Output

The Postscript Output function is used to produce output in Postscript format. Use the following commands to set the standard line width to 0.25mm and the scaling factor to 0.75, and generate a Postscript output file named **`plot1.ps`** for the currently loaded SCM sheet:



Plot Output
Plotter Pen Width
Plotter Pen Width (0.10mm) ? 0.25
Plot Scale
Plot Scale Factor (1.0) ? 0.75
Postscript Output
Plot File Name ? plot1.ps

The system issues the following message after successfully generating the Postscript output:

```
Postscript output done!
```

2.5.5 Generic Output under Windows

A generic print/plot output function is implemented with the Windows versions of the BAE PC software. I.e., *any* print/plot output feature supported by the current Windows operating system configuration is also supported with the **Schematic Editor** of the BAE Windows software.

Use the following commands to activate the Windows print/plot menu:

Plot Output
    Generic Output

The Windows printer dialog specifications for the number of copies, page sorting mode and page range are considered by the Generic Output function.

The All Pages option from the Windows printer dialog of the Generic Output function plots all elements of the currently loaded database class. This allows for the output of, e.g., all sheets of a schematic circuit. All SCM sheets are plotted according to specific plot parameter settings such as plot rotation modes.

The Selection option from the Windows printer dialog allows for the selection of the print area for generic outputs.

The generic output is automatically scaled to the print page format selected through the Windows printer setup if the size of the element to be plotted exceeds the paper size. The page aspect ratio is maintained when automatic plot scaling is applied. The status message of the Generic Output function provides information to indicate any non-default scaling factor being used for automatic plot to sheet size scaling.

2.5.6 Bitmap Plot Output to Windows Clipboard

The Output to Clipboard function from the Plot Output menu can be used under Windows for plotting a bitmap of the currently loaded element into the clipboard ready to be imported (Pasted) to other Windows applications capable of processing bitmaps. The whole element is plotted on default. The Clipping On option can be used to restrict the output to a mouse-selectable rectangle. The plot dialog box also allows for bitmap size specifications and plot rotation mode selection.

2.6 Hierarchical Circuit Design

The **Schematic Editor** provides functions for creating hierarchical schematic designs. Through these functions it is possible to define schematic sheets to be sub-blocks and to reference sub-blocks from higher level schematic plans. This high-sophisticated SCM design feature is recommended especially for experienced users designing large projects with replicated parts of the circuitry. Hierarchical circuit design usually is applied for the development of integrated circuits such as gate arrays, standard cells or ASICs.

2.6.1 Sub-block Circuit Drawing

The schematics for the sub-block can be defined on one or more sheets. The Sub-Block option of the Set Sheet Blockname function from the Settings dialog is used for assigning a sub-block name to the sub-block circuit drawing. Identical sub-block names must be set for different sheets defining the same sub-block. Figure 2-10 shows an example for a sub-block circuit drawing named **BLOCK**. Connections to higher hierarchy levels (i.e., the sub-block's pins) are defined by module ports. The system uses a standard label symbol named **port** for module ports. The Add Module Port function from the Symbols menu is used for defining and placing module ports. The Move Symbol/Label and Delete Symbol/Label functions from the Symbols menu can be used for moving and/or deleting module ports. In figure 2-10 the module ports **S**, **R**, **Q**, **/Q** and **HYPER** are defined. Within hierarchical circuit design there is a difference between global and local net names. A local net is defined by preceding the net name with an ampersand (**&**) character. Hence the nets named **&ABC** and **/&ABC** in figure 2-10 are defined locally in all sheets with the same blockname. Standard net name allocation (such as at **VCC**) introduces a global net definition which connects on all sheets, no matter what hierarchy level the net name is defined on. When using a global net name in a sub-block, all matching nets from multiple referenced sub-blocks are connected as well.

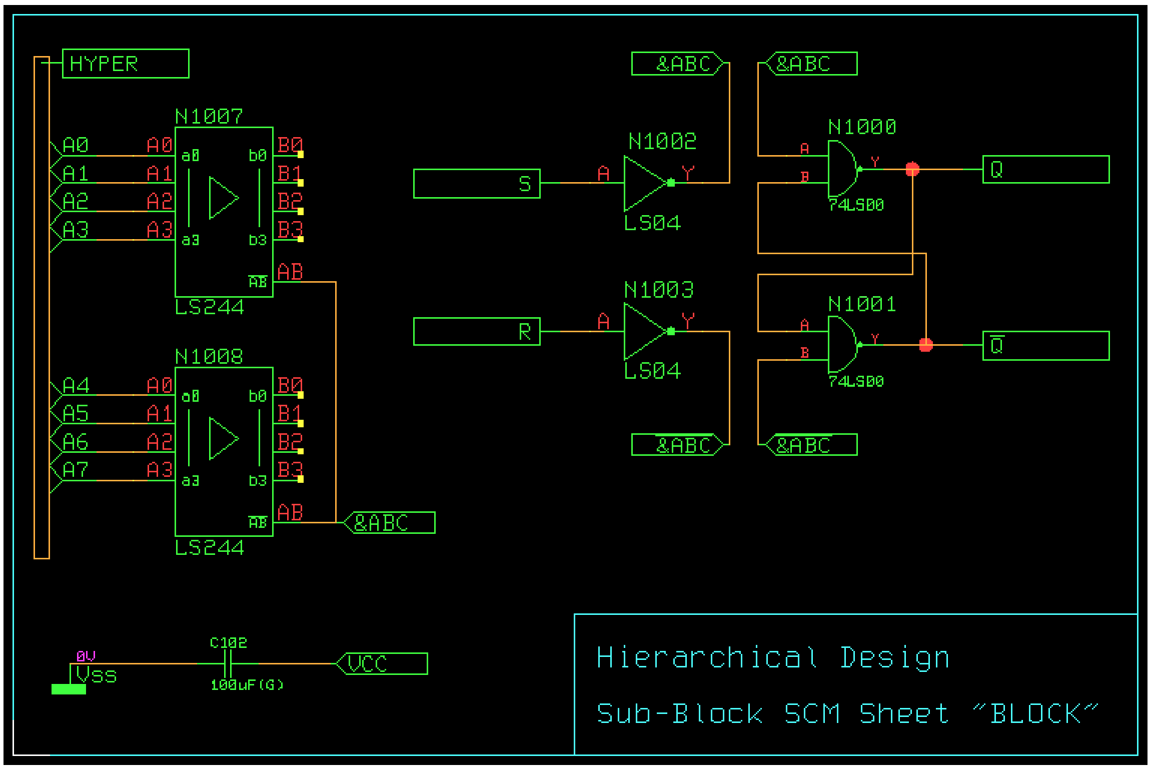


Figure 2-10: Hierarchical Circuit Design; Sub-Block SCM Sheet "BLOCK"

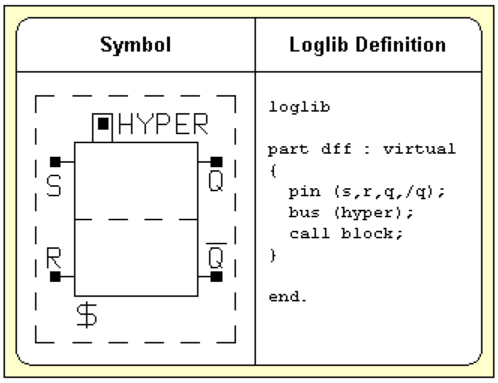
The Set Sheet Blockname parameter setting also provides the Single Sub-Block option for block diagramming support. Whilst there are multiple references to a standard Sub-Block allowed, a Single Sub-Block sheet can only be referenced once. Single sub-blocks are treated like normal SCM sheets by **Packager** and Backannotation. I.e., the **Packager** transfers symbol/part names from single sub-blocks without **[Pn]** prefices to the layout, and Backannotation transfers part name changes and pin/gate swaps on single sub-blocks back to the schematics.

2.6.2 Sub-block Reference Symbol and Logical Sub-block Reference

A special SCM symbol is required for referencing any sub-block on higher hierarchy levels. The pin names of that symbol must match the names of the module ports defined on the corresponding sub-block circuit drawing.

The **Packager** needs a logical reference for any sub-block reference symbol used throughout the schematics. This logical reference must be created in the Logical Library by defining a logical library file entry for the sub-block reference symbol. This **LOGLIB** entry must define a virtual part with the name of the sub-block reference symbol. For referencing the sub-block, the **LOGLIB** command call (with the blockname of the sub-block as argument) must be used. See also chapter 7.11 of this manual for a more detailed description of the **LOGLIB** utility program.

Figure 2-11 shows the sub-block reference symbol `DFF` and the corresponding logical sub-block reference for the sub-block circuit drawing in figure 2-10. Note that figure 2-11 also illustrates how to define busses on sub-block reference symbols.



```
loglib

part dff : virtual
{
  pin (s,r,q,/q);
  bus (hyper);
  call block;
}

end.
```

Figure 2-11: Hierarchical Circuit Design; Block Symbol "DFF" with Loglib Definition

2.6.3 Top Level Circuit Drawing

Figure 2-12 shows how the sub-block symbol **DFF** from figure 2-11 is used in a higher level schematic drawing. The circuit drawing in figure 2-12 contains three **DFF** symbols named **DFF_1**, **DFF_2** and **DFF_3**, each referencing one **BLOCK** sub-block circuit drawing.

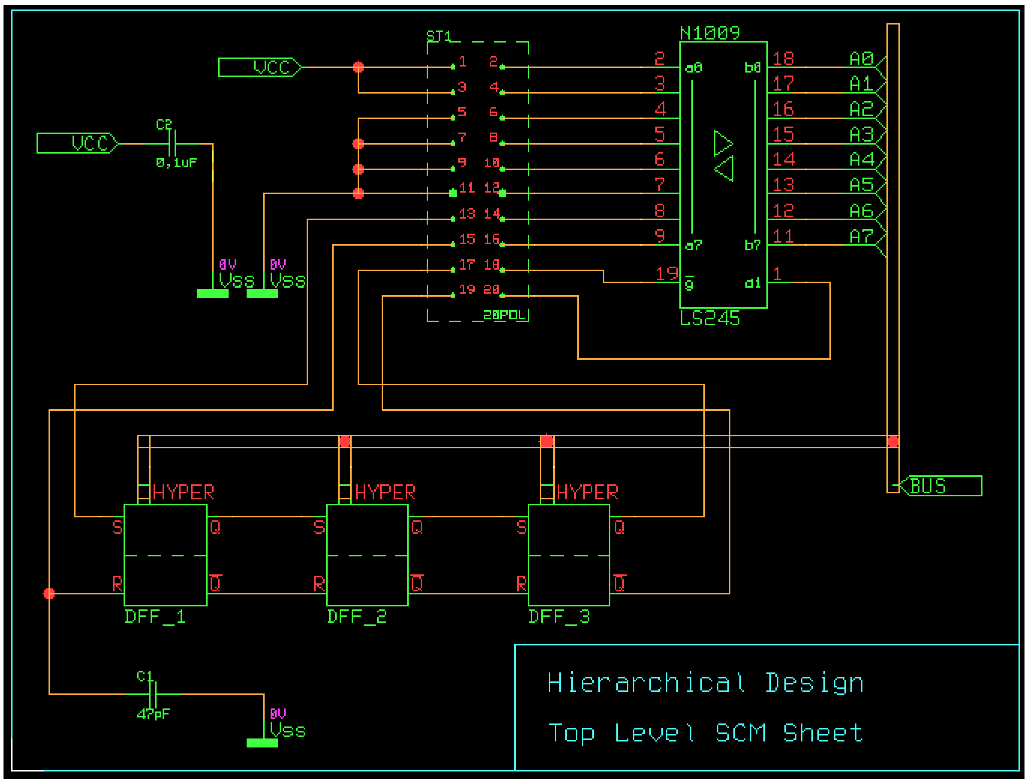


Figure 2-12: Hierarchical Circuit Design; Top Level SCM Sheet

2.7 Backannotation

Backward annotation with the Backannotation is always required after performing layout net list modifications such as part name changes, pin/gate swaps or alternate part package type assignments. Part names can be changed using the Netlist Part Name function from the menus for interactive part placement. Interactive pin/gate swaps are applied with the Pin/Gate Swap function from the manual part placement menu. Automatic pin/gate swap is performed with the Full Autoplacer, Single Pass Optimizer and Multi Pass Optimizer autoplacement functions, and can also be performed when using the Router P/G-Swap On option with the rip-up router of the **Autorouter** module. Alternate package types can be assigned using the Alternate Part submenu function during manual part placement. Note that any of the net list modifications mentioned above will get lost when re-editing the schematic without running the Backannotation.

Backannotation is fully integrated to the Schematic Editor and can be explicitely started using the Backannotation command from the Utilities menu. A feature for automatically processing pending Backannotation requests is also integrated to the Schematic Editor functions for loading schematic plans. Backannotation requests are generated when saving layouts with net list modifications such as pin/gate swaps or changed part names. Loading an SCM sheet with a pending Backannotation request automatically activates a verification menu which allows to backannotate the currently processed design. The Backannotation request is deleted after successfully running the Backannotation.

See chapter 3 for a detailed description on how Backannotation is applied.

Chapter 3
Packager / Backannotation

This chapter describes how to use the **Packager** program module and the Backannotation function from the **Schematic Editor** for performing forward and backward annotation of net list data. The examples presented with this chapter introduce the basic concepts of net list data processing and annotation in the **Bartels AutoEngineer** in a logical sequence. The circuit design of the preceding chapter will be prepared for further processing in the subsequent chapters. The reader should work through this chapter without missing any sections to gain full understanding of **Packager** and Backannotation. Once a command has been used and/or explained, the operator is assumed to have understood its function and be able to perform it again. Subsequent instructions containing this command will be less verbose for easier reading and more speedy learning.

Contents

Figures

3.1 General

The schematic capture system of the **Bartels AutoEngineer** essentially consists of an interactive **Schematic Editor** with integrated Backannotation and the **Packager** program module for converting logical into physical netlists ("Forward Annotation"). The following sections of this manual describe in detail how to use **Packager** and Backannotation.

3.1.1 Components and Features

Bartels AutoEngineer provides the **Packager** program module and the Backannotation function for transferring net list data between schematics and layout. Figure 3-1 shows the design flow of forward and backward annotation using **Packager** and Backannotation.

Editing SCM sheets with the **Schematic Editor** results in a logical net list. The **Packager** collects the interconnection information from the logical net list and generates a physical net list ready for layout. The forward annotation process is controlled by the logical library entries defining the assignments of SCM symbols to layout packages and the corresponding pin mappings (see also the description of the **LOGLIB** utility program). The **Packager** annotates the schematic with the physical pin names to replace the logical pin names and also transfers special information such as pin/gate swap definitions, part attribute settings, power supply pinning, etc.

Layout Net list modifications such as part name changes, pin/gate swaps or part package type assignments must be backward annotated to the schematics using the Backannotation function from the **Schematic Editor**. Running the Backannotation results in a modified logical net list with annotated part and/or pin names on the corresponding SCM sheet.

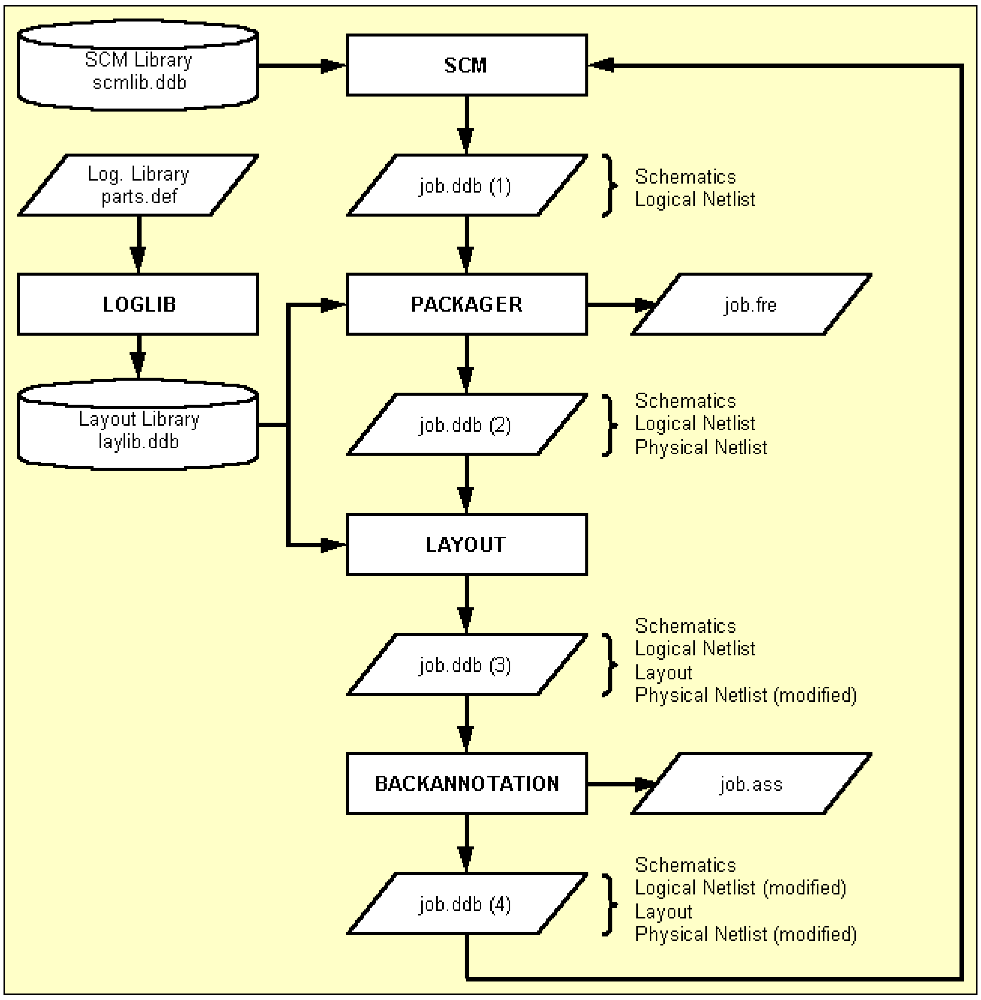


Figure 3-1: Design Flow Packager - Backannotation

3.2 Packager

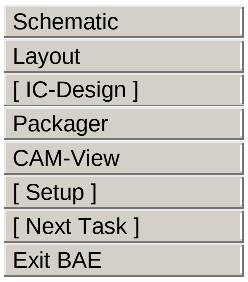
Forward annotation with the **Packager** is always required after introducing net list changes with the **Schematic Editor** such as loading symbols to the schematic, changing the connectivity on the circuit, setting attribute values, etc.

3.2.1 Starting the Packager

It is recommended to start the **Bartels AutoEngineer** from in the directory where the design files should be generated, since this considerably simplifies job file access. If you intend to process the examples provided with this manual it is recommended to move to the BAE examples directory installed with the BAE software. The **Packager** can be called from the **Bartels AutoEngineer** main shell. Start the BAE shell by typing the following command to the operating system prompt:

```
> bae
```

The **AutoEngineer** comes up with the Bartels logo and the following menu (the Setup function is only available under Windows/Motif; the IC-Design and Next Task menu items are available only with special software configurations such as **BAE HighEnd** or **BAE IC Design**):

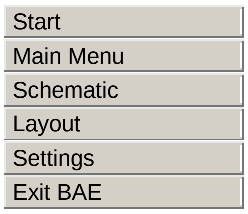


Move the menu cursor to the Packager menu item and confirm this choice by pressing the left mouse button:

Now the **Packager** program module will load. If this fails to happen then check your BAE software installation (see the Bartels AutoEngineer® Installation Guide for details on how to perform a correct installation).

3.2.2 Packager Main Menu

The **Packager** user interface provides a menu area on the right side, consisting of the main menu on top and the currently active menu below that main menu. After entering the **Packager** the Settings function with its menu is active. The main menu is always available and provides the following functions:



The Settings function is used to set the parameters for the **Packager** process to be started with the Start function. The Main Menu, Schematic and Layout functions are used to switch to the BAE Main Menu, the **Schematic Editor** or the Layout system. Exit BAE ends the BAE session. For designs without existing layout, layout element creation for the net list created by the **Packager** is automatically suggested when switching to the Layout system.

3.2.3 Running the Packager

Before starting the **Packager** process through the Start function, the user must specify the parameters to be used for the **Packager** call using the functions from the Settings menu.

The following prompts are activated after calling the All Parameter function from the Settings menu:

Settings
All Parameters
Design File Name ?
Design Library Name ?
Layout Element Name ?

Each of these parameters can also be set explicitely through the corresponding functions from the Settings menu.

The design file name is the name of the project file to be selected for packaging. The design file must be available with extension **`.ddb`** but this extension must not be included with the file name specification. If the user types an empty string (by pressing the return key ↵) to the design file name prompt, then the system will automatically use the file name of the previously processed element (i.e., the global project name).

The design library name is the name of the layout library file to be used for the PCB design. This file must also contain the logical library definitions required for packaging. The design library file must be available with extension **`.ddb`** but this extension must not be included with the file name specification. If the user types in an empty string (by pressing the return key ↵) to the design library name prompt, then the system will automatically refer to the default layout library file name defined with BAE setup (see also command **`LAYDEFLIBRARY`** of the **BSETUP** utility program).

The layout element name is the name of the layout and/or the net list to be generated. The layout element name is freely selectable when packaging a certain design for the first time. If the user types in an empty string (by pressing the return key ↵) to the layout element name prompt, then the system uses the default layout element name defined with BAE setup (see also command **`LAYDEFELEMENT`** of the **BSETUP** utility program).

After successfully processing all SCM sheets, the **Packager** generates a physical net list with the same name as specified for the layout board element. That layout board element can then be created, parts placed and traces routed. For user information purposes, the **Packager** also creates a free list file with file name extension **`.fre`**. The free list is an ASCII file containing an unconnected pins report and statistical information such as net pin counts.

The system issues the **`No errors occurred!`** message after successfully completing the packaging process, and the user can press any key to return to the BAE main menu. Errors during packaging usually result from wrong and/or missing logical library entries. In this case the logical library must be corrected before re-running the **Packager** (see the description of the **LOGLIB** utility program for details on how to define logical library entries).

Backannotation Requests

Saving a layout with pin/gate swaps and/or net list part name changes creates a design-specific Backannotation request. The **Packager** checks for Backannotation requests and activates a verification to prevent the **Packager** from forward annotation without confirmation. Confirmation to run the **Packager** will discard any layout net list changes not yet backannotated.

Attribute Transfer

The **Packager** automatically transfers SCM symbol pin names to the **`$llname`** pin attribute. This allows for the display of logical pin names in the layout by defining **`$llname`** texts on padstack level.

`$nettype` pin attributes are automatically transferred to connected nets. The **`$nettype`** value **`mixed`** is assigned to nets with different **`$nettype`** attribute values.

The **BAE HighEnd Packager** automatically transfers **`$drcblk`** pin attributes to connected nets. The **`$drcblk`** attribute value addresses a **BAE HighEnd** design rule check parameter block to be assigned to the corresponding net.

ERC (Electrical Rule Check)

The **Packager** evaluates **`$pintype`** pin attribute settings to perform electrical rule checks (ERC). It is recommended to assign fixed ERC pin types through the logical symbol/part definitions. The following pin type attribute value settings are supported:

`$pintype`	**Pin Type**
`in`	Input Pin
`out`	Output Pin
`bidi`	Bi-directional Pin
`anl`	Analog Pin
`sup`	Power Supply Pin

The ERC issues a warning message such as **`Net 'netname' has only inputs!`** if a net with one or more input pins has no (normal, bi-directional or power supply) output pin. A warning message such as **`Driver collision on net 'netname'!`** is issued if a normal output pin is connected to another output pin, a bi-directional pin or a power supply pin.

3.2.4 Example

This section describes how to run the **Packager** on the **`demo.ddb`** example project file which has been created in the previous chapter. I.e., you should change to the directory where the **`demo.ddb`** file resides.

Faulty Packager Pass

Start the BAE main menu, call the **Packager** as described in chapter 3.2.1, and use the following commands to run the **Packager** for transferring the logical net list in **`demo.ddb`** to a physical net list named **`board`** using library **`demolib.ddb`**:

Settings	
All Parameters	
Design File Name ?	demo ↵
Design Library Name ?	demolib ↵
Layout Element Name ?	board ↵
Start	

Errors will occur during packaging, and the following messages are displayed:

```
================
BARTELS PACKAGER
================
Design File Name ...........: 'demo'
Library File Name ..........: 'demolib'
Layout Element Name ........: 'board'
Active Schematic Sheet .....: 'sheet1'.
ERROR : Part 'cd4081' not in library!
Abort - Database not changed!
```

The **`Abort - Database not changed!`** message at the end of the **Packager** protocol indicates that packaging was not successful.

Under Windows, the text window for **Packager** messages is displayed with scrollbars to provide complete access to lengthy **Packager** protocols without having to view the **`bae.log`** log file (see below) through an external text editor.

Hit the return key ↵ to return to the **Packager** main menu, and use the following command to exit from BAE:

Exit BAE

You are now back on operating system level. The **Packager** report has been written to an ASCII file named **`bae.log`** in the current directory. You can examine this file with your editor (or send it to your printer) to interpret the **Packager** report. The **Packager** process stopped with the following error message:

```
ERROR : Part 'cd4081' not in library!
```

The error message above means that library **`demolib.ddb`** does not contain any logical library entry for SCM symbol **`cd4081`**. **`cd4081`** is the new SCM symbol created in chapter 2.2.2. The system only knows that this symbol is repeatedly referenced on SCM sheet **`sheet1`** of the **`demo.ddb`** project file, and that the logical pins of **`cd4081`** are named **`A`**, **`B`** and **`Y`**. The SCM symbol **`cd4081`** represents only one of the four gates of the complete CD4081 part. The **Packager** needs additional information such as the layout package associated with **`cd4081`**, logical to physical pin mapping, predefined power supply pins and pin/gate swap definitions.

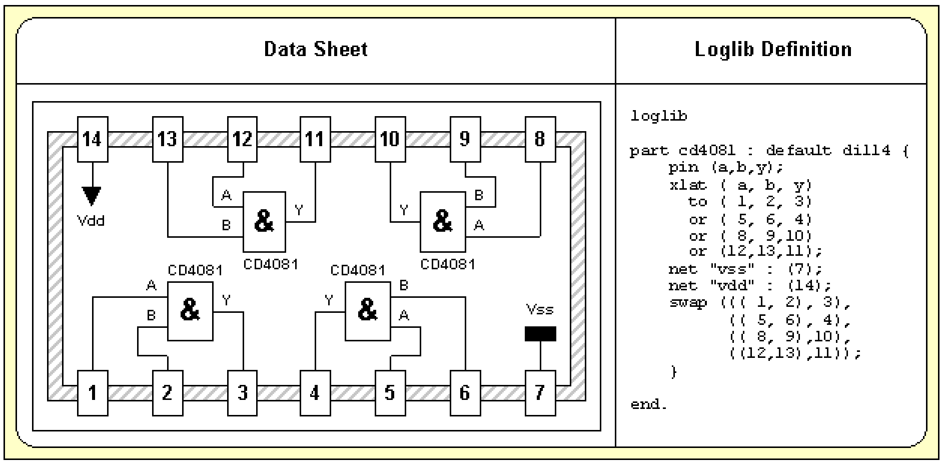
SCM Sheet Packager Error Tracking

SCM sheets which can be singled out to have caused **Packager** errors are automatically loaded when switching from the menu-driven **Packager** versions to the **Schematic Editor**. In that case, the **Schematic Editor** also triggers a Zoom Window to the first SCM symbol which caused a **Packager** error.

Correcting and/or Completing the Logical Library

A **LOGLIB** file must be edited to provide the missing information about the assignment of SCM symbol **`cd4081`** to its corresponding layout package. The **LOGLIB** utility program must then be applied to transfer the contents of that **LOGLIB** file to the **`demolib.ddb`** library file to be used for packaging.

Use your editor to prepare an ASCII file named **`cd4081.def`** containing the **LOGLIB** definition as shown in figure 3-2.



```
loglib

part cd4081 : default dil14 {
    pin (a,b,y);
    xlat ( a, b, y)
      to ( 1, 2, 3)
      or ( 5, 6, 4)
      or ( 8, 9,10)
      or (12,13,11);
    net "vss" : (7);
    net "vdd" : (14);
    swap ((( 1, 2), 3),
          (( 5, 6), 4),
          (( 8, 9),10),
          ((12,13),11));
    }

end.
```

Figure 3-2: Part CD4081 Data Sheet with Loglib Definition

The following just explains the contents of the **LOGLIB** definition in figure 3-2. See chapter 7.11 for a detailed description of the **LOGLIB** input file syntax.

The **LOGLIB** file always starts with the **`loglib`** keyword and ends with the **`end.`** keyword.

The **`part`** command is used to assign a layout package to the SCM symbol. The **`default`** keyword in the example indicates that this assignment is not imperative, i.e., it can be changed with an appropriate value assignment to the **`$plname`** attribute of the SCM symbol (e.g., for assigning an SMD instead of the standard DIL package).

The **`pin`** command defines the logical pins **`A`**, **`B`** and **`Y`** of the SCM symbol. Take care that this definition agrees with the pin definitions on the SCM symbol, or otherwise packaging will fail.

The **`xlat`** is used to allocate the gates of the physical part and to define the logical to physical pin mapping.

The **`swap`** command is used to define the rules for pin and gate swap. In the example above, the gates are mutually swapable, and the gate input pins are also swapable.

The **`net`** command is used to define power supply pins. In the example above, pin **`7`** is connected to signal **`Vss`** and pin **`14`** is connected to signal **`Vdd`**.

Translating the Loglib Definition

Use the following **LOGLIB** call to translate the logical library definition from **`cd4081.def`** and store it to the **`demolib.ddb`** library file:

```
> loglib cd4081 demolib
```

If the definitions in **`cd4081.def`** are correct, then the **LOGLIB** utility program should issue the following messages:

```
===================================
BARTELS LOGICAL LIBRARY MAINTENANCE
===================================

Program run successfully done.
```

Now **`demolib.ddb`** contains all information required for successfully packaging the project file **`demo.ddb`** (see also the contents of **LOGLIB** file **`demolib.def`** which has already been translated to **`demolib.ddb`**).

Logical Library Definition Query

The **Schematic Editor** Symbols menu provides a function named Show Symbol Logic for displaying logical library part definitions of selectable symbols of the currently loaded SCM sheet.

Show Symbol Logic, decodes and displays the internal logical library part definition stored with the **LOGLIB**. The logical library definition display header provides information on whether the **LOGLIB** definition was found in the current job file (Project) or in the default layout library (Library). The current job file is searched with higher priority. The default layout library file name is specified with the **`LAYDEFLIBRARY`** command of the **BSETUP** utility program (see also chapter 7.2).

The Show Symbol Logic function issues an error message such as **`File not found!`** on errors accessing the default library. An error message such as **`Symbol logic data ('symbolname') not found!`** is issued if the requested logical library definition is neither available in the job file nor in the default library.

Successful Packager Pass

Restart the **Packager** through the BAE main menu and use the following commands to call the **Packager** for transferring the logical net list in **`demo.ddb`** to a physical net list named **`board`** using library **`demolib.ddb`**:

Settings		
All Parameters		
Design File Name ?	demo ↵	
Design Library Name ?	demolib ↵	
Layout Element Name ?	board ↵	
Start		

The **Packager** will issue the following messages on the screen:

```
================
BARTELS PACKAGER
================

Design File Name ...........: 'demo'
Library File Name ..........: 'demolib'
Layout Element Name ........: 'board'
Active Schematic Sheet .....: 'sheet1'.
Active Schematic Sheet .....: 'sheet2'.
No error occurred!
```

The **`No error occurred!`** message means that the packaging has been successfully completed, and a (packed) physical net list named **`board`** was generated in the **`demo.ddb`** project file.

The **Packager** creates not only a physical net list, but also a logical net list with element name extension **`_log`**. In the example above a logical net list named **`board_log`** has been generated. The logical net list is not required by the system, but can be utilized for interfacing to certain foreign systems such as simulators (see also the **USERLIST** application example in chapter 3.4.3).

The **Packager** produces a free list named **`demo.fre`** for user information purposes only. You should examine the unconnected pins report and the net pin counts provided with this ASCII file (e.g., for checking on single pin signals).

The **Packager** annotates physical net list data such as part and pin names to the schematics. Check this by examining the SCM sheet elements after successfully running the **Packager**. Particularly you should have a close look at the four **`cd4081`** gates on **`sheet1`** of the example job file **`demo.ddb`**. All these gates have the same part name now (since they are all assigned to the same physical part), and the logical pin designators (**`A`**, **`B`** and **`Y`**, respectively) have been replaced by physical pin names (**`1`**, **`2`**, **`3`**, etc.).

Subsequent Tasks

After successfully completing the **Packager** process, the project is ready for layout, and you can enter the BAE **Layout Editor** to start with the PCB design. The name of the layout element to be created must match the name of the net list specified with the **Packager** run. I.e., the layout element name for the above-mentioned example project file **`demo.ddb`** should be **`board`**. Within the Layout system the library path should be set to library file **`demolib.ddb`** which has been used for packaging, and then the net list parts can be placed (e.g., by repeatedly applying the Place Next Part function from the Parts menu; see chapter 4.3.2 for details).

3.2.5 Messages

This section provides an alphabetically sorted list of all **Packager** error, warning and status messages. The `ERROR :` and `WARNING :` prefixes have been omitted since some messages are issued as errors or warnings depending on the Error Control parameter. The messages displayed on screen are logged into a file named `bae.log` and can be viewed after exiting the **Packager**.

The system automatically loads the sheet with the first erroneous SCM symbol and zooms to that symbol when switching immediately to the **Schematic Editor** after encountering error messages related to specific schematic symbols or their logical library definitions. The Symbols / Show Symbol Logic and/or Symbols / Edit Symbol Logic functions can be used to view and/or edit the logical library definition if the problem is caused by the logical library definition. These functions also take `$rlname` and/or `$rlext` attribute settings for alternative logical library definition name assignments into account.

The **Packager** copies the required logical library definitions from the selected layout library file to the project file. Local, project-specific logical library definitions have priority over external logical library definitions from the layout library in subsequent **Packager** runs. It is important to consider this behaviour when correcting logical library definitions. The Definition Update setting from Settings / Update Mode forces the **Packager** to restore and/or copy (corrected) logical library definitions from the layout library.

Any symbol pin changes such as the adding, deleting or renaming of pins require all of the affected SCM sheets to be reloaded and saved to update the logical netlists. Please note that although new and/or changed symbol pin names are displayed when re-loading a sheet, the SCM sheet has to be saved to update the logical netlist accordingly.

`[012] Abort - Database not changed!`

The **Packager** encountered one or more errors which prevented it from creating a valid layout net list. Consequently, the **Packager** was aborted without writing a layout net list to the project file. If a layout net list with the selected element name already existed in the project file, then this net list is left unchanged and can be reused. Please examine the **Packager** transcript and/or log file for the exact error cause(s).

`[019] Active Block/Schematic Sheet: '[nnn]blockname' / 'blocksheetname'.`

**** NOT YET DOCUMENTED ****

`[018] Active Schematic Sheet .....: 'sheetname'.`

The SCM sheet with the name `sheetname` is currently being processed. Subsequently listed error and warning messages relate to symbols on this sheet.

`[005] Alternate Library File Name : 'alternativelibraryname'`

This message displays the file name of the alternative layout and logical library used by the **Packager**. The **Packager** fetches and copies required layout part macros and logical library definitions from the alternative library to the project file if these definitions are neither in the project file nor in the default layout library. Since the **Packager** copies required layout part symbols to the project file, these part macros are available for subsequent part placement operations in the **Layout Editor**. The **Packager** saves the alternative library file name parameter in the project file, thus eliminating the need for specifying the same parameter again in subsequent **Packager** runs.

`[121] Assignment for $gp 'layoutpartpinname' not found!`

**** NOT YET DOCUMENTED ****

```
[076] Attribut data for symbol 'symbolname' sheet 'sheetname' is
missing!
```

**** NOT YET DOCUMENTED ****

```
[027] Block instance of net name 'netname' too long!
```

**** NOT YET DOCUMENTED ****

```
[026] Block instance of port name 'moduleportname' too long!
```

**** NOT YET DOCUMENTED ****

```
[028] Block instance of symbol name 'symbolname' too long!
```

**** NOT YET DOCUMENTED ****

```
[044] Cannot change library name for 'symbolname'. Missing default
command in logical definition!
```

**** NOT YET DOCUMENTED ****

```
[040] Cannot translate logical pin 'symbolpinname'!
```

**** NOT YET DOCUMENTED ****

```
[060] Cannot translate symbol 'symbolname'!
```

**** NOT YET DOCUMENTED ****

```
[127] Conflicting $netname net name assignments
'netname1'<>'netname2'!
```

**** NOT YET DOCUMENTED ****

```
[013] Connection list changed in layout
'layoutname'/'projectfilename'! Run Packager anyway?
```

**** NOT YET DOCUMENTED ****

```
[102] Database limit exceeded!
```

**** NOT YET DOCUMENTED ****

[094] Database read/write error (filename)!

**** NOT YET DOCUMENTED ****

[093] Database read/write error!

**** NOT YET DOCUMENTED ****

[106] Database record 'elementname' not found!

**** NOT YET DOCUMENTED ****

[105] Database record not found!

**** NOT YET DOCUMENTED ****

[098] Database structure corrupt (filename)!

**** NOT YET DOCUMENTED ****

[097] Database structure corrupt!

**** NOT YET DOCUMENTED ****

[077] Definition assigned to symbol 'symbolname' has invalid class: ('symbolmacroclass'<>'definitionclass')!

**** NOT YET DOCUMENTED ****

[003] Design File Name ....: 'projectfilename'

This message displays the currently processed project file name. The **Packager** converts the connections and symbols from the schematic sheets of the project file into a layout part and net list.

[138] Destination layout 'layoutname' loaded by user 'username'!

**** NOT YET DOCUMENTED ****

[014] Error calling program module!

**** NOT YET DOCUMENTED ****

[092] Error creating design database file 'filename'!

**** NOT YET DOCUMENTED ****

[091] Error creating design database file!

**** NOT YET DOCUMENTED ****

[110] Error writing ASCII file! Error writing free list!

**** NOT YET DOCUMENTED ****

[015] Error writing design database!

**** NOT YET DOCUMENTED ****

[100] Errors in file structure (filename)!

**** NOT YET DOCUMENTED ****

[099] Errors in file structure!

**** NOT YET DOCUMENTED ****

[011] Fatal internal error errornumber!

The **Packager** aborted due to an unexpected error (e.g., a list element search failed immediately after the element was saved in the list). There is likely to be either a hardware problem or an unknown error occurred which is not (yet) handled by the **Packager**. Please submit the project file to your Bartels support department for debugging if the error occurs with a specific project and is not sporadic (i.e., if the error can be reproduced).

[104] File 'filename' is not compatible with program version!

**** NOT YET DOCUMENTED ****

[136] File 'filename' is read only!

**** NOT YET DOCUMENTED ****

[108] File 'filename' not found!

**** NOT YET DOCUMENTED ****

```
[134] File 'filename' read access denied!
```

**** NOT YET DOCUMENTED ****

```
[103] File is not compatible with program version!
```

**** NOT YET DOCUMENTED ****

```
[135] File is read only!
```

**** NOT YET DOCUMENTED ****

```
[096] File not a database (filename)!
```

**** NOT YET DOCUMENTED ****

```
[095] File not a database!
```

**** NOT YET DOCUMENTED ****

```
[107] File not found!
```

**** NOT YET DOCUMENTED ****

```
[133] File read access denied!
```

**** NOT YET DOCUMENTED ****

```
[101] Function not available for this format!
```

**** NOT YET DOCUMENTED ****

```
[109] General database error!
```

**** NOT YET DOCUMENTED ****

```
[132] Ignored pins assigned to net 'n.c.'.
```

**** NOT YET DOCUMENTED ****

`[031] Illegal (negative) DRC block number for net 'netname'!`

**** NOT YET DOCUMENTED ****

`[029] Illegal (negative) net number for net 'netname'!`

**** NOT YET DOCUMENTED ****

`[041] Illegal net number for pin 'symbolpinname'!`

**** NOT YET DOCUMENTED ****

`[001] Invalid call!`

This error message is issued if the **Packager** (`logpack.exe` and/or `logpack`) is called directly from a **DOS** and/or **UNIX/Linux** shell. The **Packager** must be called from the BAE user interface to ensure that the required setup file and library path names are known and that the **Packager** can change (back) to the other BAE modules.

`[046] Invalid logical library name for symbol 'symbolname'!`

**** NOT YET DOCUMENTED ****

`[024] Invalid logical netlist format!`

**** NOT YET DOCUMENTED ****

`[051] Invalid logical part format : 'symbolname'!`

**** NOT YET DOCUMENTED ****

`[067] Invalid minimum distance for pin 'symbolpinname'!`

**** NOT YET DOCUMENTED ****

`[048] Invalid physical assignment for symbol 'symbolname'!`

**** NOT YET DOCUMENTED ****

`[047] Invalid physical library name for symbol 'symbolname'!`

**** NOT YET DOCUMENTED ****

`[065] Invalid power width for pin 'symbolpinname'!`

**** NOT YET DOCUMENTED ****

`[068] Invalid route priority for pin 'symbolpinname'!`

**** NOT YET DOCUMENTED ****

`[066] Invalid route width for pin 'symbolpinname'!`

**** NOT YET DOCUMENTED ****

`[058] Invalid swap data for pin 'layoutpartpinname'!`

**** NOT YET DOCUMENTED ****

`[009] Layout Element Name ........: 'layoutnetlistname'`

This message displays the name of the layout netlist to be created by the **Packager**. The same element name should be specified when (subsequently) creating the layout for this net list.

`[042] Layout Part Macro 'layoutpartmacroname' not in library!`

**** NOT YET DOCUMENTED ****

`[004] Library File Name ..........: 'libraryname'`

This message displays the file name of the default layout and logical library used by the **Packager**. The **Packager** fetches and copies required layout part macros and logical library definitions from the default library to the project file if these definitions are not yet in the project file. Since the **Packager** copies required layout part symbols to the project file, these part macros are available for subsequent part placement operations in the **Layout Editor**. The **Packager** saves the library file name parameter in the project file, thus eliminating the need for specifying the same parameter again in subsequent **Packager** runs.

`[043] Logical Definition for 'definitionname' not in library!`

**** NOT YET DOCUMENTED ****

`[052] Mainpart 'definitionname' subparts net internal deactivated!`

**** NOT YET DOCUMENTED ****

```
[045] Mainpart undeclared for definition 'definitionname'!
```

**** NOT YET DOCUMENTED ****

```
[022] Missing Schematic Netlist
```

**** NOT YET DOCUMENTED ****

```
[032] Multiple DRC block numbers for net 'netname'!
```

**** NOT YET DOCUMENTED ****

```
[070] Multiple colliding pin functions for pin 'layoutpartpinname'!
```

**** NOT YET DOCUMENTED ****

```
[030] Multiple net numbers for net 'netname'!
```

**** NOT YET DOCUMENTED ****

```
[035] Multiple placements for variant variant number
('symbolname')!
```

**** NOT YET DOCUMENTED ****

```
[126] Net 'netname' $nettype=mixed hence 'nettype1'<>'nettype2' of
'symbolname'!
```

**** NOT YET DOCUMENTED ****

```
[086] Net 'netname' is connected to inputs only!
```

**** NOT YET DOCUMENTED ****

```
[120] Net 'netname' multiple values 'attributename'
('value1'<>'value2' of 'symbolname')!
```

**** NOT YET DOCUMENTED ****

```
[056] Net assigned pin 'layoutpartpinname' not found!
```

**** NOT YET DOCUMENTED ****

`[055] Net assignment attribute 'attributename' invalid!`

**** NOT YET DOCUMENTED ****

`[054] Net assignment attribute 'attributename' not found!`

**** NOT YET DOCUMENTED ****

`[064] Net attribute missing for pin 'symbolpinname'!`

**** NOT YET DOCUMENTED ****

`[124] Net name collision 'netname'!`

**** NOT YET DOCUMENTED ****

`[125] Net name for 'layoutpartname'/'layoutpartpinname' too long!`

**** NOT YET DOCUMENTED ****

`[023] Netlist 'netlistname' not found!`

**** NOT YET DOCUMENTED ****

`[037] No assignment for symbol pin 'symbolpinname' found!`

**** NOT YET DOCUMENTED ****

`[017] No error occurred.`

**** NOT YET DOCUMENTED ****

`[071] No schematic sheet with block name 'blockname' found!`

**** NOT YET DOCUMENTED ****

`[016] Not enough memory!`

There was not enough main memory available for successfully processing the project data. It is highly unlikely that this error occurs with today's hardware configurations unless the computer's main memory is used to capacity by other processes.

[002] Not yet implemented!

The selected menu function and/or option is not available. This is usually due to the use of an intermediate and/or preliminary BAE version which displays menu functions or options which are not yet implemented. Please be patient, the selected function/option will be fully implemented in one of the next BAE versions and/or builds.

[021] Optimization End (n Parts deleted).

**** NOT YET DOCUMENTED ****

[020] Optimization Start (n Parts input).

**** NOT YET DOCUMENTED ****

[087] Output collision at net 'netname'!

**** NOT YET DOCUMENTED ****

[049] Part 'layoutpartname' ('symbolname') overloaded (List:) Logical part : 'symbolname'

**** NOT YET DOCUMENTED ****

[050] Part 'layoutpartname' ('symbolname') selected definition or layout macro incompatible with:

**** NOT YET DOCUMENTED ****

[128] Part 'layoutpartname' alternate part list deactivated by $plname!

**** NOT YET DOCUMENTED ****

[089] Part 'layoutpartname' contains unused gate (pin 'layoutpinname')!

**** NOT YET DOCUMENTED ****

[080] Part 'layoutpartname' different package assignments ('packagename1'<>'packagename2')!

**** NOT YET DOCUMENTED ****

```
[084] Part 'layoutpartname' different values for
'attributename'/variantnumber('value1'<>'value2')!
```

**** NOT YET DOCUMENTED ****

```
[033] Part 'layoutpartname' double defined!
```

**** NOT YET DOCUMENTED ****

```
[090] Part 'layoutpartname' mainpart unused!
```

**** NOT YET DOCUMENTED ****

```
[082] Part 'layoutpartname' multiple values for 'attributename'
('value1'<>'value2')!
```

**** NOT YET DOCUMENTED ****

```
[085] Pin 'layoutpartname/layoutpartpinname' diff. values for
'attributename'/variantnumber('value1'<>'value2')!
```

**** NOT YET DOCUMENTED ****

```
[083] Pin 'layoutpartname/layoutpartpinname' multiple values for
'attributename' ('value1'<>'value2')!
```

**** NOT YET DOCUMENTED ****

```
[069] Pin 'layoutpartpinname' invalid pin function
'pinfunctionspecification'!
```

**** NOT YET DOCUMENTED ****

```
[053] Pin 'layoutpartpinname' multiple section assignment!
```

**** NOT YET DOCUMENTED ****

```
[075] Pin 'layoutpartpinname' not found for attribute
'attributename' assignment!
```

**** NOT YET DOCUMENTED ****

[038] Pin 'layoutpartpinname' used more than once or differently!

**** NOT YET DOCUMENTED ****

[073] Pin 'symbolbuspinname' bus name too long!

**** NOT YET DOCUMENTED ****

[063] Pin 'symbolpinname' assignment not found!

**** NOT YET DOCUMENTED ****

[039] Pin 'symbolpinname' connected differently!

**** NOT YET DOCUMENTED ****

[074] Pin 'symbolpinname' not found for gate/group request!

**** NOT YET DOCUMENTED ****

[072] Port pin 'layoutpartpinname' not found!

**** NOT YET DOCUMENTED ****

[117] Questionable Multi Part 'layoutpartname' divergent $noplc Status :

**** NOT YET DOCUMENTED ****

[122] Questionable Part 'layoutpartname' Pin 'layoutpartpinname' ('pinfunctionname')

**** NOT YET DOCUMENTED ****

[113] Questionable Part : 'layoutpartname'

A problem occurred when processing the layout part with the name `layoutpartname`. Please examine subsequent error message for more detailed information.

[114] Questionable Symbol : 'symbolname'

A problem occurred when processing the schematic symbol with the name `symbolname`. Please examine subsequent error message for more detailed information.

[115] Questionable Variant Part : 'layoutpartname'

A problem occurred when processing the (variant) layout part with the name **`layoutpartname`**. Please examine subsequent error message for more detailed information.

[131] Respectively no pin named 'layoutpartpinname' on part macro 'layoutpartmacroname'!

**** NOT YET DOCUMENTED ****

[025] Single Sub Block 'blockname' multiple usage!

**** NOT YET DOCUMENTED ****

[057] Swap defined pin 'layoutpartpinname' not found!

**** NOT YET DOCUMENTED ****

[137] Symbol 'symbolname' 'attributename' not unique ('value1'<>'value2')!

The logical library definition of the symbol **`symbolname`** contains multiple **`newattr`** commands with different value assignments to the **`attributname`** attribute.

This problem can be fixed by removing the redundant/ambiguous **`newattr`** commands from the logical library definition of the symbol.

[123] Symbol 'symbolname' Pin 'symbolpinname'

**** NOT YET DOCUMENTED ****

[079] Symbol 'symbolname' definition assignment not allowed!

**** NOT YET DOCUMENTED ****

[078] Symbol 'symbolname' definitions inconsistent: (mainpart/subpart 'symbolmacroname'<>'alternativedefinitionname')!

**** NOT YET DOCUMENTED ****

[111] Symbol 'symbolname' double defined!

**** NOT YET DOCUMENTED ****

[036] Symbol 'symbolname' not defined!

**** NOT YET DOCUMENTED ****

[088] Symbol 'symbolname', ignoring $rlext 'definitionextension' (name too long)!

**** NOT YET DOCUMENTED ****

[130] Symbol macro 'symbolmacroname' changed after sheet 'sheetname' was saved. Any pin change will be ignored!

**** NOT YET DOCUMENTED ****

[112] Symbol pin 'symbolpinname' not found for attribute 'attributename' assignment!

**** NOT YET DOCUMENTED ****

[081] Synthetic name for part 'symbolname' too long!

**** NOT YET DOCUMENTED ****

[008] Test Point Mode .... : modespecification

**** NOT YET DOCUMENTED ****

[006] Test Point Name Prefix .... : 'partnameprefix'

**** NOT YET DOCUMENTED ****

[007] Test Points Logical Library : 'definitionname'

**** NOT YET DOCUMENTED ****

[010] The file names contain invalid characters!

Either the specified project file name or the specified library file name contains invalid (control) characters or **?** and/or ***** characters, or the project file name (Settings / Design File Name) was not set.

Please set a project file name and/or use file names without special characters to avoid this problem.

`[116] The layout element name contains invalid characters!`

The layout element (net list) name specified through Settings / Layout Element Name is either empty or contains invalid (control) characters or **?** and/or ***** characters.

Please set a valid layout element without special characters to avoid this problem.

`[062] Variant part 'layoutpartname'/'symbolname': name conflict with $rpname (List:)`

**** NOT YET DOCUMENTED ****

`[061] Variant symbol 'symbolname': xlat with gates not allowed!`

**** NOT YET DOCUMENTED ****

`[118] Variant variantnumber sub symbol 'symbolname': not placed!`

**** NOT YET DOCUMENTED ****

`[119] Variant variantnumber sub symbol 'symbolname': placed!`

**** NOT YET DOCUMENTED ****

`[034] Virtual variant parts not allowed ('symbolname')!`

**** NOT YET DOCUMENTED ****

`[059] xlat pin 'layoutpartpinname' not found on layout part macro!`

**** NOT YET DOCUMENTED ****

3.3 Backannotation

Backward annotation with the Backannotation is always required after performing layout net list modifications such as part name changes, pin/gate swaps, or alternate part package type assignments. Part names can be changed using the Netlist Part Name function from the menus for interactive part placement. Interactive pin/gate swaps are applied with the Pin/Gate Swap function from the manual part placement menu. Automatic pin/gate swap is performed with the Full Autoplacer, Single Pass Optimizer and Multi Pass Optimizer autoplacement functions, and can also be performed when using the Router P/G-Swap On option with the rip-up router of the **Autorouter** module. Alternate package types can be assigned using the Alternate Part submenu function during manual part placement. Note that any of the net list modifications mentioned above will get lost when re-editing the schematic without running the Backannotation.

3.3.1 Starting the Backannotation

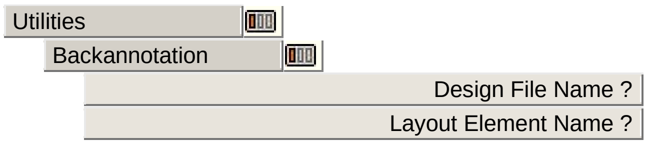
Backannotation is fully integrated to the **Schematic Editor** and can be explicitely started using the Backannotation command from the Utilities menu.

Automatic Backannotation Requests Processing

A feature for automatically processing pending Backannotation requests is integrated to the **Schematic Editor** functions for loading schematic plans. Backannotation requests are generated when saving layouts with net list modifications such as pin/gate swaps or changed part names. Loading an SCM sheet with a pending Backannotation request automatically activates a verification menu which allows to backannotate the currently processed design. The Backannotation request is deleted after successfully running the Backannotation.

3.3.2 Running the Backannotation

Design and/or project file name and layout element name prompts are activated after calling Backannotation (pressing the ESC key aborts the Backannotation):



The design file name is the name of the project file to be selected for Backannotation. The design file must be available with extension `.ddb` but this extension must not be included with the file name specification. If the user types an empty string (by pressing the return key ↵) to the design file name prompt, then the system will automatically use the file name of the currently loaded or the previously processed element (i.e., the global project name).

The layout element name is the name of the layout and/or the net list to be backannotated. If the user types in an empty string (by pressing the return key ↵) to the layout element name prompt, then the system uses the default layout element name defined with BAE setup (see also command `LAYDEFELEMENT` of the **BSETUP** utility program).

After successfully processing the layout the Backannotation will generate a modified logical net list in the project file. For user information purposes Backannotation also creates an assignments file under the design file name with extension `.ass`. The assignments file is an ASCII file containing an annotation report (part name changes, logical to physical pin mapping).

The system issues the `No errors occurred!` message after successfully completing the Backannotation process, and the user can press any key to exit from the Backannotation function. Errors during Backannotation usually result from a wrong layout element name specification and/or from missing logical net list data. Missing SCM symbols and/or parts could also prevent the Backannotation from completing successfully. The Backannotation lists missing symbols/parts. These must be replaced on the SCM sheet(s) to allow for subsequent Backannotation processes to be completed successfully (the missing parts problem could also be "solved" by simply re-running the **Packager** instead of the Backannotation, but all netlist-specific layout modifications scheduled for Backannotation such as pin/gate swaps would then be lost).

3.3.3 Example

This section describes how to run Backannotation on the **`demo.ddb`** example project file which has been created in the previous chapter. You should change to the directory where the **`demo.ddb`** file resides.

Start the **Schematic Editor** through the BAE main menu and use the following commands to call Backannotation for transferring the physical net list named **`board`** from the **`demo.ddb`** design file back to the schematics:

Utilities			
	Backannotation		
		Design File Name ?	demo ↵
		Layout Element Name ?	board ↵

Backannotation displays the following messages:

```
==============================
BARTELS BACKANNOTATION UTILITY
==============================

Design File Name ........: 'demo'
Layout Element Name .....: 'board'
No error occurred!
```

The **`No error occurred!`** message means that Backannotation was successfully completed and the logical net list in project file **`demo.ddb`** has been annotated with the physical net list named **`board`**.

Backannotation produces an assignment file named **`demo.ass`** which is not required by the BAE design system. You can use the part name changes report and the logical to physical pin mapping lists provided with this ASCII file for backannotating to foreign CAE systems.

Backannotation annotates physical net list modifications such as part name changes and pin/gate swaps to the schematic sheets. You can check this by examining the SCM sheet elements after successfully running the Backannotation.

3.4 Net List Utilities

Within the **Bartels AutoEngineer** design process the net list is usually created with the **Schematic Editor**. Besides that BAE provides the subsequently mentioned utilities for creating and/or converting various foreign ASCII net list formats. See also chapter 7 for a detailed description of these utility programs.

3.4.1 Importing Logical Net Lists

Importing logical (i.e., unpacked) net lists to the **Bartels AutoEngineer** requires a subsequent **Packager** pass to prepare for the layout design. A DDB library file for the **Packager** process including all of the requested layout symbols and logical library entries is required.

The **NETCONV** (Logical Netlist Conversion Utility) program is used to transfer logical net list data from BAE ASCII format to internal **Bartels AutoEngineer** format ready for processing with the BAE **Packager**.

3.4.2 Importing Physical Net Lists

No packaging is required when importing physical (i.e., packed) net lists to the **Bartels AutoEngineer**. Nevertheless, a DDB library file including all of the requested layout symbols must be available for the subsequent layout design process.

The **CONCONV** (Connection Conversion Utility) program is used to transfer physical net lists to the **Bartels AutoEngineer**. Supported ASCII net list formats are BAE, CALAY, MARCONI and RACAL. Net lists from other systems can often be written in one of these formats providing a convenient link between other schematic systems and the powerful BAE layout system.

The **REDASC** (REDAC ASCII Input Interface) program is used to transfer layout data from Redac MAXI system, i.e., **REDASC** converts layout symbols, part lists, net lists, placement information etc. from Redac CDI ASCII format to internal BAE DDB format.

3.4.3 Exporting Net List Data

Either the **Bartels User Language** programming facilities (see the Bartels User Language Programmer's Guide for details on how to apply **Bartels User Language**) or the **USERLIST** (User Programmable List Generator) utility program can be utilized to provide tools for exporting net list data in arbitrary formats. Use your editor to create a **USERLIST** script named **`netlist.usf`** with the following contents:

```
EXTENSION = ".nl";
PRINT("NETLIST ",PROJECTNAME,";",CR);
FOR (ALL PARTS)
{
    PRINT(PARTNAME:-5,"(",$plname:-8,") : ");
    CLEARCOUNTER;
    FOR (ALL PINS)
    {
        IF (NETPINCOUNT > 1)
        {
            PRINT(PINNAME,"(",NETNAME,")");
            COUNTUP;
            IF (COUNTVALUE > 4) { PRINT(CR,TAB);CLEARCOUNTER; }
        }
    }
    PRINT(";",CR);
}
ENDSPEC
```

Use the following **USERLIST** call to run the `netlist.usf` **USERLIST** script to export the (physical) net list `board` from job file `demo.ddb` to ASCII net list file `demo.nl`:

```
> userlist netlist demo board ↵
```

The above **USERLIST** call produces the `demo.nl` net list file with the following contents:

```
NETLIST board;
c100 (chip1210) : 1(vss)2(vdd);
c101 (chip1206) : 1(vss)2(net);
ic10 (dil14   ) : 1(@7)10(@9)11(@2)12(@3)13(@9)14(vdd)2(net)
    3(@6)4(@4)5(@5)6(@6)7(vss)8(@11)9(@4);
k1   (relais  ) : a1(vdd)a2(@14)c1(vdd)c2(vdd)nc1(bus.out1)
    nc2(bus.out3)no1(bus.out0)no2(bus.out2);
r100 (r04a25  ) : 1(@7)2(@6);
r101 (r04a25  ) : 1(@5)2(@4);
r102 (r04a25  ) : 1(@11)2(@9);
r103 (r04a25  ) : 1(@3)2(@2);
r104 (minimelf) : 1(@2)2(@8);
r105 (chip1206) : 1(net)2(vdd);
s1000(s1dilo  ) : 1(@7)2(net);
s1001(s1dilo  ) : 1(@5)2(net);
s1002(s1dilo  ) : 1(@11)2(net);
s1003(s1dilo  ) : 1(@3)2(net);
s1004(s1dilo  ) : 1(net)2(vss);
s1005(s1dilo  ) : 1(net)2(vss);
s1006(s1dilo  ) : 1(net)2(vss);
s1007(s1dilo  ) : 1(net)2(vss);
s1008(s1dilo  ) : 1(net)2(vss);
s1009(s1dilo  ) : 1(net)2(vss);
v1   (to92    ) : 1(vss)2(@8)3(@14);
v1000(d04a25  ) : a(@14)k(vdd);
x1000(xsubd9bl) : 1(vss)4(bus.out0)5(bus.out1)6(bus.out2)7(bus.out3)
    9(vdd);
```

Use the following **USERLIST** call to run the **`netlist.usf`** **USERLIST** script to export the (logical) net list **`board_log`** from job file **`demo.ddb`** to ASCII net list file **`demo.nl`**:

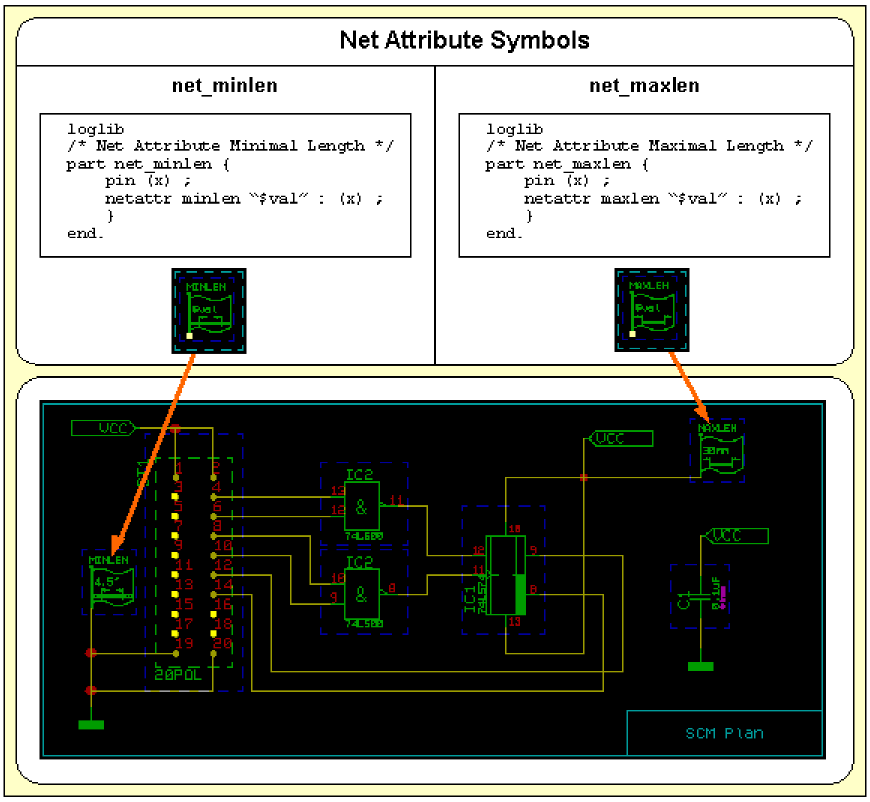
```
> userlist netlist demo board_log ↵
```

The above **USERLIST** call produces the **`demo.nl`** net list file with the following contents (note the different pin designators and **`$plname`** settings when comparing to the preceding example):

```
NETLIST board_log;
c100 (c        ) : 1(vss)2(vdd);
c101 (c        ) : 1(vss)2(net);
ic10 (cd4081   ) : a(@7)b(net)y(@6);
ic11 (cd4081   ) : a(@5)b(@6)y(@4);
ic12 (cd4081   ) : a(@11)b(@4)y(@9);
ic13 (cd4081   ) : a(@3)b(@9)y(@2);
k10  (rels     ) : a1(vdd)a2(@14);
kk100(relc     ) : c(vdd)nc(bus.out1)no(bus.out0);
kk101(relc     ) : c(vdd)nc(bus.out3)no(bus.out2);
r100 (r        ) : 1(@7)2(@6);
r101 (r        ) : 1(@5)2(@4);
r102 (r        ) : 1(@11)2(@9);
r103 (r        ) : 1(@3)2(@2);
r104 (r        ) : 1(@2)2(@8);
r105 (r        ) : 1(net)2(vdd);
s1000(s_1dil   ) : 1(@7)2(net);
s1001(s_1dil   ) : 1(@5)2(net);
s1002(s_1dil   ) : 1(@11)2(net);
s1003(s_1dil   ) : 1(@3)2(net);
s1004(s_1dil   ) : 1(net)2(vss);
s1005(s_1dil   ) : 1(net)2(vss);
s1006(s_1dil   ) : 1(net)2(vss);
s1007(s_1dil   ) : 1(net)2(vss);
s1008(s_1dil   ) : 1(net)2(vss);
s1009(s_1dil   ) : 1(net)2(vss);
v1   (tr_bc517) : b(@8)c(@14)e(vss);
v1000(d        ) : a(@14)k(vdd);
x1000(x_subd9b) : 1(vss)4(bus.out0)5(bus.out1)6(bus.out2)7(bus.out3)
    9(vdd);
```

3.4.4 Net Attributes

The BAE **Packager** provides functions for transferring arbitrary net attributes. These attributes are set with special (user-defined) net attribute symbols on SCM sheet level using the Assign Value function from the Symbols menu. Logical library entries must be defined for packaging the net attribute definitions. Figure 3-3 illustrates how to define and use net attribute symbols and corresponding logical library entries (see also chapter 7.11 for a detailed description of the **LOGLIB** utility program).



```
loglib
/* Net Attribute Minimal Length */
part net_minlen {
     pin (x) ;
     netattr minlen "$val" : (x) ;
     }
end.
```

```
loglib
/* Net Attribute Maximal Length */
part net_maxlen {
     pin (x) ;
     netattr maxlen "$val" : (x) ;
     }
end.
```

Figure 3-3: Net Attribute Definitions

Chapter 4
PCB Design /CAD

This chapter describes how to use the **Layout Editor**, the **Autorouter**, the **CAM Processor** and the **CAM View** for creating layout library symbols and designing PCB layouts and for generating and processing CAM output and manufacturing data for the PCB production. The examples presented in this chapter introduce the concepts and advanced features of the BAE PCB design system in a logical sequence and will take the user from the creation of layout library symbols and the design and modification of a PCB layout through to the generation of manufacturing data for the PCB production. The PCB design example is based on the circuit drawing and net list data prepared in the preceding chapters and will be subject to further processing in the following chapter. The reader should work through this chapter without missing any sections to gain full understanding of the BAE PCB design system. Once a command has been used and/or explained, the operator is assumed to have understood its function and be able to perform it again. Subsequent instructions containing this command will be less verbose for easier reading and more speedy learning.

Contents

Tables

Figures

4.1 General

The **Bartels AutoEngineer** PCB layout system essentially consists of an interactive **Layout Editor** with integrated layout symbol editor and functions for automatic part placement and automatic copper fill, the **Bartels AutoEngineer** for fully automated PCB routing, and **CAM Processor** and **CAM View** program modules for generating and processing CAM output and manufacturing data for PCB production. The following sections of this manual describe in detail how to use these modules.

4.1.1 Components and Features

Layout Editor

The floating point database used throughout the BAE **Layout Editor** allows for the mixed specification of metric and imperial coordinate units. All placement coordinates including rotation angles can be specified with floating point values. There are no restrictions at the definition and placement of pads, copper areas, parts, traces, texts, etc. The user interface provides a genuine display even for the more complex structures such as circles or arcs. Area visibility can be defined according to mirror modes, thus supporting SMD pins with different pad shapes for component and solder side.

The Design Rule Check (DRC) operates in grid-free mode with a precision of eight digits behind the decimal point. The DRC provides constant monitoring of copper against net list, with visual indication of distance violations and short-circuits. The system supports both Online and Batch DRC. Online DRC performs incremental checks, i.e., only the modified items are checked real-time thus saving a lot of computation time.

The Undo/Redo facilities allow to use the **Layout Editor** without fear of causing damage. Up to twenty commands can be reversed or undone by applying the Undo function and then reprocessed with the Redo function. Undo/Redo ensures data security and provides a powerful feature for estimating design alternatives.

Arbitrary parts of the layout can be selected to groups and then moved, rotated, mirrored, copied or saved (and subsequently used as templates).

Fast interactive part placement with instant connections update guarantees an optimum exploitation of the layout area. Part connections (unroutes, airlines) are dynamically calculated and instantly displayed during placement. Parts can be rotated at arbitrary angles, and they can be mirrored for SMT applications. Parts can be placed at arbitrary coordinates and polar coordinates are supported for placing parts on a circle. The part library supports part-specific preferences for part rotation and part mirroring. During manual placement, an alternate package type can be selected for the currently processed part.

The system supports genuine net-orientated Connectivity. This means that electrical connections can be realized through copper areas instead of traces connecting. Traces and copper areas can be created in grid-free mode with floating point precision. Highlight is used to indicate the selected and/or processed signal, and each point of the signal can be connected. The system also supports arc shaped traces as well as blind and buried vias with automatic via type selection.

The BAE layout system provides powerful automatic copper fill functions. Copper areas can optionally be generated with line or cross hatching. The BAE layout system also supports power layer definitions. Arbitrarily shaped active copper areas can be placed on power layers, thus featuring split power planes.

High frequency and analogous applications are supported by useful facilities for shrinking and/or enlarging copper areas at arbitrary expansion values. Trace segment lengths and copper area edge lengths can automatically be determined or measured for test or documentation purposes.

With the **Bartels User Language** integrated to the **Layout Editor** the user is able to implement enhanced CAD functions and macros, user-specific post processors, report and test functions, etc. **User Language** programs can be called by applying the Run User Script function from the File menu or by pressing a key on the keyboard (hot key).

Autoplacement

The BAE layout system is equipped with powerful **Autoplacement** functions. Parts can be selected and/or deselected for automativ placement according to set principles either by specifying part names (`c1`, `r2`, `ic15`, etc.) or by specifying library symbol names (`dil14`, `so20`, `plcc44`, etc.), where wildcards are also permitted. The user can also select parts from certain blocks of a hierarchical circuit design. The functions for designating the parts to be placed are a useful feature for controlling the placement process in a way that e.g., first the plug(s) can be placed, then the DIL packages, then the capacitors, etc.

The matrix placement facilities are used for the automatic placement of selectable part sets to definable placement grids. Matrix placement is intended especially for the initial placement of homogeneous part types such as memory devices, block capacitors, test points, etc. Of course the matrix placement function also considers the default rotation angle and mirror mode settings.

Initial placement functions are provided for performing a complete placement by automatically placing not yet placed parts inside the board outline. The parts are placed onto the currently selected placement grid, where pre-placed parts such as plugs and LEDs are considered as well as the connectivity derived from the net list. The initial placement algorithm features automatic SMD and block capacitor recognition. The solder side can optionally be used for placing SMDs. Parts are automatically rotated in 90 degree steps, where part rotation can optionally be restricted to provide fault-robust insertion data. A part expansion parameter can be set to support automatic part spacing. The initial placement process is controlled by adjustable heuristic weight factors for evaluating part segment fitting and for considering different net list structures. Intermediate rip-up/retry passes are activated during initial placement to optimize board area exploitation.

The system also provides placement optimization functions for automatic part and pin/gate swap. The part swap facility mutually exchanges identical components at their insertion position to minimize unroutes lengths. The pin/gate swap facility analogously performs an iterative exchange of gates and/or pins and pin groups, where gates or groups can also be swapped between different parts. Fixed parts are excluded from placement optimization. The admissibility of pin/gate swaps is fairly controlled with appropriate library definitions. Either single pass or multi pass optimization can be applied, with the swap method (only part swap, only pin/gate swap or both part and pin/gate swap) to be selected as required. Applying placement optimization will usually cause a significant simplification of the routing problem, thus resulting in a considerable time saving at the subsequent Autorouting process.

Bartels Autorouter

The **Bartels Autorouter®** is the culmination of years of research and practical routing experience. It has an impressive set of features and has the flexibility to route a wide range of PCB technologies, including analog, multilayer and SMT designs. The **Bartels Autorouter®** has been incorporated in leading PCB layout systems throughout the world, setting new industrial standards of success, reliability and flexibility and dramatically reducing the requirement for manual routing in all boards. Its incorporation into other systems has varied considerably. With **Bartels AutoEngineer**, all of the advanced features and capabilities of the **Bartels Autorouter®** are available. With the **Bartels Autorouter®**, an experienced BAE user should be able to complete the design of a eurocard PCB including schematic drawing and manufacturing data output in not more than one or two days.

Bartels Autorouter® is based on special backtracking/rip-up/reroute algorithms. These types of routing algorithms have been proven to be the only ones capable of achieving 100% route completion on the majority of today's challenging PCB designs. Extensive artificial intelligence features have been built into the router to produce high quality design results in a reasonable time. The **Bartels Autorouter®** includes special features for production optimization giving excellent yield in board manufacture. The complete routing process is supervised by a backtracking algorithm, which not only prevents from a deterioration of the result or a dead-lock during rip-up or optimization but also is able to exploit a new and/or better routing solution.

Both the *selective* rip-up and the cross-net optimizer passes are assisted by a unique intelligent multi-net cleanup algorithm with pattern recognition. This algorithm identifies traces blocking unrouted connections and rips up and reroutes multiple connections or even trees at a time to improve the global routing result. The **Autorouter** is capable of moving trace bunches to make space for not yet routed connections (push'n'shove routing). Cleanup passes for performing cross-net changes are applied during optimization and will considerably reduce via counts and straighten trace paths. All advanced routing features are supported by a sophisticated array of heuristic cost parameters which are dynamically adapted to produce quality routing results comparable to those made by skilled layout designers.

The **Bartels Autorouter®** is capable of simultaneously routing up to 28 layers (16 signal layers and 12 power layers; multilayer routing). The **Autorouter** automatically identifies and connects pre-placed power supply structures thus ensuring correct power and ground routing. The router also utilizes automatic routines for correctly connecting SMD pins to power layers wherever necessary. This feature also supports any split power planes placed on the power layers (split power plane routing). Routing widths are pin-specific rather than net-specific. SMD pins are routed with the anticipated pin routing widths to avoid perpendicular positioning of SMD parts during the soldering process. T-shaped connections are automatically generated (full copper sharing).

The routing progress can be watched both graphically and on statistical readout on the **Autorouter** user interface. The **Autorouter** can be stopped at any time and then continued or re-started with changed parameters on demand.

Features are provided for automatically adapting layouts to placement and/or net list changes (re-entrant routing) where the **Autorouter** evaluates pre-routed traces, identifies and removes wrong and redundant paths, and then routes open connections to achieve a correct 100% solution. Subsequently, the modified layout can be optimized again.

All of today's advanced PCB technologies are fully supported by the **Bartels Autorouter®**. The **Autorouter** is able to consider and/or connect arbitrarily shaped pads, traces, copper areas, power planes and keepout areas. The built-in off-grid recognition allows off-grid placement of parts and pre-routed traces. The **Autorouter** supports standard routing grids with optional half-grid routing (1/20 inch to 1/100 and/or 1/200 inch), arbitrary routing grids for special pin pitches and grid-free routing. Multilayer SMD technologies are supported by the **Autorouter**'s SMD via preplace and BGA (Ball Grid Array) fanout routing algorithms, which can optionally be used if via pre-allocation is required for connecting SMD and/or BGA pins to inside layers with high priority. The **Bartels Autorouter®** supports microvias (via-in-pad technology) and blind and buried vias with automatic via type selection thus increasing the routability of multilayer layouts and also supporting new PCB manufacturing technologies such as processes for producing plasma-etched vias. Area and block routing methods can be supported by defining keepout routing areas and/or prohibited layers, and the **Autorouter** is capable of considering via keepout areas.

The **Autorouter** provides the same initial placement and placement optimization functions as already known from the **Layout Editor**, allowing for automatic pre-placement and placement optimization before starting the autorouting process.

The **Autorouter** provides enhanced routing features such as placement optimization (pin/gate swap) during rip-up/retry routing, gridless routing, single net routing, component routing, area routing, mixed-grid routing, etc. The place and route function activates both the Full Autoplacer (including complete initial placement and placement optimization) and the Full Autorouter (including complete Initial Router, rip-up/retry routing and optimizer), thus performing complete placement and routing on the push of a button. The single net routing function of the **Autorouter** is used for automatically routing selectable nets and/or connections. The single net router can be applied to pre-rout power supply or critical nets with specific preferences for trace widths, clearance, layer assignments, etc. For completeness reasons the single net router also supports an option for selectively deleting already routed nets and/or connections. The **Autorouter** provides a feature for selectively routing net groups (e.g., busses of a certain circuit block) using specific routing options for preferred routing directions, trace widths, etc. The component routing function can be used to rout selectable parts. The area/block routing function of the **Autorouter** is used for routing selectable areas and/or circuit blocks. It is possible to define different routing areas according to circuit topology (I/O, memory, digital and/or analog part of the design, etc.) and perform autorouting using block-specific routing options such as routing grid, clearance, preferred routing direction, bus routing preferences, etc. During the rip-up autorouting process the **Autorouter** utilizes features for automatically performing selective component and/or pin/gate swaps in cases where this might result in better routability of swapped parts, gates or pins.

The Undo/Redo facilities allow to use the **Autorouter** without fear of causing damage. Up to twenty commands can be reversed or undone by applying the Undo function and then reprocessed with the Redo function. Since this is true even for most complex operations such as complete **Autorouter** passes, Undo/Redo ensures data security and provides a powerful feature for estimating design alternatives.

With the **Bartels User Language** integrated to the **Autorouter**, the user is able to implement enhanced CAD functions and macros, user-specific placement and autorouting procedures, report and test functions, etc. **User Language** programs can be called by applying the Run User Script function from the File menu or by pressing a key on the keyboard (hot key).

The **Autorouter** module of the **BAE HighEnd** system provides advanced autorouting technologies based on patented neural network technology (**Neural Autorouter**). The **BAE HighEnd Autorouter** supports skilled analog signal routing, automatic microwave structure generation, grid-less object-orientated routing with automatic placement optimization, etc. The **BAE HighEnd Autorouter** also provides features for routing problem recognition and/or classification and for learning and automatically applying problem-adapted routing strategies and/or rules.

CAM Processor

The **Bartels AutoEngineer** layout system allows for the definition and creation of arbitrarily shaped areas and pads which can also be rotated at any angle. All these features are supported by the intelligent **CAM Processor**, which utilizes a series of automatic fill algorithms with multiple aperture selection and area reduction. I.e., pad definitions are not necessarily restricted to some particular tool set, but may fully support the demanded technology. The **CAM Processor** automatically optimizes the use of the selected tools and aperture table definitions to generate all structures at highest possible precision with regard to the chosen plot parameters and output tolerance ranges. The system also provides appropriate error messages and highlight features to indicate objects which cannot be plotted using the selected tools and the current plot parameter settings. In-built functions are available for producing HP-GL pen plot, PCL laser print and Postscript output, and for supporting configurable drivers for Gerber photoplotting with freely definable aperture tables. The **CAM Processor** also provides programming data for auto insertion and pick and place equipment, solder mask, drill data and SMD adhesive masks. CAM output is generated considering general parameters such as tool tolerance, scaling factor, CAM rotation and/or mirroring, freely definable CAM origin, etc. Special plot parameters for generating negative power layer and split power plane plots are also supported. With the flexibility of easily preparing all of the required output data the BAE user has free choice amongst photo plot producers and PCB manufacturers.

With the **Bartels User Language** integrated to the **CAM Processor** the user is able to implement enhanced CAM functions and macros, user-specific and/or batch-driven post processors, report and test functions, etc. **User Language** provides unrestricted access to the BAE design database. The user is able to implement user-specific programs for exporting any data such as part lists, net lists, geometry data, drill data, insertion data, test data, milling data, etc. in freely definable formats. **User Language** programs can be called by applying the Run User Script function from the File menu or by pressing a key on the keyboard (hot key).

CAM View

The **CAM View** module provides features for displaying Gerber data, drilling data (Sieb&Meier and/or Excellon) and milling data (Excellon) in order to check CAM data validity and to estimate the efficiency of tool usage. **CAM View** features multiple input and sorted output with variable offsets and adjustable aperture tables to support panelization. **CAM View** also provides a powerful function for translating Gerber data to BAE layout design data, i.e., with **CAM View** the user is able to import Gerber data produced by foreign PCB layout systems.

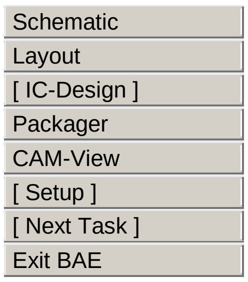
With **Bartels User Language** integrated to the **CAM View** module the user is able to implement enhanced **CAM View** functions and macros, user-specific batch procedures, report and test functions, etc. **User Language** programs can be called by applying the Run User Script function from the File menu or by pressing a key on the keyboard (hot key).

4.1.2 Starting the Layout System

It is recommended to start the **Bartels AutoEngineer** from the directory where the design files should be generated since this considerably simplifies job file access. If you intend to process the examples provided with this manual it is recommended to move to the BAE examples directory installed with the BAE software. The **Layout Editor** can be called from the **Bartels AutoEngineer** main shell. Start the BAE shell by typing the following command to the operating system prompt:

```
> bae
```

The **AutoEngineer** comes up with the Bartels logo and the following menu (the Setup function is only available under Windows/Motif; the IC-Design and Next Task menu items are available only with special software configurations such as **BAE HighEnd** or **BAE IC Design**):



Move the menu cursor to the Layout menu item and confirm this choice by pressing the left mouse button:

The **Layout Editor** program module is loaded and the **Layout Editor** menu is activated. Check your BAE software installation if this fails to happen (see the Bartels AutoEngineer® Installation Guide for details on how to perform a correct installation).

It is also possible to call the **Layout Editor** directly from the **Packager**. In this case, layout element creation for the net list created by the **Packager** is automatically suggested for designs not yet containing a corresponding layout.

4.1.3 Layout Editor Main Menu

The **Layout Editor** standard/sidemenu user interface provides a menu area on the right side, consisting of the main menu on top and the currently active menu below that main menu. After entering the **Layout Editor** the Files menu is active and the menu cursor points to the Load Element function.

The Windows and Motif versions of the **Layout Editor** can optionally be operated with a pull-down menu user interface providing a horizontally arranged main menu bar on top. The **`WINMENUMODE`** command of the **BSETUP** utility program is used to switch between **`SIDEMENU`** and **`PULLDOWN`** Windows/Motif menu configurations (see chapter 7.2 for more details).

The following main menu is always available whilst processing layout elements with the **Layout Editor**:

- Undo, Redo
- Display
- Files
- Parts
- Traces
- Areas
- Text, Drill
- Groups
- Parameter
- Utilities

Undo, Redo

The functions provided with the Undo, Redo menu allow you to use the **Layout Editor** without fear of causing damage. Up to twenty commands can be reversed or undone using Undo and then reprocessed with the Redo. This is true even for complex processing such as group functions or **User Language** program execution. Undo, Redo ensures data security and provides a powerful feature for estimating design alternatives.

Display

The View or Display menu can either be activated by selecting the corresponding main menu item or by pressing the middle mouse button. Activation through the middle mouse button is even possible whilst performing a graphical manipulation such as placing or moving an object. The View or Display menu provides useful functions for changing display options such as zoom window, zoom scale, input and/or display grids, grid and/or angle lock, color settings, etc. The View or Display menu also contains advanced display functions such as Find Part and Query Element.

Files

The Files menu provides functions for creating, loading, saving, copying, replacing and deleting DDB elements. The Files menu also allows to load and/or store color tables or to call important database management functions such as listing DDB file contents and performing library update.

Parts

The Parts provides functions for defining part placement groups, for manual part placement, for part renaming, for automatic part placement (matrix placement, initial placement) and for manual and automatic placement optimization (component swap, pin/gate swap). The function for selecting the via(s) for subsequent routing is also provided in this menu.

On part level, the Parts menu is used for placing, moving, deleting and renaming pins (i.e., padstack symbols). On padstack level, the Parts menu is used for placing, moving and deleting pads.

Traces

The Traces menu provides functions for interactive routing, i.e. for manually creating new traces and for modifying or deleting existing traces and/or trace segments.

Areas

The Areas menu is used for defining the board outline, for generating copper areas and power planes and for creating documentary lines and/or documentary areas. Existing areas can be moved, rotated, mirrored, copied and deleted. The Areas menu also provides the copper fill functions for defining copper fill workareas, for automatic copper fill area generation and/or elimination and for hatched copper area creation.

Text, Drill

The Text, Drill menu is used for creating, moving, changing and deleting texts on any layout hierarchy level. On padstack level, the Text, Drill menu provides additional functions for defining drill holes.

Groups

The Groups menu provides functions for selecting elements to group, for moving, rotating, mirroring, scaling, copying, deleting, fixing, releasing, saving and loading groups, and for replacing symbols in a group.

Parameter

The Parameter menu provides functions for selecting the layout library, setting the origin and the element boundaries of the currently loaded element, defining power layers, selecting the mincon function for the airline display, setting the spacing design rule parameters and activating the automatic design data backup feature.

Utilities

The Utilities menu provides functions for exiting BAE, returning to the BAE main shell, calling the **Autorouter** or the **CAM Processor**, starting the batch design rule check, generating net list data from current copper (Back Netlist), defining area mirror visibility and starting **User Language** programs.

4.1.4 Customized Layout Editor User Interface

Menu Assignments and Key Bindings

The BAE software comes with **User Language** programs for activating a modified **Layout Editor** user interface with many additional functions (startups, toolbars, menu assignments, key bindings, etc.). The **BAE_ST User Language** program is automatically started when entering the **Layout Editor**. **BAE_ST** calls the **UIFSETUP User Language** program which activates predefined **Layout Editor** menu assignments and key bindings. Menu assignments and key bindings can be changed by modifiying and re-compiling the **UIFSETUP** source code. The **HLPKEYS User Language** program is used to list the current key bindings. With the predefined menu assignments of **UIFSETUP** activated, **HLPKEYS** can be called from the Key Bindings function of the Help menu. Menu assignments and key bindings can be listed with the **UIFDUMP User Language** program. The **UIFRESET User Language** program can be used to reset all currently defined menu assignments and key bindings. **UIFSETUP**, **UIFDUMP** and **UIFRESET** can also be called from the menu of the **KEYPROG User Language** program which provides additional facilities for online key programming and **User Language** program help info management.

Context-sensitive Function Menus

Pressing the left mouse button in the graphic workarea activates a context-sensitive menu with specific functions for the object at the current mouse position if no other menu function is currently active. The Load Element and/or Create/New Element file management functions are provided if no element is currently loaded. This feature is implemented through an automated call to the **GED_MS User Language** program.

Cascading Windows/Motif Pulldown Menus

The Windows and Motif pulldown menu user interfaces of the **Layout Editor** provide facilities for cascading submenu definitions. I.e., submenus can be attached to other menu items. The **UIFSETUP User Language** program configures cascading submenus for the pulldown menu interfaces of the Windows/Motif **Layout Editor** modules. This allows for easy submenu function location (and activation) without having to activate (and probably cancel) submenus. The function repeat facility provided through the right mouse button supports cascading menus to simplify repeated submenu function calls.

Windows/Motif Parameter Setup Dialogs

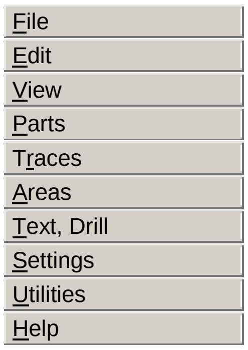
The following Windows/Motif parameter setup dialogs are implemented for the **Layout Editor**:

- Settings - Settings: General **Layout Editor** Parameters
- View - Settings: Display Parameters
- Parts - Auto Placement - Settings: Automatic Placement Parameters
- Areas - Copper Fill - Settings: Copper Fill Parameters

The **UIFSETUP User Language** program replaces the parameter setup functions of the Windows and Motif pulldown menus with the above menu functions for activating the corresponding parameter setup dialogs.

Windows/Motif Pulldown Menu Konfiguration

When using pulldown menus under Windows and Motif, the **UIFSETUP User Language** program configures the following modified **Layout Editor** main menu with a series of additional functions and features:



4.1.5 In-built Layout System Features

Automatic Parameter Backup

The **Layout Editor** provides an in-built feature for automatically saving important design and operational parameters with the currently processed SCM sheet and/or SCM library hierarchy level. The following parameters are stored to the current design file when activating the Save Element function:

- Autosave Time Interval
- Name of the currently loaded Layout Color Table
- Input Grid
- Display Grid
- Grid/Angle Lock
- Coordinate Display Mode
- Wide Line Draw Start Width
- Group Display Mode
- Part Placement Default Rotation Angle
- Part Placement Default Mirror Mode
- Default Text Size
- Part Airline Display Mode
- Standard Trace Widths
- Trace Segment Move Mode
- Library File Name
- Mincon Function Type
- Placement Matrix
- Placement Matrix Enabled Flag
- Copper Fill Isolation Distance
- Copper Fill Minimum Area Structure Size
- Copper Fill Trace Cutout Mode
- Copper Fill Island Delete Mode
- Copper Fill Heat Trap Mode
- Copper Fill Heat Trap Width
- Hatching Line Spacing
- Hatching Line Width
- Hatching Mode

Parameter sets are stored with special names according to the currently processed layout database hierarchy level. The layout element name is used for layout elements, parameter set name **`[part]`** is used for layout part symbol elements, **`[padstack]`** is used for layout padstack elements and **`[pad]`** is used for layout pad elements. When loading an element, the corresponding parameter set is automatically loaded and/or activated as well, thus providing a convenient way of activating a default parameter set suitable for processing the selected design and/or library element type.

Layer Assignments

The layer assignment is most important for the creation and modification of layout symbols and for the design of layouts. The system provides the following layers and/or display items:

- Signal Layers 1 - 100
- Signal Layer "Top Layer"
- Signal Layer "All Layers"
- Signal Layer "Middle Layer"
- Power Layers 1 - 12
- Documentary Layers 1 - 100
- Board Outline
- Airlines
- Drill Holes
- Workarea
- Origin
- Errors
- Highlight
- Drill "-", "A" - "Z"
- Fixed
- Glued

The signal layers are subject to the design rule check. On the signal layers the electrical conductivity of the PCB is defined by creating and/or placing traces, pads and passive or active copper areas. Top Layer, All Layers and Middle Layers are special signal layers.

The Top Layer can be dynamically assigned to a certain signal layer during the layout design process. On default, the Top Layer is assigned to signal layer 2. The Top Layer is most useful when generating library symbols for multilayer designs with an arbitrary number of layers. The Set Top Layer function from the Settings menu is used to assign the PCB top layer, thus defining the number of PCB layers. The mirror functions (e.g., for mirroring parts) will always consider the current top layer setting, i.e., signal layers beyond the top layer are not affected by mirror functions.

Note that the menu item text for selecting the Top Layer can be changed with the BAE setup to provide a user-specific menu text such as Component Side or Insertion Layer. See the description of the **BSETUP** utility program for more details on customizing the layout signal layer menus.

The All Layers signal layer includes all signal layers from signal layer 1 (solder side) to the currently defined top signal layer (component side). Objects defined on All Layers are considered to be placed on all signal layers and are checked and plotted accordingly. The All Layers signal layer is most useful for the definition of library symbols such as drilled pins with identical pad shapes on all signal layers.

The Middle Layers signal layer matches all signal inside layers, i.e., the Middle Layers signal layer includes all layers between signal layer 1 (solder side) and the currently defined top signal layer (component side). Objects defined on Middle Layers are considered to be placed on all signal inside layers and are checked and plotted accordingly. The Middle Layers signal layer considerably simplifies pin definitions for multilayer layouts where inside layer pad shapes differ from component and solder side pad shapes.

Up to twelve power layers can be defined with each layout, all of which being negatively displayed and plotted by the system. The Set Power Layers function from the Settings menu is used to assign power layer signal names such as **`gnd`**, **`0v`** or **`vcc`** to define the power layers of the currently loaded layout. The **Layout Editor** allows for the DRC-controlled definition of split power planes by (hierarchically) placing active copper on power layers. The **CAM Processor** automatically generates isolations and heat traps when plotting power layers.

The documentary layers are used to store and/or include documentary information (graphic, text) and special keepout definitions with the layout and/or the layout library symbols. The **BSETUP** utility program can be used to define up to 100 different (or 12 in **BAE Professional** simultaneously accessible) documentary layers with individual names and characteristics according to special requirements. The system allows for the definition of customer-specific documentary layers for, e.g., silk screen, insertion data, solder resist, glue spots, drill plan and drill overlap control, plot markers, part spacing check, comments, measuring, etc. Each documentary layer consists of the following sides to support double-sided SMD design:

Side 1
Side 2
Both Sides

Each documentary layer side is a sub-layer of the corresponding documentary layer. The Side 1 and Side 2 sub-layers are mutually interchanged when performing mirroring. The **CAM Processor** can optionally plot the Both Sides sub-layer together with Side 1 and/or Side 2. Documentary layer definitions have fundamental influence on what can be defined in the layout and which CAM output can be produced. Documentary layer definition changes introduced by **BSETUP** only apply to subsequently defined library symbols to enable data exchange between different configurations. It is strongly recommended to become familiar with the features for defining documentary layers (and to define documentary layers according to your specific requirements) before starting with the design of real layouts or layout symbols for productive use. See chapter 7.2 of this manual for a description of the **BSETUP** utility program and for the default documentary layer definitions provided with initial BAE software installations.

The Board Outline layer is used for defining and/or displaying the board outline of the currently loaded layout. The Airlines layer is used for displaying unroutes on the currently loaded layout. The Drill Holes layer is used for displaying drill holes on the currently loaded layout, part or padstack. Workarea, Origin, Errors and Highlight are layers for displaying special layout user interface elements. The Drill layers are used for assigning colors to drill classes. The Fixed and Glued layers are used for the selection of patterns for the display of fixed and/or glued layout elements.

Pick Preference Layer and Element Selection

The pick preference layer is used to resolve ambiguities when selecting objects which are placed at the same position but on different layers (e.g., traces, areas, SMD parts, etc.). The pick functions use the pick preference layer to designate the element to be selected, if more than one object of interest is placed at the pick position. The pick preference layer can be selected with the Set Edit Layer function from the View menu. On default, the pick preference layer is set to signal layer 1 (solder side).

The Settings dialog from the View menu provides the Pick Mode parameter for selecting the element pick method for multiple elements at the same pick position. The Preference Layer default option picks an element from the currently selected preference layer. The Element Selection option provides an element selection facility if more than one element is found at the pick position. A loop for highlighting the selectable elements with status line short description is activated. The highlighted element can be selected through return key or left mouse button click. This element selection can be aborted through escape key or right mouse button input. Any other key switches to the next element at the pick position.

Net List

A net list is usually required for the layout design. In BAE, the net list is commonly created with the Schematic Editor and will then be transferred to the layout using the **Packager**. Alternatively, ASCII net lists can be imported to the BAE layout system using utility programs such as **CONCONV** or **REDASC**. See chapter 3.4 of this manual for more details on net list processing.

The function for loading a layout will not only load the pertinent data from the lower hierarchy levels (parts, padstacks and pads), but also the corresponding net list. The net list data is correlated with all geometrical data on the layout ("Connectivity Generation"). Please note that the element names for both the layout and the net list must be identical for this to work.

After successfully generating the connectivity, the system is capable of instantly controlling and/or correlating each layout modification with the net list definitions. This high-sophisticated layout design feature is called "Full Copper Sharing" or "True Connectivity". Full copper sharing enables real-time recognition of electrical connections, no matter whether connections are created with traces, copper areas or vias. I.e., the system supports advanced routing features such as cross connection recognition, genuine T-connections, lining up traces created with group copying, implementing routes with arbitrarily shaped copper areas, etc.

The **BAE HighEnd** layout connectivity evaluation system uses special data structures for storing additional information about the structure of electric connections between copper elements. This requires more memory compared to **BAE Professional**, but results in dramatically shortened response times when manually editing large nets or moving parts since only the elements in the neighborhood of the currently processed object have to be considered by the DRC.

Mincon Function

Not yet routed connections (unroutes) are displayed as airlines. During part placement, the Mincon function performs dynamic (real-time) recalculation of these airlines in order to display minimum length airline sets. This makes the Mincon function an indispensable utility for achieving an optimum placement. The Mincon Function menu item from the Settings menu is used to set the type of airline display, i.e., to choose the method of calculating unroutes from pin-to-pin or corner-to-corner minimization where also the unroute components to be calculated can be selected (horizontal only, vertical only, horizontal + vertical sum, etc.).

BAE HighEnd utilizes modified data structures for internal connectivity representation and airline calculations. The **BAE HighEnd** Mincon functions run at much higher performance compared to **BAE Professional**, however, at the cost of higher memory requirements.

Design Rule Check

The BAE design rule check utilizes full copper sharing features to perform real-time recognition of open connections, short-circuits and distance violations. Open connections (unroutes) are displayed as airlines, short-circuits are indicated with highlight, and distance violations are enclosed by a rectangle. The corresponding colors for displaying Airlines, Highlight and Errors can be selected with the Change Colors function from the View menu.

Only modified items are checked real-time by the incremental online design rule check, thus saving a lot of computing time. As board parameters could have changed until post-processing, it is recommended to use the Batch DRC function from the Utilities menu for a final batch design rule check with the current layout parameter settings before releasing the board. The incremental connectivity, however, does not have parameters and thus is always correct. The Report function from the Utilities menu should be used after Batch DRC to check all error counts.

The spacing parameters for the distance checks can either be set from the Settings menu (functions Distance TR/TR, Distance TR/CO and Distance CO/CO) or can be defined as net-specific parameters in the net list using the `MINDIST` net attribute). Note, however, that these parameter settings need not be the same as for the autorouting since the **Autorouter** can automatically set minimum distance parameters according to the routing grid selection. Net-specific spacing parameters are individually considered by the **Autorouter**.

Groups

The **Layout Editor** group functions provide powerful features for design data manipulation. An arbitrary set of items of the currently loaded layout or part can be selected to a group which then can then be saved, moved, rotated, mirrored, scaled, copied, deleted, fixed or unfixed.

The group functions are featuring set principles. Elements can be added (selected) to or removed (deselected) from the currently defined group. Highlight display is used to indicate group-selected items. The Group Polygon function is used to select and/or deselect parts, traces, areas, texts, visible and/or invisible elements or elements of any type by defining an area around the items to be selected. The Group Elements function is used to select and/or deselect single elements of a certain type (part, trace, area or text) by picking the desired items. The Group Elements function is active as long as valid pick elements are selected, thus allowing for the selection of a series of objects of a certain type). The Reset Group function is used to deselect *all* items from the current group.

All group-selected elements are subject to subsequent group functions such as Save Group, Move Group, Copy Group and Delete Group. The Fix Group and Unfix Group functions only work on group-selected parts, traces and vias.

The number of modified elements is displayed with the feedback messages of Reset Group, Group Polygon, Delete Group, Fix Group and Unfix Group functions. This information can be used to check whether the group function was applied as intended.

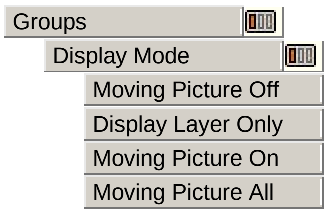
The Save Group function is used to save the currently defined group. The Save Group function prompts for a group origin selection which becomes the origin of the new element> The group origin is also used as the reference point for group load commands. Saved groups are stored as an element of the same type as that from which the group was selected. To prevent from unintentionally overwriting existing database elements, Save Group prompts for confirmation if the specified element already exists in the destination file. The Load Group function can be used to reload previously saved groups (as well as layout and/or part elements) to different layouts and/or parts. The save and load group facilities can be used for a variety of tasks such as replicating tracking, saving and loading PCB templates, stealing from existing and proven designs, etc.

The Copy Group and Load Group functions do not perform automatic part and/or pin renaming when copying and/or loading groups including parts (on layout level) or padstacks (on part level). Naming conflicts are resolved by assigning # names to parts and/or pins, and manual renaming is required to correct the name list (or some special **User Language** program must be implemented for solving such name list problems automatically).

Pressing the right mouse button during group movement operations such as Move Group, Copy Group or Load Group activates a submenu with functions for placing the group to relative or absolute coordinates (Jump Relative, Jump Absolute), rotating the group (Rotate Left, Rotate Right, Set Angle), mirroring the group (around the X-axis using either Mirror Off and/or Mirror On) or even scaling the group element dimensions and placement coordinates (Scale, applied after placing the currently processed group). The Move Group submenu provides the Set Quadrant option for automatically rerouting traces between the moved group and the rest of the layout after selecting the start point for the group move operation. Set Quadrant first expects an interactive selection of a quadrant origin and then prompts for the quadrant (Upper Right, Upper Left, Lower Left, Lower Right). Subsequent group move operations are only applied on group-selected elements and/or points placed within the selected quadrant. It is possible to move parts of a layout and to reroute the connections to the moved group. Note, however, that the algorithm applied by the group move functions for rearranging traces is not a genuine routing algorithm and is designed to be applied on horizontal and/or vertical move operations only. For more complex group shift operations it is recommended to discard the routing and perform a re-routing using the appropriate **Autorouter** procedures.

The Load Group function first resets the currently defined group to deselect all group elements before performing the load operation. After successfully loading a group, all loaded group elements are automatically group-selected. I.e., only loaded group elements are subject to subsequent group functions.

The **Layout Editor** supports different options for displaying group-selected elements during group movement operations. The behavior of this feature can be controlled with the Display Mode function which provides the following display modes:



Moving Picture Off deactivates the group movement display. Display Layer Only displays upper level group-selected elements which are assigned to the group display layer; the group display layer is a BAE setup parameter defined with the **LAYGRPDISPLAY** command of the **BSETUP** utility program (see chapter 7.2 for a description of **BSETUP**). Moving Picture On displays all group-selected elements except for traces, vias and drill holes during group movement. Moving Picture All displays all group-selected elements including traces, vias and drill holes during group movement. Moving Picture On is the default setting.

The Group Macro function is used to replace group-selected part macros (on layout level) or padstack macros (on part level). This function is most useful for fast pin type replacement (technology change) on part level. On layout level, the Group Macro function supports alternate part package type assignments for net list parts which include the specified part macro in their alternate part package type list. Package types for non-netlist parts, however, are changed without this check.

Fixing/Releasing Elements

Certain layout elements such as parts or traces can be fixed. Fixed objects are not changed by subsequent automatic processes such as placement optimization, automatic pin/gate swap or autorouting. It is strongly recommended to fix critical elements such as parts (e.g., plugs, switches, LEDs, etc.) which are to be excluded from the placement optimization or manually pre-routed traces (e.g., for power supply) which must not be changed by the **Autorouter**. A specific highlight for indicating fixed elements is activated during fix and release operations.

All functions for manipulating and/or copying fixed elements (parts, traces, vias) preserve and/or copy existing fixed modes. This also means that the **Layout Editor** trace functions for interactive routing keep traces fixed flags when processing pre-routed fixed traces. I.e., when manipulating pre-routed fixed traces, there is no need of re-fixing such traces to prevent the **Autorouter** from re-routing and/or rejecting pre-routed traces.

Polygons, Areas

In the **Layout Editor**, the functions from the Areas menu are used to create and/or modify polygons and/or pad shapes. Each polygon definition is associated with a layer and/or a polygon type and can contain an arbitrary number of arcs. The definition of an arc is applied by selecting either the Arc Left or the Arc Right option from the submenu to be activated after specifying the arc start point; subsequently, the system expects the arc center point specification, and the arc definition is completed with the arc end point selection. The system provides instant display of the resulting circle and/or arc during interactive input of the arc center and end points. At the definition of full circles, the input of the arc end point is superfluous, i.e., the full circle definition can be completed with the Done submenu function to be activated after setting the arc center point.

The algorithm for picking polygon corners and polygon segments picks the polygon element with minimum distance from the pick point if more than one possible pick element is within snapping distance. This allows for reliable selection of polygon elements even when working in (small) zoom overview display modes.

Project and Version Control Attributes

The `$pltfname` (element file path name), `$pltfsname` (element file name) and `$pltename` (element name) texts can be placed to display the project file path name, the project file name (without directory path) and the element name of the currently loaded element.

The `$plttime` (current time), `$pltdatede` (current date, German notation) and `$pltdateus` (current date, US notation) system attributes are substituted with the current time and/or date when displayed and/or plotted on layout level.

The `$pltstime` (save time), `$pltsdatede` (save date, German notation) and `$pltsdateus` (save date, US notation) system attributes are substituted with the time and date at which the currently loaded layout was last saved.

Automatic time and date substitutions are applied throughout all database hierarchy levels (pad, padstack, part, layout), unless other attribute values are explicitly set for these attributes.

Measuring

The **Layout Editor** automatically activates a measuring function when placing new text with the hash character as text string (`#`). This measuring function determines the distance between the two input coordinates subsequently to be specified. The resulting text string is built from the corresponding distance value with length units (inch or mm) retrieved from the current coordinate display mode as selected with the Coordinate Display function from the Settings menu.

Considering CAM Output Restrictions

The **Layout Editor** supports the definition of arbitrarily shaped areas and pads which can also be rotated at any angle. All these features are supported by the intelligent **CAM Processor**, which utilizes a series of automatic fill algorithms with multiple aperture selection and area reduction. The **CAM Processor** automatically optimizes the use of the selected tools and aperture table definitions in order to generate all structures at highest possible precision with regard to the chosen plot parameters and output tolerance ranges. The system also provides appropriate error messages and highlight to indicate objects which cannot be plotted using the selected tools and the current plot parameter settings.

As mentioned above, the **CAM Processor** is able to construct arbitrary polygons. When defining pads, consideration should, however, be given to the way in which Gerber photo plot data is generated. A flash code containing only one set of coordinates is used for areas that match apertures such as circles and squares. For areas that don't match apertures efficient drawing techniques generating few coordinates are used where possible (e.g., with rectangles), but with some shapes and sizes much more coordinates are needed. With careful design of pads and appropriate aperture tables and plot parameter settings, much smaller Gerber data files can be generated, making them quicker to copy, cheaper to send by modem, less expensive to plot and require less media space, thus considerably reducing the costs incurred by the CAM and manufacturing process.

User Language

The **Bartels User Language Interpreter** is integrated to the **Layout Editor**, i.e., **User Language** programs can be called from the **Layout Editor**, and it is possible to implement any user-specific **Layout Editor** function required such as status display, parameter setup, reports and test functions, CAD/CAM input/output functions, symbol library management utilities, automatic or semi-automatic placement and/or routing functions, customer-specific batch procedures, etc.

The **Layout Editor** provides both explicit and implicit **User Language** program call facilities. **User Language** programs can be started with explicit program name specification using the Run User Script function from the File menu (empty string or question-mark (**?**) input to the program name query activates a **User Language** program selection menu).

User Language programs can also be called by simply pressing special keys of the keyboard. This method of implicit **User Language** program call is supported at any time unless another interactive keyboard input request is currently pending. The name of the **User Language** program to be called is automatically derived from the pressed key, i.e. pressing a standard and/or function key triggers the activation of a **User Language** program with a corresponding name such as **ged_1** for digit key 1, **ged_r** for standard key r, **ged_#** for standard key #, **ged_f1** for function key F1, **ged_f2** for function key F2, etc.

The **Layout Editor User Language Interpreter** environment also features event-driven **User Language** program calls, where **User Language** programs with predefined names are automatically started at certain events and/or operations such as **GED_ST** at **Layout Editor** module startup, **GED_LOAD** after loading a design element, **GED_SAVE** before saving a design element, **GED_TOOL** when selecting a toolbar item and **GED_ZOOM** when changing the zoom factor. The module startup **User Language** program call method is most useful for automatic system parameter setup as well as for key programming and menu assignments. The element save and load program call methods can be used to save and restore element-specific parameters such as the zoom area, color setup, etc. The toolbar selection event must be used to start **User Language** programs which are linked to toolbar elements. The zoom event can be used to apply an update request to a design view management feature.

Bartels User Language also provides system functions for performing key programming, changing menu assignments and defining toolbars. These powerful features can be applied for user interface modifications. Please note that a large number of additional functions included with the **Layout Editor** menu are implemented through the **User Language** programs delivered with the BAE software.

See the Bartels User Language Programmer's Guide for a detailed description of the **Bartels User Language** (chapter 4.2 lists all **User Language** programs provided with the BAE software).

Neural Rule System

A series of advanced **Bartels AutoEngineer** features are implemented through the integrated **Neural Rule System**. See chapter 6.3.2 for the rule system applications provided with the PCB layout system.

4.2 Layout Library Symbol Design

The **Bartels AutoEngineer** is shipped with an extensive layout library. Nevertheless, you might require a certain layout symbol which has not yet been defined in these libraries. This section shows in detail how to create layout library symbols. The example symbols are created starting with the lowest DDB hierarchy level. I.e., first of all some pad and padstack symbols are defined, and, subsequently, some layout part symbols are defined. All these symbols will be stored to a DDB file named `demo.ddb`. Use the following commands to move to the BAE examples directory (e.g., `c:\baejobs`) and start the **Bartels AutoEngineer**:

```
>  C: ↵
>  cd c:\baejobs ↵
>  bae ↵
```

The BAE main menu is activated, and you can start the **Layout Editor** with the following command:

Layout

The **Layout Editor** is activated, and you can create layout library elements. You should become familiar with the conventions used for layout symbol design before generating your own symbols. Technology-dependent and manufacturing-specific conventions are usually to be considered with regard to pad shape definitions, part insertion pick points, identification of part pin `1`, pin placement grids, part placement origin, minimum text sizes, drill symbols, part spacing parameters for SMDs, etc. The layer assignment is most important for the creation and modification of layout symbols and for the design of layouts in order to provide correct and complete CAM data output later (see chapter 4.1.5 for more details).

Figure 4-1 shows the layout library symbols to be created in the following sections.

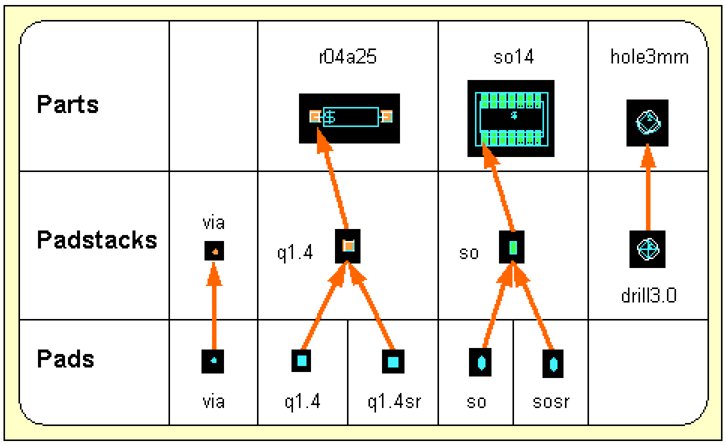


Figure 4-1: Layout Library Symbols

4.2.1 Creating Layout Pads

On layout pad level the pad shapes (i.e., the pin contact areas) are defined by creating passive copper areas. Different pad symbols can be assigned to different layers on a single padstack symbol thus defining a particular layout pin type.

It is recommended to refrain from assigning layers to the copper areas created on pad level, i.e., layer assignment should be disabled on pad level (this is the default setting introduced by **BSETUP**). The layer assignment can later be applied on padstack level to minimize the expenditure for adapting layout libraries to different manufacturing technologies.

Creating a Pad Symbol

Use the following commands to create a new pad symbol named **`via`** with an element size of 1*1mm in DDB file **`demo.ddb`**:

```
File
  New
    Pad
                         File Name ?  demo
                      Element Name ?  via
               Element Width (mm/") ?  1
              Element Height (mm/") ?  1
```

The display now shows a square frame with a cross in the middle. The frame describes the element boundaries of the pad, and the cross marks the position of the element origin.

Defining a Copper Area

Use the following commands to define a circle-shaped copper area with a diameter of 0.9mm on the currently loaded pad element:

```
Areas
  Add Passive Copper
    
      Jump Absolute
            Absolute X Coordinate (mm/") ?  0.45
            Absolute Y Coordinate (mm/") ?  0
    
      Arc Left
    
      Jump Absolute
            Absolute X Coordinate (mm/") ?  0
            Absolute Y Coordinate (mm/") ?  0
    
      Done
```

Setting the Element Boundaries

The pad symbol's element boundaries should be reduced to enclose the pad definition as densely as possible. This is accomplished with the following commands:

- Settings
  - Upper/Right Border
    - Jump Absolute
      - Absolute X Coordinate (mm/") ? 0 ↵
      - Absolute Y Coordinate (mm/") ? 0 ↵
  - Lower/Left Border
    - Jump Absolute
      - Absolute X Coordinate (mm/") ? 0 ↵
      - Absolute Y Coordinate (mm/") ? 0 ↵

Saving the Element

You can also place documentary information such as text, documentary lines or documentary areas on pad level. However it is recommended to refrain from placing documentary items on pad level since this would result in a very specific pad definition which could be used in quite few superior layout symbols. Use the following commands to save the pad symbol:

- File
  - Save Element

Now the new pad symbol named **`via`** is completely defined and stored to the DDB file **`demo.ddb`**. This pad will later be loaded to a padstack thus defining a via for manual and automatic routing.

Defining Square Pads

Use the following commands to define a pad named `q1.4` with a square-shaped copper area (edge length 1.4mm) in DDB file `demo.ddb`:

```
File 🖱
  New 🖱
    Pad 🖱
                                  File Name ? | demo ↵
                               Element Name ? | q1.4 ↵
                        Element Width (mm/") ? | 2 ↵
                       Element Height (mm/") ? | 2 ↵
Areas 🖱
  Add Passive Copper 🖱
    🖱
      Jump Absolute 🖱
                   Absolute X Coordinate (mm/") ? | 0.7 ↵
                   Absolute Y Coordinate (mm/") ? | 0.7 ↵
    🖱
      Jump Relative 🖱
                   Relative X Coordinate (mm/") ? | -1.4 ↵
                   Relative Y Coordinate (mm/") ? | 0 ↵
    🖱
      Jump Relative 🖱
                   Relative X Coordinate (mm/") ? | 0 ↵
                   Relative Y Coordinate (mm/") ? | -1.4 ↵
    🖱
      Jump Relative 🖱
                   Relative X Coordinate (mm/") ? | 1.4 ↵
                   Relative Y Coordinate (mm/") ? | 0 ↵
    🖱
      Done 🖱
File 🖱
  Save Element 🖱
```

With the commands above, a pad named `q1.4` has been defined which will be used on a padstack for parts with plated-through pins. This pin definition will require a corresponding pad symbol for the solder resist mask, which should be same as `q1.4` except for a slightly expanded copper area. Use the following commands to copy the (still loaded) pad named `q1.4` to a pad symbol named `q1.4sr` (square 1.4mm, solder resist), then load pad `q1.4sr` and enlarge the copper area of `q1.4sr` by 0.1mm:

```
File 🖱
  Save Element As 🖱
                             File Name ? | ↵
                          Element Name ? | q1.4sr ↵
  Load Element 🖱
    Pad 🖱
                                     File Name ? | ↵
                                  Element Name ? | q1.4sr ↵
Areas 🖱
  Resize Area 🖱
                  Move to Area Corner/Edge | 🖱
          Expansion Distance (mm/") ? | 0.1 ↵
File 🖱
  Save Element 🖱
```

Defining Finger Pads

Use the following commands to define a pad named **so** with a finger-shaped copper area (width 0.7mm, length 1.7mm) in the **demo.ddb** DDB file:

```
File
    New
        Pad
            File Name ?                        demo ↵
            Element Name ?                     so ↵
            Element Width (mm/") ?             1 ↵
            Element Height (mm/") ?            2 ↵
Areas
    Add Passive Copper
        Jump Absolute
            Absolute X Coordinate (mm/") ?     0.35 ↵
            Absolute Y Coordinate (mm/") ?     0.5 ↵
        Arc Left
        Jump Relative
            Relative X Coordinate (mm/") ?     -0.35 ↵
            Relative Y Coordinate (mm/") ?     0 ↵
        Jump Relative
            Relative X Coordinate (mm/") ?     -0.35 ↵
            Relative Y Coordinate (mm/") ?     0 ↵
        Jump Relative
            Relative X Coordinate (mm/") ?     0 ↵
            Relative Y Coordinate (mm/") ?     -1 ↵
        Arc Left
        Jump Relative
            Relative X Coordinate (mm/") ?     0.35 ↵
            Relative Y Coordinate (mm/") ?     0 ↵
        Jump Relative
            Relative X Coordinate (mm/") ?     0.35 ↵
            Relative Y Coordinate (mm/") ?     0 ↵
        Done
File
    Save Element
```

With the commands above, a pad named **so** has been defined which will be used to define a standard SMD pin for SO (small outline) package types. Use the following commands to copy the (still loaded) pad named **so** to a pad symbol named **sosr** (small outline pad, solder resist), load the **sosr** pad and enlarge the copper area of **sosr** by 0.05mm:

```
File
    Save Element As
        File Name ? ↵
        Element Name ? sosr ↵
    Load Element
        Pad
            File Name ? ↵
            Element Name ? sosr ↵
Areas
    Resize Area
        Move to Area Corner/Edge
        Expansion Distance (mm/") ? 0.05 ↵
File
    Save Element
```

Checking the File Contents

Within this section we have created a series of pad symbols which we stored to DDB file **demo.ddb**. Use the following commands to list the pad(s) defined in the **demo.ddb** DDB file:

```
File
    File Contents
        Pad
            File Name ? ↵
```

An empty string input (i.e., pressing the return key ↵) to the file name prompt causes the system to use the file name of the currently loaded element, which in our example is **demo.ddb**. The system now produces the following listing of the pads contained in **demo.ddb**:

```
Type : Pad / File : demo.ddb


: q1.4           : q1.4sr          : so               : sosr
: via            - End -
```

Hit the spacebar to continue.

4.2.2 Creating Layout Padstacks

On layout padstack level the layout pin symbols and vias are defined by placing symbols from the subordinate pad level. Each pad can be assigned to a signal and/or documentary layer thus designating contact areas for the routing or defining pad shapes for solder resist, SMD masks, etc. A drill hole and drill plan info can be created optionally for the definition of vias or drilled pins. Keepout areas can be utilized for controlling the pin contact mode. Documentary lines or areas can serve as pin designators on the silk screen or insertion plan, and reference texts can be used for displaying pin names on part and/or layout level.

In this section the pad symbols defined in the previous section are used to generate some padstack symbols. To provide visual input control it is recommended to use the following commands to change the color setup in order to display the Drill Holes as well as the Drill Plan (Both Sides) and the Solder Mask (all sides) documentary layers during padstack definition:

```
Change Colors
    Drill Holes
        Move to Desired Color, white
    -> Doc.-Layer
    Drill Plan/Both Sides
        Move to Desired Color, light blue
    Solder Mask/Both Sides
        Move to Desired Color, gray
    Solder Mask/Side 1
        Move to Desired Color, dark gray
    Solder Mask/Side 2
        Move to Desired Color, light gray
    Exit
```

Changing some item-specific color is accomplished by selecting the desired display item using the left mouse button and then selecting the desired color button from the Change Colors function from the View menu. In the layout system, the Change Colors menu provides a feature for fast display item fade-out/fade-in. Activating and/or deactivating some item-specific display is accomplished by selecting the desired display item entry with the right mouse button which works as a toggle between fade-out and fade-in. The system won't loose information on currently defined colors of faded-out display items; strike-through color buttons are used for notifying currently faded-out display items.

Use the following commands to store the currently defined color table with name **`stackedit`** to the **`ged.dat`** system file (in the BAE programs directory):

```
View
    Save Colors
        Element Name ?  stackedit ↵
```

Once a color table has been saved it can be reloaded at any time using the Load Colors function from the View menu. Special color tables (e.g., for library edit, for finding unroutes, etc.) can be defined and reloaded on request. The default color table to be loaded after startup is the one named **`standard`**.

Creating a Padstack Symbol

Use the following commands to create a padstack symbol named **`via`** with an element size of 1*1mm in DDB file **`demo.ddb`**:

```
File
    New
        Padstack
                                  File Name ? demo ↵
                               Element Name ? via ↵
                        Element Width (mm/") ? 1 ↵
                       Element Height (mm/") ? 1 ↵
```

The display now shows a square frame with a cross in the middle. The frame describes the element boundaries of the padstack, and the cross marks the position of the element origin.

Loading Pads

The Add Part function from the Parts menu is used for placing pads onto the current padstack symbol. The system prompts for the library element name, i.e., the name of the pad symbol to be loaded. Popup menus are provided with the library element name query for selecting the library file and the pad symbol name. The library file name list is derived from the layout library path defined with the BAE setup, i.e., all DDB files available in the directory of the layout library path are listed. Pad symbols can optionally be selected by typing both the library file name (i.e., one of the names displayed with the library file name popup), a slash (**`/`**), and the pad symbol name to the library element name prompt (typing **`?`** for the element name will activate a popup menu providing the list of pad symbols from the specified library file). An empty string input to the library element name query causes the system to use the pad symbol previously selected with the Add Part function (if there was already one specified).

Use the following commands to load the pad symbol **`via`**, place it at the padstack origin and assign it to the All Layers signal layer:

```
Parts
    Add Part
                     Library Element Name ? via ↵
        Select Input Layer
            All Layers
        Done
```

Defining the Drill Hole

Use the following commands to define a drill hole with a diameter of 0.5mm:

```
Text, Drill
    Place Drill Hole
                          Drill size ( 0.00mm) ? 0.5 ↵
        Done
```

The Drilling Class option from the Place Drill Hole function allows for the assignment of a non-default drilling class to support and/or process blind and buried via definitions (see also chapter 4.6.11). The Mirr. Drill. Class option allows for the assignment of a mirror mode drill class definition. The mirror mode drill class is activated when the part on which the drill hole is defined is mirrored, thus supporting mirroring of parts with blind and buried pins. On padstack level, the mirror mode drill class indicator is displayed below the standard drill class. On layout level, only the currently active drill class is displayed.

Defining a Drill Symbol

Use the following commands to define a drill symbol by creating a documentary line on the Drill Plan (Both Sides) documentary layer (it is recommended to select the inch coordinate display mode and to set the input grid to 1/80 inch; during polygon definition the polygon corner point coordinates can be taken from the info field displayed on the right top of the user interface):

```
Settings
    Coordinate Display
        Display Inch

    Grids/Rotation
        Set Input Grid
            1/80 Inch
            Grid+Rotation On
Areas
    Add Document Line
        Drill Plan
                            Move to [0.0125",0.0125"]
                           Move to [-0.0125",0.0125"]
                           Move to [0.0125",-0.0125"]
                          Move to [-0.0125",-0.0125"]
                            Move to [0.0125",0.0125"]

            Done
```

Defining a Keepout Area

Use the following commands to create a circle-shaped keepout area with a diameter of 0.9mm on the Drill Plan (Both Sides) documentary layer:

```
Areas
    Add Keep Out Area
        Document Layer
            Drill Plan

            Jump Absolute
                Absolute X Coordinate (mm/") ? 0.45
                Absolute Y Coordinate (mm/") ? 0

            Arc Left
                                  Move to Origin [0,0]

            Done
```

The Design Rule Check will mutually check keepout areas on documentary layers. I.e., the keepout area definition from the example above will cause the DRC to indicate distance errors at via drill hole overlaps inadvertently introduced by manual routing.

Saving the Element

The definition of the `via` padstack symbol is completed now. Do not forget to *save* this *symbol* with the following commands:

```
File
    Save Element
```

Creating a Padstack Symbol for plated-through Pins

Use the following commands to create a padstack symbol named **`q1.4`** with an element size of 2*2mm in DDB file **`demo.ddb`**:

```
File
    New
        Padstack
                                  File Name ? demo ↵
                               Element Name ? q1.4 ↵
                        Element Width (mm/") ? 2 ↵
                       Element Height (mm/") ? 2 ↵
```

Use the following commands to place the pad symbols **`q1.4`** (on All Layers signal layer) and **`q1.4sr`** (on Solder Mask / Both Sides documentary layer):

```
Parts
    Add Part
                    Library Element Name ? q1.4 ↵
        Select Input Layer
            All Layers
        Done
    Add Part
                    Library Element Name ? q1.4sr ↵
        Select Input Layer
            Document Layer
                Solder Mask
                    Both Sides
        Done
```

Use the following commands to create a documentary line on the Insertion Plan (Side 2) documentary layer for indicating the pin outline:

```
Areas
    Add Document Line
        Insertion Plan
                 Move to [0.025",0.025"]
                Move to [-0.025",0.025"]
               Move to [-0.025",-0.025"]
                Move to [0.025",-0.025"]
                 Move to [0.025",0.025"]

            Done
```

Use the following commands to define a drill hole with a diameter of 0.9mm:

```
Text, Drill
    Place Drill Hole
                  Drill size ( 0.00mm) ? 0.9 ↵
        Done
```

Use the following commands to define a drill symbol by creating a documentary line on the Drill Plan (Both Sides) documentary layer:

```
Areas
    Add Document Line
        Drill Plan
                            Move to [0.0125",0.0125"]
                           Move to [-0.0125",0.0125"]
                          Move to [-0.0125",-0.0125"]
                           Move to [0.0125",-0.0125"]
                            Move to [0.0125",0.0125"]

            Done
```

Use the following commands to set the input grid to 1/200 inch, and define four small square-shaped keepout areas (on signal layer All Layers) at the corners of the pin:

```

    Grids/Rotation
        Set Input Grid
        1/200 Inch
        Grid+Rotation On
Areas
    Add Keep Out Area
        All Layers
                            Move to [0.0250",0.0250"]
                            Move to [0.0300",0.0250"]
                            Move to [0.0300",0.0300"]
                            Move to [0.0250",0.0300"]

            Done
    Copy Area
                            Move to [0.0250",0.0250"]
                           Move to [-0.0300",0.0250"]
    Copy Area
                           Move to [-0.0300",0.0250"]
                          Move to [-0.0300",-0.0300"]
    Copy Area
                          Move to [-0.0300",-0.0300"]
                           Move to [0.0250",-0.0300"]
```

The keepout areas created with the commands above define a special pin contact mode for padstack `q1.4` to allow only orthogonal connections. This restriction is considered by the **Autorouter**.

The definition of padstack symbol `q1.4` is completed now. Do not forget to *save* this *symbol* with the following commands:

```
File
    Save Element
```

Creating a Padstack Symbol for SMD Pins

Use the following commands to create a padstack symbol named **so** with an element size of 1*2mm in DDB file **demo.ddb**:

```
File
  New
    Padstack
                     File Name ?  demo
                  Element Name ?  so
         Element Width (mm/") ?  1
        Element Height (mm/") ?  2
```

Use the following commands to place the pad symbols **so** (on component side, Layer n (Parts.)) and **sosr** (on Solder Mask, Side 2 documentary layer):

```
Parts
  Add Part
            Library Element Name ?  so
    Select Input Layer
      Layer n (Parts.)
    Done
  Add Part
            Library Element Name ?  sosr
    Select Input Layer
      Document Layer
        Solder Mask
          Side 2
    Done
```

Use the following commands to create a documentary line on the Insertion Plan (Side 2) documentary layer for indicating the pin outline:

```
Areas
  Add Document Line
    Insertion Plan
                 Move to [0.0150",0.0350"]
                Move to [-0.0150",0.0350"]
               Move to [-0.0150",-0.0350"]
                Move to [0.0150",-0.0350"]
                 Move to [0.0150",0.0350"]

      Done
```

The definition of padstack symbol **so** is completed now. Do not forget to *save* this *symbol* with the following commands:

```
File
  Save Element
```

Creating a Padstack Drill Symbol

Use the following commands to create a padstack symbol named **`drill3.0`** with an element size of 3*3mm in DDB file **`demo.ddb`**:

- File
  - New
    - Padstack

File Name ?	demo ↵
Element Name ?	drill3.0 ↵
Element Width (mm/") ?	3 ↵
Element Height (mm/") ?	3 ↵

Use the following commands to create a drill hole with a diameter of 3.0mm and assign that drill hole to drilling class **`Z`** (this drilling class can be utilized in the **CAM Processor** to perform selective output of non-plated drills; see chapter 4.7.13 for details on producing drill data output):

- Text, Drill
  - Place Drill Hole
    - Drill size (0.00mm) ? 3.0 ↵
    - Drilling Class
      - New Drilling Class (-,A..Z) (-) ? Z ↵
    - Done

A dash string (**`-`**) input to the drilling class prompt can be used to refrain from any drilling class assignment (this is the default for drill hole definitions).

Use the following commands to define a drill symbol by creating a documentary line on the Drill Plan (Both Sides) documentary layer:

- Areas
  - Add Document Line
    - Drill Plan
    - Move to [0.00",0.05"]
    - Move to [-0.05",0.00"]
    - Move to [0.00",-0.05"]
    - Move to [0.05",0.00"]
    - Move to [0.00",0.05]
      - Done

Use the following commands to define a keepout area on the All Layers signal layer with size and position corresponding with the drill hole definition:

- Areas
  - Add Keep Out Area
    - All Layers
    - Jump Absolute
      - Absolute X Coordinate (mm/") ? 1.5
      - Absolute Y Coordinate (mm/") ? 0
    - Arc Left
    - Jump Absolute
      - Absolute X Coordinate (mm/") ? 0
      - Absolute Y Coordinate (mm/") ? 0
    - Done

The keepout area defined with the commands above will prevent the **Autorouter** from routing over the drill hole.

The definition of padstack symbol **`drill3.0`** is completed now. Do not forget to *save* this *symbol* with the following commands:

- File
  - Save Element

4.2.3 Creating Layout Parts

On layout part level the layout part symbols (i.e., the part package types) are defined (and stored to a layout part library). A particular layout part symbol is usually defined by placing elements from the subordinate padstack level in order to define the types and positions of the physical pins of the corresponding part. Keepout areas (for performing clearance checks on placed parts, defining via keepout areas, etc.), copper areas, drawing information (component outline on insertion plan) and text (for part name reference, insertion data pick point, attribute value display, etc.) can be created optionally.

The BAE layout system also allows for placing traces and vias on part hierarchy level. The layout level design rule check is deactivated between different traces on part level to allow for the correct representation of special devices such as printed inductors. I.e., pin connections on printed inductors can be created by introducing two connecting traces. Short-circuits and distance violations to other traces on layout level will still be recognized by the design rule check. Since the layout level design rule check will not perform checks between traces defined on the same part, it is strongly recommended to apply part level design rule checks at the definition of such parts with the clearance parameters set to the smallest minimum distance(s) intended for the use of such parts on layout level.

In this section, a resistor part with drilled pins, a 14-pin SMD part and a constructive part symbol representing a drill hole are created. The padstack symbols from the preceding section are used for pin definitions.

For the following operations it is recommended to set the input grid to 1/20 inch (use the Grids/Rotation and Set Input Grid functions from the View menu), and to set the coordinate display mode to inch (use function Coordinate Display from the Settings menu).

Creating a Part Symbol

Use the following commands to create a new part symbol named **`r04a25`** with an element size of 0.6*0.2 inch in DDB file **`demo.ddb`**:

- File
  - New
    - Part

File Name ?	demo ↵
Element Name ?	r04a25 ↵
Element Width (mm/") ?	0.6" ↵
Element Height (mm/") ?	0.2" ↵

The display now shows a rectangle-shaped frame with a cross at the lower left corner. The frame describes the element boundaries of the part, and the cross marks the position of the element origin.

Placing the Pins

The Add Part function from the Parts menu is used for placing pins on the current layout part symbol. The system prompts for the pin name and the library element name, i.e., the name of the padstack symbol to be loaded. Note that part pin names must be unique, i.e., when specifying a pin name already used on the current part, then the user must confirm to replace the existing pin. Popup menus are provided with the library element name query for selecting the library file and the padstack symbol name. The library file name list is derived from the layout library path defined with the BAE setup, i.e., all DDB files available in the directory of the layout library path are listed. Padstack symbols can optionally be selected by typing both the library file name (i.e., one of the names displayed with the library file name popup), a slash (`/`), and the padstack symbol name to the library element name prompt (typing **`?`** for the element name activates a popup menu providing the list of padstack symbols from the specified library file). An empty string input to the library element name query causes the system to use the padstack symbol previously selected with the Add Part function (if there was already one specified).

Use the following commands to place two **`q1.4`** padstack symbols to define pins **`1`** and **`2`** of the part:

```
Parts                          ▮▯▯
  Add Part                     ▮▯▯
               Part Name ?  1 ↵
    Library Element Name ?  q1.4 ↵
         Move to [0.1",0.1"]  ▮▯▯
  Add Part                     ▮▯▯
               Part Name ?  2 ↵
    Library Element Name ?  ↵
         Move to [0.5",0.1"]  ▮▯▯
```

The padstack **`q1.4`** is one of the symbols created in chapter 2.2.2. An empty string input (i.e., pressing the return key ↵) to the library element name prompt causes the system to use the name of the previously loaded padstack symbol. With all required pins placed on the part the part package definition is basically completed. What still might be missing is drawing and/or text information required for documentation purposes or CAM output.

Creating Drawing Information for the Insertion Plan

Use the following commands to create a documentary line on the Insertion Plan (Side 2) documentary layer for indicating the part contour:

```
Areas                          ▮▯▯
  Add Document Line            ▮▯▯
    Insertion Plan             ▮▯▯
         Move to [0.15",0.05"]  ▮▯▯
         Move to [0.45",0.05"]  ▮▯▯
         Move to [0.45",0.15"]  ▮▯▯
         Move to [0.15",0.15"]  ▮▯▯
         Move to [0.15",0.05"]  ▮▯▯
    ▯▯▮
      Done                     ▮▯▯
```

Use the following commands to create two Insertion Plan (Side 2) documentary lines connecting the part contour with the pins:

```
Areas                          ▮▯▯
  Add Document Line            ▮▯▯
    Insertion Plan             ▮▯▯
         Move to [0.10",0.10"]  ▮▯▯
         Move to [0.15",0.10"]  ▮▯▯
    ▯▯▮
      Done                     ▮▯▯
  Copy Area                    ▮▯▯
         Move to [0.10",0.10"]  ▮▯▯
         Move to [0.45",0.10"]  ▮▯▯
```

Defining the Part Name Reference

Use the following commands to place the text string **$** on the Insertion Plan (Side 2) documentary layer:

```
Text, Drill
    Add Text
        Document Layer
            Insertion Plan
                                         Text ? $
                           Move to [0.15",0.05"]
```

The **$** text string is utilized as a variable for indicating the name of corresponding references on superior hierarchy levels. I.e., a **$** text string placed on layout part level will display pertinent part reference names (e.g., **`IC01`**, **`R20`**, **`V2`**) on the PCB layout. Such reference text definitions can also be placed on a special layer for generating insertion data. The **CAM Processor** Insertion Output function can then be used to print all insertion data layer texts with coordinates and rotation angles. I.e., the insertion data text position must exactly match the appropriate part pick point for the automatic insertion equipment, and it is also recommended to disable reference text movement by setting the **`PHYSICAL`** text mode for the insertion data documentary layer (see chapter 7.2 for the description of the **BSETUP** utility program and how to define documentary layers).

Defining the Origin

Use the following commands to set the part symbol origin to pin **`1`**:

```
Settings
    Set Origin
                   Move to Pin "1",[0.1",0.1"]
```

The part origin is the reference point for placing the part on the layout. On packages with drilled pins usually the first pin position is used, whilst on SMD packages the origin is usually set to the part center point (i.e., the pin gravity point). It is recommended to refrain from off-grid origin settings in order to avoid off-grid pin placement on the layout since on-grid items make the job much easier for the **Autorouter** (and of course for the manufacturing process).

Saving the Element

The definition of the **`r04a25`** part symbol is completed now. Do not forget to *save* this *symbol* with the following commands:

```
File
    Save Element
```

Creating an SMD Part

Use the following commands to create a new part symbol named **so14** (14-pin small outline package) with an element size of 0.5*0.4 inch in DDB file **demo.ddb**:

File	🖱
New	🖱
Part	🖱
File Name ?	demo ↵
Element Name ?	so14 ↵
Element Width (mm/") ?	0.5" ↵
Element Height (mm/") ?	0.4" ↵

Use the following commands to define the pins **1** through **7** by loading and placing the **so** padstack symbol which was created in chapter 4.2.2:

Parts	🖱
Add Part	🖱
Part Name ?	1 ↵
Library Element Name ?	so ↵
Move to [0.10",0.10"]	🖱
Add Part	🖱
Part Name ?	2 ↵
Library Element Name ?	↵
Move to [0.15",0.10"]	🖱
Add Part	🖱
Part Name ?	3 ↵
Library Element Name ?	↵
Move to [0.20",0.10"]	🖱
Add Part	🖱
Part Name ?	4 ↵
Library Element Name ?	↵
Move to [0.25",0.10"]	🖱
Add Part	🖱
Part Name ?	5 ↵
Library Element Name ?	↵
Move to [0.30",0.10"]	🖱
Add Part	🖱
Part Name ?	6 ↵
Library Element Name ?	↵
Move to [0.35",0.10"]	🖱
Add Part	🖱
Part Name ?	7 ↵
Library Element Name ?	↵
Move to [0.40",0.10"]	🖱

Use the following commands to define the pins **8** through **14** by loading and placing the **so** padstack symbol:

```
Parts                              ▮▯▯
    Add Part                       ▮▯▯
                    Part Name ?    8 ↵
         Library Element Name ?    so ↵
           Move to [0.40",0.30"]   ▮▯▯
    Add Part                       ▮▯▯
                    Part Name ?    9 ↵
         Library Element Name ?    ↵
           Move to [0.35",0.30"]   ▮▯▯
    Add Part                       ▮▯▯
                    Part Name ?    10 ↵
         Library Element Name ?    ↵
           Move to [0.30",0.30"]   ▮▯▯
    Add Part                       ▮▯▯
                    Part Name ?    11 ↵
         Library Element Name ?    ↵
           Move to [0.25",0.30"]   ▮▯▯
    Add Part                       ▮▯▯
                    Part Name ?    12 ↵
         Library Element Name ?    ↵
           Move to [0.20",0.30"]   ▮▯▯
    Add Part                       ▮▯▯
                    Part Name ?    13 ↵
         Library Element Name ?    ↵
           Move to [0.15",0.30"]   ▮▯▯
    Add Part                       ▮▯▯
                    Part Name ?    14 ↵
         Library Element Name ?    ↵
           Move to [0.10",0.30"]   ▮▯▯
```

Use the following commands to create two Insertion Plan (Side 2) documentary lines for indicating the part contour:

```
Areas                              ▮▯▯
    Add Document Line              ▮▯▯
        Insertion Plan             ▮▯▯
         Move to [0.100",0.100"]   ▮▯▯
         Move to [0.425",0.100"]   ▮▯▯
         Move to [0.425",0.300"]   ▮▯▯
         Move to [0.075",0.300"]   ▮▯▯
         Move to [0.075",0.125"]   ▮▯▯
         Move to [0.100",0.100"]   ▮▯▯
        ▯▯▮
            Done                   ▮▯▯
    Add Document Line              ▮▯▯
        Insertion Plan             ▮▯▯
         Move to [0.075",0.125"]   ▮▯▯
         Move to [0.425",0.125"]   ▮▯▯
        ▯▯▮
            Done                   ▮▯▯
```

Frequently certain distances between SMD parts are required by the manufacturing process to avoid solder bridging. An appropriate SMD part clearance check can be supported by defining corresponding keepout areas on a certain documentary layer. The **Layout Editor** DRC will then mutually check these keepout areas and indicate distance violations in case of overlaps. Use the following commands to define a keepout area on documentary layer Insertion Plan (Side 2) for the SMD part clearance check:

```
Areas
    Add Keep Out Area
        Document Layer
            Insertion Plan
        Move to [0.05",0.05"]
        Move to [0.45",0.05"]
        Move to [0.45",0.35"]
        Move to [0.05",0.35"]

            Done
```

Use the following commands to set the part origin to the part center point (i.e., to the pin gravity point):

```
Settings
    Set Origin
        Move to [0.25",0.20"]
```

Use the following commands to place the text string **$** at the part origin with a text size of 0.05 inch on the Insertion Plan (Side 2) documentary layer:

```
Text, Drill
    Add Text
        Document Layer
            Insertion Plan
        Text ? $

            Text Size
                Text Size ( 2.54mm) ? 0.05"
        Move to [0.0",0.0"]
```

The definition of the **so** part symbol is completed now. Do not forget to *save* this *symbol* with the following commands:

```
File
    Save Element
```

Creating a Constructive Part Symbol

Use the following commands to create a new part symbol named **`hole3mm`** (drill hole with 3mm diameter) with an element size of 4*4mm in DDB file **`demo.ddb`**:

- File
  - New
    - Part
      - File Name ? demo
      - Element Name ? hole3mm
      - Element Width (mm/") ? 4
      - Element Height (mm/") ? 4

Use the following commands to set the part origin and to change the upper/right element boundary:

- Settings
  - Set Origin
    - Move to [0.075",0.075"]
  - Upper/Right Border
    - Move to [0.075",0.075"]

Use the following commands to place the drill hole padstack symbol **`drill3.0`** which was created in chapter 4.2.2 ("dummy" pin name **`x`** can be used since the **`hole3mm`** part symbol is a constructive definition without any logical counterpart):

- Parts
  - Add Part
    - Part Name ? x
    - Library Element Name ? drill3.0
    - Move to [0,0]

Use the following commands to create a circle-shaped documentary line on the Insertion Plan (Side 2) documentary layer for indicating the part contour:

- Areas
  - Add Document Line
    - Insertion Plan
    - Jump Absolute
      - Absolute X Coordinate (mm/") ? 1.6
      - Absolute Y Coordinate (mm/") ? 0
    - Arc Right
    - Jump Absolute
      - Absolute X Coordinate (mm/") ? 0
      - Absolute Y Coordinate (mm/") ? 0
    - Done

The definition of the **`hole3mm`** part symbol is completed now. Do not forget to *save* this *symbol* with the following commands:

- File
  - Save Element

4.3 Designing PCB Layouts

The layout level is the top hierarchy level of the BAE PCB design system database. On layout level, the PCB contour is defined, the parts (from the subordinate layout part level) are placed, keepout areas, power planes and copper areas are defined, the traces are routed and, finally, the CAM output is generated. Drawing items and text can be created on layout level for such things as plot registration markers, measurement, project identification, etc.

This section explains how to use the basic functions for creating layouts, placing parts, creating text and documentary graphic and performing manual routing. A layout named **`board`** will be created in the **`demo.ddb`** DDB file, and the layout symbols generated in the preceding chapters will be used on that layout.

Use the following commands to move to the BAE examples directory (i.e. where the example job file **`demo.ddb`** resides, e.g., **`c:\baejobs`**) and start the **Bartels AutoEngineer**:

```
>  C: ↵
>  cd c:\baejobs ↵
>  bae ↵
```

Move the menu cursor to the Layout menu item and confirm this choice by pressing the left mouse button:

Layout

The **Layout Editor** is activated and you can start to edit PCB layouts.

4.3.1 Creating and Editing PCB Layouts

Creating a new Layout

Use the following commands to create a new layout in the **`demo.ddb`** DDB file with an element size of 3.2*2.9 inch:

- File
  - New
    - Layout

File Name ?	demo ↵
Element Name ?	board ↵
Element Width (mm/") ?	3.2" ↵
Element Height (mm/") ?	2.9" ↵

A frame designating the element boundaries of the layout appears on the screen. If the system issues an error message such as **`Element does already exist!`**, then a layout with the specified element name has already been generated. In this case, the requested element cannot be created, but must be loaded (see below). It is strongly recommended to specify a layout element name identical to the name of the net list for which the layout should be designed, because otherwise the system won't be able to correlate the layout data with the net list definitions.

Use the following commands to store the new layout element and return to the BAE main shell:

- File
  - Save Element
- File
  - Main Menu

Editing an existing Layout

Call the **Layout Editor** again, and use the following commands to reload the previously created layout element (you are now accessing an existing element; thus you can select the file and element name via popup menu and mouse-pick):

- File
  - Load Element
    - Layout
      - File Name ? demo
      - Element Name ? board

The system should load the previously created layout. An error message such as **`File not found!`** is issued if the requested DDB file is not available. An error message such as **`Element not found!`** is issued if the requested element is not available in the selected DDB file.

When entering the **Layout Editor**, the system knows the name of the file previously processed in any other BAE module. On subsequent file name queries, this global project name can alternatively be specified by selecting the Project button of the file name popup menu or by entering an empty string (i.e., by pressing the return key) to the file name prompt.

Selecting the Project button of the element name popup menu or entering an empty string (i.e., pressing the return key) to the element name prompt causes the system to use the default layout element name defined with the setup (see also chapter 7.2 for a description of the **BSETUP** utility program and how to use the **`LAYDEFELEMENT`** command).

The function for loading a layout does not only load the pertinent data from the lower hierarchy levels (parts, padstacks and pads) but also the corresponding net list. The net list data is correlated with all geometrical data on the layout ("Connectivity Generation"). This indeed presupposes identical names for both the layout and the net list. After successfully generating the connectivity, the system is capable of instantly controlling and/or correlating each layout modification with the net list definitions. This highly sophisticated layout design feature is called "Full Copper Sharing" or "True Connectivity". Full copper sharing enables real-time recognition of missing parts, open connections, short-circuits, etc.

The load layout function issues the

```
Connected pins missing!
```

warning message if certain net list parts are not yet placed on the layout.

Use the following commands to set the coordinate display mode to inch (which is preferable for the subsequent operations):

- Settings
  - Coordinate Display
    - Display Inch

From now on, Inch units are used during graphic cursor movement for displaying coordinates in the info field.

Board Outline

The design of a new layout should start with the definition of the board outline. This is a continuous polygon line defining the perimeter of the PCB. Use the following commands to create a board outline as shown in figure 4-2:

```
Areas
  Add Board Outline
    Move to [0.1",0.3"]
    Move to [0.2",0.3"]

      Arc Left
    Move to [0.4",0.3"]
    Move to [0.6",0.3"]
    Move to [2.8",0.3"]
    Move to [2.8",2.2"]
    Move to [2.3",2.7"]
    Move to [0.4",2.7"]

      Arc Left
    Move to [0.4",2.5"]
    Move to [0.2",2.5"]
    Move to [0.1",2.5"]

      Done
```

The board outline must not be confused with the element boundaries. The element boundaries are used to define the size of the layout data element. The board outline is a continuous line defining the perimeter of the PCB thus describing the area where parts and traces can be placed. The board outline is required by the Autorouter, i.e., the **Autorouter** call will fail if no board outline is defined or if parts and/or pins are placed outside the board outline.

Plot Markers

Use the following commands to define two plot markers as shown in figure 4-2:

```
Areas
  Add Document Line
    Plot Markers
                         Move to [0.1",0.1"]
                         Move to [0.1",0.2"]
                         Move to [0.2",0.1"]
                         Move to [0.2",0.2"]
                         Move to [0.1",0.1"]

      Done
  Copy Area
                         Move to [0.1",0.1"]
                         Move to [2.7",2.6"]
```

Plot markers (or film registration marks) are plot control symbols which are intended for adjusting the plots and/or films later. The plot markers documentary layer can be defined with the **BSETUP** utility program, thus providing a useful feature of including design information on all plot layers with only one definition (see chapter 7.2 for more details).

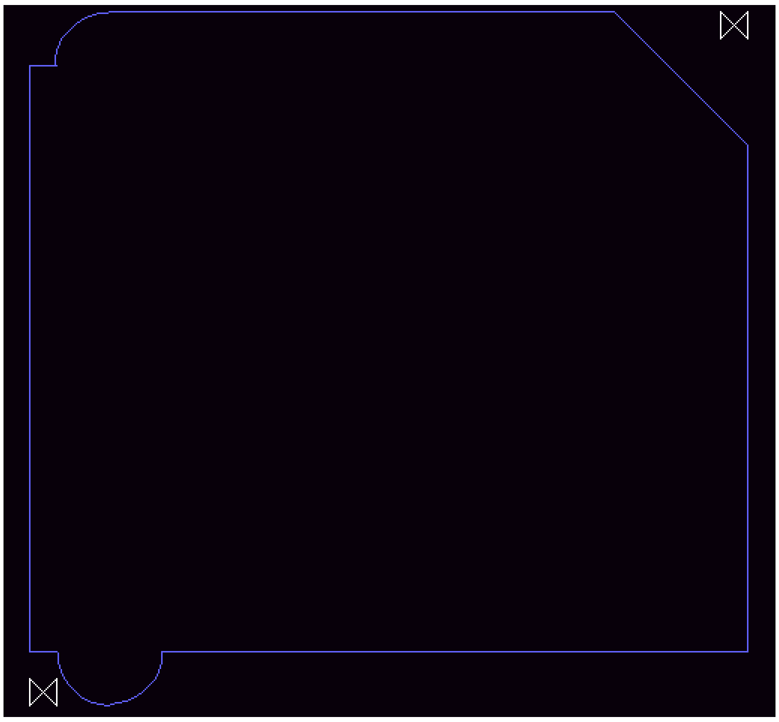
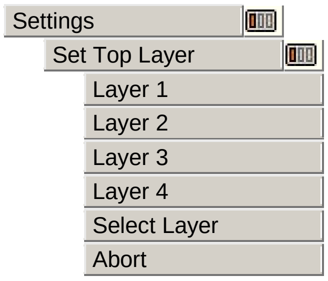


Figure 4-2: Layout with Board Outline and Plot Markers

Top Layer

The Set Top Layer function from the Settings menu is used to set the Top Layer (the component side signal layer) of the layout. This layer assignment is most useful when defining (SMT) library symbols for multilayer layouts where the number of layers is not yet determined. The Set Top Layer function can be used to assign a Top Layer for the currently loaded layout:

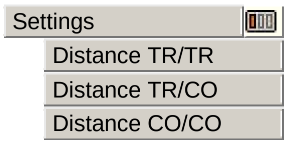


The default Top Layer setting for new layouts is signal layer 2. The Top Layer will also be the mirror position for mirroring parts, i.e., objects placed beyond the Top Layer will not be affected by mirror functions. The default routing layer count for the **Autorouter** is automatically set to the Top Layer (but can afterwards be modified in the **Autorouter**).

The **Bartels AutoEngineer** layout system supports two methods for setting the layout Top Layer. The first is to assign the desired Top Layer to each layout. This method requires all Top Layer library elements such as SMD pads to be assigned to the component side layer and all signal inside layer assignments to be correct. The All Layers and Middle Layers signal layers are provided to simplify such library definitions. The second method of defining the Top Layer is to use the default signal layer 2 setting for each layout and to interpret the layers beyond layer 2 as signal inside layers. This might slightly reduce the expense for creating layout elements, but the user must then explicitly set the **Autorouter** routing layer count when routing layouts with more than two signal layers.

Clearance Check Parameters

The Distance functions from the Settings menu are used to set the clearance check parameters for the currently loaded layout. Different Distance functions can be activated from the Settings menu:



The Distance TR/TR function is used to set the minimum trace-to-trace distance, Distance TR/CO is used to set the minimum trace-to-copper distance and Distance CO/CO is used to set the minimum copper-to-copper distance checking parameter. When creating new layouts, the default spacing value is set to 0.3mm for each clearance check parameter. The distance check parameters are applied by the Design Rule Check and should be set according to manufacturing requirements before starting with the layout design. Note that too tolerant spacing parameters will produce wasted PCBs, thus involving expensive redesign and re-manufacturing requirements.

Non-default net-specific minimum distance settings introduced with the `MINDIST` net attribute are also considered by the Design Rule Check when performing clearance checks on traces and/or active copper areas.

4.3.2 Parts, Placement

On layout hierarchy level, the Parts menu provides functions for placing, moving and deleting parts, for changing part names and for manually performing pin/gate swaps.

Input Grid

BAE supports arbitrary grids and grid-free input. Nevertheless, it is recommended to use a reasonable grid for part placement, thus avoiding off-grid pin placement and enabling pin channel routing to make the routing job easier and to yield higher routing density. Use the following commands to set a 1/10 inch input grid with grid lock:

```
View
    Grids/Rotation
        Set Input Grid
            1/10 Inch
            Grid+Rotation On
```

Library Access

The Select Library function from the Settings menu is used for selecting the layout library from which the part symbols should be loaded. Use the following commands to check the current library path selection:

```
Settings
    Select Library
        Library Name ('c:/baelib/laylib.ddb') ?
```

The name of the currently selected library is displayed with the library file name prompt. When calling the **Layout Editor**, a default library path is automatically set according to the system setup (see also chapter 7.2 of this manual for a description of the **BSETUP** utility program).

An empty string input to the library name query does not change the current library path setting. A dash string input (`-`) resets the library path setting (no library selected). **`!`** and/or **`.`** input restores the default library path defined with the setup. Under Windows, the library file name is specified through file dialog box. Use the following commands to check the library path settings:

```
Settings
    Select Library
        Library Name ('c:/baelib/laylib.ddb') ?  -
    Select Library
        Library Name ('') ?  !
```

The **Bartels AutoEngineer** supports a highly flexible hierarchical database concept. This allows for any DDB file to be used as symbol library. When loading a part symbol, the system first checks the job-specific library (i.e., the currently edited DDB file) for the requested library element. If the requested element is not available in the the project file, then the search is expanded to the currently selected library. The **Layout Editor** automatically creates a complete copy of the requested library symbol in the current project file when loading a part symbol from an external library. Subsequent requests for the same symbol will then refer to the job-specific library. Figure 4-3 illustrates the **Layout Editor** library access concept.

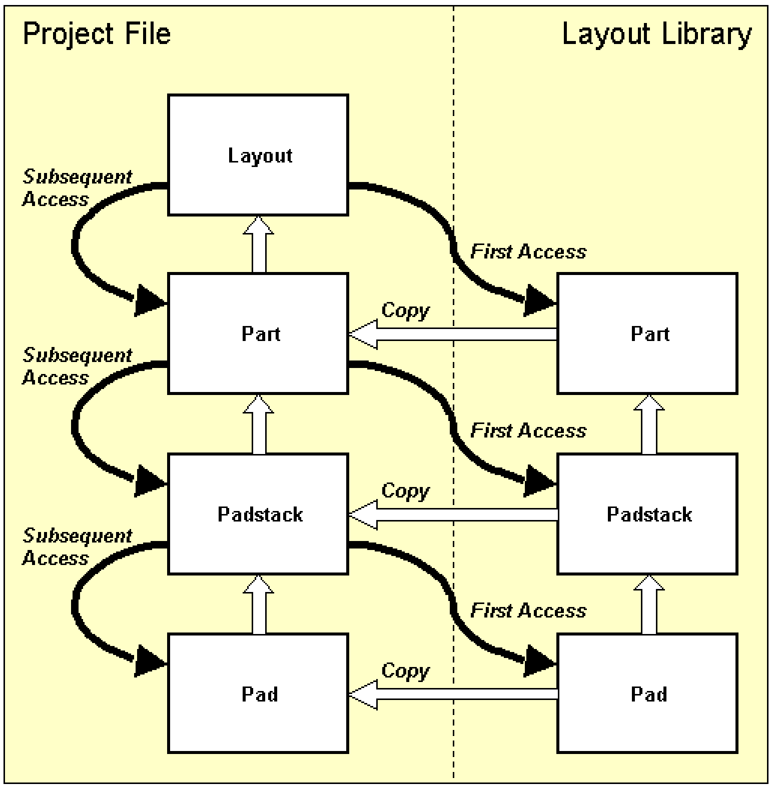


Figure 4-3: Layout Library Access

Changing the library path setting is meaningful only if a series of different part symbols are to be loaded from a library file which is not accessible through the predefined library path. Use the following commands to set the library path to the `demolib.ddb` DDB file of the current directory:

Settings
    Select Library
        Library Name ('c:/baelib/stdsym.ddb') ? | demolib

With the commands above, the library path is set to the `demolib.ddb` DDB file of the working directory, i.e., requested part symbols on subsequent load operations will be copied from `demolib.ddb`.

Placing Parts

The Add Part, Place Next Part and Place Parts functions from the Parts menu are used for placing parts. These functions work on the selected part set. Initially after loading a layout, all parts are selected, i.e., the system behaves as usual. The functions from the Part Set submenu can be used to restrict and/or redefine the set of parts selected for placement. The Place Next Part function issues a **`All parts have been placed already!`** message if all parts are already placed or a **`All selected parts have been placed!`** message if all <u>selected</u> parts are already placed, but there are still <u>unselected</u> parts which have not yet been placed. The part set selection popup window displays placed selected parts as **`[ name ]`**, placed unselected parts as **`( name )`**, unplaced selected parts as **`: name`** and unplaced unselected parts as **`< name >`**. See chapter 4.4.1 for more information on part set definitions.

The Add Part function is used to load and place parts on the current layout. On layout hierarchy level, the Add Part function activates a net list part name popup menu with the part name prompt. This popup menu displays placed net list parts in brackets (**`[`**, **`]`**). Unplaced net list parts are preceded by a colon (**`:`**). When selecting or specifying an unplaced net list part name, this part is loaded for placement. Selecting an already placed net list part prompts for confirmation to replace the placed part. Entering an empty string to the part name prompt or selecting the Unplc. button loads the next unplaced net list part. This feature is also provided with the Place Next Part function and works until all net list parts (from the part set) are placed. The system prompts for the library element name when specifying a part name which is not defined in the net list (e.g., when placing a constructive part). Popup menus are provided with the library element name query for selecting the library file and the library element name. The library file name list is derived from the layout library path defined with the BAE setup, i.e., all DDB files available in the directory of the layout library path are listed. The Lib. button or **`>`** input to the library file name prompt selects the standard layout library defined through the Select Library function from the Settings menu. The Project button in the library file name popup selects the current project file. A library element name query is activated after selecting the library file. Library elements can optionally be selected by typing both the library file name (i.e., one of the names displayed with the library file name popup), a slash (**`/`**), and the library element name to the library element name prompt (typing **`?`** for the element name activates a popup menu providing the list of library elements from the specified library file). An empty string input to the library element name query causes the system to use the library element previously selected with the Add Part function (if there was already one specified).

The net list part name selection popups are also provided with the Change Part Name and Netlist Part Name functions from the Parts menu. The library element name selection of the Add Part function is also implemented for placing pins (i.e., selecting padstacks) on layout part symbol level and for loading pads onto layout padstack level.

Use the following commands to load the part named **`IC10`** and place it at coordinate [1.2",1.8"]:

Parts	🖱		
	Add Part	🖱	
		Part Name ?	ic10 ⏎
		Move to [1.2",1.8"]	🖱

The Add Part function checks whether a part with the specified name exists in the net list. If the part is defined in the net list, then the system knows the package to be used for this part and automatically loads the corresponding part symbol (i.e., **`dil14`** in the example above). The system issues an error message such as

```
Element not found!
```

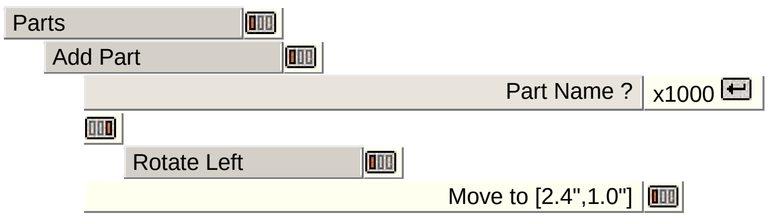
if the requested part symbol is not available in the current job file or in the selected library. In this case you should use the Select Library function from the Settings menu in to check whether the correct library is accessed.

Use the following commands to load and place two parts named **`hole1`** and **`hole2`** using the **`hole3mm`** package type which was created in chapter 4.2.3 in DDB file **`demo.ddb`**:

Parts	🖱		
	Add Part	🖱	
		Part Name ?	hole1 ⏎
		Library Element Name ?	hole3mm ⏎
		Move to [0.4",0.3"]	🖱
	Add Part	🖱	
		Part Name ?	hole2 ⏎
		Library Element Name ?	⏎
		Move to [0.4",2.5"]	🖱

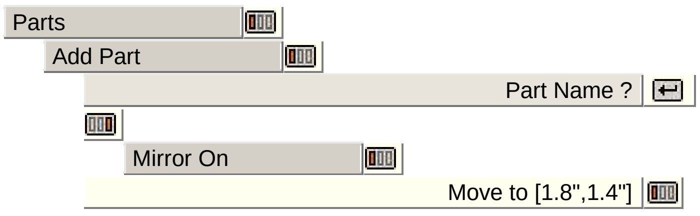
The (constructive) parts **`hole1`** and **`hole2`** are not defined in the net list, i.e., the system does not know the corresponding package types. The Add Part function activates a library element name query with a popup menu providing access to all library files available through the predefined default library path. The Project button of this menu can be used for accessing the job-specific library of the currently processed DDB file. After selecting the library file, another popup menu with a list of all symbols available in the selected library file is activated. An empty string input (i.e., just pressing the return key) to the library element name query causes the system to use the part symbol loaded with the preceding Add Part call.

Use the following commands to load and place the plug named **`x1000`**:



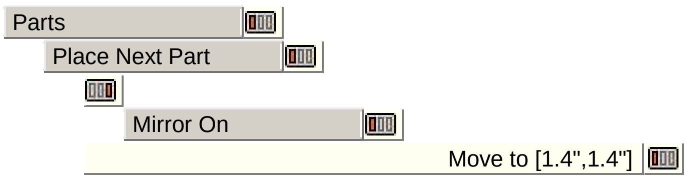
Pressing the right mouse button during part placement activates a submenu which provides functions for specifying absolute placement coordinates, part rotation (even at arbitrary rotation angles) and part mirroring (for placing SMDs on solder side).

Use the following commands to place the next part:



An empty string input (i.e., just pressing the return key) to the part name query causes the Add Part function to load the next unplaced net list part. In the example above, the system automatically loads the SMD capacitor named **`c100`**, which then is placed on solder side.

The facility for loading the next net list part (without knowing the corresponding part name) can also be activated with the Place Next Part function. Use this command to place the part named **`c101`** as in



Use the following commands to place the relay part named **`k1`**, the resistor parts named **`r100`**, **`r101`**, **`r102`**, **`r103`**, **`r104`**, **`r105`** and the switch named **`s1000`**:

```
Parts ▮▯▯
    Place Next Part ▮▯▯
        Move to [0.8",0.8"] ▮▯▯
    Place Next Part ▮▯▯
        Move to [1.0",2.4"] ▮▯▯
    Place Next Part ▮▯▯
        Move to [1.6",2.4"] ▮▯▯
    Place Next Part ▮▯▯
        ▯▯▮
            Rotate Left ▮▯▯
        Move to [1.6",1.2"] ▮▯▯
    Place Next Part ▮▯▯
        ▯▯▮
            Rotate Left ▮▯▯
        Move to [1.4",1.2"] ▮▯▯
    Place Next Part ▮▯▯
        Move to [2.1",2.1"] ▮▯▯
    Place Next Part ▮▯▯
        Move to [2.1",1.8"] ▮▯▯
    Place Next Part ▮▯▯
        ▯▯▮
            Rotate Right ▮▯▯
        Move to [0.2",2.3"] ▮▯▯
```

The `Set Default Angle` function can be utilized for subsequently placing a series of parts at certain rotation angles and/or mirror modes (e.g., SMD parts). Use the following commands to set the standard placement rotation angle to 270 degrees and place the switches named **`s1001`** through **`s1009`**:

```
Parts ▮▯▯
    Set Default Angle ▮▯▯
        270 Degrees Left ▮▯▯
    Place Next Part ▮▯▯
        Move to [0.2",2.1"] ▮▯▯
    Place Next Part ▮▯▯
        Move to [0.2",1.9"] ▮▯▯
    Place Next Part ▮▯▯
        Move to [0.2",1.7"] ▮▯▯
    Place Next Part ▮▯▯
        Move to [0.2",1.5"] ▮▯▯
    Place Next Part ▮▯▯
        Move to [0.2",1.3"] ▮▯▯
    Place Next Part ▮▯▯
        Move to [0.2",1.1"] ▮▯▯
    Place Next Part ▮▯▯
        Move to [0.2",0.9"] ▮▯▯
    Place Next Part ▮▯▯
        Move to [0.2",0.7"] ▮▯▯
    Place Next Part ▮▯▯
        Move to [0.2",0.5"] ▮▯▯
```

Use the following commands to complete the placement, i.e., place part `v1` (with absolute placement coordinates and 45 degree rotation angle) and part `v1000`:

```
Parts
  Place Next Part
    Set Angle
      Angle (Deg./(R)ad) ? -45
    Jump Absolute
      Absolute X Coordinate (mm/") ? 0.825"
      Absolute Y Coordinate (mm/") ? 1.225"
  Place Next Part
    Move to [1.2",1.6"]
  Place Next Part
```

The Place Next Part function issues the

```
All parts have been placed already!
```

message once all net list parts are placed. I.e., the Place Next Part function can also be used to check on placement completion.

After correctly executing all work steps of this chapter, the PCB layout example should look like the one shown in figure 4-4.

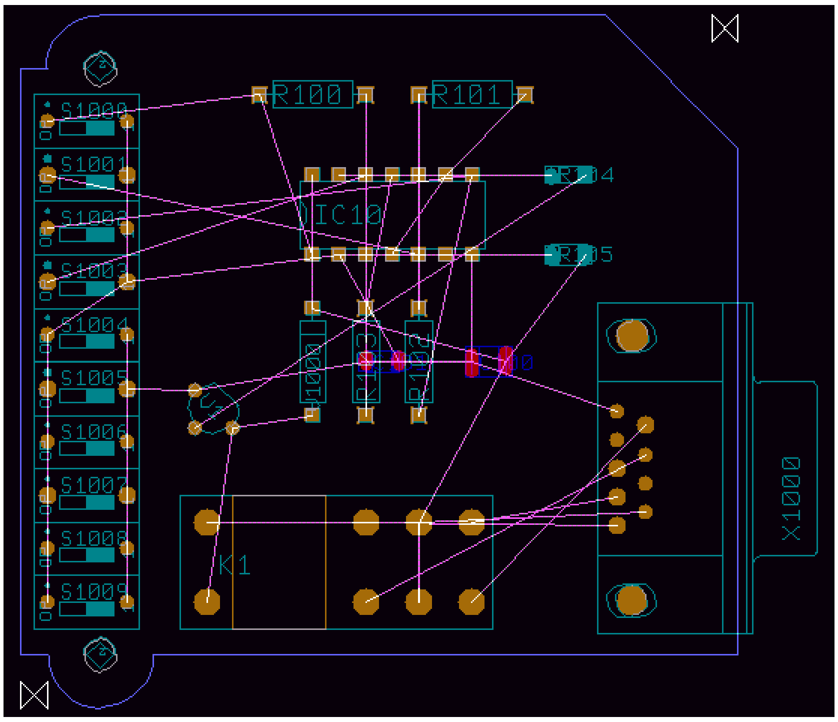


Figure 4-4: Layout with Part Placement

Moving and Deleting Parts

The Move Part function is used to move parts which are already placed on the layout. Pressing the right mouse button during part movement activates a submenu with special options for rotating and mirroring, absolute or relative placement coordinates input, etc. The Delete Part function is used to delete placed parts from the layout. You should test these functions on the currently loaded layout and use the Undo and Redo functions for estimating design modifications and returning to the original design stage as shown in figure 4-4.

Alternate Part Package Type Assignment

The Alternate Part option is provided with the **Layout Editor** submenus which can be called by pressing the right mouse button during interactive part placement. This option can be used to change the part package type of the currently placed part. The alternate package type can be selected from a list of layout part macros. The alternate part list is either pre-defined in the logical library using the **LOGLIB** utility program (see chapter 7.11) or can be defined through an SCM symbol **`$plname`** attribute value assignment such as **`[package1,package2,...]`**. An error message such as **`No alternate part macros defined!`** is issued if no alternate package types are defined for the currently processed part. An error message such as **`Connected pins missing!`** is issued if the pin definitions on the selected alternate package type do not match the net list, i.e., if net list pins are missing on the selected macro. Alternate part package type assignments create a backannotation request which triggers automatic Backannotation when loading SCM plans to the **Schematic Editor** (see chapter 2.7). Backannotation requests also prevent the **Packager** from forward annotation without confirmation (see chapter 3.2.3).

Mincon Function, Airlines Display

Unrouted net list connections are displayed as airlines. During part placement, the Mincon function performs dynamic (real-time) recalculation of these airlines to display minimum length airline sets. I.e., the Mincon function is a most useful utility for achieving an optimum manual placement. Use the following commands to test different airline calculation methods supported through the Mincon function and note how the Mincon function type affects the way airlines are displayed:

- Settings
  - Mincon Function
    - Corners Horizontal
  - Mincon Function
    - Pins Vertical
  - Mincon Function
    - Mincon Off
  - Mincon Function
    - Corners H+V Sum

The Airlines Display function from the Parts menu is used to set the airline display mode to be applied during part movement. No Airlines switches off the airlines display during part movement. Static Airlines does not recalculate the nearest connection points during part movement. Dynamic Airlines dynamically recalculates the closest connection points (note how the airlines jump to the nearest pins or copper corners during part movement). Dynamic Airlines is the default Airlines Display option.

The Net Visible and Net Invisible Mincon options allow for a net-specific airline display, i.e., it is possible to fade-out (Net Invisible) and/or fade-in (Net Visible) net-specific airlines by selecting the desired net(s) from a net name popup menu which optionally allows for multiple net selections through net name pattern specifications.

The All Nets Visible Mincon option forces the graphical display of all airlines, whilst All Nets Invisible fades out all airlines.

The context menu which is available through the right mouse button during part placement provides the All Nets and Part Nets options for selecting either part-specific or global airline display.

4.3.3 Text and Graphic

On layout hierarchy level, text (on documentary layers or on signal layers) and graphic (i.e., polygons such as documentary lines, documentary areas, passive copper areas, active copper areas, keepout areas, copper fill workareas) can be defined. For the following operations it is recommended to set the input grid to 1/20 inch (use the Grids/Rotation and Set Input Grid functions from the View menu) and to set the coordinate display mode to inch (use function Coordinate Display from the Settings menu).

Placing Text

Use the following commands to place the **DEMO** text string adjacent to the right top of the board outline with a text size of 4mm on the Insertion Plan (Side 2) documentary layer:

```
Text, Drill ▮▯▯
    Add Text ▮▯▯
        Document Layer ▮▯▯
            Insertion Plan ▮▯▯
                                        Text ? DEMO ↵
        ▯▯▮
            Set Angle ▮▯▯
                             Angle (Deg./(R)ad.) ? -45 ↵
        ▯▯▮
            Text Size ▮▯▯
                              Text Size ( 2.54mm) ? 4 ↵
                             Move to [2.4",2.6"] ▮▯▯
```

Use the following commands to place the **Component Side** text string with text size 2.54mm on the PCB component side signal layer (Top Layer, Layer n (Parts.)):

```
Text, Drill ▮▯▯
    Add Text ▮▯▯
        Layer n (Parts.) ▮▯▯
                                        Text ? Component Side ↵
        ▯▯▮
            Text Size ▮▯▯
                              Text Size ( 4.00mm) ? 2.54 ↵
                            Move to [1.00",2.55"] ▮▯▯
```

The Text Size function which can be activated whilst placing texts sets the default text size for subsequent calls to the Add Text function. Use the following commands to place a rotated and mirrored text with string **Solder Side** on the PCB solder side (signal layer 1):

```
Text, Drill ▮▯▯
    Add Text ▮▯▯
        Layer 1 ▮▯▯
                                        Text ? Solder Side ↵
        ▯▯▮
            Set Angle ▮▯▯
                             Angle (Deg./(R)ad.) ? 180 ↵
        ▯▯▮
            Mirror On ▮▯▯
                            Move to [1.80",2.55"] ▮▯▯
```

Texts placed on signal layers are subject to the design rule check. I.e., the DRC will issue distance errors for intersections between signal layer texts and other objects (pads, traces, copper areas, texts, etc.) placed on the same signal layer.

Moving Part Names and Part Attribute Texts

The Move Name function from the Text and/or Parts menu is used to move part names. Use the following commands to move the names of the SMD resistors named `r104` and `r105` on top of the parts:

```
Text, Drill
    Move Name
        Move to "r104",[2.1",2.1"]
        Move to [2.1",2.15"]
    Move Name
        Move to "r105",[2.1",1.8"]
        Move to [2.1",1.85"]
```

The Move Attribute function from the Parts and/or Text menu can be used for moving and/or placing selectable part attributes. Attribute text offsets defined through Move Attribute override the global text offset set with Move Name. The attribute to be moved is selected through a mouse-click on the attribute text. Since part names are internally stored as **$** attributes, the Move Attribute function can also be used to move part names without changing the placement of other part attributes.

Part names and part attribute texts of group-selected parts can be reset to their library-defined default positions using the Part Texts Reset function implemented with the **GEDGROUP User Language** program.

Measuring

The **Layout Editor** automatically activates a measuring function when placing new text with string **#**. This function determines the distance between the two input coordinates subsequently to be specified. The resulting text string emerges from the corresponding distance value; the length units (inch or mm) result from the current coordinate display mode selected with the Coordinate Display Settings menu function. Use the following commands to apply measuring on the bottom right board outline edge and place the resulting distance value (2.2") on the Insertion Plan (Side 2) documentary layer:

```
Text, Drill
    Add Text
        Document Layer
            Insertion Plan
        Text ? #
        Move to Left Coordinate,[0.6",0.3"]
        Move to Right Coordinate,[2.8",0.3"]
        Move to [1.50",0.15"]
```

Documentary Lines, Documentary Areas

Use the following commands to create a documentary line and two arrow-shaped documentary areas on the Insertion Plan (Side 2) documentary layer to indicate the previously defined measuring:

```
Areas                     ▮▯▯
    Add Document Line     ▮▯▯
        Insertion Plan    ▮▯▯
                            Move to [0.6",0.15"]   ▮▯▯
                            Move to [0.8",0.15"]   ▮▯▯
        ▯▯▮
            Done          ▮▯▯
    Add Document Area     ▮▯▯
        Insertion Plan    ▮▯▯
                            Move to [0.6",0.15"]   ▮▯▯
                            Move to [0.65",0.2"]   ▮▯▯
                            Move to [0.65",0.1"]   ▮▯▯
        ▯▯▮
            Done          ▮▯▯
    Add Document Area     ▮▯▯
        Insertion Plan    ▮▯▯
                            Move to [2.8",0.15"]   ▮▯▯
                            Move to [2.75",0.2"]   ▮▯▯
                            Move to [2.75",0.1"]   ▮▯▯
        ▯▯▮
            Done          ▮▯▯
```

Passive Copper Areas

Use the following commands to create a circle-shaped passive copper area on signal layer 2:

```
Areas                     ▮▯▯
    Add Passive Copper    ▮▯▯
        Layer 2           ▮▯▯
                            Move to [0.95",1.6"]   ▮▯▯
        ▯▯▮
            Arc Left      ▮▯▯
                            Move to [0.8",1.6"]    ▮▯▯
        ▯▯▮
            Done          ▮▯▯
```

Active Copper Areas

Use the following commands to create an active copper area on signal layer 1 and assign this copper area to net **Vdd**:

```
Areas                     ▮▯▯
    Add Active Copper     ▮▯▯
                            Net Name ?   vdd ↵
        Layer 1           ▮▯▯
                            Move to [2.4",1.7"]    ▮▯▯
                            Move to [2.7",1.7"]    ▮▯▯
                            Move to [2.7",2.2"]    ▮▯▯
                            Move to [2.6",2.3"]    ▮▯▯
                            Move to [2.4",2.3"]    ▮▯▯
        ▯▯▮
            Done          ▮▯▯
```

Keepout Areas

Use the following commands to define a keepout area on signal layer 1 and a keepout area on signal layer All Layers (keepout areas define layer-specific regions on the PCB where no traces, no copper areas, no texts, etc. are allowed to be placed):

```
Areas
  Add Keep Out Area
    Layer 1
                        Move to [1.8",1.2"]
                        Move to [2.2",1.2"]
                        Move to [2.2",0.9"]
                        Move to [1.7",0.9"]

      Done
  Add Keep Out Area
    All Layers
                        Move to [0.6",2.6"]
                        Move to [0.6",2.1"]
                        Move to [0.9",2.1"]
                        Move to [0.9",2.6"]

      Done
```

4.3.4 Traces, Routing

The Traces menu provides functions for interactive routing on layout and/or part hierarchy level, i.e., for manually creating new traces and for modifying or deleting existing traces and/or trace segments. Arbitrary track widths can be defined for individual trace segments or complete routing paths. Special or critical nets such as the power supply can be pre-routed and then fixed to prevent the **Autorouter** from rerouting such nets.

Graphic Control Functions, Input Grid, Display

During manual routing, the system provides a highlight on all objects of the currently routed signal. After completing the routing of a certain connection, the corresponding airline is removed from the display. Short-circuits and distance violations introduced by manual routing are immediately indicated by the Online DRC using highlight and/or distance error boxes.

If you fail to connect certain (off-grid) pins with the current input grid settings, then you should use the View functions for selecting a smaller input grid (or even release the grid lock).

The middle mouse button can be used to activate the View menu at any time during manual routing. I.e., important display options such as zoom scale, grids, colors, etc. can be changed without cancelling the currently active trace edit function.

The View menu also provides the Set Edit Layer function for selecting the pick preference layer. The pick preference layer is used to resolve ambiguities at the selection of trace elements placed at the same position but on different layers. The pick preference layer is also the default routing layer when starting to route from a drilled ("All Layer") pad or from a copper-free position. On default, the pick preference layer is set to signal layer 1 (solder side).

The algorithm for picking trace corners and trace segments picks the trace element with minimum distance from the pick point if more than one possible pick element is within snapping distance. This allows for reliable selection of trace elements even when working in (small) zoom overview display modes. The trace pick snapping distance is automatically adjusted to the trace width. I.e., trace elements (of wide traces or in high zoom display modes) can easily be picked without having to hit the trace center line.

Highlight Net

The Highlight Net function from the **Layout Editor** Traces menu is used to mark all connections of the selected net with a special ("Highlight") color. Any power layer heat-trap circles and isolations of the selected net is included with the highlight display, and isolated pins and active copper areas can also be selected for highlight. Starting a new trace on a passive copper area triggers a highlight for the nets connected to that copper area. The Highlight Net function works as a toggle. To reset the highlight of a certain net, pin or active copper area, simply re-select that object with the Highlight Net function. Advanced color support and highlight reset facilities are provided through the Highlight Net (default mode), Color Nets (with highlight color selection) and Reset All (for clearing all nethighlight) options of the Highlight Net function.

In **BAE HighEnd**, the Highlight Net function causes a highlight and/or de-highlight of the selected nets in *all* currently loaded plans of the current project file on layout board and schematic sheet level (global net highlight, cross-probing).

Selecting the Via

The **Bartels AutoEngineer** layout system utilizes special padstack symbols (vias) for connecting different layers of the PCB. At least one via assignment is required before any routing function (i.e. either a trace function or the **Autorouter**) can be activated. Use the following commands to insert the `estk` padstack symbol to the currently active via list:

```
Parts
    Select Via
        New/Delete: Via Name ?  estk
        New/Delete: Via Name ?
```

The `estk` padstack symbol is used as pin symbol on the already placed part `v1000` (package type `d04a25`). I.e., `estk` is available in the current job file. A series of via padstack definitions are provided with the layout libraries shipped with the BAE software. Usually the padstack symbol named `via` can be used as standard via. The via assignments introduced with Select Via are stored with the layout element. The Add Trace function issues the following error message if no via is selected:

```
Default via not defined!
```

The system is capable of processing several different via definitions at a time in order to support blind and buried vias. The via assignment list can be changed at any time. Previously placed vias will not be affected by via list modifications, i.e., the currently defined via list will apply for the interactive routing of new traces. When changing layers during manual routing the **Layout Editor** will automatically place vias with the least possible layer usage (if padstacks are defined and selected to the via list accordingly). The current via assignment list will also be used by the **Autorouter**, and, with the use of blind and buried vias, the **Autorouter** will minimize layer occupancy as well (this is true for all traces except for fixed traces which will not be processed by the **Autorouter**).

Standard Trace Widths

The trace function submenus provide options for toggling between thin and wide default track widths during manual routing (necking). The standard values for these default track widths can be changed with the Set Trace Width function from the Settings menu, where the options Set Thin Default (default value 0.3mm) and Set Wide Default (default value 1.0mm) are provided. The routing of a new trace always starts with the thin default track width.

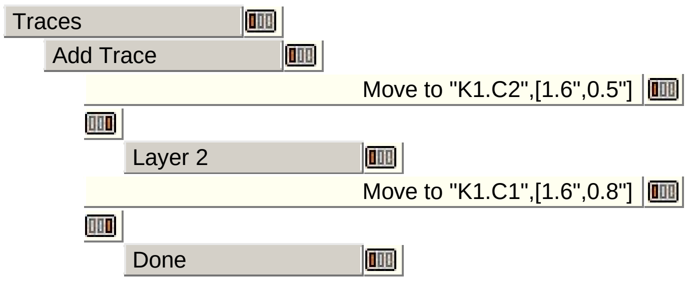
Manual Routing

Use the following commands to route from pin `A1` of part `K1` to pin `C1` of the same part (both pins are connected to net `Vdd`):

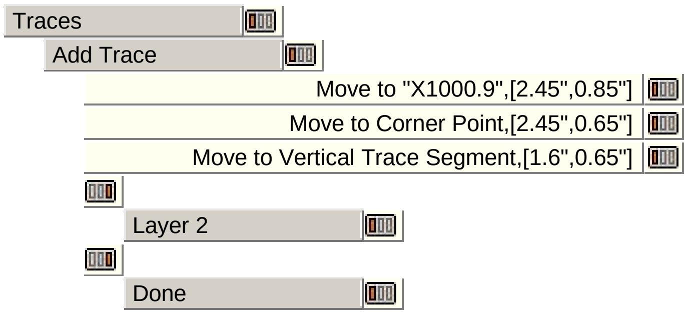
```
Traces
    Add Trace
        Move to "K1.A1",[0.8",0.8"]
        Move to Corner Point,[0.8",0.95"]
        Move to Corner Point,[1.6",0.95"]
        Move to "K1.C1",[1.6",0.8"]

            Done
```

Each trace corner point is set by pressing the left mouse button. The routing of a trace path is completed with the Done option from the submenu available with the right mouse button. This submenu is also available during trace editing with functions such as Insert Corner or Move Segment where the routing layer or the track width can be changed, arc-shaped trace segments can be created, relative and absolute coordinate input can be specified, etc.

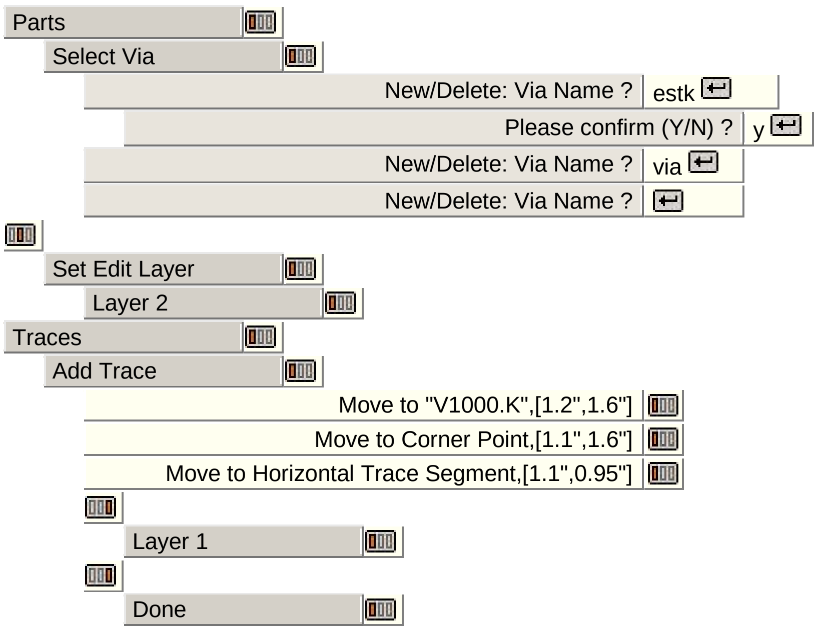
Use the following commands to connect the pins **C2** and **C1** of part **K1** with a trace segment on signal layer 2:



Use the following commands to create a trace segment on signal layer 1 starting at pin **9** of the plug named **X1000** and ending with a via to connect to the previously generated trace:


Traces
Add Trace
Move to "X1000.9",[2.45",0.85"]
Move to Corner Point,[2.45",0.65"]
Move to Vertical Trace Segment,[1.6",0.65"]
Layer 2
Done

Use the following commands to deselect **estk** from the via list, select the padstack symbol **via** (which has been created in chapter 4.2.2 in the **demo.ddb** DDB file) to the via list, and then route a trace on signal layer 2 starting at pin **K** of the diode named **V1000** and ending with a via connecting to the horizontal trace between the pins **A1** and **C1** of part **K1**:



You should now gain more practice by using the trace functions on the currently loaded layout in order to create and/or modify more traces (move, cut and delete trace segments; insert, move and delete trace corner points; change track widths, create arc-shaped traces, etc.). Note the Move without Neighbour, Move with Neighbour and Adjust neighbours options provided with the submenu of the Move Segment function. The Move without Neighbour default mode just moves the corner points of the processed trace segment, whilst the Move with Neighbour mode also rearranges adjacent trace segments in order to keep the angles between the processed trace and the adjacent segments constant. Adjust neighbours considers intersection points between new and neighbouring segments and rearranges neighbouring segments accordingly. This feature is especially useful for moving diagonal segments at trace corners. The Move w.out neighb. option is applied if neighbouring segments cannot be rearranged. Use the Undo and Redo functions to estimate design modifications and return to the original design stage.

Fixing Traces

It is recommended to fix pre-routed traces which must not be modified and/or rerouted by the **Autorouter**. Use the following commands to fix all prerouted traces of net **Vdd**:

Traces
Fix Trace/Net
Net
Move to Signal/Net "Vdd",e.g.[2.0",0.65"]

The Unfix Trace/Net function is used to release fixed traces or nets. During fix and release operations the **Layout Editor** provides a specific highlight for indicating currently fixed elements.

The **Layout Editor** trace functions for interactive routing keeps traces and vias fixed when processing pre-routed fixed traces. I.e., when manipulating pre-routed fixed traces, there is no need of re-fixing such traces to prevent the **Autorouter** from re-routing and/or rejecting pre-routed traces.

4.4 Autoplacement

The BAE layout system is equipped with powerful **Autoplacement** functions. The part set functions are used for selecting and/or deselecting parts for subsequent placement operations. The matrix placement facilities are used for placing selectable part sets onto definable placement grids. Initial placement functions are provided for performing fully automated part placement. The system also provides placement optimization functions for automatic part and pin/gate swap.

The **Autoplacement** functions are available from the Part Set, Auto Placement and Matrix Placement submenus of the **Layout Editor** Parts menu.

Start the **Layout Editor** if it is not yet active and load the layout **`board`** from the **`demo.ddb`** DDB file.

Use the Delete Part function from the Parts menu to delete the resistors named **`r100`** and **`r101`**, and apply group functions to delete the switches named **`s1000`** through **`s1009`**:

```
Parts
    Delete Part
                      Move to "r100",[1.0",2.4"]
    Delete Part
                      Move to "r101",[1.6",2.4"]
Groups
    Group Polygon
        Parts
            Select
                              Move to [0.1",0.4"]
                              Move to [0.6",0.4"]
                              Move to [0.6",2.4"]
                              Move to [0.1",2.4"]
                
                    Done
    Delete Group
```

Autoplacement features and functions are used throughout the following sections to place the deleted parts back onto the layout.

4.4.1 Part Set

The Part Set submenu provides features for selecting the parts to be subsequently placed. Parts can be selected and/or deselected according to set principles either by specifying part names (**`c1`**, **`r2`**, **`ic15`**, etc.) or by specifying package types (**`dil14`**, **`so20`**, **`plcc44`**, etc.), where wildcards are also permitted. The user can also select parts from certain blocks of a hierarchical circuit design. With the Part Set functions the user is able to control the placement process in a way that e.g., first the plug(s) can be placed, then the DIL packages, then the capacitors, etc.

Use the following commands to deselect all unplaced parts:

```
Parts
  Part Set
    All
      Deselect
        Deselect all parts ? | j ⏎
    Abort
```

Now you might want to use the List Parts command to have a look at the part list:

```
Parts
  Part Set
    List Parts
```

The List Parts function provides verbose output of all net list parts:

```
File : demo.ddb  Layout : board  Parts : 23


 : c100   chip1210  P  : c101   chip1206  P  : ic10   dil14     P
 : k1     relais    P  : r100   r04a25    U  : r101   r04a25    U
 : r102   r04a25    P  : r103   r04a25    P  : r104   minimelf  P
 : r105   chip1206  P  : s1000  s1dilo    U  : s1001  s1dilo    U
 : s1002  s1dilo    U  : s1003  s1dilo    U  : s1004  s1dilo    U
 : s1005  s1dilo    U  : s1006  s1dilo    U  : s1007  s1dilo    U
 : s1008  s1dilo    U  : s1009  s1dilo    U  : v1     to92      P
 : v1000  d04a25    P  : x1000  xsubd9bl  P  - End -
```

Each part is listed with part name, library name and placement status, respectively. The placement status is indicated using either of the characters **`U`**, **`P`** or **`S`**. **`U`** indicates that the part is neither placed nor selected for placement. **`P`** indicates that the part is already placed on the layout. **`S`** indicates that the part is not yet placed but selected for subsequent placement operations. Press x and the return key ⏎ to exit from the parts list.

Use the following commands to select all unplaced parts with part names starting with an **`r`**:

```
Parts
  Part Set
    Part(s)
      Select
        Part Name ? | r* ⏎
        Part Name ? | ⏎
```

The Select and Deselect part set selection options from the Part(s) function also provide a popup menu for part selections. With the Select option, the popup menu only displays unselected parts, whilst with the Deselect option, only selected parts are displayed. A part name pattern can be entered for part set selections.

The above commands selected the two resistors **`r100`** and **`r101`** for subsequent placement operations. You can check this with the List Parts function (see placement status **`S`** for these two parts).

The All function can be used to select all unplaced parts for placement. The Macro(s) function can be used to select parts with library names (such as **`so16`** for all unplaced 16 pin small outline package types or **`dil*`** for all unplaced dual-in-line package types). The Deselect option can be used to deselect parts from the set of parts to be subsequently placed.

The List Blocks and Block functions from the Part Set menu can be used for selecting and/or deselecting parts which are defined in certain blocks of a hierarchical circuit design. I.e., parts are selected or deselected through block names. Part block names are automatically transferred and stored by the **Packager** utilizing the internal **`$blkname`** part attribute.

It has already been mentioned that the Add Part and Place Next Part functions work on the part set (see section Placing Parts in chapter 4.3.2). Use the following commands to set the coordinate display mode to inch, and place all parts of the currently defined part set (i.e., the two resistors named **`r100`** and **`r101`**):

- Settings
  - Coordinate Display
    - Display Inch
- Parts
  - Place Next Part
    - Move to [1.0",2.4"]
- Part
  - Place Next Part
    - Move to [1.6",2.4"]
- Parts
  - Place Next Part

The Place Next Part function issues the

```
All selected parts have been placed already!
```

message if no more part is selected for subsequent placement operations anymore (i.e., all parts of the previously defined part set are already placed).

4.4.2 Matrix Placement

Matrix placement is a special initial placement process for automatically placing selected parts on certain matrix-defined insertion positions.

Use the following commands to select all parts with library name **`s1dilo`** (i.e., the switches named **`s1000`** through **`s1009`**) for subsequent placement operations:

```
Parts
  Part Set
    Select Macro
      Select
                      Library Element Name ?  s1dilo
                      Library Element Name ?
```

Set the default placement rotation angle to 270 degree (without mirroring (the Default Angle option is probably located in the Parts group of the Settings dialog from the Settings menu when working with customized BAE user interfaces):

```
Parts
  Default Angle
    270 Degrees Left
```

Use the following commands to define a placement matrix, place all selected parts onto this matrix and delete the placement matrix definition again:

```
Parts
  Matrix Placement
    Define Matrix
                    Move to Matrix Origin,[0.2",2.3"]
       Move to Matrix Element Size Designation,[0.3",2.1"]
                Move to Matrix Size Designation,[0.2",0.5"]
    Place Matrix
    Delete Matrix
```

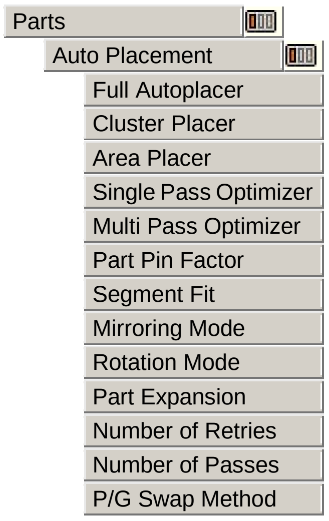
The Place Matrix function automatically places all parts of the currently defined part set onto the placement matrix, also considering the current default placement rotation angle and mirror mode settings.

Now all net list parts are placed. You might want to check this with the List Parts function.

4.4.3 Initial Placement

The system provides full automatic initial placement algorithms. The initial placement functions automatically place all unplaced net list parts inside the board outline onto the currently selected input (placement) grid, considering pre-placed parts (connectors, LEDs, etc.), keepout areas and net list preferences, and also featuring automatic block capacitor and SMD device recognition. Automatic SMD part mirroring can optionally be enabled to allow SMD placement on the solder side. Either unrestricted or restricted automatic part rotation in 90 degree steps can be applied; restricted part rotation can be used to simplify subsequent insertion processes. A generally applicable part expansion value can optionally be defined to ensure minimum clearance between adjacent parts. The placement algorithms are guarded by adjustable strategy parameters for net list preference control and for the consideration of part segment matching/fitting. Rip-up and retry passes are automatically applied during the placement process to optimize board area yield.

The initial placement routines are integrated to the Auto Placement submenu which provides the following functions:



The Full Autoplacer, Cluster Placer and Area Placer function can be used to perform automatic part placement. Only the parts from the current part set (see chapter 4.4.1) are placed. Placed parts which are not selected to the current part set are treated like fixed parts. This feature allows for automatic placement of selectable part groups.

The first part selected for placement will be positioned at the placement start point. The Full Autoplacer function automatically designates the placement start point. The Cluster Placer and Area Placer functions prompt the user for the placement start point (prompt `Select placement start point!`). The placement of the first part determines the placement of subsequent parts. I.e., the selected placement start point position has crucial meaning for the complete placement process (with regard to 100% completion success and/or the quality of the achieved placement). If the first part cannot be placed at the placement start point, then no subsequent part can be placed either. If the placement start point is selected outside the board outline, then no part can be placed at all. If no board outline is defined then there are no restrictions on selecting the placement start point, however parts could then be placed even outside the boundaries of the currently loaded layout element.

After starting an initial placement function the placement progress is indicated in the status line by a progress message such as `Pass : 1/1 Part : <p>/<n>`. The autoplacement process can be stopped by keystroke. Each autoplacement stop request is verified with user confirmation and will be denoted by a message such as `Autoplace aborted!`. Note that there might be a short delay on stop requests since any currently active autoplacer rip-up and retry pass must be completed.

A message such as `Operation completed without errors.` is issued after completing an initial placement function to indicate that all parts have been successfully placed. If the autoplacer fails to complete 100% part placement using the current parameter settings, then a message such as `<n> parts could not be placed!` is issued, denoting the number `<n>` of yet unplaced parts. On incomplete initial placement it is recommended to change placement parameters (use smaller placement grid, reduce part expansion, use unrestricted part rotation, allow SMD mirroring; see below on how to set placement parameters) and restart the autoplacer (after Undo and/or selecting a different placement start point) to achieve 100% part placement.

Full Autoplacer

The Full Autoplacer function performs cluster placement (see below) with automatic placement start point selection and subsequently applies multi-pass placement optimization (see chapter 4.4.4 below) using the currently selected placement optimization parameters (number of passes and the pin/gate swap method). The placement start point results from the board outline gravity point. No part can be placed using the Full Autoplacer function, if the board outline gravity point is outside the board outline (e.g., on L-shaped PCBs) then ; in this case either Cluster Placer or Area Placer must be used instead of Full Autoplacer.

If no board outline is defined, then the placement start point results from the absolute origin of the currently loaded layout element. In this case it is recommended to zoom to overview after completing the placement process since parts can be placed beyond the currently defined and/or visible layout element boundaries.

Cluster Placer

After activating the Cluster Placer function the user is expected to select the placement start point interactively. The first part selected for placement is positioned at the placement start point.

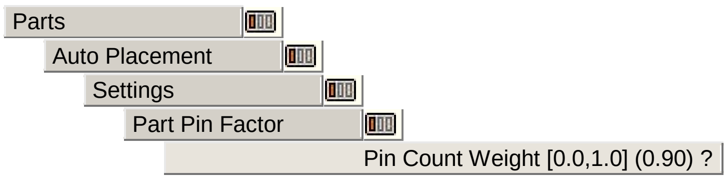
The Cluster Placer function analyzes the net list to classify the nets and parts contained in the net list. Parts are grouped to clusters, and these clusters are subsequently placed on the layout with respect to pre-calculated cluster-dependent placement priorities. Each cluster is grouped by selecting a multi-pin part and a series of connected parts with less than four pins. Parts which are connected to power signals only (such as block capacitors) are postponed during cluster generation and will be assigned to clusters with corresponding power connections afterwards. This method of cluster building automatically involves block capacitor recognition, thus featuring appropriate block capacitor placement nearby corresponding ICs. Parts and/or part clusters are assigned and/or processed due to alphabetical part name order. It is recommended to introduce part type grouping by choosing appropriate type-specific part name patterns, i.e., to insert connectors at the end of the part name list to avoid block capacitor assignment to connectors and/or plugs (e.g., by using name pattern **`x???`** for connectors and/or plugs, **`ic???`** for integrated circuits, etc.).

Area Placer

The Area Placer function performs area placement with interactively selectable placement start point. Area placement does not consider any net list preferences, i.e., only part dimensions are considered as placement criteria. The Area Placer function can be used to estimate part placement area requirements, i.e., to check whether the board area is large enough to place all parts.

Part Pin Factor

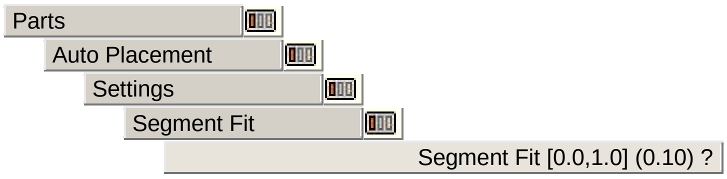
The part pin factor is used to control the sequence of parts and/or clusters to be placed. The next part to be placed is selected by applying a combination of two strategies. The first strategy simply selects the part with the maximum number of pins connected to already placed parts. The second strategy selects the part with the maximum ratio of number of part pins connected to already placed parts to total part pin count. The part pin factor value designates which strategy has higher priority. Use the



function to set the desired part pin factor value in the range 0.0 (for considering connection counts only) to 1.0 (for considering connection count to total pin count ratios only). On default the part pin factor value is set to 0.9. A high part pin factor value usually results in a better distribution of nets, but could cause excessive placement area fragmentation on high-density layouts by early placement of small parts, which might prevent from placing larger parts afterwards.

Segment Fit

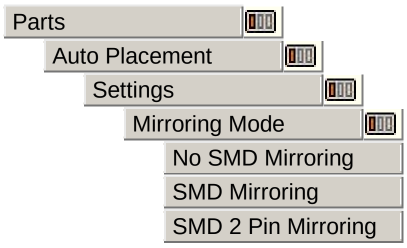
The segment fit value is used to control on how much part edge length matching should be considered by the placement algorithm, i.e., whether preference should be given on placing parts with equal edge lengths side by side. Use the



function to set the weight factor for considering part edge length matching. Valid segment fit values are in the range 0.0 (no segment fit preference) to 1.0 (high segment fit preference). On default the segment fit value is set to 0.1. High segment fit values usually result in better looking layouts, and can increase the routability of layouts with many bus connections. Better routability of high-density layouts with a "random" connection distribution however can be achieved using smaller segment fit values.

Mirroring Mode

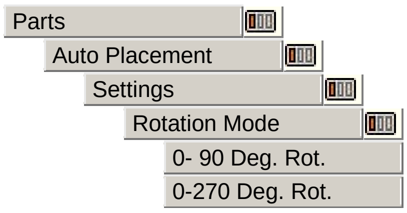
The part mirroring mode has effect on the placement of SMD parts only. With the No SMD Mirroring default option, SMD parts can only be placed on the PCB part side. SMD Mirroring is used to allow unrestricted placement of SMD parts on both the part side and the solder side. The SMD 2 Pin Mirroring option restricts automatic part mirroring to 2-pin SMDs only, thus allowing for solder side placement of small parts such as block capacitors whilst placement of SMDs with more than 2 pins is forced onto the PCB part side. Use the



function to select the desired SMD mirroring option.

Rotation Mode

The part rotation mode is used to apply either unrestricted or restricted automatic part rotation in 90 degree steps. With the 0-270 Degree Rotation default option all parts can be placed at arbitrary 90 degree rotation steps. Option 0-90 Degree Rotation can be used to restrict part rotation, i.e., to allow no rotation or 90 degree left rotation only. Restricted part rotation can be used to simplify subsequent insertion processes which will also result in a reduction of CPU time required by the placement algorithms. Use the



function to select the desired part rotation mode.

Part Size, Part Expansion, Block Capacitors, Placement Grid

The parts are placed onto the currently selected input grid (to be set using the Input Grid function from the View menu). The placement area required for each part (i.e., the part size) emerges from the element boundaries of the corresponding part macro. This should be taken into consideration at the definition of element boundaries when creating part macro symbols. It is recommended to refrain from defining unnecessarily large part macro element boundaries to avoid redundant placement area occupancy.

A generally applicable part expansion value can optionally be defined in order to ensure minimum clearance between adjacent parts (on low-density layouts). Use the

Parts
Auto Placement
Settings
Part Expansion
Expansion Value (0.00mm) ?

function to set the desired part expansion value. On default the part expansion value is set to 0.0mm (i.e., no part expansion is applied). Note that the specified part expansion value with no regard to different part macro type dimensions is equally applied on each part macro type except for block capacitors. Block capacitors are automatically recognized and will be placed as close as possible to the ICs to be supplied. The Full Autoplacer preferably places block capacitors either on top or on the right-hand side of the integrated circuits, according to the ICs orientation.

The Full Autoplacer features a function for automatically reducing part expansion settings by 25 percent steps until either complete placement is achieved or part expansion is reduced to zero.

Rip-up and Retry Passes

The initial placement algorithms are designed to consider only connections to/between already placed parts. Intermediate placement results might show up with alternate positions to be more optimum for a series of previously autoplaced parts. For this reason, rip-up and retry passes are activated for replacing already placed parts in order to find a more optimum placement. Note that preplaced unfixed parts are also processed by the rip-up and retry passes. It is strongly recommended to fix those preplaced parts which must not be replaced. Parts can be fixed using **Layout Editor** group functions. Use the

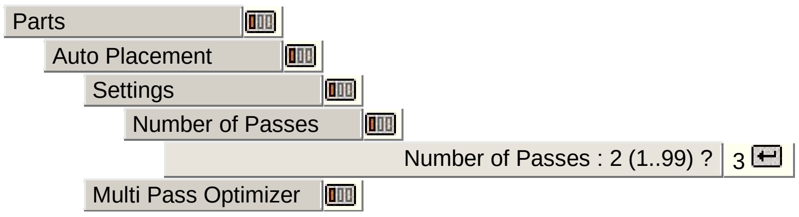
Parts
Auto Placement
Settings
Number of Retries
Number of Retries [0,9] (2) ?

function to specify the number of retry passes to be applied during the autoplacement process. Two retry passes are used on default; i.e., one rip-up and retry pass is applied after placing half of the unplaced parts and one rip-up and retry pass is applied at the end of the autoplacement process. It is recommended to refrain from specifying too many retry passes since this not only increases CPU time requirements but also can result in a deterioration of intermediate placement results since small parts such as block capacitors might not be correctly replaced with their clusters.

4.4.4 Placement Optimization

The placement optimization features of the BAE layout system include functions for automatic part and pin/gate swap. The part swap facility mutually exchanges identical components at their insertion position in order to minimize unroutes lengths. The pin/gate swap facility analogously performs an iterative exchange of gates and/or pins and pin groups, where gates or groups can also be swapped between different parts. The admissibility of pin/gate swaps is fairly controlled with appropriate library definitions. Either single pass or multi pass optimization can be applied. Different swap methods can be selected through the P/G Swap Method function, with the default Both Swap Methods option to perform both component and pin/gate swap, Only Part Swap to switch off pin/gate swap and Only Pin/Gate Swap to switch off component swap. Applying placement optimization usually causes a significant simplification of the routing problem, thus considerably reducing runtime at the subsequent Autorouting process.

Use the following commands to perform a triple-pass placement optimization with both component and pin/gate swaps:



```
Parts
  Auto Placement
    Settings
      Number of Passes
        Number of Passes : 2 (1..99) ? 3
    Multi Pass Optimizer
```

During optimization a status line message instantly reports the number of parts processed by the current optimizer pass. Placement optimization can be interrupted at any time by pressing an arbitrary key on the keyboard.

Wait until all optimizer passes are completed and use the Undo function to estimate how placement optimization did change the complexity of the "ratsnest" (unroutes).

Fixed parts are excluded from placement optimization. It is strongly recommended to fix critical parts *before* running the placement optimization. Critical parts are those which must not be replaced (e.g., plugs, switches, LEDs, etc.) or where pin/gate swaps are not allowed (e.g., relays, multi-opamps, etc.). Parts can be fixed using **Layout Editor** group functions. Please note that automatic pin/gate swap should only be performed if all of the swap definitions stored with the logical library are permitted. Select the Only Part Swap option from the P/G Swap Method function to switch off automatic pin/gate swap if you are in doubt about the correctness of swap definitions. Consider please that usually no pin/gate swap is allowed for plugs. Swaps for parts with special attribute values (e.g., resistor networks with attribute **`$val`** must be defined internal to prevent the system from swapping gates between parts with different values. See chapter 7.11 of this manual for a description of the **LOGLIB** utility program and how to introduce pin/gate swap definitions to the logical library.

4.5 Autorouter

With real jobs you should make sure that all of the required power layers are defined (use the Set Power Layers function from the **Layout Editor** Settings menu), the top signal layer setting is correct (use the Set Top Layer function from the **Layout Editor** Settings menu) and pre-routed critical traces such as power supply are fixed, before starting the autorouting process. After finishing the autorouting, you should always run a Batch DRC from the **Layout Editor** before generating manufacturing data with the **CAM Processor**.

The user interface of the **Bartels Autorouter** is similar to the one of the **Layout Editor**. The **Bartels Autorouter** provides standard autorouting functions (full and initial routing, rip-up and retry routing, optimizer) and a series of enhanced placement and routing features such as automatic pre-placement with placement optimization, single net routing, net-group routing, component routing, area and/or block routing, mixed-grid routing, selective component and pin/gate swap during rip-up routing, etc. The **Autorouter** version provided with the **BAE HighEnd** system provides powerful autorouting technologies based on patented neural network technology. The **BAE HighEnd Autorouter** supports skilled analogue signal routing, automatic microwave structure generation, grid-less object-orientated routing with automatic placement optimization, etc. The **BAE HighEnd Autorouter** also provides features for routing problem recognition and/or classification and for learning and automatically applying problem-adapted routing strategies and/or rules.

4.5.1 Starting the Autorouter

The Autorouter function from the **Layout Editor** File menu is used to start the **Autorouter**. After activating this function, the system automatically saves the currently loaded layout and the **Autorouter** program module starts. The **Autorouter** automatically re-loads any previously loaded layout. The layout requires a board outline definition (to be defined with the Add Board Outline function from the **Layout Editor** Areas menu) and a valid via assignment list (use function Select Via from the **Layout Editor** Parts menu). Before starting any autorouter procedure, all net list parts must be placed with correct package types inside the board outline (no part or pin neither any pre-routed trace or via can be placed outside the board outline).

The following error messages might be issued when activating an **Autorouter** procedure:

Board outline not defined!	
Cause:	PCB board outline definition is missing
Fix:	**Layout Editor** - Areas - Add Board Outline - ...
Default via not defined!	
Cause:	no via assignment
Fix:	**Layout Editor** - Parts - Select Via(s) - ...
Parts not found or of different type!	
Cause:	not all net list parts are placed on the layout or there are net list parts placed with wrong package types
Fix:	use **Layout Editor** - Parts - Delete Update to remove net list parts with wrong package types; then use manual or interactive placement functions to complete the part placement
Pin out of border (<partname>) !	
Cause:	net list part(s) and/or net list part pin(s) placed outside board outline
Fix:	correct part placement
Via out of border!	
Cause:	pre-placed, fixed via(s) placed outside board outline
Fix:	unfix and/or delete via(s) outside board outline
Trace out of border!	
Cause:	pre-routed, fixed trace(s) placed outside board outline
Fix:	unfix and/or delete trace(s) outside board outline
Invalid via padstack (cannot use it)!	
Cause:	via list contains invalid via definition(s)
Fix:	correct via definition(s); there must be at least one via for connecting all signal layers, and each via must contain a drill hole and pads for at least two adjacent signal layers
Double defined padstack!	
Cause:	layout contains ambiguous library part definition(s) such as part(s) with more than one padstack placed at the same pin position
Fix:	correct library part definition(s)
Double defined pad!	
Cause:	layout contains ambiguous library pad and/or padstack definition(s) such as pad(s) with more than one copper/connection area or padstack(s) with more than one pad placed on same layer
Fix:	correct library pad and/or padstack definition(s)

Other error messages might be issued if the layout contains short circuits caused by pre-routed fixed traces.

4.5.2 Autorouter Main Menu

The **Autorouter** standard/sidmenu user interface provides a menu area on the right side, consisting of the main menu on top and the currently active menu below that main menu. After entering the **Autorouter** the menu cursor points to the Load Element function.

The Windows and Motif versions of the **Autorouter** can optionally be operated with a pull-down menu user interface providing a horizontally arranged main menu bar on top. The **`WINMENUMODE`** command of the **BSETUP** utility program is used to switch between **`SIDEMENU`** and **`PULLDOWN`** Windows/Motif menu configurations (see chapter 7.2 for more details).

The following main menu is always available whilst processing layouts with the **Autorouter**:

Undo, Redo
Display
Preplacement
Autorouter
Interaction
Options
Control
Strategy
Parameter
Utilities

The functions provided with the Undo, Redo menu allow to use the **Autorouter** without fear of causing damage. Up to twenty commands can be reversed or undone using Undo and then reprocessed with Redo. This is true even for complex processing such as complete autorouting passes or **User Language** program execution. Undo, Redo ensures data security and provides a powerful feature for estimating design alternatives.

The View or Display menu can either be activated by selecting the corresponding main menu item or by pressing the middle mouse button. Activation through the middle mouse button is even possible whilst performing a graphical manipulation such as selecting a net or component for routing. The View or Display menu provides useful functions for changing display options such as zoom window, zoom scale, input and/or display grids, grid and/or angle lock, color settings, etc.

The Preplacement menu of the **Autorouter** is equal to the Autoplacement menu of the Layout Editor (see chapters 4.4.3 and 4.4.4). The **Autorouter** provides the same initial placement and placement optimization functions as already known from the **Layout Editor**, allowing for initial placement and placement optimization to be applied in the **Autorouter** without having to go back to the **Layout Editor**.

The Autorouter menu provides the functions for activating autorouting procedures such as Full Autorouter, Optimizer, Load Layout, Batch Setup and Batch Start. The Batch Start function is used to run a series of router passes previously defined with the Batch Setup function.

The Interaction menu provides the functions for activating special routing operations such as single net routing, net group routing, component routing, area/block routing, etc.

The Options menu is used to set fundamental **Autorouter** options for subsequent router passes. These parameters define the design rules and technology requirements to be considered by the **Autorouter** (signal layer count, layer assignment, routing grid with half-grid routing option, standard trace width, standard minimum distance, maximum via count, via grid, on-grid or off-grid trace bending, pin contact mode). The parameters defined from the Options menu are stored with the layout. Changes to basic routing option parameters (signal layer count, routing grid, toggle half-grid option, standard trace width, standard minimum distance, pin contact mode) cause an automatic router restart (with discard of current routing results) when re-entering some standard autorouting process such as Full Autorouter, initial routing, SMD fan out routing, rip-up routing, optimizing and Load Traces.

The Control menu provides basic functions for controlling the routing process such as setting the optimizer passes count, activate/deactivate multi-net pattern recognition during rip-up and/or optimization, set the persistence of the rip-up router, activate automatic SMD fan out routing and turn on or turn off automatic security copy.

The Strategy menu provides functions for setting the strategy parameters and heuristic cost factors to be used by subsequent router and optimizer passes (optimizer routing direction, via costs, pin channel costs, counter direction costs, direction change costs, packing costs, dynamic density cost factor, bus bending costs, rip-up distance costs, trace crossing costs, diagonal routing costs, off-grid routing costs).

The Parameter menu provides functions for selecting the layout library, setting the coordinate display mode, selecting the Mincon function for the airline display and activating the automatic design data backup feature.

The Utilities menu provides functions for exiting BAE, returning to the BAE main shell, calling the **Layout Editor** and starting **User Language** programs. This menu also provides important file management functions such as load and save layout elements or list DDB file contents.

4.5.3 Customized Autorouter User Interface

The BAE software comes with **User Language** programs for activating a modified **Autorouter** user interface with many additional functions (startups, toolbars, menu assignments, key bindings, etc.). The **BAE_ST** **User Language** program is automatically started when entering the **Autorouter** **BAE_ST** calls the **UIFSETUP** **User Language** program which activates predefined **Autorouter** menu assignments and key bindings. Menu assignments and key bindings can be changed by modifying and re-compiling the **UIFSETUP** source code. The **HLPKEYS** **User Language** program is used to list the current key bindings. With the predefined menu assignments of **UIFSETUP** activated, **HLPKEYS** can be called from the Key Bindings function of the Help menu. Menu assignments and key bindings can be listed with the **UIFDUMP** **User Language** program. The **UIFRESET** **User Language** program can be used to reset all currently defined menu assignments and key bindings. **UIFSETUP**, **UIFDUMP** and **UIFRESET** can also be called from the menu of the **KEYPROG** **User Language** program which provides additional facilities for online key programming and **User Language** program help info management.

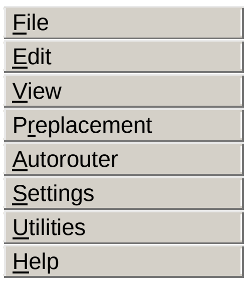
The Windows and Motif pull down menu user interfaces of the **Autorouter** provide facilities for cascading submenu definitions. I.e., submenus can be attached to other menu items. The **UIFSETUP** **User Language** program configures cascading submenus for the pulldown menu interfaces of the Windows/Motif **Autorouter** modules. This allows for easy submenu function location (and activation) without having to activate (and probably cancel) submenus. The function repeat facility provided through the right mouse button supports cascading menus to simplify repeated submenu function calls.

The following Windows/Motif parameter setup dialogs are implemented for the **Autorouter**:

- Settings - Settings: General Autorouter Parameters
- View - Settings: Display Parameters
- Preplacement - Settings: Automatic Placement Parameters
- Autorouter - Options: Autorouting Options
- Autorouter - Control Parameters: Autorouting Control Parameters
- Autorouter - Strategy: Autorouting Strategy Parameters
- Autorouter - Batch Setup: Autorouting Batch Setup

The **UIFSETUP** **User Language** program replaces the parameter setup functions of the Windows and Motif pulldown menus with the above menu functions for activating the corresponding parameter setup dialogs.

When using pulldown menus under Windows and Motif, the **UIFSETUP** **User Language** program configures the following modified **Autorouter** main menu with a series of additional functions and features:



4.5.4 In-built Autorouter System Features

Automatic Parameter Backup

The **Autorouter** provides an in-built feature for automatically saving important design and operational parameters with the currently processed SCM sheet and/or SCM library hierarchy level. The following parameters are stored to the current design file when activating the Save Element function:

- Autosave Time Interval
- Name of the currently loaded Layout Color Table
- Input Grid
- Display Grid
- Grid/Angle Lock
- Coordinate Display Mode
- Wide Line Draw Start Width
- Part Airline Display Mode
- Library File Name
- Mincon Function Type

Parameter sets are stored using the current layout element name. When loading a layout, the corresponding parameter set is automatically loaded and/or activated, thus providing a convenient way of activating a default parameter set suitable for processing the selected design.

Graphical Output and Status Displays

The current routing result is displayed graphically and through statistical readouts whilst routing is in progress. Graphic output and statistical readout during routing can be deactivated by pressing key d. The **`Router working. Press 'd' to activate display...`** message is displayed whilst router graphic output is deactivated. Pressing the 'd' key again reactivates the graphic output and the statistical readout. The routing process accelerates by up to 10 percent if graphic output is deactivated. A screen redraw is automatically performed after the routing process has finished.

The number of currently routed connections (compared to the total connection count) and the global via count are continuously reported in the status line whilst during Autorouting processes. Additionally, a routing pass info window for displaying internal routing procedure information is provided on the right side of the **Autorouter** user interface. This info window contains a status line for identifying the currently active router pass type (**`L`** - Load Layout, **`S`** - SMD Via Preplacement Pass, **`I`** - Initial Routing Pass, **`R`** - Rip-Up Pass, **`P`** - Optimizer Pattern Search Pass, **`O`** - Optimizer Pass) and the number **`n`** of processed elements in relation to the total number **`m`** of elements to be processed by the current router pass (display **`n/m`**). The status line ends with the current pass number **`c`** and the total number **`p`** of passes to be processed (display **`c/p`**). The values displayed with the router pass status line cannot be used for accurate total routing time predictions since the required time for completing the routing of each net and/or connection strongly depends on the current routing complexity.

The View menu is used to set parameters for controlling graphical output. The zoom functions (Zoom All, Zoom In, Zoom Out) are used to select the workspace to be displayed. Usually, one would use Zoom All to display the complete **Autorouter** work area (i.e. the workspace designated by the board outline). Zoom All is the default setting after starting the **Autorouter**. The functions for changing the zoom factor (Zoom In and Zoom Out) and the Redraw function can only be executed if layout data is already loaded.

The Change Colors function is used to change the current color setup. The Load Colors function is used to load a predefined color table. On default, the **Autorouter** uses the color table named **`standard`** (from the **`ged.dat`** system file of the BAE programs directory). It is recommended to use a color setup which displays all routing layers. When routing with blind and buried vias, it is also a good idea to change the color setup to distinguish different via types.

The Set Clipping function is used to set the width at which circuit traces are displayed with their true widths. All traces having a screen width greater than the clipping width are displayed with their true widths. All traces having a screen width less than the clipping width are displayed as center lines. The default clipping width value is 1.5mm.

The Potential Display function is used to display connections to active copper areas using either cross-shaped markers on connected pins (option Cross) or airlines connecting pins and copper area gravity points (option Box). On default the Cross option is used.

User Language

The **Bartels User Language Interpreter** is integrated to the **Autorouter**, i.e., **User Language** programs can be called from the **Autorouter**, and it is possible to implement any user-specific **Autorouter** function required such as status display, parameter setup, reports and test functions, CAD/CAM input/output functions, automatic or semi-automatic placement and/or routing functions, customer-specific batch procedures, etc.

The **Autorouter** provides both explicit and implicit **User Language** program call facilities. **User Language** programs can be started with explicit program name specification using the Run User Script function from the File menu (empty string or question-mark (**?**) input to the program name query activates a **User Language** program selection menu).

User Language programs can also be called by simply pressing special keys of the keyboard. This method of implicit **User Language** program call is supported at any time unless another interactive keyboard input request is currently pending. The name of the **User Language** program to be called is automatically derived from the pressed key, i.e. pressing a standard and/or function key triggers the activation of a **User Language** program with a corresponding name such as **ar_1** for digit key 1, **ar_r** for standard key r, **ar_#** for standard key #, **ar_f1** for function key F1, **ar_f2** for function key F2, etc.

The **Autorouter** environment also features event-driven **User Language** program calls, where **User Language** programs with predefined names are automatically started at certain events and/or operations such as **AR_ST** at **Autorouter** module startup, **AR_LOAD** after loading a design element, **AR_SAVE** before saving a design element, **AR_TOOL** when selecting a toolbar item and **AR_ZOOM** when changing the zoom factor. The module startup **User Language** program call method is most useful for automatic system parameter setup as well as for key programming and menu assignments. The element save and load program call methods can be used to save and restore element-specific parameters such as the zoom area, color setup, etc. The toolbar selection event must be used to start **User Language** programs which are linked to toolbar elements. The zoom event can be used to apply an update request to a design view management feature.

Bartels User Language also provides system functions for performing key programming, changing menu assignments and defining toolbars. These powerful features can be applied for user interface modifications. Please note that a large number of additional functions included with the **Autorouter** menu are implemented through the **User Language** programs delivered with the BAE software.

See the Bartels User Language Programmer's Guide for a detailed description of the **Bartels User Language** (chapter 4.2 lists all **User Language** programs provided with the BAE software).

Neural Rule System

A series of advanced **Bartels AutoEngineer** features affecting **Autorouter** behaviour and procedures are implemented through the **Neural Rule System**. See chapter 6.3.2 for the rule system applications provided with the PCB layout system.

4.5.5 Autorouter Options

The Options menu is used to define the design rules and technology requirements to be considered by the **Autorouter**. Option parameters can only be set *before* starting the Autorouting process (unless otherwise mentioned). On **Autorouter** security copies the current option parameter settings are stored to the currently processed job file. I.e., **Autorouter** options need not be redefined on subsequent **Autorouter** calls, unless parameter changes are really required for the layout to be automatically routed.

Routing Grid, Standard Trace Width, Standard Minimum Distance

The **Bartels AutoEngineer** works as a grid-based router unless the gridless router of the **Autorouter** (see below) is activated. The Routing Grid function from the Options menu is used to define the routing grid before starting the Autorouting process. Table 4-1 lists the selectable standard routing grids.

Table 4-1: Autorouter Grids

		Standard Trace Width		Standard Minimum Distance	
Routing Grid	**Via Offset**	**[mm]**	**[mil]**	**[mm]**	**[mil]**
1/20 Inch std.	-	0.37	14.6	0.29	11.4
1/40 Inch std.	x	0.32	12.6	0.29	11.4
1/50 Inch std.	-	0.25	9.9	0.23	9.1
1/60 Inch std.	x	0.21	8.3	0.19	7.5
1/80 Inch std.	x	0.16	6.3	0.13	5.1
1/100 Inch std.	-	0.13	5.1	0.10	3.9
1/40 Inch no ofs.	-	0.32	2.6	0.29	1.4
1/60 Inch no ofs.	-	0.21	8.3	0.19	7.5
1/80 Inch no ofs.	-	0.16	6.3	0.13	5.1
1/100 Inch w. ofs.	x	0.13	5.1	0.10	3.9

The default routing grid setting at the first **Autorouter** call for a particular layout is 1/40 Inch std.. Each routing grid change is stored with the layout and also sets the standard trace width and minimum clearance distance values to the default settings for that grid (see table 4-1 for the grid-specific default values). Both the standard trace width and the standard minimum distance can be changed *after* specifying a new routing grid (use functions Trace Width and Minimum Distance, respectively). Note however that the sum of trace width + minimum distance must not exceed the current routing grid; otherwise the **Autorouter** issues an `Incompatible options selected!` error message when starting the routing process (note warnings such as `Accepted for smaller default trace width!` and/or `Accepted for smaller minimum distance!`, and modify trace width and/or minimum distance until either of the messages `Minimum distance value accepted!` or `New default trace width accepted!` is issued). You should set the trace width and minimum distance values to 0.2mm each if you intend to route an SMD layout with 1/40 inch routing grid, since this allows the **Autorouter** to use the pin channels of SO packages.

Note that the 1/60 inch routing grid (i.e., the complete routing matrix) is internally shifted by 1/120 inch, thus enabling the **Autorouter** to place two traces between adjacent pins of DIL packages which are placed on 1/10 or 1/20 inch grid.

The Other Grid option of the Routing Grid function can be used to specify arbitrary routing grids such as some metric grid for special pin grids. Arbitrary routing grid settings will set both the standard trace width and the minimum distance to half the value of the routing grid. I.e., a routing grid specification of e.g., 1.1mm will set the trace width and the minimum distance to 0.55mm each.

The built-in off-grid recognition of the **Bartels AutoEngineer** allows for off-grid placement of pins and pre-routed traces. I.e., the **Autorouter** is able to connect objects which are not placed on the routing grid (e.g., plug pins on metric grid). Note however that on-grid items make the job much easier for the **Autorouter** since off-grid routing is quite time-consuming and could even prevent the **Autorouter** from connecting certain items in a simple way. It is recommended to use a reasonable grid for part placement to avoid off-grid pin placement and to enable pin channel routing for better routing results.

Please note that smaller routing grids result in quadratic growth of memory requirements for the routing matrix, and that CPU time usage during the Autorouting process rises even more dramatically (due to a non-polynomial growth of the number of possible routing solutions). It is recommended to refrain from selecting unnecessary small routing grids.

Some routing grid options support routing with via offset (see table 4-1). No via offset means that vias are placed in-line with the traces (i.e., on the routing grid). Routing with via offset means that vias can be shifted by half the routing grid at their placement. The choice to offset or not must be calculated by the user taking into consideration the routing grid, trace width, clearance distance and via size. Routing with via offset may well allow to use channels which would otherwise be occupied by in-line placed vias on adjacent grid channels. This can have a significant negative effect on the routing success (see figure 4-5 for an illustration). The via offsets are calculated for via diameters smaller than 1mm. It is recommended to refrain from routing with via offset when using larger vias since this could cause poor routing results.

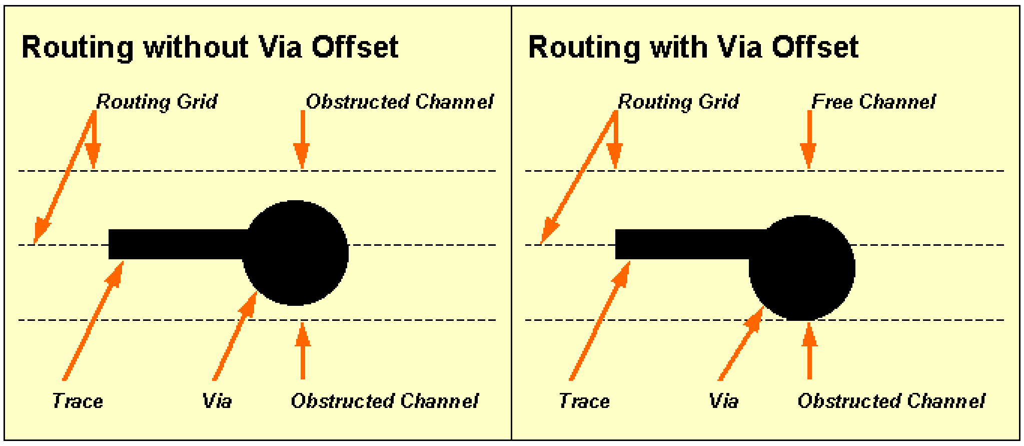


Figure 4-5: Routing with or without Via Offset

The standard trace width is the track width to be used for routing the connections. The standard trace width applies for all nets except for those where certain net attributes are defined. If a **`routwidth`** net attribute value is set for a particular net, then the entire net is routed with that routing width. If a **`powwidth`** value is set for a particular net, then all library-defined power supply pins of that net are connected with that power width. The standard minimum distance sets the default minimum clearance distance to be considered by the **Autorouter** when routing the connections. The standard minimum distance applies for all nets, except for those where non-default minimum distance values are assigned with the **`mindist`** net attribute. The **Autorouter** automatically performs high priority processing of nets with net attribute values set. The **`priority`** net attribute can be used to specify explicit net-specific routing priorities. See the **LOGLIB** utility program description in chapter 7.11 of this manual for more details on how to use net attributes for controlling the routing process.

Signal Layer Count and Layer Assignment

The Signal Layer Count function from the Options menu is used to specify the number of signal layers to be simultaneously routed by the **Autorouter**. The signal layer count can range from 2 to 12. The default signal layer count for a particular layout corresponds with the top layer setting of that layout.

The Layer Assignment function is used to set the preferred routing direction (horizontal, vertical or all directions) for each routing layer. With the Layer Assignment function it is also possible to define trace keepout layers (i.e., layers where routing is prohibited) or to remove layers from the routing layer list (which will decrement the signal layer count). The default layer assignments are horizontal for layer 1, vertical for layer 2, horizontal for layer 3, vertical for layer 4, etc. Changing the routing layer count with function Signal Layer Count resets the layer assignments.

Routing layer numbering starts with signal layer 1 (solder side). The **Autorouter** simultaneously routes all routing layers. Single-layer boards can be routed with routing layer count 2 and routing layer 1 defined as prohibited layer.

All objects placed on prohibited layers are considered by the **Autorouter**. I.e., via keepout areas on a double-sided layout can be defined by placing keepout areas on signal layer 3 of the layout, setting the routing layer count to 3, and defining routing layer 3 to be prohibited. The **Autorouter** would then use signal layers 1 and 2 for routing the traces and refrain from placing vias at positions where via pads would intersect with keepout areas on layer 3.

The layer assignments (except for the signal layer count) can be changed between different router passes, i.e., without the need of restarting the complete routing process. Note however that restrictions introduced with layer assignment changes will never cause a deterioration of the current routing result at subsequent router passes. I.e., the **Autorouter** will try to but not necessarily remove all previously routed traces from prohibited layers, if those layers were available for routing before.

Changing layer assignments between different router passes introduces highest flexibility at the routing of complex PCB technologies. Best success at the routing of certain multilayer SMT boards is often achieved when using outside layers (solder and component side) with highest priority for connecting SMD pins to inside signal layers. This can be accomplished with SMD Via Pre-Place passes to be subsequently activated with dynamically adapted layer assignments and increasing routing channel widths (use the Batch Setup and Batch Start functions to apply single router passes). After completing the SMD Via Pre-Placement some Initial passes (with via and/or channel width restrictions) can be applied to route the signal inside layers only (prohibit the outside layers). Finally the outside layers can be released again for subsequent router passes such as Complete Initial pass, Optimizer, rip-up routing, etc.

The **Bartels AutoEngineer** is capable of routing up to 12 power layers in addition to the signal layers. I.e., the **Bartels AutoEngineer** can rout multilayer designs with a total of up to 24 layers. The power layer routing algorithms are equipped with intelligent power plane detection features for split power plane routing. When routing power layers, the **Autorouter** correctly connects SMD pins to power layers and/or active copper areas defined on such power layers.

Maximum Via Count

The Maximum Via Count function is used to set the maximum number of vias per circuit trace. The default maximum via count is 20. A maximum via count of 0 forces the **Autorouter** to rout the layout without vias. The maximum via count can be changed between different router passes, i.e., without the need of restarting the complete routing process. Note however that restrictions introduced with maximum via count changes will never cause a deterioration of the current routing result at subsequent router passes. I.e., the **Autorouter** will try to but not necessarily remove all previously placed vias.

Via Grid

The Via Grid function is used to set the grid for placing vias to 1/10 or 1/20 inch. The No Grid option is used on default, thus allowing unrestricted via placement according to the current Routing Grid setting (see above). The via grid can be changed between different router passes without the need of restarting the complete routing process. Note however that restrictions introduced with the via grid changes will never cause a deterioration of the current routing result at subsequent router passes, i.e., the **Autorouter** will try to but not necessarily replace previously placed vias.

Sub-Grid Routing

The Routing Sub-Grid function is used to activate half-grid routing (option Half Grid 1:2). On default half-grid routing is deactivated (option Standard 1:1). Half-grid routing means that the **Autorouter** can alternatively use a routing grid shifted by half of the selected routing grid, e.g., half-grid routing with 1/40 inch routing grid can also utilize the 1/80 inch grid routing grid. The current standard trace width and minimum distance settings are not affected by the sub-grid routing option. Half-grid routing yields better use of regions nearby off-grid placed objects such as pin channels at off-grid placed parts, thus considerably increasing routability of dense layouts.

Gridless Routing Option

A rule-driven gridless router is integrated to the **Autorouter**. Gridless routing is deactivated on default (option Gridbased Routing of the Gridless Routing function from the Options menu). The rule-driven gridless router can be activated through the Gridless Routing option from the Gridless Routing function. The gridless router performs *selective* gridless routing, i.e., gridless routing is only applied locally where this yields better results in terms of routability and manufacturing optimization. With gridless routing activated, there are much more options for using off-grid pin channels, which might significantly increase the routability of complex layouts. Gridless routing generates straighter connections to off-grid placed pins, thus preventing from blocking adjacent pin channels and also performing more optimization for manufacturing.

The Gridless Routing mode allows for traces to leave the routing grid under virtually any condition and for minimum distance routing between off-grid pins, thus achieving significantly better routing results especially for dense SMT boards.

The Gridless Pins/Vias mode straightens connections to gridless placed pins and vias only locally and resumes and/or applies gridbased routing for normal traces. This mode is not capable of routing between offgrid placed pins where this otherwise would be possible in Gridless Routing mode.

Warning

Note that gridless routing requires more main memory and computing power due to additional data structures (gridless priority tree) to be maintained during the routing process.

Traces On-Grid/Off-Grid

The Traces On-Grid function is used to set the trace corner cutting mode. On default trace corner cutting falls on half grid points (option Traces On-Grid Off). The Traces On-Grid On option forces trace corner angles onto the routing grid only, thus ensuring correct spacing between diagonal trace segments and pad corners. Figure 4-6 illustrates the effects of off-grid and on-grid trace corner cutting.

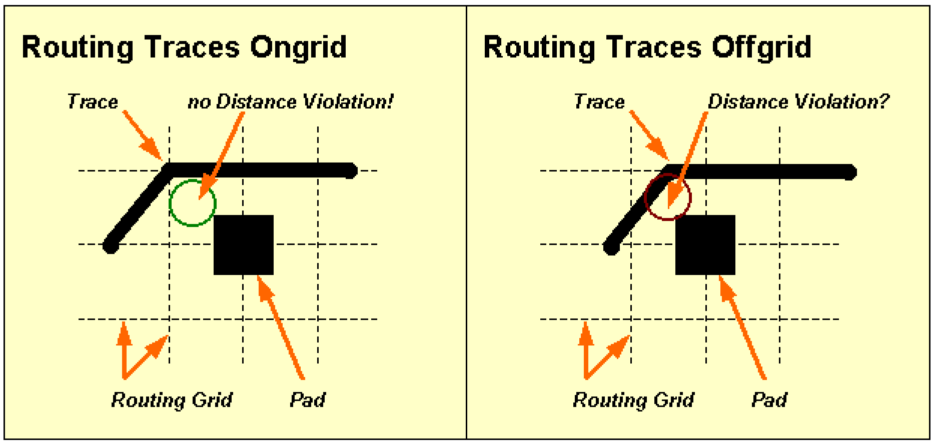


Figure 4-6: Routing Traces Ongrid/Offgrid

Pin Contact Mode

The Pin Contact Mode function is used to allow (option Use Pin Corners, default) or avoid (option Lock Pin Corners) pin corner routing. This feature works on approximately rectangle-shaped pads and controls whether traces can exit such pads at 45 degree angles or not. Routing with pin corner obstruction can produce better looking layouts, but could also impede 100% routing. Note that connecting pins with a size approximately equal to or smaller than the trace width could fail. The Lock Pin Corners option should not be used with designs that include thick traces. Note also that bus routing can produce unpredictable results when switched to pin corner obstruction.

4.5.6 Autorouter Control

The Control menu provides a series of functions for controlling the routing process.

Optimizer Passes

The Optimizer Passes function is used to set the number of Optimizer routing passes to be automatically activated by the Full Autorouter function after obtaining a 100% routing. The Optimizer passes count can range from 0 to 99; 2 Optimizer passes are activated on default.

Router Cleanup, Optimizer Cleanup

The Router Cleanup function is used to activate (default option Rip-Up Cleanup On) or deactivate (option Rip-Up Cleanup Off) cleanup passes during rip-up routing. The Optimizer Cleanup function is used to activate (default option Optimizer Cleanup On) or deactivate (option Optimizer Cleanup Off) cleanup passes during optimization. When running cleanup passes (note message `Pattern Search` issued by the routing progress report), the **Autorouter** makes use of a unique pattern search recognition algorithm for identifying disturbing traces during rip-up and cross-net optimization. The **Autorouter** is able to select and remove disturbing traces during rip-up and perform cross-net changes during optimization. Cleanup during cross-net optimization requires more computing time, but dramatically reduces via counts and thus is also called if the rip-up router (temporarily) fails to find an acceptable solution. It is recommended to refrain from turning off cleanup when running rip-up on complex designs or optimizing dense boards. Turning cleanup off results in sequential processing of the connections and can produce contenting results when running final Optimizer passes on certain layouts (however, more passes are then required for pushing and/or straightening trace bunches).

Rip-Up Trees, Rip-Up Depth, Rip-Up Retries

The Rip-Up Trees function sets the maximum number of traces allowed to be simultaneously ripped up per rip-up cycle. The rip-up trees number is set to 2 on default, and can range from 1 to 9. The Rip-Up Depth function is used to control the persistence of the rip-up process. A high value will result in higher persistence. The rip-up depth value can range from 1 to 999, and is set to 50 on default. The Rip-Up Retries function sets the maximum number of rip-up retries for routing a particular trace, thus defining the local rip-up router intensity. The rip-up retries number is set to 2 on default, and can range from 0 to 99. Higher rip-up control parameters increase the persistence and intensity of the rip-up routing process, and thus can be used for special problems such as completing 100% routing without intermediate optimizer passes or for routability check.

Once 99.5% routing completion is achieved, the Rip-Up parameters are automatically increased to Rip-Up Trees 6, Rip-Up Level 200 and Rip-Up Retries 10, unless higher values are already set. This helps to avoid the time-consuming Optimizer cleanup passes between Rip-Up passes if only a few open connections are left.

SMD Via Pre-Place

The SMD Via Pre-Place function is used to activate (option Via Pre-Place On) or deactivate (default option Via Pre-Place Off) the initial routing algorithm for connecting SMD pins to signal inside layers. With SMD via pre-placement activated the Full Autorouter function will start with the SMD via pre-place initial routing pass. The SMD via pre-placer will - as far as possible and/or meaningful - generate short trace connections to vias for those SMD pins to be wired. At SMD fanout routing, routing directions result from the shapes and positions of the corresponding SMD pads; i.e., the SMD fanout routing algorithm ignores layer-specific routing direction preferences to refrain from blocking PLCC/SMD pin channels and or PLCC/SMD pins. The SMD via pre-routing pass is intended for preventing the **Autorouter** from extensively using the SMD outside layers at an early stage of the routing process, thus involving earlier 100% routing success in a wide range of SMT designs. The SMD via router does not (re-)rout SMD pins which are already connected to fixed traces, and redundant SMD via connections are later eliminated by the Optimizer.

Placement Optimization during Rip-Up Routing

A feature for performing placement optimizations is integrated to the rip-up router of the **Autorouter**. The Router P/G-Swap Control menu function can be used to activate (option Router Pinswap On) or deactivate (option Router Pinswap Off) placement optimization during rip-up routing. With Router Pinswap On, the rip-up router performs pin/gate/groups swaps to increase routability. This can significantly reduce the time required for achieving a 100% autorouting completion. Placement optimization is selective, i.e., swaps with a high potential to simplify the routing problem are carried with higher priority. Swap operations which cause a deterioration of the current routing result rejected by the router's backtracking process. Only unfixed parts are subject to swap operations, and the admissibility of any pin/gate swap is controlled through the corresponding logical library part definitions.

Pin/gate swaps create a backannotation request which triggers automatic Backannotation when loading SCM plans to the **Schematic Editor** (see chapter 2.7). Backannotation requests also prevent the **Packager** from forward annotation without confirmation (see chapter 3.2.3).

Security Copy

The Security Copy function is used to activate (default) or deactivate automatic security copy of intermediate routing results.

4.5.7 Autorouter Strategy

The Strategy menu provides functions for setting routing strategies such as via cost, pin channel cost, packing cost, bus bending cost, dynamic density cost, etc. These settings should be used with care. Only change one or two at a time. Poor routing results are often found to be caused by random strategy parameter settings. The default settings will work best in the vast majority of cases. It is strongly recommended to change strategy parameters in special cases only and to refrain from using extreme values.

There are strong mutual dependencies between different **Autorouter** strategy parameters. A high via cost value (for eliminating vias) will necessarily result in more ignorance of routing direction preferences, thus compensating the cost factor for keeping preferred routing directions. Note also that strategy parameters only define subordinate options for the Autorouting process, due to the fact that it is much more important to achieve a 100% routing result instead of, e.g., keeping preferred routing directions. I.e., some cost factors may be completely ignored during initial routing and rip-up and will only be considered by the Optimizer. Tabelle 4-2 provides an overview to all of the routing parameters which can be set from the Strategy menu.

Table 4-2: Autorouter Strategy Parameters

Strategy Parameter	Value Range	Default Value	Effect on Router	Effect on Optimizer
Optimize Direction	Normal Preferred Diagonal	Normal	-	x
Via Cost	2..20	10	x	x
Pin Channel Cost	0..10	3	x	-
Counter Dir. Cost	0..5	1	x	x
Dir. Change Cost	0..5	2	-	x
Packing Cost	0..5	1	x	-
Dyn. Density Cost	0..50	10	x	-
Bus Bending Cost	0..5	2	x	-
Distance-1 Cost	0..10	5	x	-
Distance-2 Cost	0..10	2	x	-
Trace Cross. Cost	2..100	20	x	x
Diagonal Cost	0..10	5	-	x
Off-Grid Cost	0..5	2	x	x
Prefered Grid	0..7	0	x	x
Anti-Prefered Grid Cost	0..10	1	x	x
Outside Net Type Area Cost (**BAE HighEnd**)	0..5	1	x	x

Optimize Direction

The Optimizer Direction function is used to designate the Optimizer strategy. The Normal default option causes the Optimizer to ignore layer-specific preferred routing directions in order to obtain the greatest reduction of vias. The Preferred option causes the Optimizer to consider layer-specific preferred routing directions, which could increase the number of vias. The Diagonal option causes the Optimizer to prefer diagonal (45 degree) routing where appropriate.

Via Cost

The Via Cost setting is used by the Router and the Optimizer. A high via cost factor results in fewer vias but more complex circuit traces. A low via cost factor permits more vias within the restraint of the maximum via count (see above). The via cost value can range from 2 to 20; the default value is 10.

Pin Channel Cost

The Pin Channel Cost setting is used by the Router only. A high pin channel cost factor results in infrequent use of pin channels. A low pin channel cost factor permits the frequent use of pin channels. Pin channels are the regions between adjacent part pins. The pin channel cost value can range from 2 to 20; the default value is 10.

Counter Direction Cost

The Counter Dir. Cost setting is used by the Router and the Optimizer. A high counter direction cost factor results in strict adherence to the layer-specific preferred routing directions. A low cost factor permits frequent variations from the preferred direction. The counter direction cost value can range from 0 to 5; the default value is 1.

Direction Change Cost

The Dir. Change Cost setting is used by the Optimizer only. A high direction change cost factor results in less circuit corners. A low direction change cost factor permits frequent changes in routing directions. The direction change cost value can range from 0 to 5; the default value is 2.

Packing Cost

The Packing Cost setting is used by the Router only. A high packing cost factor results in high bundling of circuit traces. A low packing cost factor will result in wider distribution of circuit traces. The packing cost value can range from 0 to 5; the default value is 1.

Dynamic Density Cost

The Dyn. Density Cost setting is used by the Router only. The dynamic density cost factor controls the global distribution of circuit traces over the entire layout. A high dynamic density cost factor results in a more even distribution of the circuit traces. A low cost factor gives more influence to routing costs. The dynamic density cost value can range from 0 to 50; the default value is 10.

Bus Bending Cost

The Bus Bending Cost setting is used by the Router only. The bus bending cost factor controls the bending of traces after passing a pin channel. A high bus bending cost factor results in high priority of bending. A low bus bending cost factor results in less bending. The bus bending cost value can range from 0 to 5; the default value is 2.

Rip-Up Distance Cost

The Distance-1 Cost and Distance-2 Cost settings are used by the Router during rip-up.

The rip-up distance-1 cost factor controls the use of channels left by ripped up traces in the near distance (0 to 1 grid point). A high distance-1 cost factor results in less use of these channels, thus forcing more local changes during rip-up and retry routing. The rip-up distance-1 cost value can range from 0 to 10; the default value is 5.

The rip-up distance-2 cost factor controls the use of channels left by ripped up traces in the far distance (2 grid points). A high distance-2 cost factor results in less use of these channels, thus forcing more global changes during rip-up and retry routing. The rip-up distance-2 cost value can range from 0 to 10; the default value is 2.

Trace Crossing Cost

The Trace Cross. Cost is used to set the trace transition cost factor, which is considered by the Router and the Optimizer to control cleanup pattern recognition during multi-net optimization. A high trace crossing cost factor allows for a more complex routing with more traces crossing each other, thus also producing more vias. A low trace crossing cost factor leads to increased (and more time-consuming) analysis during cross-net optimization, thus eliminating more vias. The trace crossing cost value can range from 2 to 100; the default value is 10.

Diagonal Routing Cost

The Optimizer considers the Diagonal Cost setting on routing layers where the Diagonal routing option is selected (see Optimize Direction function above). A high diagonal routing cost factor causes the Optimizer to use more diagonal routes. A low diagonal routing cost factor results in less diagonal routing. The diagonal cost value can range from 0 to 10; the default value is 5.

Off-Grid Routing Cost

The Off-Grid Cost setting is used by the Router and the Optimizer, and is considered when routing with the half-grid option (see above). A high off-grid routing cost factor results in less use of the sub-grid. A low off-grid routing cost factor permits frequent use of the sub-grid. The off-grid routing cost value can range from 0 to 5; the default value is 2.

Prefered Grid, Anti-Prefered Grid Cost

The Prefered Grid parameter is used for setting a prefered routing grid. Prefered Grid values range from 0 to 7; 0 switches the prefered grid of, higher values cause the router to pefer the use of every 2nd, 4th, 8th, 16th, 32nd, 64th or 128th grid point. The Anti-Prefered Grid Cost parameter (value range 1 to 10, default value 1) is used to set a cost factor for routing outside the prefered grid. Prefered grid routing spreads the routing density and takes only effect on layouts with (large) unoccupied areas. Prefered grid routing parameters should be set before the first **Autorouter** run. Later changes to these parameters require multiple time-consuming optimizer runs. Note also that high cost factors could cause detouring routes.

Outside Net Type Area Cost (BAE HighEnd)

The Outside Net Type Area Cost strategy parameter is only supported in **BAE HighEnd**. A high cost factor prevents nets from being routed outside their net-specific routing areas, a low cost factor allows for nets to be routed outside their routing areas more frequently. The net area cost factor can range from 0 to 5; the default value is 1.

4.5.8 Autorouter Functions

The **Autorouter** supports various routing algorithms and autorouter procedures such as Full Autorouter, initial routing, SMD fanout routing, rip-up/retry routing, Optimizer, re-entrant routing, etc. These routing procedures can be activated from the Autorouter menu (functions Full Autorouter, Optimizer, Load Layout, Batch Setup, Batch Start), the Control menu (SMD Via Pre-Place function) and the Interaction menu (Route Single Net, Route Single Part Place and Route). Activating an autorouting procedure *after* changing basic routing parameters such as signal layer count, routing grid, half-grid option, standard trace width, standard minimum distance, or pin contact mode discards the currently unfixed traces and vias and (re-)routes the layout with changed options.

The autorouting procedures in **BAE HighEnd** are some 30 percent faster than in **BAE Professional**. This performance boost is mainly achieved by optimized internal data structures (HighSpeed Kernel) in **BAE HighEnd**.

The current routing result is displayed graphically and through statistical readouts whilst routing is in progress. The routing process can be stopped at any time by pressing a key, causing the **Autorouter** to revert to the (currently best) routing result.

Single-Pass Initial Routing

The single-pass Initial Router performs signal trace routing considering preferred routing directions, a certain channel width and a maximum via count per two-point connection. The channel width is the maximum permitted deviance from preferred directions. The channel width is specified in routing grid steps. A zero channel width removes the restraints on deviating from the preferred direction, i.e., the entire board area is then released for routing. The maximum via count used by the Initial Router will never exceed the value set with the Maximum Via Count function from the Options menu (see above). The Initial Router places traces close together, thus using minimum space in order to leave more room for subsequent traces (trace hugging). The Initial Router also uses advanced techniques of copper sharing where appropriate. The Initial Router processes power layer connections and attributed nets (with non-default routing widths, minimum distance settings and routing priorities) with highest priority.

Complete Initial Routing

The Complete Initial Router automatically activates four Initial Router passes to rout all open connections which can be routed without rip-up and retry. With each Initial Router pass the channel width and the maximum via count is increased. The first Initial Router pass runs with maximum via count zero. The last Initial Router pass runs with channel width zero and a maximum via count according to the parameter set with the Maximum Via Count function from the Options menu (see above).

Rip-Up/Retry Router

The Rip-Up/Retry Router attempts to route all open connections until the board is completely routed. Connections which can be routed without rip-up are routed first. Then the Rip-Up/Retry Router selects and eliminates traces (rip-up) and re-routes them to create space for the unroutes. For this purpose, the router gathers information on dense board areas and increases the cost of routing in such areas. The Rip-Up/Retry Router is supported by a sophisticated array of heuristic strategy parameters. The cost factors can be dynamically adapted to the current routing problem, thus controlling the "price" of strategies such as via placement, routing against preferred directions, using pin channels, etc. It is strongly recommended to refrain from modifying these strategies, unless the routing success is not what would be expected. When changing cost factors, slight adjustments to a few can make significant improvements or make things much worse. The Rip-Up/Retry Router is guarded by a unique backtracking algorithm, which not only prevents from a deterioration of the result or a dead-lock during rip-up or optimization but also is able to exploit a new and/or better routing solution. The Rip-Up/Retry Router automatically activates intermediate Optimizer passes if a single rip-up pass fails to achieve 100% routing success.

Optimizer

The Optimizer function is used to start a single Optimizer pass. The Optimizer usually is applied after 100% routing to optimize the layout for manufacturing. The Optimizer eliminates redundant vias, smoothes traces and attempts to rout open connections. Channel width zero is used during optimization and the maximum via count is set to the same value as defined with the Maximum Via Count function from the Options menu (see above).

SMD Via Pre-Placement

SMD Via Pre-Place is a special initial routing algorithm for connecting SMD pins to signal inside layers. The SMD via pre-placer generates short trace connections to vias for those SMD pins to be wired. With SMD fanout routing passes, a channel width setting is required and routing directions result from the shapes and positions of the corresponding SMD pads. The SMD via pre-routing pass prevents the **Autorouter** from extensively using the SMD outside layers at an early stage of the routing process. This strategy achieves earlier 100% routing success in a wide range of SMT designs. Redundant via connections created by SMD Via Pre-Place are later eliminated by the Optimizer.

Full Autorouter

The Full Autorouter main menu function is used to start a complete Autorouting process including all router passes such as SMD Via Pre-Placement (optionally), complete initial routing, rip-up/retry routing (if necessary) and optimization. This is the standard procedure for performing complete routing of a board. The number of final Optimizer passes can be set with the Optimizer Passes function from the Control menu (see above). The SMD Via Pre-Placer will only be activated if the Via Pre-Place On option has been selected with the SMD Via Pre-Place function from the Control menu.

Batch Setup and Batch Start

The Batch Start function from the **Standard Autorouter** main menu is used to start a routing process with different user-defined router passes. The Batch Setup function is used to schedule up to ten of the following autorouting procedures to be subsequently processed with the Batch Start function:

Command	Routing Pass/Procedure
`L`	Load Traces
`F`	Full Autorouter
`I`	Single-Pass Initial Routing
`C`	Complete Initial Routing
`R`	Rip-Up/Retry Router
`O`	Optimizer
`S`	SMD Via Pre-Placement
`-`	none (remove router pass from batch)

Single-Pass Initial Routing pass requires a routing channel width specification and a maximum via count setting. SMD Via Pre-Placement requires a routing channel width specification. Optimizer requires the number of required Optimizer passes (up to 999).

Single Net Routing

The Route Single Net function from the **Autorouter** Interaction menu activates the single net router. The net and/or connection to be autorouted by the single net router can be selected by mouse-clicking some pin or pre-routed trace of the desired net. The single net router is commonly used for pre-routing power supply signals or critical nets with specific preferences for trace widths, clearance, layer assignments, etc.

Component Routing

The Route Single Part function from the Interaction menu can be used to rout mouse-selectable parts, i.e., to autoroute all nets connected to the selected component.

Fully Automated Placement and Routing

The Place and Route function from the Interaction can be used to activate the Full Autoplacer and Full Autorouter function sequence. thus allowing for fully automated layout placement and routing.

Net-specific Airline Display and Net Group Routing

The Mincon Function from the Settings menu provides options for controlling net-specific airline display. Nets which are faded-out from airline display are excluded from the autorouting process. Net-specific airline display settings can be changed between router passes. This allows for consecutive router passes to route different groups of nets such as busses of a certain circuit block or signals with certain attributes with specific routing options for preferred routing directions, maximum via counts, etc..

Area/Block Routing in BAE HighEnd

The **Autorouter** of **BAE HighEnd** considers net-specific routing areas to be defined as documentary areas on signal layers with rule assignments containing a **`net_type`** predicate for the net type. See the **`nettype.rul`** file in the **User Language** programs directory (**`baeulc`**) for net type rule definitions. Up to eight different routing area net types can be assigned to each layout. Routing areas and rule assignments (with the **GEDRULE** **User Language** program) must be defined in the **Layout Editor**. Within a routing area, the **Autorouter** can only route nets with a matching **`$nettype`** attribute. Nets without **`$nettype`** value or with a **`$nettype`** value different to the **`net_type`** assignment of a certain routing area must not be routed in that area unless they connect to pins inside the routing area. Nets of a certain net type are not restricted to net-specific routing areas, however, they are preferably routed in the corresponding routing areas. Routing outside the net-specific routing area can be controlled with the new Outside Net Type Area Cost Factor (0..5) option to be set through the Strategy menu.

4.5.9 Using the Autorouter

Starting the Autorouter

You should first start the **Layout Editor** and load the layout `board` from the `demo.ddb` DDB file. The Autorouter command from the **Layout Editor** File menu is used to call the **Autorouter**:

- File
  - Autorouter

Setting Autorouter Options

The following operations will show how to apply certain router passes for routing our example job using three signal layers. Use the following commands to set the signal layer count to 3:

- Options
  - Signal Layer Count
    - 3 Signal Layers

Use the following commands to select 1/40 inch routing grid without via offset, and set the standard trace width and the minimum distance to 0.3mm each:

- Options
  - Routing Grid
    - 1/40 Inch no ofs.
  - Trace Width
    - New Default Trace Width (0.32mm) ? 0.3
  - Minimum Distance
    - New Minimum Distance (0.29mm) ? 0.3

View, Layout Display

Use the following commands to change trace clipping to 0.1mm and activate the Zoom All function:

- V|iew
  - Set Clipping
    - Display Traces wider than (1.50mm) ? 0.1
  - Zoom All

Router Batch

Use the following commands to define a router batch process consisting of SMD Via Pre-Placement with channel width 4, Complete Initial Routing, and a single Optimizer pass:

- Autorouter
  - Batch Setup
    - 1: ---
      - New Batch Pass (L/F/I/S/C/R/O/-) ? s ↵
      - Initial Pass Channel Width ? 4 ↵
    - 2: ---
      - New Batch Pass (L/F/I/S/C/R/O/-) ? c ↵
    - 3: ---
      - New Batch Pass (L/F/I/S/C/R/O/-) ? o ↵
      - Optimizer Passes ? 1 ↵
    - No Change

A dash string input ("-") to the **New Batch Pass** prompt can be used to remove the selected router pass from the batch. The sequence of router passes to be processed by Batch Start corresponds with the sequence of router passes defined with Batch Setup. Use the Batch Start command to run all router passes previously defined with the Batch Setup function:

- Autorouter
  - Batch Start

Router batches can also be started with the Start button from the Batch Setup dialog without having to leave the Batch Setup dialog and subsequently calling the Batch Start function.

The routing process for our example starts with SMD Via Pre-Placement. Subsequently, a Complete Initial Routing pass is performed. The Initial Router will end up with 100% routing success. Finally, one Optimizer pass is activated. Each routing pass automatically saves its routing result. The following message is displayed in the status line after finishing the last pass:

```
Max. 53 of 53 Routes (Pins: 76, Vias: 5)
```

The layout is now completely routed with 5 vias placed by the **Autorouter**.

Optimizer

Use the following commands to run an additional Optimizer pass (without pattern recognition, but with high priority for considering preferred routing directions):

- Control
  - Optimizer Cleanup
    - Optimizer Cleanup Off
- Strategy
  - Optimizer Direction
    - Preferred
  - Counter Dir. Cost
    - Counter Direction Cost Factor : 1 (0..5) ? 4 ↵
- Autorouter
  - Optimizer

Note how the Optimizer pass routes traces according to preferred routing directions, albeit at the price of an increased via count of 13.

Exiting the Autorouter

You can exit the **Autorouter** using one of the Exit BAE, Main Menu or Layout Editor functions from the File menu.

Exit BAE exits to the operating system. Main Menu saves the currently processed layout and returns to the BAE main shell. Layout Editor saves the currently processed layout and returns to the **Layout Editor**.

Use the Layout Editor function from the File menu to return to the **Layout Editor**:

- File
- Layout Editor

Re-entrant Routing

Using three routing layers for our simple example appears to be rather wasteful. We want to route the layout again, now with two routing layers instead of three. We also want to apply Load Traces to keep as much of the current routing result as possible.

Use the following commands to switch back to the **Autorouter**:

- File
  - Autorouter

Use the following commands to reduce the signal layer count to 2 and reload the data produced by the preceding router pass:

- Options
  - Signal Layer Count
    - 2 Signal Layers
- Autorouter
  - Load Traces

The Load Traces function fails to load the traces previously routed on layer 3 since the signal layer count is now set to 2. I.e., the router progress message in the status line will indicate 9 open connections (max. 44 of 53 routes) and a via count of 8 (instead of 13) after finishing the Load Traces function.

Use the following commands to switch off preferred routing direction optimization and start the Full Autorouter to complete the routing:

- Strategy
  - Optimizer Direction
    - Normal
- Autorouter
  - Full Autorouter

The Full Autorouter completes the routing, now on two instead of three signal layers and with 6 vias. Use the following commands to return to the **Layout Editor**:

- File
  - Layout Editor

After correctly performing all operations of this chapter, the PCB layout example should look like the one shown in figure 4-7.

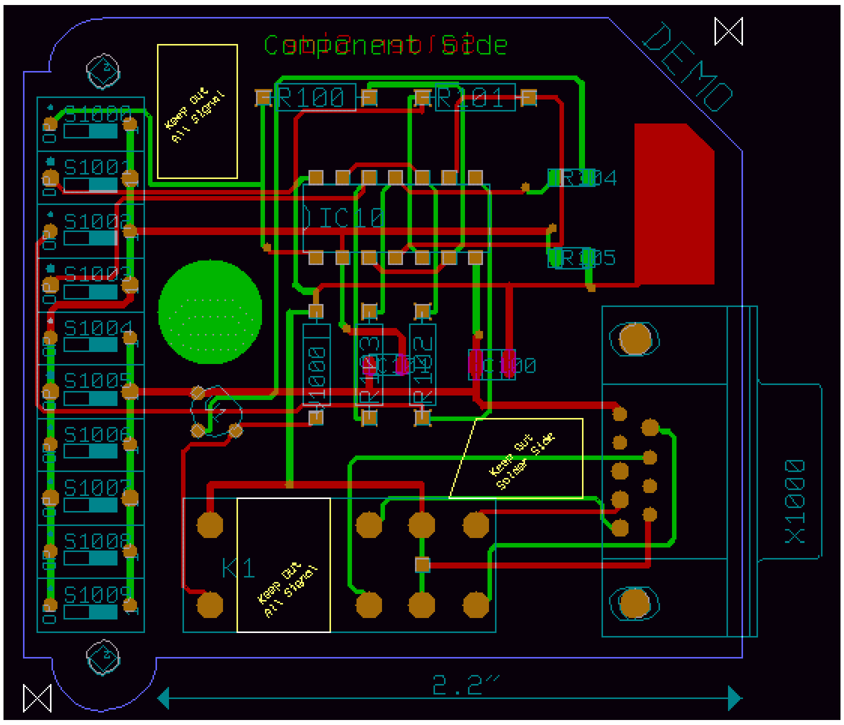


Figure 4-7: Layout after Autorouting

4.6 Special Layout Features

This section describes advanced features and special functions provided with the BAE PCB design system.

4.6.1 Batch Design Rule Check, Report

The Batch DRC function from the **Layout Editor** Utilities menu is used to run a complete design rule check on the currently loaded layout. It is strongly recommended to use Batch DRC before generating CAM data and/or passing CAM data to the PCB manufacturer to avoid design rule violations such as short-circuits, unrouted nets or clearance violations. Use the following commands to perform a batch design rule check on the currently loaded layout **`board`** of the **`demo.ddb`** DDB file:

Utilities
Batch DRC
Please confirm (Y/N) ? y

After performing the design rule check, the Batch DRC function implicitly activates the Report function from the Utilities menu to display the checking results and the design state. The Report function can also be called explicitly using the following commands:

Utilities
Report

The following listing is displayed by the Report function or after Batch DRC (zero error lines might be omitted):

```
File : demo.ddb
Type : Layout / Element : board


Number of Nets .................: 22
Number of Open Connections .....: 0
Number of Short Circuits .......: 0
Copper Distance Violations .....: 0
Documentary Distance Violations : 0
Number of Power Layer Errors ...: 0
Number of Missing Parts ........: 0
Number of Wrong Type Parts .....: 0
Number of Missing Netlist Pins .: 0
Height Distance Violations .....: 0
Power Layers in Use.............: -
Signal Layers in Use............: 1-2
```

The **`Number of Nets`** report entry displays number of nets defined in the net list of the currently loaded layout. The **`Number of Open Connections`** entry denotes the number of not yet routed (two-point) connections on the currently loaded layout. The **`Number of Short Circuits`** entry denotes the number of short-circuits encountered by the Design Rule Check. The **`Copper Distance Violations`** entry denotes the number of copper layer clearance distance violations online-encountered by the Design Rule Check on the currently loaded element. The **`Documentary Distance Violations`** entry denotes the number of clearance distance violations on documentary layers. The **`Number of Power Layer Errors`** entry denotes number of copper area cross-intersections on split power planes. The **`Number of Missing Parts`** report entry denotes the number of net list parts not yet placed on the currently loaded layout. The **`Number of Wrong Type Parts`** report entry denotes the number of net list parts placed with a wrong part package type (macro) on the currently loaded layout. The **`Number of Missing Netlist Pins`** entry denotes the number of unplaced and/or missing net list pins. Missing net list pins can cause an erroneous open connections count. An indicator is added to the open connections count if net list pins are missing. The **CHECKLNL** **User Language** program can be used to track missing net list pins. The **`Height Distance Violations`** entry denotes the number of (part) height design rule violations. The **`Power Layers in Use`** and **`Signal Layers in Use`** report entries display the used power and signal layers. Signal layer 1 to top layer are always assumed to be in use. These entries help to identify power layers without global net assignment, hence with split power planes to be considered by connectivity.

DRC Error Display

The Settings dialog from the **Layout Editor** View menu provides the DRC Error Display parameter for selecting DRC distance and height rule violation error display modes and/or colors. The Error Color option displays error boxes using the error color selected with the Change Colors function, the default of which is white. The Highlight Layer setting displays error rectangles using the color selected for the layer on which the erroneous element is placed. Errors on layers which are faded-out through Change Colors are not displayed.

The **Layout Editor** Utilities menu provides the DRC Error List function for displaying the DRC distance and height rule violation error lists. The DRC errors are listed in a popup menu, indicating error type, error layer and error coordinates for each error. A Zoom Window to the error position ist triggered when selecting an error from the list using the left mouse button. The + and - keys can be used to move the zoom window through the error position list in either direction.

Note that the modified **BAE HighEnd** data structure for storing the layout connectivity data allows for a *selective* shortcut display, i.e., shortcuts between two nets are displayed by highlighting only those elements causing the shortcut, whilst **BAE Professional** highlights the whole connection tree affected by the short-circuit.

Hit the spacebar to return to the **Layout Editor** menu.

4.6.2 Color Setup, Color Tables, Pick Preference Layer

The Change Colors function from the View menu activates a popup menu for modifying the current color settings. This color setup menu simultaneously can be utilized for displaying the current color assignments. Changing some item-specific color is accomplished by selecting the desired display item using the left mouse button and then selecting the desired color button from the Change Colors function. In the layout system, Change Colors provides a feature for fast display item fade-out/fade-in. Activating and/or deactivating some item-specific display is accomplished by selecting the desired display item entry with the right mouse button which works as a toggle between fade-out and fade-in. The system won't loose information on currently defined colors of faded-out display items; strike-through color buttons are used for notifying currently faded-out display items.

At overlaps of different elements the resulting mixed color is displayed. The highlight color is also mixed with the color of the element to be marked, thus resulting in a brighter display of that element.

The Save Colors function from the View menu is used to save the current color settings with a user-specified name to the **`ged.dat`** system file in the BAE programs directory. When starting the **Layout Editor** (or any other layout program module), the color table named **`standard`** is automatically loaded. Any other color table available in **`ged.dat`** can be loaded using the Load Colors function from the View menu.

It is a good idea to define color tables for certain tasks such as **`stackdef`** for padstack editing (i.e., with drill holes and drill plan visible) or **`unroutes`** for fast open connections recognition (i.e., with airlines visible only). At the definition of color tables it is to be considered that screen redraw functions will take longer if more objects must be displayed. It is recommended to define task-specific color tables to display only those objects which are important for the corresponding task.

The Set Edit Layer function is usually used to allow selection on a specific layer. It also has the useful hidden function of loading a layer-specific color table. These color tables have specific names, and if they don't exist there will be no change in the color display. See table 4-3 for the color table names assigned to the pick preference layers (**`<n>`** is the layer number, respectively). Table 4-3 also lists short layer names which can optionally be used in certain layer menus for selecting layers via keyboard input.

Table 4-3: Pick Preference Layer Color Tables and Short Layer Names

Layer	**Pick Preference Layer Color Table Name**	**Short Layer Name**
Signal Layer <n>	layer_<n>	<n>
Signal Layer All Layers	layer_all	a
Signal Layer Middle Layers	layer_def	m
Signal Layer Top Layer	layer_def	
Power Layer <n>		p<n>
Documentary Layer <n> Side 1	layer_d<n>_1	d<n>s1
Documentary Layer <n> Side 2	layer_d<n>_2	d<n>s2
Documentary Layer <n> Both Sides	layer_d<n>_a	d<n>sa
Board Outline		b
Airlines		u

When selecting a certain pick preference layer (e.g., signal layer 2) using the Set Edit Layer function, the color table with the corresponding name (e.g., **`layer_s2`**) is automatically loaded if available in **`ged.dat`**. This feature is most useful for e.g., manual routing.

4.6.3 Layout Net List Changes

The **Layout Editor** provides functions for performing pin/gate swaps to simplify the routing problem. Alternate package types can be assigned to layout parts, and net list part names can be changed for better legibility of the insertion plan (for manual insertion). These modifications are net list changes which must be backward annotated to the schematics using the Backannotation function from the **Schematic Editor** (see also chapter 3.3).

In this section we will apply some pin/gate swaps and change a couple of part names. First of all, use the following commands to set the coordinate display mode to inch and apply group functions to delete all traces from the currently loaded layout:

- Settings
  - Coordinate Display
    - Display Inch
- Groups
  - Group Polygon
    - Traces
      - Select
        - Move to [0.1",0.1"]
        - Move to [2.8",0.1"]
        - Move to [2.8",2.7"]
        - Move to [0.1",2.7"]
        - Done
  - Delete Group

The Pin/Gate Swap function from the Parts menu is used to perform manual pin/gate swaps. Use the following commands to swap the pins `1` and `2` of the switches named `s1000`, `s1001`, `s1002` and `s1003`, respectively:

- Parts
  - Pin/Gate Swap
    - Move to "s1000.1",[0.2",2.3"]
    - Move to "s1000.2",[0.5",2.3"]
  - Pin/Gate Swap
    - Move to "s1001.1",[0.2",2.1"]
    - Move to "s1001.2",[0.5",2.1"]
  - Pin/Gate Swap
    - Move to "s1002.1",[0.2",1.9"]
    - Move to "s1002.2",[0.5",1.9"]
  - Pin/Gate Swap
    - Move to "s1003.1",[0.2",1.7"]
    - Move to "s1003.2",[0.5",1.7"]

Note how the Pin/Gate Swap function provides graphical pin/gate swap indicators. A circle (0.6mm diameter, Workspace color) is displayed for every pin/gate swap enabled pin when the Pin/Gate Swap function is activated. The indicator for the first selected pin turns into a filled square, leaving only swappable pins marked by circles and character codes for the possible swap operations. The character codes are **P** for pin swap, **G** for gate swap and **A** (Array) for gate group swap. Selecting black for the Workspace color deactivates the pin/gate swap indicator display.

Each pin/gate swap is performed using the corresponding pin/gate swap definition from the Logical Library (see also chapter 7.11 for a description of the **LOGLIB** utility program and chapter 3.2 for a description of the **Packager** program module). The following error message is issued if no swap is allowed for the selected pins and/or gates:

```
Not allowed to swap these pins!
```

Use the following commands to swap the gates (1,2,3) and (5,6,4) as well as the pins **`12`** and **`13`** of the part named **`IC10`**:

```
Parts
    Pin/Gate Swap
                     Move to "ic10.(1,2,3)",[1.4",1.8"]
                     Move to "ic10.(5,6,4)",[1.5",1.8"]
    Pin/Gate Swap
                          Move to "ic10.12",[1.4",2.1"]
                          Move to "ic10.13",[1.3",2.1"]
```

Use the following commands to perform a part swap for the resistor parts named **`R101`** and **`R103`** (this is allowed because the same attribute values are assigned to these parts):

```
Parts
    Pin/Gate Swap
                             Move to "r101",[1.6",2.4"]
                             Move to "r103",[1.4",1.2"]
```

The Netlist Part Name function from the Parts menu is used to change part names in the net list. Use the following commands to change the net list part name of connector **`X1000`** to **`filename`**X1, and also change the net list part name of diode **`V1000`** to **`V2`**:

```
Parts
    Netlist Part Name
                            Move to "x1000",[2.4",1.5"]
                                 Part Name (X1000) ?  X1
    Netlist Part Name
                            Move to "v1000",[1.2",1.2"]
                                 Part Name (V1000) ?  V2
```

The Netlist Part Name function issues the following error message if you try to assign a part name which is already in use for another part:

```
Part name already in use!
```

Use the following commands to apply the Change Part Name function to change the name of part **`IC10`** to **`IC1`**:

```
Parts
    Change Part Name
                             Move to "ic10",[1.2",1.8"]
                                  Part Name (IC10) ?  IC1
```

The airlines previously connected to **`IC10`** have been disappeared from **`IC1`**. I.e., the Change Part Name function not only performs a part name change, but also *replaces the selected part*. With the commands above the part named **`IC10`** has been replaced with the part **`IC1`** which is not defined in the net list. Use the following commands to reset this part name change and apply the Netlist Part Name function to change the *net list* part name to **`IC1`** (note how the airlines connected to **`IC1`** will appear again):

```
Parts
    Change Part Name
                              Move to "ic1",[1.2",1.8"]
                                   Part Name (IC1) ?  ic10
    Netlist Part Name
                             Move to "ic10",[1.2",1.8"]
                                  Part Name (IC10) ?  ic1
```

It is recommended to use the Change Part Name function with caution. Multiple misuse of the Change Part Name function can cause short circuits on routed layouts, and it might get very laborious to backtrack part name changes for design corrections.

The layout net list changes accomplished with the previous operations must be backannotated to the schematics and the logical net list. Use the following commands to save the currently loaded layout and return to the BAE main menu:

- File
  - Save Element
- File
  - Main Menu

The BAE main menu is activated. Use the following commands to switch to the **Schematic Editor** and run Backannotation to transfer the physical net list named **`board`** in in **`demo.ddb`** back to the schematics:

- Schematic
  - Utilities
    - Backannotation
      - Design File Name ? demo
      - Layout Element Name ? board

Backannotation displays the following messages:

```
==============================
BARTELS BACKANNOTATION UTILITY
==============================


Design File Name ........: 'demo'
Layout Element Name .....: 'board'
No error occurred!
```

The **`No error occurred!`** message means that Backannotation has been successfully completed, and the logical net list in project file **`demo.ddb`** has been annotated with the physical net list named **`board`**. Hit any key to return to the **Schematic Editor** main menu and use the following commands to load the SCM sheet **`sheet1`** from DDB file **`demo.ddb`**:

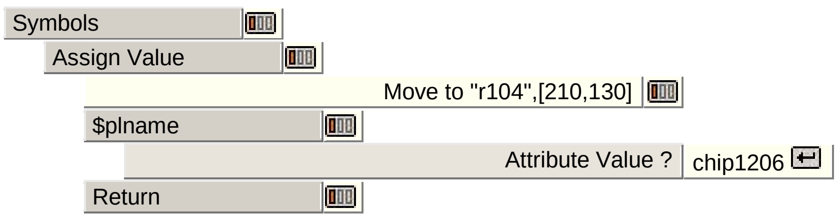
- File
  - Load
    - Sheet
      - File Name ? demo
      - Element Name ? sheet1

You can now examine the currently loaded SCM shee for modifications introduced by Backannotation. Particularly note the part name changes (**`IC1`** instead of **`IC10`**, **`V2`** instead of **`V1000`**, etc.) and the pin assignment changes (e.g., at the gates of **`IC1`** or at the switches **`S1000`** through **`S1003`**).

4.6.4 SCM Changes, Redesign

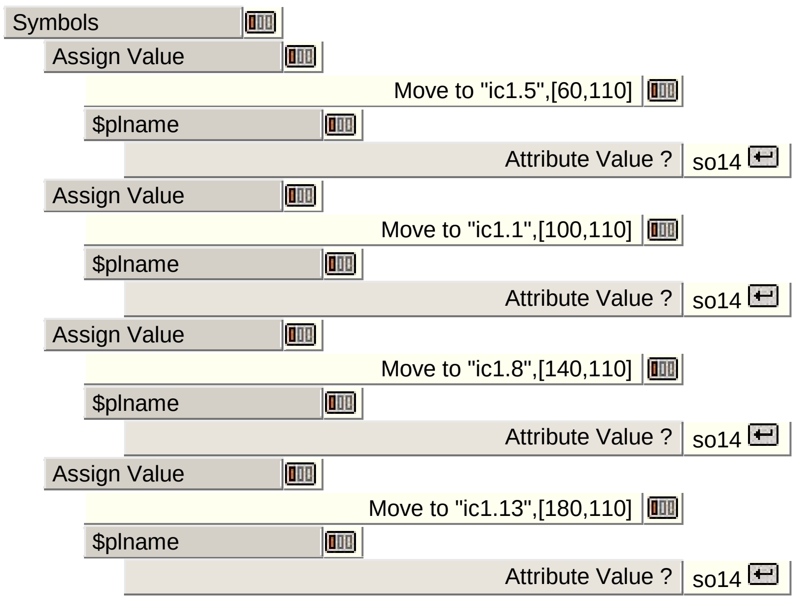
Bartels AutoEngineer provides features for modifying SCM sheets of an already completed design, without the need to prepare a completely new layout. This section describes how perform a redesign by making changes to the currently loaded SCM sheet.

Use the following commands to change the **`$plname`** attribute value of the resistor **`R104`** from **`minimelf`** to **`chip1206`**:



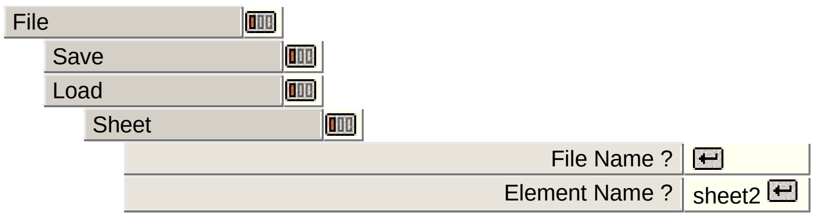
```
Symbols
  Assign Value
                          Move to "r104",[210,130]
    $plname
                                 Attribute Value ? chip1206
    Return
```

With the commands above the package type assignment for the resistor part **`R104`** has been changed. Use the following commands to assign value **`so14`** to the **`$plname`** attribute of each gate of part **`IC1`**, thus defining a non-default package type for **`IC1`** (the default package type was **`dil14`**):



```
Symbols
  Assign Value
                          Move to "ic1.5",[60,110]
    $plname
                                 Attribute Value ? so14
  Assign Value
                          Move to "ic1.1",[100,110]
    $plname
                                 Attribute Value ? so14
  Assign Value
                          Move to "ic1.8",[140,110]
    $plname
                                 Attribute Value ? so14
  Assign Value
                          Move to "ic1.13",[180,110]
    $plname
                                 Attribute Value ? so14
```

Use the following commands to save the currently loaded SCM sheet and load **`sheet2`**:



```
File
  Save
  Load
    Sheet
                                 File Name ?
                              Element Name ? sheet2
```

The currently loaded SCM sheet contains some **Autorouter** control parameter settings (net attribute assignments), which should be modified. Use the following commands to change the **`ROUTWIDTH`** net attribute of **`NET`** from 0.5mm to 0.3mm, change the **`ROUTWIDTH`** attribute of net **`Vss`** from 0.6mm to 0.45mm and change the **`MINDIST`** attribute for net **`Vdd`** from 0.4mm to 0.3mm:

```
Symbols
  Assign Value
    Move to "NET"/Routwidth Symbol
    $val
      Attribute Value ? 0.3
  Assign Value
    Move to "Vss"/Routwidth Symbol
    $val
      Attribute Value ? 0.45
  Assign Value
    Move to "Vdd"/Mindist Symbol
    $val
      Attribute Value ? 0.3
```

Use the following commands to return to the BAE main menu (the currently loaded SCM element is automatically saved):

```
File
  Main Menu
```

Now the BAE main menu is activated. Use the following commands to run the **Packager** to transfer the SCM changes to the layout, i.e., to the physical net list named **`board`** (the **`demo.ddb`** job file can be used as design library since it contains all of the information required for packaging):

```
Packager
    Design File Name ? demo
    Design Library Name ? demo
    Layout Element Name ? board
```

The **Packager** issues the **`No error occurred!`** message after successfully completing the forward annotation to the layout. Hit any key to return to the BAE main menu and use the following command to start the **Layout Editor**:

```
Layout
```

Now the **Layout Editor** is activated. Use the following commands to load the annotated layout:

```
File
  Load Element
    Layout
      File Name ? demo
      Element Name ? board
```

A connectivity generation is accomplished after loading the layout. Note that there are no airlines connected anymore to the parts **`IC1`** and **`R104`**, since the package types have been changed for these parts. Use the following commands to delete all parts with changed package type assignments introduced by net list modifications:

```
Parts
  Delete Update
    Please confirm (Y/N) ? y
```

The confirm prompt is only activated if there are parts placed with wrong package types. The Delete Update function can be used for checking the layout for wrong package types (simply type **`n`** to the confirm prompt if you want to abort the Delete Update function). The Delete Update call from above deletes the parts **`R104`** and **`IC1`** from the layout and issues the following message:

```
2 Parts deleted!
```

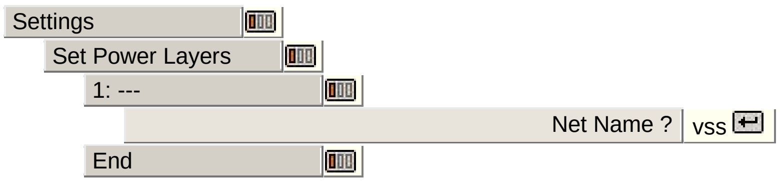
Use the following commands to place the previously deleted parts with correct package types:

- Parts
  - Add Part
    - Part Name ? ic1
    - Jump Absolute
      - Absolute X Coordinate (mm/") ? 1.6"
      - Absolute X Coordinate (mm/") ? 2.0"
  - Place Next Part
    - Mirror On
    - Rotate Right
    - Jump Absolute
      - Absolute X Coordinate (mm/") ? 0.35"
      - Absolute Y Coordinate (mm/") ? 2.0"

Now all parts are placed again on the layout. You can check this with the Place next Part function which should issue the `All parts have been placed already!` message.

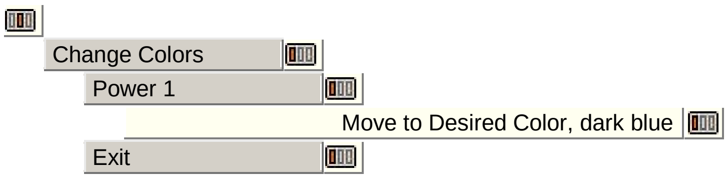
4.6.5 Defining and Editing Power Layers

The **Bartels AutoEngineer** layout system provides features for defining power layers and/or power planes. Use the following commands to define a power layer for signal **`Vss`**:



Note how the above-mentioned power layer definition removes the airlines connected to the drilled pins of signal **`Vss`**. Drilled pins are automatically connected to the corresponding power layer (i.e., the **CAM Processor** will later generate heat traps at the drillings of power layer pins). The airlines connected to the SMD pins of signal **`Vss`** won't be deleted, i.e., the **Autorouter** will automatically connect these pins to the power layer using vias.

Power layer definitions can be displayed in the BAE layout system. Use the following commands to set the color for power layer 1 to dark blue:



On the display of drilled pin definitions there is a distinction whether the pin is connected to a power layer or not. Power layer connections are displayed as circle outlines; isolations, i.e., drills which are not to be connected to any power layer, are displayed as filled circles.

The layer selection menus of the Add Active Copper function from the Areas menu allow for the selection of power layers. Power planes or isolated areas (net name **`-`**) can be placed on power layers to perform split power plane editing, i.e., to define more than one signal on a single power layer. Power planes are displayed with their outlines which will later be interpreted as isolation line by the **CAM Processor** (see also chapter 4.7.6 of this manual). There is a restriction that no power plane can overlap any other power plane partially since this would cause an ambiguity in the power plane tree detection algorithms. Partial power plane overlaps will cause the design rule check to issue power layer errors; the number of power plane errors detected by the DRC is shown with the **`Power Layer Errors`** entry of the Report Utilities function. Power planes completely enclosed by other power planes are allowed; the power plane tree detection will always match the "innermost" power plane for connecting.

The layer selection menus of the Add Text function from the Text, Drill menu allow for the selection of power layers. Text can be placed on power layers to add documentary information to the power layers. When placing text on power layers, the design rule check will perform distance checking against the All Layers and Middle Layers signal layers. Power layer text is visible on the PCB only if the corresponding power layer is configured as board outside layer (i.e., as either solder side or component side layer). Usually power layers are defined as board inside layers of a multilayer design. Text placed on power layers can serve as control information (e.g., for the layout designer or as plot or film archive information).

4.6.6 Autorouter Via Keepout Areas

There is often the need to define areas where the **Autorouter** is allowed to rout traces but must not place vias (e.g., underneath special parts). Via keepout areas can be defined on a routing layer which is not required for the real layout. This "via keepout layer" should then be prohibited with the **Autorouter** layer assignment. The **Autorouter** considers all objects placed on prohibited layers, and vias and/or via drill holes are assumed to be defined on all layers. I.e., the **Autorouter** refrains from placing vias at positions where via pads would intersect with keepout areas on prohibited routing layers.

Use the following commands to place a keepout area on layer 3 matching the region underneath the switches `s1000` through `s1009`:

```
Areas
  Add Keep Out Area
    Layer 3
                    Move to [0.25",0.4"]
                    Move to [0.45",0.4"]
                    Move to [0.45",2.4"]
                    Move to [0.25",2.4"]

      Done
```

Use the following commands to start the **Autorouter** (the currently loaded layout will automatically be saved):

```
File
  Autorouter
```

Use the following commands to define routing layer 3 to be prohibited and start the Full Autorouter (the **Autorouter** options such as routing grid, trace width, clearance, etc., have already been defined with the previous **Autorouter** session):

```
Options
  Layer Assignment
              Select Layer Number ?  3
     Select Layer Type (P/H/V/A/-) ?  p
              Select Layer Number ?
Autorouter
  Full Autorouter
```

The Full Autorouter will a find 100% routing solution with no vias placed inside the via keepout area. Note also how the SMD pins of the previously defined power layer `Vss` (see above) are automatically connected using short traces with vias.

Use the following command to return to the **Layout Editor** after successfully completing the autorouting:

```
File
  Layout Editor
```

4.6.7 Area Mirror Mode

The Mirror Display function from the **Layout Editor** Areas menu is used to set special display attributes for selectable areas. The Visible always option is the default display mode for any area. Visible always defines the selected area to be always visible. The Visible unmirrored option defines the selected area to be visible only when unmirrored (i.e., when placed on the component side). The Visible mirrored option defines the selected area to be visible only when mirrored (i.e., when placed on the solder side). The Mirror Display function can be used on part, padstack and pad level. It is possible to define SMD parts with differently shaped pads or keepout areas depending on which side of the board the part is placed (for supporting different soldering processes on solder and/or component side).

Use the following commands to load the pad symbol **`p1206`** from the **`demo.ddb`** DDB file:

```
File
  Load Element
    Pad
                              File Name ?  demo
                           Element Name ?  p1206
```

There is finger-shaped contact area defined on the currently loaded **`p1206`** pad. Use the following commands to define this copper area to be visible only when unmirrored (i.e., when placed on the component side):

```
Areas
  Mirror Display
                       Move to Area Edge
    Visible unmirrored
```

Now the component side pad shape is defined. Use the following commands to define a rectangle-shaped passive copper area to be visible only when mirrored (i.e., when placed on the solder side):

```
Areas
  Add Passive Copper
    
      Jump Absolute
             Absolute X Coordinate (mm/") ?  0.6
             Absolute Y Coordinate (mm/") ?  0.9
    
      Jump Relative
             Relative X Coordinate (mm/") ?  -1.2
             Relative Y Coordinate (mm/") ?  0
    
      Jump Relative
             Relative X Coordinate (mm/") ?  0
             Relative Y Coordinate (mm/") ?  -1.8
    
      Jump Relative
             Relative X Coordinate (mm/") ?  1.2
             Relative Y Coordinate (mm/") ?  0
    
      Done
Areas
  Mirror Display
                Move to Area Corner/Edge
    Visible mirrored
```

Use the following commands to save the currently loaded pad symbol and re-load the layout:

```
File
  Save Element
  Load Element
    Layout
                                   File Name ?  ↵
                                Element Name ?  board ↵
```

The system accomplishes a connectivity generation to correlate the library modifications with the current net list definition. The **`p1206`** pad is used on the **`s1206`** padstack symbol, which, in turn, is used on the **`chip1206`** part symbol. Note how the layout system uses different pad shapes for the mirrored parts **`R104`** and **`C101`** and for the unmirrored part **`R105`**.

The Mirror Display function can be applied on any copper, keepout or documentary polygon defined on a library element which can be mirrored. You can also use this feature to define part keepout areas on documentary layers to perform different part clearance checks depending on which side of the board the parts are placed.

Reflow-Reflow SMT/SMD Soldering Support

A control for setting the Area Mirror Display parameter is provided in the Settings dialog from the Settings menu to support reflow-reflow SMD/SMT soldering techniques. On default, the Mirror Display parameter is activated, and polygons with the Visible unmirrored, attribute are only visible when not mirrored, whilst polygons with the Visible mirrored attribute are only visible when mirrored. However, with the Mirror Display, parameter deactivated, all polygons declared as Visible unmirrored, are always visible, and all polygons declared as Visible mirrored, are never visible, independently of any (part) mirroring. Mirror Display, deactivation allows for SMD libraries designed for conventional SMD soldering to be re-used for reflow-reflow SMD soldering.

4.6.8 Automatic Copper Fill

The BAE layout system provides powerful automatic copper fill functions with user-definable minimum structure size and isolation distance. The copper fill algorithm support isolated copper area elimination and automatic heat trap generation with adjustable connection widths and heat-trap-specific isolation distances.

The copper fill functions are available through the Copper Fill submenu of the **Layout Editor** Areas menu.

Copper Fill Parameters

Depending on the current BAE menu setup, copper fill parameters can either be set from the Settings dialog of the Copper Fill submenu or through dedicated menu functions in that same menu.

The Set Fill Clearance function is used to set the copper fill isolation distance parameter for the copper fill algorithm (default 0.3mm). The copper fill clearance distance is applied on default unless some net-specific minimum distance settings are defined through **MINDIST** net attributes (see chapter 7.11). Net-specific minimum distance attribute settings are considered individually.

The Min. Fill Size function is used to define the minimum structure size for area generation (default 0.1mm). This value should correspond with the smallest Gerber aperture size to ensure valid Gerber photoplot data generation without overdraw errors.

The Round Corners option of the Traces Cut Mode function causes the copper fill algorithm to generate arc-shaped concave area borders during trace segment isolation; on default octagonal circle interpolation is applied (option Octagonal Corners).

The Keep Islands option of the Insol. Area Mode function switches off isolated copper area recognition (i.e., the copper washes over the pad that it is to connect to); on default, isolated copper areas are automatically eliminated during active copper generation (option Delete Islands). The Select Islands option keeps isolated areas and automatically selects them to the current group as they are created.

The Heat Trap Mode options also support for different processing modes for pins and vias. It is possible to decide whether heat-trap connections should be generated for both pins and vias (option Pin & Via Heat Traps), for pins only (option Pin Heat Traps), for vias only (option Via Heat Traps), or if direct connections only should be generated (option Direct Connect).

The Direct Connect option of the Heat Trap Mode function deactivates automatic heat trap generation when filling active copper. The default Heat Trap Mode option Use Heat Traps is used to activate automatic heat trap generation, i.e., to create thermal relieves for copper area connections. With Use Heat Traps selected, the system prompts for the heat-trap junction width and an optional heat-trap-specific clearance distance, where invalid inputs won't change current settings. On default, the heat trap junction width is set to 0.3mm and the heat-trap clearance is set to zero (i.e., heat-traps are isolated using standard minimum clearance settings).

BAE HighEnd provides the HT-Junction option for specifying the maximum heat-trap connections count (1, 2, 3 or 4). The default heat-trap generation sequence is left, right, bottom, top.

Copper Fill Workarea Definition

Copper fill workareas are required for copper fill functions to operate. Use the following **Layout Editor** commands to define a rectangle-shaped copper fill workarea on layer 1 for the signal named `net`:

```
Areas
  Copper Fill
    Add Cop.-Fill Area
                                   Net Name ?  net
      Layer 1
                            Move to [0.6",0.4"]
                            Move to [1.7",0.4"]
                            Move to [1.7",1.7"]
                            Move to [0.6",1.7"]

        Done
```

A dash string input (`-`) to the net name prompt of the Add Cop.-Fill Area function creates a copper fill workarea which is not assigned to any net. This feature can be used for generating pure shielding areas.

Passive copper areas with signal net connections (e.g., teardrops created as passive copper) are treated like active copper areas of that net and won't be isolated from copper fill areas of that net.

Automatic Copper Fill

The Fill all areas, Fill single area, Clear all areas and Clear single area functions are used to activate automatic copper fill procedures. Fill all areas performs automatic copper fill on all copper fill workareas. Clear all areas deletes copper areas from all copper fill workareas. With Fill single area and Clear single area, the user is expected to select the desired copper fill workarea.

Use the following commands to set the isolation distance to 0.35mm, specify a minimum structure size of 0.15mm, deactivate the automatic isolated copper area elimination and fill the predefined workarea:

```
Areas
  Copper Fill
    Set Fill Clearance
                 Copper Clearance Distance ( 0.30mm) ?  0.35
    Min. Fill Size
                    Min. Fill Structure Size ( 0.10mm) ?  0.15
    Insol. Area Mode
      Keep Islands
    Fill single area
                            Move to Copper Fill Workarea
```

The system fills the selected workarea with active copper assigned to the signal named `net` and isolated from other signal levels. The copper fill process is very laborious and might last a few moments to complete. You should perform a screen redraw and examine the results after the copper fille procedure is completed. You can reset the copper fill, change the copper fill parameters and re-apply the copper fill function if you are not satisfied with the results (maybe because of either too small or too large structures being generated).

Use the following commands to reset the copper fill from above and re-apply the copper fill with automatic isolated area generation:

```
Areas
  Copper Fill
    Clear single area
    Insol. Area Mode
      Delete Islands
    Fill all areas
```

The minimum structure size (i.e., the smallest area size to be generated with copper fill) has basic meaning for subsequent **CAM Processor**. The copper fill algorithm uses the minimum structure size for automatically rounding off convex corners to avoid acute-angled areas. You should specify the minimum structure size according to the size of the smallest round aperture defined for Gerber output to ensure that Gerber photoplot data can be generated without overdraw errors (see also chapter 4.7 of this manual).

Copper Fill Complexity

The CPU time and memory requirements of the copper fill algorithm strongly depend on the quantity and complexity of the structures to be isolated and/or generated. Orthogonal structures make the job much easier than e.g., arc-shaped objects since a lot more time consuming geometric distance calculations and complex floating point operations are required for the latter. Figure 4-8 elucidates how the complexity of the objects to be isolated and particularly their mutual positioning strongly affects memory requirements during copper fill. Note, however, that figure 4-8 does not show what large amount of temporary data is required for the area reduction and expansion algorithms used for generating correct copper fill areas. Defining more smaller copper fill workareas instead of a few large ones can often be the workaround when running into memory problems using the copper fill function on main memory limited PC systems.

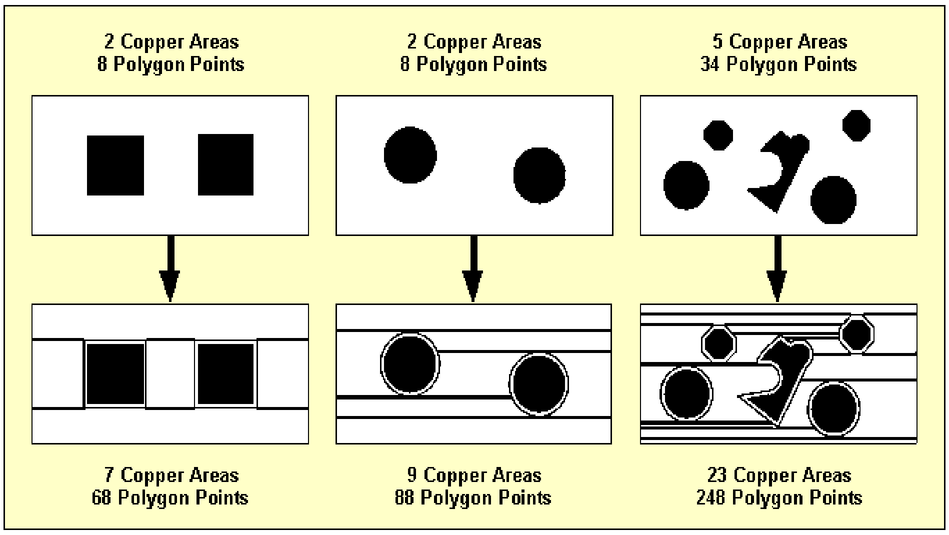


Figure 4-8: Automatic Copper Fill Complexity

You can use one of the Octagonal Corners, Octagonal Circles and Octagonal Corners+Circles options instead of the Round Corners default option from the Traces Cut Mode function to reduce the complexity and number of fill areas. This allows for trace corners and/or full circles to be isolated like octagons, thus also reducing the amount of Gerber data if no Gerber arc commands can be used.

To avoid long response times after unintentionally activating the copper fill function, automatic copper fill can be canceled by pressing any key and confirming the abort request with a verification menu. Note, however, that canceling is not possible anymore at the final stage of connectivity generation, and that areas generated before the abort request must be explicitly eliminated using the Undo function.

Using the copper fill function considerably increases the number of copper areas and polygon points. This raises the CPU time requirements for the Mincon function if the Mincon function type is set to a corner to corner airline calculation (see chapter 4.3.2 of this manual). In such cases it can be worthwhile to use a pin to pin calculation method to reduce Mincon CPU time requirements. The Mincon Function from the **Layout Editor** Settings menu is used to set the Mincon function type.

Copper Area Hatching

The Copper Fill submenu provides the Hatch Copper Area function for transforming copper areas into line or cross hatched areas with user-definable hatching width and hatching clearance.

Hatching is accomplished through trace segments generation. The width of the produced trace segments can be set with the Hatching Width function (default 0.3mm) whilst the spacing between produced trace segments can be set with the Hatching Spacing function (default 1/20 inch).

The Hatching Mode function is used to designate the hatching type. The default Line Hatching option generates hatch areas with diagonal trace segments. The Grid Hatching option generates hatch areas with crosswise intersecting diagonal trace segments.

The hatching process is activated by calling the Hatch Copper Area function and selecting the copper area to be hatched.

Use the following commands to set the coordinate display mode to inch, and transform the circle-shaped copper area on signal layer 2 of the layout to a line-hatched area with a hatching distance of 1mm:

```
Settings
    Coordinate Display
        Display Inch
Areas
    Copper Fill
        Hatching Distance
            Hatching Line Spacing ( 1.27mm) ?   1
        Hatch Copper Area
            Move to Passive Copper Area/Signal Layer 2,
                                       [0.8",1.8"]
```

Use the following commands to set the hatching width to 0.2mm and transform the active copper area (signal **vdd**) on signal layer 1 of the layout into a cross-hatched area:

```
Areas
    Copper Fill
        Hatching Width
            Hatching Line Width ( 0.30mm) ?   0.2
        Hatching Mode
            Grid Hatching
        Hatch Copper Area
            Move to Active Copper Area "vdd"/Signal Layer 1,
                                       [2.5",2.0"]
```

The Hatch Copper Area function generates a special layout polygon type called hatched copper area. The hatching and the hatching area outline are generated using traces with a trace width according to the hatching width setting. The traces are strongly connected with the hatched copper polygon to support general **Layout Editor** polygon functions such as Move Area or Delete Area.

The PCB layout example should now look like the one shown in figure 4-9 if you correctly executed all operations.

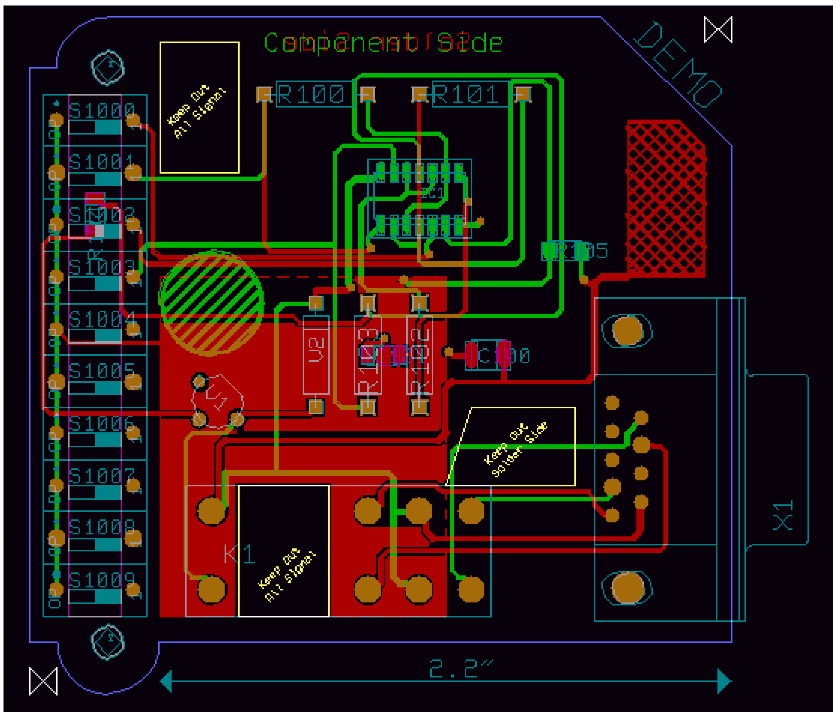


Figure 4-9: Layout with Filled Copper Areas

This might be a good time to save the layout:

File
    Save Element

4.6.9 Library Update

The Update Library function is one of the most powerful features of the **Bartels AutoEngineer**. Update Library is usually used to update a job-specific library in order to correlate it with the contents of some master library.

Use the following commands to copy the currently loaded layout to the **`democopy.ddb`** DDB file (with default layout element name), and load the copied layout:

```
File
  Save Element As
                    File Name ?  democopy ↵
                 Element Name ?  ↵
  Load Element
    Layout
                    File Name ?  democopy ↵
                 Element Name ?  ↵
```

Now the copied layout appears on the screen. However, this layout contains only those objects which are directly defined on the layout. The lower level library elements (parts, padstacks, pads) are not displayed since they have not been copied with the Save Element As function to prevent from overwriting existing library elements in the **`democopy.ddb`** destination file. The system will issue either the

```
Connected pins missing!
```

message or the

```
Cannot load all elements (display not complete)!
```

message in the status line. Use the following commands to correlate the job-specific library of DDB file **`democopy.ddb`** with the contents of DDB source file **`demo.ddb`**:

```
File
  Update Library
    Layout
                    File Name ?  ↵
                 Element Name ?  ↵
             Source File Name ?  demo ↵
```

After a few moments the system should issue a **`Library elements replaced!`** message which means that all library elements referenced from the currently loaded layout have been copied from the specified library source file. Use the following commands to re-load the layout in order to display the library update (Update Library works on DDB file level, it doesn't affect elements in main memory):

```
File
  Load Element
    Layout
                    File Name ?  ↵
                 Element Name ?  ↵
```

The Update Library function can also be used to correlate the job-specific library with the contents of a different layout library file. Use the following commands to perform a library update for the currently loaded layout using the library source file **`demolib.ddb`**, and re-load the layout (note how the **`r04a25`** resistor package type is updated with the definition from **`demolib.ddb`**):

```
File
  Update Library
    Layout
                  File Name ?  ↵
               Element Name ?  ↵
           Source File Name ?  demolib ↵
  Load Element
    Layout
                  File Name ?  ↵
               Element Name ?  ↵
```

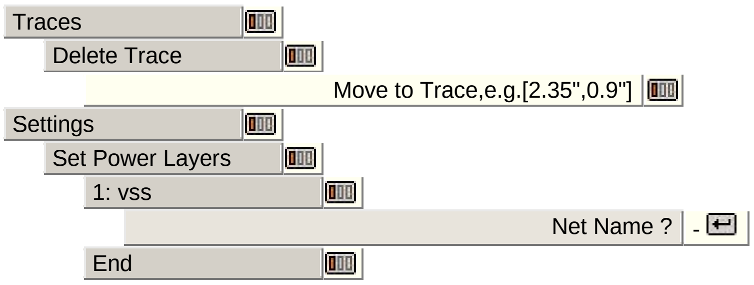
The Replace Element function is used for substituting selectable library elements with definitions from a certain library source file. Use the following commands to replace the **`r04a25`** part symbol and the **`p1206`** pad symbol of the current **`democopy.ddb`** DDB file with the corresponding definitions from the **`demo.ddb`** DDB file, and re-load the layout (Replace Element works on DDB file level, it doesn't affect elements in main memory):

```
File
  Replace Element
    Part
                  File Name ?  ↵
               Element Name ?  r04a25 ↵
           Source File Name ?  demo ↵
       Please confirm (Y/N) ?  y ↵
  Replace Element
    Pad
                  File Name ?  ↵
               Element Name ?  p1206 ↵
           Source File Name ?  demo ↵
       Please confirm (Y/N) ?  y ↵
  Load Element
    Layout
                  File Name ?  ↵
               Element Name ?  ↵
```

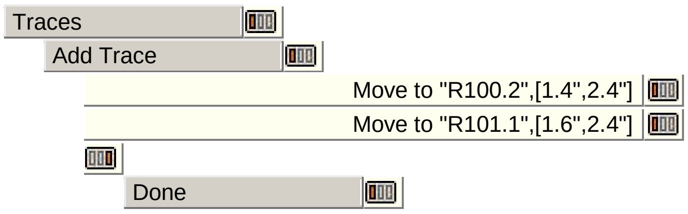
4.6.10 Back Net List

The Back Netlist function from the Utilities menu is most useful for certain applications such as high frequency design, where net list data is requested to be automatically generated from the copper located on the PCB.

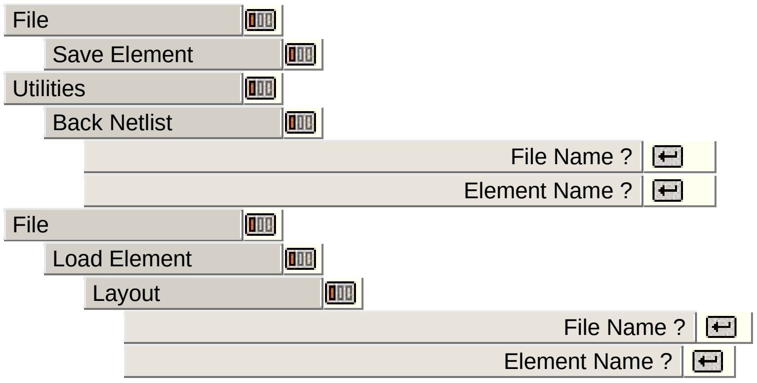
The mode of operation of this function should be demonstrated using the currently loaded layout from DDB file **democopy.ddb**. Delete the trace which connects the pins **NO1** and **4** of the parts named **K1** and **X1**, and reset the definition of power layer **vss**:



Note how the deleted trace and power layer definition are replaced with airlines. Use the following commands to add a trace to connect the pins **2** and **1** of the parts named **R100** and **R101**:



The system now indicates a short-circuit error. Use the following commands to save the layout, run Back Netlist to generate a net list with the same file and element name, and immediately re-load the layout again (Back Netlist stores to DDB file without changing the currently loaded net list data):



Now the system performs a connectivity generation to correlate the layout with the net list which was previously generated with Back Netlist. Note that there are no airlines displayed anymore after completing the connectivity generation. The layout does not contain any open connections and/or short-circuits anymore (you can check this with Batch DRC and Report from the Utilities menu). Deleting the trace which was previously causing a short-circuit would now result in an airline display.

Another useful application of the Back Netlist function net list generation from Gerber data which was previously loaded with the **CAM View** module (for more details see chapter 4.8 of this manual).

4.6.11 Blind and Buried Vias

Blind and buried vias are partial vias which can be used for layer changes when routing multilayer layouts. Routing with blind and buried vias can considerably increase the routability of multilayer layouts with four or more signal layers.

The BAE **Layout Editor** features arbitrary pad layer assignment (see also chapter 4.2.2 for more details on creating padstacks) and drill classes to support the definition of blind and buried vias. Drill classes should be assigned to certain layer-sets to support CAM drill data output for selectable layer-sets (see also chapter 4.7 for more details on drill data output). There are no system-imposed restrictions on how to define and/or assign drill classes. You can, e.g., use drill class **`A`** for layer-set 1-2, drill class **`B`** for layer-set 2-3, drill class **`C`** for layer-set 3-4, etc. Note, however, that the drill class assignment must be considered for correct drill data output by the **CAM Processor**, and that the layout top layer setting (see also chapter 4.3.1) gains special meaning when using blind and buried vias.

A typical configuration for 4-layer layouts is the definition of the vias **`via`** (for all layers), **`via_12`** (for layer-set 1-2), **`via_23`** (for layer-set 2-3) and **`via_34`** (for layer-set 3-4). All these vias are simultaneously available for routing when selected to the via list (see also chapter 4.3.4 for details on how to select vias)

The Change Layer **Layout Editor** command used during interactive routing automatically selects the via with the least possible layer occupancy. The same principle is applied by the **Autorouter**. At least one via for all signal layers required for autorouting. Additionally, the **Autorouter** can e.g., simultaneously use vias for the layer-sets 1-2-3, 1-2, 2-3, etc. However, to avoid backtracking ambiguities, the **Autorouter** is restricted in that it can not use vias with multiple mutually intersecting layers. I.e., the **Autorouter** refuses to use a via for layer-set 1-2-3 and another via for layer-set 2-3-4 at the same time (error message **`Invalid via padstack (cannot use it)!`**).

Using the **CAM Processor** for generating drill data output for the drill holes defined on partial vias requires corresponding drill class(es) to be specified.

The production of PCBs with blind and buried vias is usually more expensive than the production of standard multilayer boards. However, using blind and buried vias increases the routability of multilayer layouts and also supports advanced PCB manufacturing technologies such as plasma-etched via production processes.

4.6.12 Exiting the Layout System

Don't forget to save the currently edited layout element before exiting the **Layout Editor**:

```
File
    Save Element
```

Returning to Main Menu

The following commands can be used from each program module of the BAE layout system (except for the **Autorouter**) to return to the BAE Main Menu (BAE Shell):

```
File
    Main Menu
```

The Main Menu function automatically saves the currently loaded layout element. From the BAE Shell, the following command can be used to return to the operating system:

```
Exit BAE
```

Exiting BAE

The following commands can be used from each program module of the BAE layout system to return to the operating system, i.e., to exit the **Bartels AutoEngineer**:

```
File
    Exit BAE
```

The Exit BAE function activates a confirmation request if the currently loaded element has not yet been saved. In this case you should abort the Exit BAE function, save the current element, and then call Exit BAE again, as in:

```
File
    Exit BAE
                          Please confirm (Y/N) ? n ↵
File
    Save Element
File
    Exit BAE
```

4.7 CAM Processor

4.7.1 Starting the CAM Processor

Before starting the **CAM Processor** for generating real project manufacturing data, you should always perform a complete design rule check using the Batch DRC function from the **Layout Editor** Utilities menu. Subsequently the DRC result should be examined with the Report function. It is strongly recommended to refrain from generating and/or releasing any **CAM Processor** output if clearance violations, short circuits or open connections are indicated since this will (almost certainly) result in the production of faulty PCBs.

It is recommended to start the **Bartels AutoEngineer** from in the directory where the design files should be generated since this considerably simplifies job file access. If you intend to process the examples provided with this manual it is recommended to move to the BAE examples directory installed with the BAE software. The **Layout Editor** can be called from the **Bartels AutoEngineer** main shell. Start the BAE shell by typing the following command to the operating system prompt:

```
> bae ↵
```

Move the menu cursor to the Layout menu item and confirm this choice by pressing the left mouse button:

Layout

Now the **Layout Editor** program module is loaded and the **Layout Editor** menu is activated. If this fails to happen then check your BAE software installation (see the Bartels AutoEngineer® Installation Guide for details on how to perform a correct installation).

Use the CAM Processor command from the **Layout Editor** File menu to call the **CAM Processor**:

File
  CAM Processor

If a layout element was previously processed in the **Layout Editor**, then this element will automatically be saved and re-loaded to the **CAM Processor**. Within the **CAM Processor** the Load Element function from the File menu is used to load layout elements of any hierarchy level. Use the following commands to load the layout named `board` from DDB file `demo.ddb`:

File
  Load Element
    Layout
      File Name ? demo ↵
      Element Name ? board ↵

The system issues the `Cannot load character font!` error message, if the font used in the loaded element is not available in the `ged.fnt` font file from the BAE programs directory.

Without character font data, the **CAM Processor** fails to plot any text defined on the currently loaded layout element. You should either supply the requested character font (e.g., use the **FONTCONV** utility program) or assign an existing font to the layout element using the Select Font function from the **Layout Editor** Text menu before producing any CAM output.

4.7.2 CAM Processor Main Menu

The **CAM Processor** standard/sidemenu user interface provides a menu area on the right side, consisting of the main menu on top and the currently active menu below that main menu. After entering the **CAM Processor** the Utilities menu is active and the menu cursor points to the Load Element function.

The Windows and Motif versions of the **CAM Processor** can optionally be operated with a pull-down menu user interface providing a horizontally arranged main menu bar on top. The **`WINMENUMODE`** command of the **BSETUP** utility program is used to switch between **`SIDEMENU`** and **`PULLDOWN`** Windows/Motif menu configurations (see chapter 7.2 for more details).

The following main menu is always available whilst processing layout data with the **CAM Processor**:

Display
Control Plot
Gerber Photoplot
Drilling+Insertion
Plot Parameters
Utilities

Display

The View or Display menu can either be activated by selecting the corresponding main menu item or by pressing the middle mouse button. The View or Display menu provides useful functions for changing display options such as zoom window, zoom scale, input grid, color settings, etc.

Control Plot

The Control Plot menu provides the functions for generating HP-GL pen plots, PCL HP-laser prints and Postscript output.

Gerber Photoplot

The Gerber Photoplot menu provides the functions for generating Gerber photo plots, also including facilities for defining and managing Gerber aperture tables.

Drilling+Insertion

The Drilling+Insertion menu provides the functions for generating drill data output (format Sieb&Meier and/or Excellon) and insertion data output (generic format).

Plot Parameters

The Plot Parameters menu is used to set general plot parameters (plot tolerance, CAM origin, CAM rotation, CAM mirror mode) and to define special parameters for generating negative power layer plots. The Plot Parameters menu also provides features for controlling the processing of special layers such as the All Signal, Middle Layers, Board Outline and Plot Markers layers.

Utilities

The Utilities menu provides functions for exiting BAE, returning to the BAE main shell, calling the **Layout Editor** and starting **User Language** programs. The Utilities menu can also be used for loading layout elements, listing DDB file contents, setting the display grid and the coordinate display mode, or removing error markers.

4.7.3 Customized CAM Processor User Interface

Menu Assignments and Key Bindings

The BAE software comes with **User Language** programs for activating a modified **CAM Processor** user interface with many additional functions (startups, toolbars, menu assignments, key bindings, etc.). The **BAE_ST User Language** program is automatically started when entering the **CAM Processor**. **BAE_ST** calls the **UIFSETUP User Language** program which activates predefined **CAM Processor** menu assignments and key bindings. Menu assignments and key bindings can be changed by modifiying and re-compiling the **UIFSETUP** source code. The **HLPKEYS User Language** program is used to list the current key bindings. With the predefined menu assignments of **UIFSETUP** activated, **HLPKEYS** can be called from the Key Bindings function of the Help menu. Menu assignments and key bindings can be listed with the **UIFDUMP User Language** program. The **UIFRESET User Language** program can be used to reset all currently defined menu assignments and key bindings. **UIFSETUP**, **UIFDUMP** and **UIFRESET** can also be called from the menu of the **KEYPROG User Language** program which provides additional facilities for online key programming and **User Language** program help info management.

Cascading Windows/Motif Pulldown Menus

The Windows and Motif pulldown menu user interfaces of the **CAM Processor** provide facilities for cascading submenu definitions. I.e., submenus can be attached to other menu items. The **UIFSETUP User Language** program configures cascading submenus for the pulldown menu interfaces of the Windows/Motif **CAM Processor** modules. This allows for easy submenu function location (and activation) without having to activate (and probably cancel) submenus. The function repeat facility provided through the right mouse button supports cascading menus to simplify repeated submenu function calls.

Windows/Motif Parameter Setup Dialogs

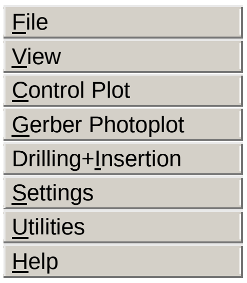
The following Windows/Motif parameter setup dialogs are implemented for the **CAM Processor**:

- Settings - Settings: General CAM/Plot Parameters
- View - Settings: Display Parameters
- Control Plot - Settings: Control Plot Parameters
- Gerber Photoplot - Settings: Gerber Photoplot Parameters
- Drilling+Insertion - Settings: Drilling Data Output Parameters

The **UIFSETUP User Language** program replaces the parameter setup functions of the Windows and Motif pulldown menus with the above menu functions for activating the corresponding parameter setup dialogs.

Windows/Motif Pulldown Menu Konfiguration

When using pulldown menus under Windows and Motif, the **UIFSETUP User Language** program configures the following modified **CAM Processor** main menu with a series of additional functions and features:



4.7.4 In-built CAM Processor System Features

Automatic Parameter Backup

The **CAM Processor** provides an in-built feature for automatically saving important operational and plot parameters with the currently processed layout and/or part symbol. The following parameters are stored to the current design file when changing to the BAE main menu or to the **Layout Editor** (i.e., CAM parameter backup can only be suppressed when exiting BAE):

- Name of the currently loaded Layout Color Table
- Input Grid
- Display Grid
- Coordinate Display Mode
- Wide Line Draw Start Width
- Output Mode Board Outline
- Output Mode All Signal Layers
- Output Mode Plot Marker Layer
- Plot/CAM Accuracy/Symbol Tolerance
- Plot/CAM Origin Coordinate
- Plot/CAM Rotation Mode
- Plot/CAM Mirroring Mode
- Power Layer Heat Trap Minimum Distance
- Power Layer Isolation Minimum Distance
- Power Layer Heat Trap Range/Tolerance
- Power Layer Isolation Range/Tolerance
- Power Layer Border Width
- Power Layer Isolation Drawing Width
- Control Plot Output File Name/Device
- Control Plot Scaling Factor
- HP-GL Plot Pen Width/Standard Line Width
- HP-GL Plot Speed
- HP-GL Plot Filling Mode
- Gerber Output File Name/Device
- Gerber Aperture Table Name
- Gerber Format/Plotter Units
- Gerber Standard Line Width
- Gerber Fill Mode
- Gerber Arc Plotting Mode
- Drilling Data Output File Name/Device
- Drilling Tool Table Output File Name/Device
- Drilling Tool Tolerance/Accuracy
- Insertion Data Output File Name/Device

Parameter sets are stored with special names according to the currently processed layout database hierarchy level. The layout element name is used for layout elements, parameter set name `[part]` is used for layout part symbol elements, `[padstack]` is used for layout padstack elements and `[pad]` is used for layout pad elements. When loading an element, the corresponding parameter set is automatically loaded as well, thus providing a convenient way of activating a default parameter set suitable for processing the selected design and/or library element type.

Output Device

The Plot Device functions from the Control Plot and Gerber Photoplot menus are used to specify the CAM data output device. The output can either be written to a file, or directly to the plotter and/or printer. When plotting to a file, the desired output file name must be specified. When writing directly to the plotter, the name of the output port where the plotter is connected (e.g., `com2`, `lpt1` under DOS) must be specified. Take care that enough disk space is available when writing to a file and that the corresponding interface is correctly initialized when directing output to a hardware device. CAM output aborts with one of the following error message if neither of these requirements are met:

```
Error creating file!
Error writing ASCII file!
```

The **CAM Processor** will prompt for the output device name after activating the desired output function, if no default plot device has been been specified with the Plot Device function. Popup menus for fast output file selection are integrated to the Plot Device, HP-GL Output, Postscript Output, HP Laser Output, Dump Aperture Table, Gerber Output, Drill Data Device, Tool Table Device, Drill Output, Tool Table Output and Insertion Output functions of the Control Plot, Gerber Photoplot and Drilling+Insertion menus. Files ending on `.ass`, `.con`, `.ddb`, `.def`, `.exe`, `.fre`, `.ulc` and `.usf` are faded out from the output file menus for data security reasons. New output file names can optionally be typed in via keyboard.

User Language

The **Bartels User Language Interpreter** is integrated to the **CAM Processor**, i.e., **User Language** programs can be called from the **CAM Processor**, and it is possible to implement any user-specific **CAM Processor** function required such as status display, parameter setup, reports and test functions, special CAD/CAM output and plot functions, customer-specific batch procedures, etc.

The **CAM Processor** provides both explicit and implicit **User Language** program call facilities. **User Language** programs can be started with explicit program name specification using the Run User Script function from the File menu (empty string or question-mark (**?**) input to the program name query activates a **User Language** program selection menu).

User Language programs can also be called by simply pressing special keys of the keyboard. This method of implicit **User Language** program call is supported at any time unless another interactive keyboard input request is currently pending. The name of the **User Language** program to be called is automatically derived from the pressed key, i.e. pressing a standard and/or function key triggers the activation of a **User Language** program with a corresponding name such as **cam_1** for digit key 1, **cam_r** for standard key r, **cam_#** for standard key #, **cam_f1** for function key **F1**, **cam_f2** for function key **F2**, etc.

The **CAM Processor User Language Interpreter** environment also features event-driven **User Language** program calls, where **User Language** programs with predefined names are automatically started at certain events and/or operations such as **CAM_ST** at **CAM Processor** module startup, **CAM_LOAD** after loading a design element, **CAM_SAVE** before saving a design element, **CAM_TOOL** at the selection of a toolbar item and **CAM_ZOOM** at the change of the zoom factor. The module startup **User Language** program call method is most useful for automatic system parameter setup as well as for key programming and menu assignments. The element save and load program call methods can be used to save and restore element-specific parameters such as the zoom area, color setup, etc. The toolbar selection event must be used to start **User Language** programs which are linked to toolbar elements. The zoom event can be used to apply an update request to a design view management feature.

Bartels User Language also provides system functions for performing key programming, changing menu assignments and defining toolbars. These powerful features can be applied for user interface modifications. Please note that a large number of additional functions included with the **CAM Processor** menu are implemented through the **User Language** programs delivered with the BAE software.

See the Bartels User Language Programmer's Guide for a detailed description of the **Bartels User Language** (chapter 4.2 lists all **User Language** programs provided with the BAE software).

Neural Rule System

A series of advanced **Bartels AutoEngineer** features are implemented through the integrated **Neural Rule System**. See chapter 6.3.2 for the rule system applications provided with the **Bartels AutoEngineer**. Some of these applications allow for special manufacturing data outputs and for extended control over manufacturing data generation processes.

4.7.5 Plot Parameters

The **CAM Processor** supports a series of general parameters for controlling the CAM output such as plot tolerance, CAM rotation, CAM mirror mode, etc. These parameters can be changed using the functions provided with the Plot Parameters menu. It is strongly recommended to set appropriate plot parameters before generating CAM output.

Each input prompt at the specification of plot parameters includes a parentheses-enclosed display of the current parameter value setting. On invalid parameter input values the **CAM Processor** issues an error message such as `Invalid numeric value!`, and the current parameter value is left unchanged.

Plot Parameter Save/Load

CAM plot parameter settings are automatically stored with the currently processed design file when switching to another BAE module (see above, chapter 4.7.4). The **CAM Processor** File menu also provides the Save Parameters and Load Parameters functions. Save Parameters can be used to save the current CAM plot parameter settings to a different DDB file. Load Parameters can be used to load CAM parameter settings from a selectable DDB file.

Plot/CAM Origin

The Plot/CAM Origin function is used to set the origin of the output coordinate system. When loading a layout element the CAM origin is automatically set to the lower left corner of that element. When selecting the plot rotate or the plot mirror option (see below), then the CAM origin is automatically moved from the lower left to the upper left corner of the currently loaded element (or vice versa). The Plot/CAM Origin function can be used to set the CAM origin to an arbitrary point inside the element boundaries. It is always possible to avoid negative output coordinates which certain output devices might not be able to understand and/or process.

The plot parameter dialog of the **CAM Processor** Windows and Motif versions provides a button for resetting the CAM origin to its default position.

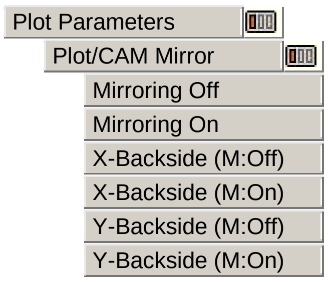
Plot/CAM Rotate

The Plot/CAM Rotate function provides the options No Rotate and Left Rotate. The No Rotate option sets the 0 degree plot rotation (which is the default). The Rotate Left option produces a 90 degree counter-clockwise rotated plot output. Negative output coordinates might be produced when rotating plots. Some printers or plotters refuse to process negative coordinates. The Rotate Left option automatically moves the CAM origin from the lower left to the upper left corner of the element (or vice versa), thus ensuring positive output coordinates only. However, Rotate Left does not change arbitrary CAM origin settings, i.e., positive output coordinates must then be enforced by replacing the CAM origin manually (see above).

To indicate CAM rotation mode settings, a mirror mode specific example text and an arrow are placed at the CAM origin marker if CAM mirroring and/or rotation is selected.

Plot/CAM Mirror

The Plot/CAM Mirror function provides options for mirroring the CAM output:



Mirroring Off produces output as displayed on the screen. Mirroring On produces output mirrored at the X-axis crossing the CAM origin. X-Backside (M:Off) mirrors all coordinates at the X-axis crossing the CAM origin, except for the texts which are plotted unmirrored. X-Backside (M:On) performs X-axis text mirroring only. Y-Backside (M:Off) mirrors all coordinates at the X-axis crossing the CAM origin, and additionally performs Y-axis text mirroring. Y-Backside (M:On) performs Y-axis text mirroring only. Figure 4-10 illustrates the effects of the different mirror options.

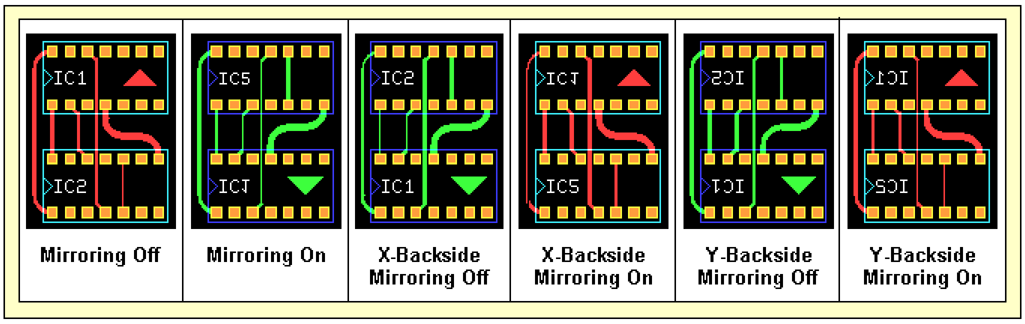


Figure 4-10: CAM Mirror Modes

With all mirror modes, the position of the CAM origin must be considered. With mirroring activated, the CAM origin should be set to the upper left corner of the layout element to ensure positive coordinates output. With the CAM origin set to default, Plot/CAM Mirror automatically replaces the CAM origin to produce positive output coordinates only.

To indicate CAM mirror mode settings, a mirror mode specific example text and an arrow are placed at the CAM origin marker if CAM mirroring and/or rotation is selected.

All Layers Mode

The All Layers Mode function is used to designate how objects should be treated, which are placed on signal layers All Layers and Middle Layers or on both sides of documentary layers. With documentary layers, the All Layers Mode refers to side 1 and side 2 and corresponds with the Both Side options of the documentary layer menus. With signal layers, the All Layers Mode refers to the All Layers and Middle Layers signal layer selections. With option Plot Separate, all layer objects are only plotted when selecting the corresponding plot layer (All Layers, Middle Layers or Both Sides with documentary layers). With option Plot Together, all layer objects are plotted together with the selected plot layer. I.e., when plotting signal layer 1 with Plot Together the objects on signal layer All Layers are also plotted, when plotting Side 2 of documentary layer Insertion Plan the objects on Insertion Plan - Both Sides are also plotted, etc. On default, the Plot Together option is used.

The All Layer Mode plot parameter of the **BAE HighEnd CAM Processor** also provides the Plot Connected, Pins & connected Vias and Vias & connected Pins options. These modes restrict the output of inside layer pin and/or via pads to only those pads which have connections to other elements on the selected inside layer. Plot connected plots only connected inside layer pin and via pads. Pins & connected Vias plots all inside layer pin pads and only connected inside layer via pads. Vias & connected Pins plots all inside layer via pads and only connected inside layer pin pads. Inside layers are all signal layers except for layer 1 and the signal layer selected with the Set Top Layer function.

Symbol Tolerance

The Symbol Tolerance function is used to set the plot accuracy for filling areas and/or creating structures using the available set of tools. It allows to plot elements to be smaller by the specified plot tolerance value if this creates more efficient output. E.g., when photoplotting a round pad with 0.062 inch diameter and the plot tolerance set less than 0.002 inch, then the 0.060 inch aperture is used to draw in a circle to produce the correct size; if the tolerance is set greater than 0.002 inch then the 0.060 inch aperture is flashed, since it is smaller than the required size but within the tolerance and much more efficient.

The plot tolerance is only calculated to the inside of the structures to be plotted since, otherwise, short-circuits might be produced. The default plot tolerance value is 0.15mm. For structures which cannot be plotted correctly using the smallest aperture and/or pen width, the **CAM Processor** uses the smallest available tool to generate that structure. Such structures produce overdraw plot errors which are reported and/or indicated with highlight. It is strongly recommended to refrain from passing CAM output with overdraw errors to the PCB manufacturer since this could produce PCBs containing short-circuits. The Remove Error Marks function from the Utilities menu can be used to reset the overdraw error display.

Board Outline and Plot Markers

The Border Mode and Reg. Marks Mode functions are used to designate whether the board outline and/or the plot markers should be plotted together with other layers. With the Plot Border Off option of the Border Mode function the board outline can be plotted only when selecting the Border plot layer. With the Plot Border On option, the board outline is plotted together with any other plot layer. With the Plot Reg.Marks Off option of the Reg. Marks Mode function, the plot markers can be plotted only when selecting the corresponding plot markers documentary layer. With the Plot Reg.Marks On option, the plot markers are plotted together with any other plot layer. The plot markers documentary layer can be defined with the **`LAYPLTMARKLAY`** command of the **BSETUP** utility program (see chapter 7.2 for more details). On default the Plot Border On and the Plot Reg.Marks On options are used.

4.7.6 Power Layers

When plotting power layers, the **CAM Processor** automatically generates negative plots using special power layer plot parameters. Text placed on power layers is plotted using the standard line width or the selected pen width.

When editing power layers, the **Layout Editor** obviously does not know about the power plot parameters which will later be set in the **CAM Processor**. I.e., the **Layout Editor**'s DRC cannot perform complete electric checking on power layer plot data generated in the future. It is strongly recommend to perform extensive visual checks on power layer plots before passing such plot data to the film and/or PCB manufacturer (use the **CAM View** module). Connectivity-changing isolations might be produced, if the power plot isolation width parameters are set too large.

Power Layer Border

The Power Layer Border function from the Plot Parameters menu is used to define the PCB power layer border width, i.e., the copper-free PCB boundary area to be generated when plotting power layers. The default value for the power layer border width is 2.1mm. If the PCB board outline contains arc-shaped segments, then an appropriate tool (aperture, standard line width or pen) for drawing the board outline with the specified power layer border width is required.

Split Power Planes and Power Plane Isolation

Both active and passive power planes placed on power layers are generated by plotting the power plane outline which then is the isolation line of the power plane. Processing active power planes with specific net assignment generates split power planes, whilst passive power planes without net assignment result in isolated areas on the power layer. The width of the power plane isolation can be adjusted with the Power Layer Isolation function from the Plot Parameters menu. The default power plane isolation width is 0.3mm.

Power Layer Pin Isolation

Pins not connected to the power layer and/or the power plane net are isolated by placing filled circles at the corresponding pin drill holes. The isolation circle diameters are calculated from the drill hole diameters and the plot parameters to be set with the P-IS Min. Distance and P-IS Tolerance functions from the Plot Parameters menu. The P-IS Min. Distance function is used to specify the minimum distance between the edge of the drill hole and the outside of the isolation circle placed in the power layer. The default value for the power layer isolation minimum distance is 0.4mm. The P-IS Tolerance function is used to specify the outward tolerance of the isolation area. The default value for the power layer isolation is 0.5mm. The minimum isolation circle diameter is calculated as in

```
Drill Diameter + 2 × (P-IS Min. Distance)
```

The maximum isolation circle diameter is calculated as in

```
Drill Diameter + 2 × (P-IS Min. Distance) + 2 × (P-IS Tolerance)
```

Figure 4-11 illustrates the effects of the minimum isolation distance and the isolation tolerance power plot parameter settings.

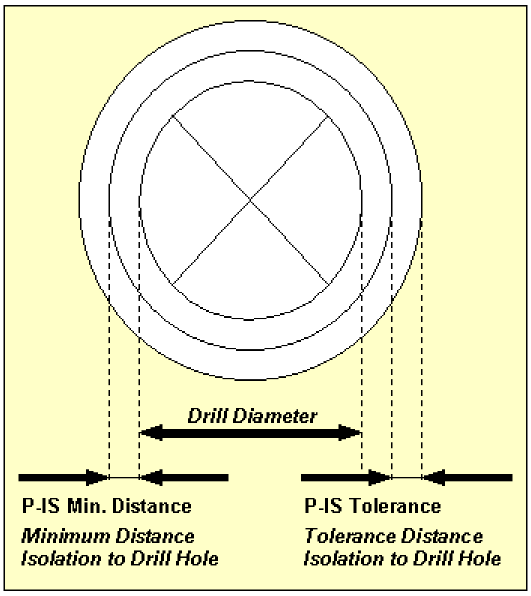

Drill Diameter
P-IS Min. Distance
Minimum Distance Isolation to Drill Hole
P-IS Tolerance
Tolerance Distance Isolation to Drill Hole

Figure 4-11: CAM Power Layer Isolation

Heat Traps

Pin connections into power layers or to power planes placed on power layers are automatically generated with heat trap construction. A heat trap is constructed with four 45 degree isolation circle segments placed round the drill hole of the pin, thus preventing from too much heat flow. The heat trap circle diameters are calculated from the drill hole diameters and the plot parameters to be set with the P-HT Min. Distance and P-HT Tolerance functions from the Plot Parameters menu. The P-HT Min. Distance function is used to specify the minimum distance between the edge of the drill hole and the outside of the heat trap circle. The default value for the heat trap minimum distance is 0.4mm. The P-HT Tolerance function is used to specify the outward tolerance of the heat trap. The default value for the heat trap tolerance is 0.5mm. The heat trap circle diameters are calculated analogously to the isolation circles (see above).

With Gerber photoplot, the **CAM Processor** checks if there is a thermal aperture available with a diameter matching the calculated tolerance range, which can be used to flash the heat trap. Otherwise, the heat trap is drawn using the smallest round aperture valid for line strokes.

Use the following commands to set the power layer border width to 0.3mm, the power plane isolation width to 0.25mm, the minimum pin isolation width to 0.35mm with a tolerance of 0.4mm (i.e., maximum pin isolation width 0.75mm) and the heat trap distance to 0.35mm with 0.45 tolerance (i.e. maximum heat trap distance 0.8mm):

```
Plot Parameters
    Power Layer Border
        Power Layer Border Width ( 2.10mm) ?  0.3
    Power Layer Isol.
        Power Layer Isolation Width ( 0.30mm) ?  0.25
    P-IS Min. Distance
        Min. Distance Isolation to Drill Hole ( 0.40mm) ?  0.35
    P-IS Tolerance
        Tolerance Isolation to Drill Hole ( 0.50mm) ?  0.4
    P-HT Min. Distance
        Min. Distance Heat Trap to Drill Hole ( 0.40mm) ?  0.35
    P-HT Tolerance
        Tolerance Heat Trap to Drill Hole ( 0.50mm) ?  0.45
```

4.7.7 HP-GL Output

The HP-GL Output function from the Control Plot menu is used to produce pen plots in HP-GL (Hewlett-Packard Graphics Language) format.

The Plotter Pen Width function is used to define the pen width for generating HP-GL output data. Structures smaller than the plotter pen width cannot be drawn without overdraw errors. The default plotter pen width is 0.3mm. Pen width value specifications can range from 0.01mm to 10.0mm.

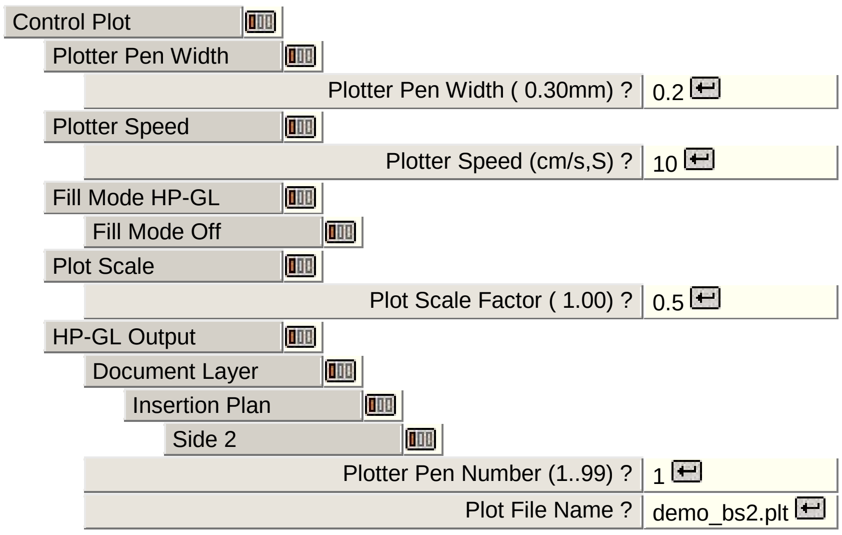
The Plotter Speed function is used to control the pen plotter speed, i.e., the speed at which the pen can be moved by the HP-GL plotter. The system accepts either `S` input for maximum speed or non-negative integer plotter speed values in cm/s units. The default plotter speed corresponds with `S` (maximum speed). The maximum speed value specification is 99cm/s.

The HP-GL coordinates are written in plotter units where 40.2 plotter units are 1 millimeter. Some HP-GL "compatible" plotters are based on 40 plotter units per millimeter. The Plot Scale function can be used to specify a scale factor for producing true to scale output with non-default plotter units. The default plot scale value is 1.0 for true to scale output. Plot scale specifications can range from 0.1 to 100.0.

The Fill Mode HP-GL function provides the options Fill Mode Off and Fill Mode On. Fill Mode Off causes the system to refrain from filling the plot structures, i.e., only outlines are drawn for filled areas. Switching off the fill mode is most useful for fast control plots. Fill Mode On is used to select the default fill mode.

The HP-GL Output function is used to produce HP-GL pen plots. The user is prompted for the plot layer and the pen number (1..99), and for the name of the output file (if no default output device was specified with Plot Device). Note the Multiple Layers option provided with the layer selection menu of the HP-GL Output function. This option can be used for writing the plot data of *multiple* selectable layers to a single output file. With the Multiple Layers option a popup menu is provided for selecting and/or deselecting the plot data output layers using the right mouse button. Layer selections with the left mouse button also allow for layer-specific pen number selections (on default pen 1 is used for each layer). The Col. button is used to select all currently visible layers for output, with the pen numbers automatically set to modulo eight of the corresponding layer colors.

Use the following commands to set the plotter pen width to 0.2mm, set the plotter speed to 10cm/s, switch off the fill mode and plot the documentary layer Insertion Plan - Side 2 of the currently loaded layout to an HP-GL plot file named `demo_bs2.plt` using pen number 1:



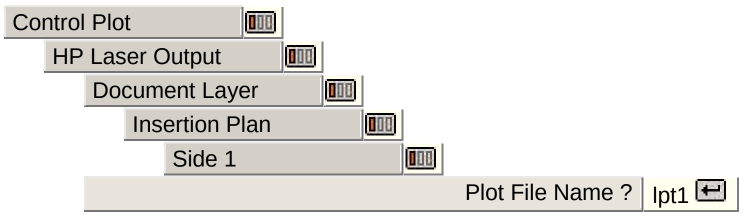
Control Plot
Plotter Pen Width
Plotter Pen Width (0.30mm) ? 0.2
Plotter Speed
Plotter Speed (cm/s,S) ? 10
Fill Mode HP-GL
Fill Mode Off
Plot Scale
Plot Scale Factor (1.00) ? 0.5
HP-GL Output
Document Layer
Insertion Plan
Side 2
Plotter Pen Number (1..99) ? 1
Plot File Name ? demo_bs2.plt

After successfully generating the HP-GL plot file, the **CAM Processor** issues a `HP-GL plot done (<n> errors)!` message where `<n>` is the number of overdraw errors. Overdraw errors are highlighted on the layout.

A `PG` HPGL command is added to the end of HPGL output files to force a page eject and to prevent from subsequent plots being plotted onto the same sheet. The Multiple Layers option must be used for plotting multiple layers onto the same sheet.

4.7.8 HP Laser Output

The HP Laser Output function is used to produce HP Laser plots in PCL (Printer Command Language) format. HP Laser plots are automatically scaled to A4 paper size, i.e., neither plot scaling factor nor pen width settings have effect. Use the following commands to generate an HP Laser plot for documentary layer Insertion Plan - Side 1 of the currently loaded layout, and direct the output to **`lpt1`** (e.g., for interfacing an appropriate DOS-connected laser writer device):



Note the Multiple Layers menu item provided with the layer selection menu of the HP Laser Output function. This option can be used for writing the plot data of *multiple* selectable layers to a single output file. With the Multiple Layers option a popup menu is provided for selecting and/or deselecting the plot data output layers using the left or right mouse button. The Col. button is used to select all currently visible layers for output.

The **CAM Processor** issues a **`HP Laser output done (scale 1:...)!`** message after successfully generating the PCL plot. The scale information in the status message indicates any non-default scaling factor being used for automatic plot to sheet size scaling.

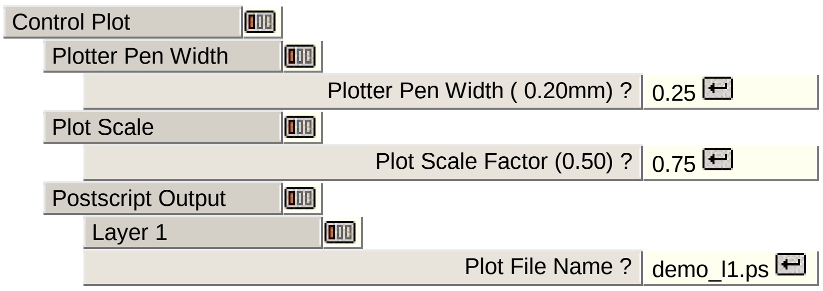
Binary copy mode is required for transferring PCL plot data to the desired output device. I.e., the **`/b`** option must be used when sending PCL files from hard disk to laser printer with the DOS COPY command as in

```
> copy pclplot lpt1 /b
```

with **`pclplot`** being the name of the previously written PCL plot file and **`lpt1`** designating the output device.

4.7.9 Postscript Output

The Postscript Output function from the Control Plot menu is used to produce output in Postscript format. Use the following commands to generate a Postscript output file named **demo_l1.ps** for signal layer 1 of the currently loaded layout with the standard line width set to 0.25mm and the scaling factor set to 0.75:



```
Control Plot
  Plotter Pen Width
    Plotter Pen Width ( 0.20mm) ? 0.25
  Plot Scale
    Plot Scale Factor (0.50) ? 0.75
  Postscript Output
    Layer 1
    Plot File Name ? demo_l1.ps
```

Note the Multiple Layers option provided with the layer selection menu of the Postscript Output function. This option can be used for writing the plot data of *multiple* selectable layers to a single output file. With the Multiple Layers option a popup menu is provided for selecting and/or deselecting the plot data output layers using the left or right mouse button. The Col. button is used to select all currently visible layers for output.

The **CAM Processor** issues a **Postscript output done!** message after successfully generating the Postscript output.

4.7.10 Windows Generic Output

A generic print/plot output function is implemented with the Windows versions of the BAE PC software. I.e., *any* print/plot output feature supported by the current Windows operating system configuration is also supported with the **CAM Processor** of the BAE Windows software.

Use the following commands to activate the Windows print/plot menu:

Control Plot
Generic Output

Note the Multiple Layers option provided with the layer selection menu of the Generic Output function. This option can be used for writing the plot data of *multiple* selectable layers to a single output file. With the Multiple Layers option, a popup menu is provided for selecting and/or deselecting the plot data output layers using the right mouse button. Layer selections with the left mouse button also allow for layer-specific color selections for color plots by assigning pen numbers which correspond with indices to the current color table. The Col. button is used to select all currently visible layers for output, with the color table indices automatically set to modulo-8 layer color numbers.

The Windows printer dialog specifications for the number of copies, page sorting mode and page range are considered by the Generic Output function.

The All Pages option from the Windows printer dialog of the Generic Output function plots all elements of the currently loaded database class. This allows for the output of, e.g., all layouts of a project. All layouts are plotted according to layout-specific plot parameter settings such as plot rotation modes.

The Selection option from the Windows printer dialog allows for the selection of the print area for generic outputs.

The generic output is automatically scaled to the print page format selected through the Windows printer setup if the size of the element to be plotted exceeds the paper size. The page aspect ratio is maintained when automatic plot scaling is applied. The status message of the Generic Output function provides information to indicate any non-default scaling factor being used for automatic plot to sheet size scaling.

4.7.11 Bitmap Plot Output to Windows Clipboard

The Output to Clipboard function from the Control Plot menu can be used under Windows for plotting a bitmap of the currently loaded element into the clipboard ready to be imported (e.g., with Paste) to other Windows applications capable of processing bitmaps. The whole element is plotted on default. The Clipping On option can be used to restrict the output to a mouse-selectable rectangle. The plot dialog box also allows for bitmap size specifications and plot rotation mirror mode selections.

A black on white plot is generated for single output layer selections. With Multiple Layers plot output, the selected layers are plotted with their currently assigned colors (and mixed color display) on black background.

4.7.12 Gerber Photoplot

Aperture tables are required for Gerber photoplot outputs. The Gerber Photoplot menu of the **CAM Processor** provides functions for the definition and administration of Gerber aperture tables. Aperture tables are stored to the `cam.dat` system file in the BAE programs directory. When installing new BAE software versions you should take care not to overwrite `cam.dat` (either use update install mode or make a backup of the `cam.dat` system file before installing the BAE software; see the Bartels AutoEngineer® Installation Guide for more details).

Gerber Construction Techniques

It is important to make the Gerber aperture tables as efficient as possible to minimize the Gerber photoplot data. The reason for this is that an unnecessarily large Gerber file takes up more media space, takes longer to transfer by modem, and takes longer to copy. If the plot is produced by a vector plotter it will also take longer to produce, and cost more. To make the Gerber data more efficient it is important to take into consideration the way the photoplotter plots pads and tracking. The photoplotter has two drawing modes, called line draw and flash. A flash is one single flash of light of a size specified with the aperture, which produces an image of the same size on the plot film. For areas that match apertures, a flash code containing only one set of output coordinates is used. For areas that don't match apertures efficient drawing techniques are used where possible, generating few coordinates but with some shapes and sizes more coordinates are needed. With careful design of pads and appropriate Gerber apertures selected very much smaller Gerber data files can be generated making them quicker to copy, cheaper to send by modem, and require less media space.

For orthogonally placed square and rectangular pads that have a matching aperture a flash is used. For rectangular pads that have sides along the X and Y axis and the smaller side matching a square aperture, that aperture is used to draw the pad. For rectangular pads (and square pads that don't have an aperture that matches) that have sides along the X and Y axis and a square aperture exists that is bigger than half the shortest side, that aperture is used to draw the pad with two strokes. For rectangular pads that don't have their sides along the X and Y axis, area fill techniques are used. For round pads that have an aperture that matches a flash is used. For round pads where no aperture matches, area fill techniques are used where line fill will use the nearest smaller aperture to draw the pad by moving in a circular path. Finger pads (rectangles with semi-circles at two opposite sides) are plotted like trace segments, i.e., line draws with matching circular apertures are applied on finger pads.

Area fill techniques are applied for pads which don't meet the above specification. A draw is used for trace widths matching a round aperture. For trace widths that don't match an aperture, overlapping lines are drawn using the next smallest round aperture. Irregularly shaped areas are always generated using fill techniques.

Gerber Aperture Table

The Load Aperture Table, Save Aperture Table and Del. Aperture Table functions from the Gerber Photoplot menu are used to load, save and delete Gerber aperture tables. The List Aperture Tables function provides a list of all of the available Gerber aperture tables. The Load Aperture Table function issues an error message such as

```
Gerber aperture table overflow!
```

when trying to load aperture tables containing more than 900 apertures. Such errors indicate a corrupt `cam.dat` file since the **Bartels AutoEngineer** it is not capable of creating and/or processing aperture tables with more than 900 apertures. When starting the **CAM Processor**, the aperture table named `standard` is automatically loaded. Table 4-4 lists all apertures defined with the `standard` aperture table.

Table 4-4: Gerber Aperture Table "standard"

D-Code	Aperture Type	Aperture Size [mil]	Aperture Size [mm]	Drawing Mode
D10	round	7.87	0.200	ALL
D11	round	8.27	0.210	ALL
D12	round	9.84	0.250	ALL
D13	round	11.81	0.300	ALL
D14	round	15.75	0.400	ALL
D15	round	19.69	0.500	ALL
D16	round	23.62	0.600	ALL
D17	round	27.56	0.700	ALL
D18	round	31.50	0.800	ALL
D19	round	35.43	0.900	ALL
D20	round	39.37	1.000	ALL
D21	round	43.31	1.100	ALL
D22	round	47.24	1.200	ALL
D23	round	51.18	1.300	ALL
D24	round	59.06	1.500	ALL
D25	round	62.99	1.600	ALL
D26	round	66.93	1.700	ALL
D27	round	78.74	2.000	ALL
D28	round	90.55	2.300	ALL
D29	round	98.43	2.500	ALL
D30	round	102.36	2.600	ALL
D31	round	110.24	2.800	ALL
D32	round	118.11	3.000	ALL
D33	round	129.92	3.300	ALL
D34	round	137.80	3.500	ALL
D35	round	149.61	3.800	ALL
D36	round	157.48	4.000	ALL
D37	round	169.29	4.300	ALL
D38	square	15.75	0.400	ALL
D39	square	19.69	0.500	ALL

		Aperture Size		
D-Code	**Aperture Type**	**[mil]**	**[mm]**	**Drawing Mode**
D40	square	23.62	0.600	ALL
D41	square	29.53	0.750	ALL
D42	square	31.50	0.800	ALL
D43	square	39.37	1.000	ALL
D44	square	43.31	1.100	ALL
D45	square	47.24	1.200	ALL
D46	square	51.18	1.300	ALL
D47	square	59.06	1.500	ALL
D48	square	62.99	1.600	ALL
D49	square	78.74	2.000	ALL
D50	square	86.61	2.200	ALL
D51	square	118.11	3.000	ALL
D52	square	129.92	3.300	ALL
D53	thermal	70.87	1.800	ALL
D54	thermal	86.61	2.200	ALL
D55	thermal	98.43	2.500	ALL

The Edit Aperture Table function is used to edit entries in the currently loaded aperture table. After activating the Edit Aperture Table function, the system provides a listing of the currently loaded aperture table where + and - keys can be used to scroll forward and/or backward in this list, and the return key ↵can be used to return to the **CAM Processor** main menu. For changing a certain aperture table entry the appropriate aperture table index must be entered. Empty string inputs (i.e., pressing the return key ↵) on subsequent aperture specification prompts causes the system to accept predefined values. The first prompt is used to specify the aperture type and/or shape, where **`r`** defines a round aperture, **`q`** defines a square-shaped aperture, **`a`** defines a rectangular aperture, **`t`** defines a thermal aperture and **`s`** defines a special aperture. Note that special apertures are not utilized by the **CAM Processor** (special aperture types are reserved for future functions). A dash string input (**`-`**) to the aperture type prompt causes the system to delete the selected entry from the aperture table. After specifying the aperture type, the system prompts for the diameter and/or the edge length(s) of the aperture. The aperture size can be input in either mm or inch units and is automatically be converted to mm units in the aperture table. The next prompt is used for specifying the aperture drawing mode, where **`f`** (Flash) defines the aperture to be used for flashes only, **`l`** (Line) defines the aperture to be used for line drawing only and **`a`** (All) defines for either flash or line drawing whatever is appropriate. The last aperture definition prompt is used for specifying the Gerber D-Code number of the aperture, where valid D-Codes range from 10 to 999. It is strongly recommended to ensure unique D-Codes in each Gerber aperture table to prevent the **CAM Processor** from random use of multiple defined D-Codes.

Note that square-shaped and rectangular aperture can be used for flashing orthogonal placed pads only. It is also strongly recommended to check whether the PCB manufacturer supports rectangular apertures.

Use the following commands to define a round line aperture with 0.3mm diameter and D-Code 10 at aperture table index 1:

Gerber Photoplot	▥
Edit Aperture Table	▥
Gerber Aperture Table Index (1..900,+,-) ?	1 ↵
Aperture (R)ound/S(Q)uare/(T)hermal/(S)pecial/(A)rea/(-) ?	r ↵
Aperture Diameter/Side Length (0.127mm) ?	0.3 ↵
Drawing Mode (A)ll/(F)lash/(L)ine ?	l ↵
Gerber D-Code Number (10-999) ?	10 ↵
Gerber Aperture Table Index (1..900,+,-) ?	↵

An empty string input (i.e., pressing the return key ↵) to the aperture table index prompt causes the system to return to the **CAM Processor** main menu. The Dump Aperture Table function is used to generate an ASCII listing of the currently loaded aperture table. The Save Aperture Table function is used to save the currently loaded aperture table with a selectable name in the `cam.dat` system file.

Gerber Format, Optimized Gerber Output

The Gerber Format function is used to specify the Gerber output format. The Gerber 2.3 Format option produces 1/1000 inch integer coordinates. The Gerber 2.4 Format option produces 1/10000 inch integer coordinates. Gerber 2.3 Dormat is selected on default.

The Gerber optimized 2.3 and Gerber optimized 2.4 options are provided for optionally generating optimized Gerber output with redundant D01 ("light off") plotter control commands eliminated, thus significantly reducing the amount of Gerber output plot data. However, it is strongly recommended to check whether the PCB manufacturer supports optimized Gerber format.

Standard Line Width

The Default Line Width function is used to define the standard line width for Gerber output. The standard line width is used for drawing graphic lines and texts. A round aperture for line drawing is required for the standard line width; otherwise the **CAM Processor** issues an error message such as

```
Cannot realize the default line width!
```

when trying to generate Gerber photoplots. The standard line width is set to 0.3mm on default.

Gerber Fill Mode

The Gerber Fill Mode function is used to select the technique for filling irregularly shaped areas (i.e., those areas not matching any aperture for flash or line stroke).

Option Line Fill is used to select the single-aperture line fill technique which uses the smallest line aperture for filling irregularly shaped areas with line strokes.

Option Multi Aperture Fill selects the multi-aperture fill technique, which is the default. The algorithm used by the multi-aperture fill technique fills areas from the area outline using the smallest aperture and proceeds using larger apertures when moving to the inner of the area. This results in distinct reduction of Gerber data output when using appropriate Gerber aperture tables for layouts with many irregularly shaped areas.

With the G36/G37 Fill mode option activated, non-flashable structures are stored with their outline contour, and will be filled by the photo plotter. This feature significantly reduces the amount of Gerber plot data and also eliminates plot overdraw errors. However, it is strongly recommended to check with the PCB manufacturer whether the photo plotter is capable of processing G36/G37 Gerber data.

Gerber Arc Mode

The Gerber Arc Mode function is used to enable Gerber I/J arc commands (option Gerber Arcs) instead of applying arc interpolation (default option Approx. Arcs). Gerber arc commands should always be used if the photo plotter is capable of interpreting these commands since this feature results in considerably reduced amounts of Gerber output data.

Extended Gerber

The Extended Gerber function can be used for optionally generating RS-274-X format Gerber output (Extended Gerber with Embedded Apertures). Extended Gerber provides the options No Extended Gerber, Extended Gerber fixed and Extended Gerber dynamic.

No Extended Gerber is the default mode for generating standard Gerber data without embedded apertures. With Extended Gerber mode selected, the Gerber aperture table and the Gerber format selection are included (i.e., embedded) with the Gerber output plot file. The Gerber aperture table to be used and/or embedded can either be the currently active aperture table (option Extended Gerber fixed) or a dynamically (i.e., automatically) generated aperture table (option Extended Gerber dynamic). Note that the dynamic aperture table is generated for the whole layout data rather than for the currently plotted layer only. With Extended Gerber the aperture table is automatically transferred to the PCB manufacturer as an integral part of the plot file, and no extra aperture table information file must be generated and/or transferred. With the Extended Gerber dynamic option, there is also no need for defining and/or activating an aperture table before starting the Gerber plot output.

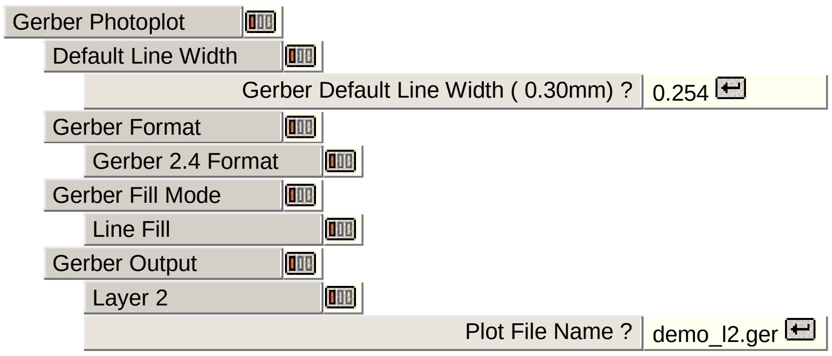
Warning

It is strongly recommended to check whether the PCB manufacturer supports Extended Gerber format before applying this feature. It is also recommended to refrain from using the Extended Gerber dynamic option with multi-aperture fill mode activated since automatically generated aperture tables usually are not appropriate for multi-aperture filling.

Gerber Photoplot Output

The Gerber Output function is used to start Gerber photoplot output for a selectable plot layer. Note the Multiple Layers option provided with the layer selection menu of the Gerber Output function. This option can be used for writing the photoplot data of *multiple* selectable layers to a single output file. With the Multiple Layers option a popup menu is provided for selecting and/or deselecting the plot data output layers using the left or right mouse button. The Col. button is used to select all currently visible layers for output.

Use the following commands to generate a Gerber photoplot output file named **`demo_l2.ger`** for signal layer 2 of the currently loaded layout, using the currently loaded Gerber aperture table with standard line width 0.254mm, Gerber format 2.4 and line fill mode:


Gerber Photoplot
Default Line Width
Gerber Default Line Width (0.30mm) ? 0.254
Gerber Format
Gerber 2.4 Format
Gerber Fill Mode
Line Fill
Gerber Output
Layer 2
Plot File Name ? demo_l2.ger

The following report is issued by the Gerber Output function after successfully generating the Gerber plot file:

```
Number of Flash Structures .............: <f>
Number of Rectangle Drawn Areas ........: <r>
Number of Circle Drawn Areas ...........: <c>
Number of Multi Aperture Areas .........: <m>
Number of Line Filled Areas ............: <l>
Number of Heat Traps ...................: <w>
Number of Overdraw Errors ..............: <e>
```

`<f>` is the number of structures which have been generated using a flash aperture. **`<r>`** is the number of rectangular-shaped areas which have been generated using a square line aperture. **`<c>`** is the number of circles which have been generated with a circular draw of a smaller round line aperture. **`<m>`** is the number of structures which have been generated using multi-aperture fill techniques. **`<l>`** is the number of structures which have been generated using line fill techniques. **`<w>`** is the number of heat traps which have been drawn using the smallest round line aperture. **`<e>`** is the number of structures which could not be generated within the specified plot tolerance using the smallest aperture.

It is strongly recommended to refrain from passing Gerber photoplot output with overdraw errors to the PCB manufacturer since there might be PCBs produced containing short-circuits.

The **CAM View** module of the **Bartels AutoEngineer** can be used to perform visual checks and panelization on the generated Gerber data (see chapter 4.8 for a description of the **CAM View** module).

4.7.13 Drill Data

The Drilling+Insertion menu provides functions for generating drilling data output in Excellon II and/or Sieb&Meier format. The **CAM View** module can be used to perform visual checks, coordinates sorting and panelization on the generated drilling data (see chapter 4.8 for more details).

Excellon

The Excellon Drill Out function from the Drilling+Insertion menu is used for generating drilling data in Excellon II format. With Excellon II format, both the drilling data and the drilling tool table are stored to a single output file.

The Drill Data Device function can be used to specify a default name for the Excellon output file. The Tool Range function is used to define the tolerance for selecting drill tools. The default value for the drill tolerance is 0.10mm.

With the Excellon Drill Out function, the drill class must be specified, thus providing a feature for selecting different types of drill holes for output such as plated and/or non-plated drill holes or drill holes defined on blind and buried vias (whichever is assigned with the selected drill class). A dash string input ("-") to the drilling class prompt selects those drill holes for output which are not assigned to any specific drill class.

Sieb & Meier

The Drill Output and Tool Table Output functions from the Drilling+Insertion menu are used for generating drilling data and drilling tool table output in Sieb&Meier format.

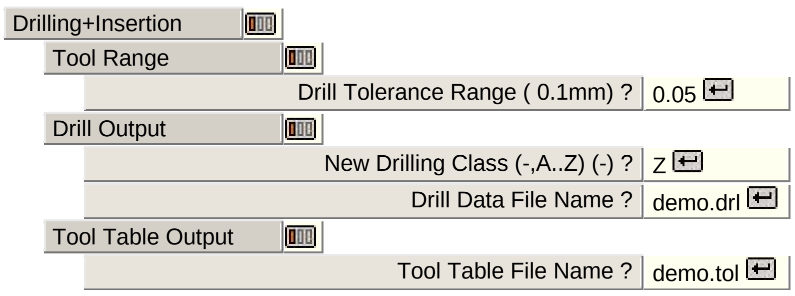
The Drill Data Device and Tool Table Device functions can be used to specify default names for the Sieb&Meier output files. The Tool Range function is used to define the tolerance for selecting drill tools. The default value for the drill tolerance is 0.10mm.

The tool table file to be generated with the Tool Table Output function provides a listing of all drill diameters used for output, with a drill tool number assigned to each drill diameter. Up to 99 different drill diameters can be processed. The output function aborts with the **`Too many drill diameters defined!`** error message if more than 99 different drill diameters are defined on the current layout.

With Sieb&Meier format, each drill hole is listed in a single line providing drill coordinates in 1/100mm integer units. With each drill diameter change, the appropriate drill tool number is appended to the end of the corresponding drill data line, thus representing a command for selecting and/or changing the drill tool.

With the Drill Output function, the drill class must be specified, thus providing a feature for selecting different types of drill holes for output such as plated and/or non-plated drill holes or drill holes defined on blind and buried vias (whichever is assigned with the selected drill class).

Use the following commands to set the drill tool tolerance to 0.05mm, generate the drill data output file **`demo.drl`** for the drill class **`Z`** which is used on the currently loaded layout, and write the drill tool table file **`demo.tol`**:



```
Drilling+Insertion
    Tool Range
        Drill Tolerance Range ( 0.1mm) ?   0.05
    Drill Output
        New Drilling Class (-,A..Z) (-) ?  Z
        Drill Data File Name ?             demo.drl
    Tool Table Output
        Tool Table File Name ?             demo.tol
```

A dash string input ("-") to the drilling class prompt selects those drill holes for output which are not assigned to any specific drill class. The **CAM Processor** issues a **`Drill data write done.`** message after successfully generating the drill data output.

With the commands above, a drill tool table file named **`demo.tol`** with the following contents is generated:

```
/* Drill(Number) Drill Diameter(mm) */
1     0.50
2     0.80
3     0.90
4     1.00
5     1.30
6     3.00
```

With the commands above, a drill data file named **`demo.drl`** with the following contents is generated:

```
%
X1016Y762T6
X1016Y6350
M30
```

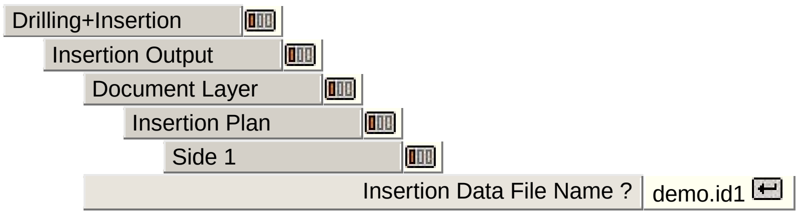
4.7.14 Insertion Data

The Drilling+Insertion menu provides functions for generating output data for automatic insertion equipment and pick and place machines. The **CAM Processor** produces insertion data in generic format where all texts placed on a selectable insertion data layer are listed with coordinates (in 1/100mm integer units) and rotation angles (in degree units).

It is recommended to define a special documentary layer for insertion data output and to place the **`$`** text string on that layer on each part symbol, thus representing the part name placed at the pick point for the automatic insertion equipment (e.g., at pin 1 for parts with drilled pins or at the part pin gravity point for SMDs). The insertion data documentary layer should be defined with the **`PHYSICAL`** text attribute to fix the positions of insertion data text strings. See also chapter 7.2 of this manual for a description of the **`LAYDOCLAYER`** command of the **BSETUP** utility program.

The Insertion Output function from the Drilling+Insertion menu is used to generate insertion data output. After activating the Insertion Output function, the user is prompted for the insertion data layer and the insertion data output file name. Note the Multiple Layers option provided with the layer selection menu of the Insertion Output function. This option can be used for writing the insertion data of *multiple* selectable layers to a single output file. With the Multiple Layers option, a popup menu is provided for selecting and/or deselecting insertion data output layers using the left or right mouse button. The Col. button is used to select all currently visible layers for output.

Use the following commands to generate insertion data output for the solder side of the currently loaded layout (select documentary layer Insertion Plan - Side 1 for output, and direct the output data to a file named **`demo.id1`**):



Insertion Data File Name ? demo.id1

The **CAM Processor** issues the

```
Insertion data write done!
```

message after successfully generating the insertion data output, and the insertion data file **`demo.id1`** should have the following contents:

```
/* Name X(mm) Y(mm) Angle(Degrees) */

C100           4635    3651       0
C101           3619    3651       0
R104            984    5016     270
```

4.8 CAM View

The **CAM View** module provides features for displaying Gerber data ("Gerber View"), Sieb&Meier and/or Excellon drilling data and Excellon milling data in order to check CAM data validity and to estimate the efficiency of tool usage. **CAM View** features multiple input, selective data set movement and sorted output with variable offsets, optional mirroring and adjustable aperture tables, thus supporting panelization. **CAM View** also provides a powerful function for translating Gerber data to BAE layout design data, i.e., with **CAM View** the user is able to import Gerber data produced by foreign PCB layout systems.

4.8.1 Starting the CAM View Module

It is recommended to start the **Bartels AutoEngineer** from the directory where the design files should be generated since this considerably simplifies job file access. If you intend to process the examples provided with this manual it is recommended to move to the BAE examples directory installed with the BAE software. The **CAM View** module can be called from the **Bartels AutoEngineer** main shell. Start the BAE shell by typing the following command to the operating system prompt:

```
> bae ↵
```

Move the menu cursor to the **CAM View** menu item and confirm this choice by pressing the left mouse button:

The **CAM View** program module is loaded and the **CAM View** menu is activated. If this fails to happen then check your BAE software installation (see the Bartels AutoEngineer® Installation Guide for details on how to perform a correct installation).

4.8.2 CAM View Main Menu

The **CAM View** standard/sidemenu user interface provides a menu area on the right side, consisting of the main menu on top and the currently active menu below that main menu. After entering **CAM View**, the Files menu is active and the menu cursor points to the Load Gerber Data function.

The Windows and Motif versions of the **CAM View** module can optionally be operated with a pull-down menu user interface providing a horizontally arranged main menu bar on top. The **WINMENUMODE** command of the **BSETUP** utility program is used to switch between **SIDEMENU** and **PULLDOWN** Windows/Motif menu configurations. See chapter 7.2 for more details.

The following main menu is always available whilst processing CAM data with **CAM View**:

Display
Files
Parameter
Utilities

Display

The View or Display menu can either be activated by selecting the corresponding main menu item or by pressing the middle mouse button. The View or Display menu provides useful functions for changing display options such as zoom window, zoom scale, input and display grid, color settings, drawing mode, coordinate display mode, etc.

Files

The File menu provides functions for loading, moving and writing Gerber, drilling or milling data sets, for generating layouts from Gerber data and for deleting the currently loaded CAM data from main memory.

Parameter

The Parameter menu provides functions for setting input offsets and mirror modes for subsequent data load processes, defining the via D-Code for generating layouts from Gerber data, selecting the Gerber format, the circle mode, the coordinate format, and activating and/or deactivating Extended Gerber format or Gerber optimization when loading and/or writing Gerber data.

Utilities

The Utilities menu provides functions for exiting BAE, returning to the BAE main shell, sorting drill data, reporting aperture usage and listing DDB file contents.

4.8.3 Customized CAM View User Interface

Menu Assignments and Key Bindings

The BAE software comes with **User Language** programs for activating a modified **CAM View** user interface with many additional functions (startups, toolbars, menu assignments, key bindings, etc.). The **BAE_ST** **User Language** program is automatically started when entering the **CAM View** module. **BAE_ST** calls the **UIFSETUP** **User Language** program which activates predefined **CAM View** menu assignments and key bindings. Menu assignments and key bindings can be changed by modifiying and re-compiling the **UIFSETUP** source code. The **HLPKEYS** **User Language** program is used to list the current key bindings. With the predefined menu assignments of **UIFSETUP** activated, **HLPKEYS** can be called from the Key Bindings function of the Help menu. Menu assignments and key bindings can be listed with the **UIFDUMP** **User Language** program. The **UIFRESET** **User Language** program can be used to reset all currently defined menu assignments and key bindings. **UIFSETUP**, **UIFDUMP** and **UIFRESET** can also be called from the menu of the **KEYPROG** **User Language** program which provides additional facilities for online key programming and **User Language** program help info management.

Cascading Windows/Motif Pulldown Menus

The Windows and Motif pulldown menu user interfaces of the **CAM View** module provide facilities for cascading submenu definitions. I.e., submenus can be attached to other menu items. The **UIFSETUP** **User Language** program configures cascading submenus for the pulldown menu interfaces of the Windows/Motif **CAM View** modules. This allows for easy submenu function location (and activation) without having to activate (and probably cancel) submenus. The function repeat facility provided through the right mouse button supports cascading menus to simplify repeated submenu function calls.

Windows/Motif Parameter Setup Dialogs

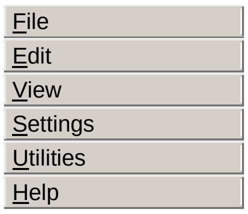
The following Windows/Motif parameter setup dialogs are implemented for the **CAM View** module:

- Settings - Settings: General **CAM View** Parameters
- View - Settings: Display Parameters

The **UIFSETUP** **User Language** program replaces the parameter setup functions of the Windows and Motif pulldown menus with the above menu functions for activating the corresponding parameter setup dialogs.

Windows/Motif Pulldown Menu Konfiguration

When using pulldown menus under Windows and Motif, the **UIFSETUP** **User Language** program configures the following modified **CAM View** main menu with a series of additional functions and features:



4.8.4 In-built CAM View System Features

User Language

The **Bartels User Language Interpreter** is integrated to the **CAM View** module, i.e., **User Language** programs can be called from **CAM View**. It is possible to implement any user-specific **CAM View** function required such as status display, parameter setup, reports and test functions, customer-specific batch procedures, etc.

CAM View provides both explicit and implicit **User Language** program call facilities. **User Language** programs can be started with explicit program name specification using the Run User Script function from the File menu (empty string or question-mark (**?**) input to the program name query activates a **User Language** program selection menu).

User Language programs can also be called by simply pressing special keys of the keyboard. This method of implicit **User Language** call is supported at any time unless another interactive keyboard input request is currently pending. The name of the **User Language** program to be called is automatically derived from the pressed key, i.e. pressing a standard and/or function key triggers the activation of a **User Language** program with a corresponding name such as **cv_1** for digit key 1, **cv_r** for standard key r, **cv_#** for standard key #, **cv_f1** for function key F1, **cv_f2** for function key F2, etc.

The **CAM View User Language Interpreter** environment also features event-driven **User Language** program calls, where **User Language** programs with predefined names are automatically started at certain events and/or operations such as **CV_ST** at **CAM View** module startup, **CV_TOOL** when selecting a toolbar item and **CV_ZOOM** when changing the zoom factor. The module startup **User Language** program call method is most useful for automatic system parameter setup as well as for key programming and menu assignments. The toolbar selection event must be used to start **User Language** programs which are linked to toolbar elements. The zoom event can be used to apply an update request to a design view management feature.

Bartels User Language also provides system functions for performing key programming, changing menu assignments and defining toolbars. These powerful features can be applied for user interface modifications. Please note that a large number of additional functions included with the **CAM View** menu are implemented through the **User Language** programs supplied with the BAE software.

See the Bartels User Language Programmer's Guide for a detailed description of the **Bartels User Language** (chapter 4.2 lists all **User Language** programs provided with the BAE software).

4.8.5 Processing Gerber Data

General Parameters

The Input Offset, Gerber Format, Circle Mode, Mirror Mode, Trailing/Leading Zeros, Extended Gerber, Optimization and Coord.-Spec. functions from the Settings menu are used to set global parameters for loading and/or writing Gerber and/or drilling/milling data sets.

The input offset is added to the input coordinates when loading CAM data. Different input offsets can be specified to load the same layout at different positions, and, subsequently, the whole data can be written to Gerber and/or drill data output files to support panelization. **CAM View** can hold Gerber data from different layers and drill data in main memory at the same time. I.e., only one input offset specification is required for each panel element. The Clear Memory function from the File menu can be used to delete previously loaded CAM data from memory before loading new CAM data for panelization.

The Gerber Format function from the Settings menu is used to specify the Gerber format for loading and/or writing Gerber data. The default option Gerber 2.3 Format selects 1/1000 inch integer units. Option Gerber 2.4 Format selects 1/10000 inch integer units. With option Other Format the user is prompted for the length of one plotter unit, thus supporting arbitrary Gerber format specifications.

The Circle Mode function from the Settings menu is used to select the mode of processing arc segments when loading Gerber data. With the All Arc Values default option, arbitrary arc definitions can be loaded without modification. This is valid for the Gerber data generated by the BAE **CAM Processor** and most of the known foreign systems. Some foreign systems can only generate arcs of up to 90 degrees where the length of the arc center point vector is also specified. In this case, the Max. 90 Degree Arcs option of the Circle Mode function is only furnished for correct Gerber data processing.

The Mirror Mode function from the Settings menu is used for optionally mirroring the input data set(s) to be subsequently loaded. The mirror modes provided with the Mirror Mode function are Mirroring Off (default), Mirror at X-Axis, Mirror at Y-Axis and Mirror at Origin, respectively. Note that input data mirroring is always done *before* any input data offset specification applies.

The Trailing/Leading Zeros function from the Settings menu is used to specify whether Gerber coordinate specifications are to be interpreted with trailing and/or leading zeros suppressed. On default, the Trailing Zeros mode for expecting trailing zeros and suppressing leading zeros is used. The Leading Zeros mode for suppressing trailing zeros and expecting leading zeros is only valid with Gerber Format 2.3 and Gerber Format 2.4, i.e., any other Gerber format specification defaults to the Trailing Zeros mode.

The Extended Gerber function can be used for optionally generating RS-274-X format Gerber output (Extended Gerber with Embedded Apertures). On input, RS-274-X Gerber format is automatically recognized by **CAM View**.

The Optimization function can be used for optionally generating optimized Gerber output with redundant D01 ("light off") plotter control commands eliminated, thus significantly reducing the amount of Gerber output plot data. On input, optimized Gerber data is automatically recognized by **CAM View**.

The Coord.-Spec. function can be used to select the coordinate input mode for Gerber and/or Excellon data load operations. With the Abs. Coordinates option, input coordinates are assumed to be absolute, whilst with option Inc. Coordinates, input coordinates are assumed to be relative and/or incremental.

Aperture Tables

For correctly loading and displaying Gerber data the same aperture table is required as was used when generating the Gerber plot. Both the **CAM Processor** and the **CAM View** module provide functions for the definition and administration of Gerber aperture tables. Aperture tables are stored to the **`cam.dat`** system file in the BAE programs directory. The aperture table named **`standard`** is loaded automatically when starting **CAM View**. The Load Aperture Table function from the File menu can be used to load any other aperture table available in **`cam.dat`**. The Edit Aperture Table function from the Settings menu can be used to change aperture definitions in the currently loaded aperture table. The Load Aperture Table and Edit Aperture Table functions can also be used *after* loading Gerber data, which would cause a Gerber data display update according to changed aperture definitions. The Save Aperture Table function from the File menu can be used to save the currently loaded aperture table with a selectable name to the **`cam.dat`** system file.

Use the following commands to define a round line aperture with 0.3mm diameter and D-Code 10 at aperture table index 1:

Edit Aperture Table	🖱
Gerber Aperture Table Index (1..900,+,-) ?	1 ↵
Aperture (R)ound/S(Q)uare/(T)hermal/(S)pecial/(A)rea/(-) ?	r ↵
Aperture Diameter/Side Length (0.127mm) ?	0.3 ↵
Drawing Mode (A)ll/(F)lash/(L)ine ?	l ↵
Gerber D-Code Number (10-999) ?	10 ↵
Gerber Aperture Table Index (1..900,+,-) ?	↵

Loading Gerber Data

The Load Gerber Data function from the File menu is used to load Gerber photoplot data files. The user is prompted for the Gerber plot file name after specifying layers for line drawn structures and flashed structures, respectively. The line draw and flash layer assignment is required by the features for decoding Gerber data to BAE layout data (see below). Gerber input layer assignments can also be utilized for panelization where Gerber data from different layers must be written to separate data files.

Layer assignments can also be used for comparing different Gerber file versions of the same layout. When loading different Gerber files versions to different layers, then the common structures are displayed with mixed colors resulting from the colors assigned to the input layers (if the Layer Assignment option is selected with the Color Assignment function from the View menu).

Use the following commands to select Gerber format 2.4, and load the Gerber plot file **`demo_l2.ger`** which has been generated in chapter 4.7.11 (select layer 2 for both the line drawn and the flashed structures):

Settings	🖱
Gerber Format	🖱
Gerber 2.4 Format	🖱
File	🖱
Load Gerber Data	🖱
Layer 2	🖱
Layer 2	🖱
Plot File Name ?	demo_l2.ger ↵

Use the following commands to repeat the Gerber load from above with an input offset of 3.2 inch in X direction:

Settings	🖱
Input Offset	🖱
X-Offset (0.000 mm) ?	3.2" ↵
Y-Offset (0.000 mm) ?	0 ↵
File	🖱
Load Gerber Data	🖱
Layer 2	🖱
Layer 2	🖱
Plot File Name ?	demo_l2.ger ↵

View, Display

The View menu provides functions for changing general display options such as zoom window, zoom scale, display grid, color settings, etc.

The Display Mode function is used to select the mode for displaying Gerber data. The Display Areas Only default option displays structures as pure areas. Option Display Outlines also displays aperture outlines, thus providing most information about how areas have been filled.

The Color Assignment function is used to select the mode of color assignment. With default option D-Code Assignment colors can be assigned to D-Codes and aperture drawing modes. With option Layer Assignment colors can be assigned to layers.

The Change Colors function operates according to the current Color Assignment. With Layer Assignment, the Change Colors function provides a layer selection menu. With D-Code Assignment, the Change Colors function prompts for a D-Code number and a drawing mode, where **f** (Flash) selects flashed structures, **l** (Line) selects line drawn structures and **b** (Border) selects drawing area outlines. The default D-Code color setup is light gray for flashed structures, gray for line drawn structures and red for border lines.

Use the following commands to select the yellow color for the line drawn structures which have been generated using D-Code 10:

View
Color Assignment
D-Code Assignment
Change Colors
Gerber D-Code Number (10-999) ? 10
Mode (f)lash/(l)ine/(b)order ? l
Move to Desired Color,yellow

Black color assignments can be used to suppress the display of selected elements.

Report

The Report function from the Utilities menu provides a statistical report on what apertures and/or D-Codes are how often used for what kind of structures.

Use the following commands to activate the Report function:

Utilities
Report

After activating the command above the following listing should appear in the graphic workarea:

```
D10  0.30 mm  Rnd. :
      Lines   : 58
D11  0.25 mm  Rnd. :
      Lines   : 454
D16  1.52 mm  Rnd. :
      Flashed : 64
D18  2.54 mm  Rnd. :
      Flashed : 16
      Lines   : 4
D29  1.40 mm  Sqr. :
      Flashed : 20 ...
```

Structures reported with a `Lines of length zero` mode are unexposed Gerber coordinates.

Moving Gerber Data

The Move Data Set function from the File menu can be used for moving previously loaded Gerber data sets. The data to be moved can be selected from a popup menu where available data sets are listed according to the load sequence. Each data set entry of the popup menu displays the layer(s), the offset to origin, the mirror mode and the input data file name. After selecting the desired data set, the user must specify the movement vector by selecting the start and end point of the movement. During these selections, the input grid corresponds with the current display grid settings. Pressing the right mouse button during data set movement activates a context menu with functions for performing absolute and/or relative jumps.

Writing Gerber Data

The Save Gerber Data function from the File menu is used to write currently loaded Gerber data of a selectable layer to a named file.

Use the following commands to select Gerber format 2.3, and write the Gerber data currently loaded to layer 2 to a Gerber file named **`demo_l2.g25`**:

```
Settings
    Gerber Format
        Gerber 2.3 Format
File
    Save Gerber Data
        Layer 2
        Plot File Name ? demo_l2.g25
```

The Clear Memory function from the File menu is used to delete the currently loaded data from memory and to re-initialize the display to prepare for loading other data. Use the following commands to clear the memory:

```
File
    Clear Memory
```

4.8.6 Processing Drilling and Milling Data

Loading Drilling/Milling Data

The Load Drill Data function from the File menu is used to load drilling and/or milling data. The Load Tool Table function for loading a drill tool table must be applied *before* loading Sieb&Meier drill data; otherwise the Load Drill Data function issues a **`No tool table loaded!`** error. Take care that the tool table corresponds with the drill data to ensure complete and correct drill data management. Note that the Load Tool Table function deletes previously loaded drill data from memory.

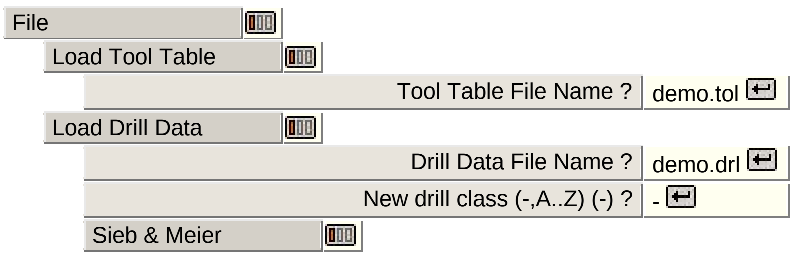
The Load Drill Data function allows for the selection of either Sieb&Meier or Excellon format. Both input units and processing of trailing and/or leading spaces are automatically derived from the input data file. The current Gerber input parameter and/or format specifications are automatically used for input data not providing corresponding format commands. Excellon input Inch unit specifications are processed using the currently selected Gerber format (Gerber Format 2.3, Gerber Format 2.4, etc.).

The function for loading Excellon data features automatic milling data recognition, i.e., Excellon milling data is automatically recognized and/or processed. With Excellon output function the output data type (i.e., either drilling data or milling data) must be specified since each drill class can contain both milling and drilling data.

The drilling and/or milling data sets are loaded and saved with drill class specification, respectively. It is possible to load drilling and/or milling data sets with different drill classes and to generate drilling and/or milling data output with selectable drill classes. With milling data the drill class can be used to differ between plated and non-plated. Drill holes and milling puncture points not assigned to the standard (**`-`**) drill class are indicated with the corresponding drill class letter (**`A`**, **`B`**, ..., **`Z`**).

The Load Drill Data function automatically activates the color table entry for the drilling data display. This color table entry is set to white if it is still undefined (i.e., black and/or invisible). This ensures that drilling data load operations can be visually monitored.

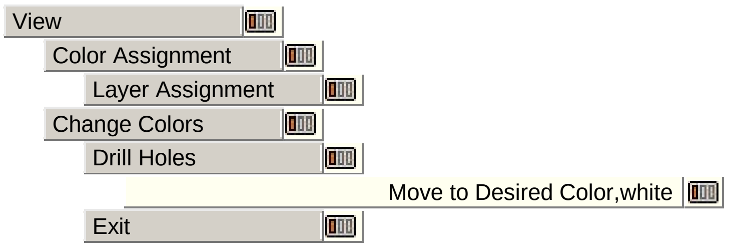
Use the following commands to load the Sieb&Meier drill data file **`demo.drl`** to drill class **`-`** using the corresponding drill tool table **`demo.tol`** (see also chapter 4.7.12, where these files have been generated):



```
File
  Load Tool Table
                          Tool Table File Name ?  demo.tol ↵
  Load Drill Data
                          Drill Data File Name ?  demo.drl ↵
                     New drill class (-,A..Z) (-) ?  - ↵
    Sieb & Meier
```

View, Display

Currently loaded drilling and/or milling data can be displayed using the layer color assignment with a non-black color (e.g., white) selected for the drill holes, as in:



```
View
  Color Assignment
    Layer Assignment
  Change Colors
    Drill Holes
                Move to Desired Color,white
    Exit
```

Moving Drilling/Milling Data

The Move Data Set function from the File menu can be used for moving previously loaded drilling and/or milling data sets. The data to be moved can be selected from a popup menu where available data sets are listed according to the load sequence. Each data set entry of the popup menu displays the layer(s), the offset to origin, the mirror mode and the input data file name. After selecting the desired data set the user must specify the movement vector by selecting the start and end point of the movement. During these selections, the input grid corresponds with the current display grid settings. Pressing the right mouse button during data set movement activates a context menu with functions for performing absolute and/or relative jumps.

Writing Drilling/Milling Data

The Save Drill Data function from the File menu is used to write currently loaded drilling and/or milling data to a named file.

Use the following commands to write the currently loaded drilling data in Excellon format with drill class **`-`** to output file **`demo.exc`**:

```
File
  Save Drill Data
                         Drill Data File Name ?  demo.exc
                 New drill class (-,A..Z) (-) ?  -
    Excellon
    Drill Data
```

Sorting Drill Data

The Sort Drill Data function from the Utilities menu is used to activate an algorithm for sorting the currently loaded drill data to minimize the distances between subsequent drill holes. Note that Sort Drill Data performs *separate* sorting on each of the currently loaded drilling data sets. Overall sort on multiple drill data sets can be applied by loading all data sets, save them as single data set, clear the memory, reload the single data set and run the Sort Drill Data function on the reloaded data set. Note also that the complexity of the drill sort algorithm grows quadratically with the number of drillings per tool, thus raising CPU time requirements enormously when working on large layouts and/or panels (>1000 drill holes).

Use the following commands to sort the currently loaded drill data, and write the sorted drill data to a file named **`demosort.drl`**:

```
Utilities
  Sort Drill Data
File
  Save Drill Data
                         Drill Data File Name ?  demosort.drl
```

4.8.7 Retrieving Layouts from Gerber Data

Loading Gerber Data

All of the required Gerber data must be loaded to the **CAM View** module before generating a layout from that Gerber data. When loading Gerber data for layout generation, the line drawn structures should be loaded to their respective layers whilst the flashed structures can be loaded to either signal layers or documentary layers. In most cases it is advisable to load the flashed structures onto documentary layers since flashed structures usually represent pads which, when placed on signal layers, would create redundant copper at the layout pin positions once the corresponding parts are loaded. Pad info retrieved from the flashed structures should rather serve as an aid for placing the parts on the layout. It is a good idea to select a special documentary layer such as Part Groups for loading flashed structures. Certain documentary Gerber plots such as the Insertion Plan can also be loaded to the corresponding BAE documentary layer to support subsequent part placement procedures.

Creating a Layout

After loading all Gerber data to **CAM View**, the Create Layout function from the File menu can be used to generate a layout from that Gerber data. With the Create Layout function first a via D-Code must be specified. Use the Report function and/or appropriate D-Code color assignments to find out which D-Code is used for vias. Flashed structures generated with the specified via D-Code are translated to via definitions. Automatic via translation is only carried out for the first Gerber data input file to avoid multiple via placement at the same coordinates. The Create Layout function automatically generates a via padstack using an all layer pad with the size of the selected via aperture and a drill hole with half that aperture size.

After specifying the via D-Code, the Create Layout function prompts for the file and element name where the layout should be written to. Note that Create Layout overwrites previously existing layout elements (nevertheless, you can use the **Layout Editor** group functions to mix different layouts retrieved from Gerber data).

Further Layout Processing

The sequence of the following operations is crucial for the creation of valid layout data.

After successfully running the Create Layout function from the **CAM View** File menu, you should return to the BAE main menu (use function Main Menu from the File menu) and start the **Layout Editor**. Within the **Layout Editor** you should load the layout and place all of the required parts (use the Add New Part function from the Parts menu). Note that **User Language** programs can be used for automatic placement instead of interactive placement if placement data ASCII files are available with appropriate formats. After completing the placement the layout should be saved, and then the Back Netlist function from the Utilities menu should be used to generate a net list with the same file and element names as were specified when creating the layout. Back Netlist generates a net list from the copper placed on the currently loaded layout. However, Back Netlist works only on DDB file level, i.e., it won't change the currently loaded net list. You should re-load the layout immediately after running Back Netlist (refrain from activating any function which could store the still empty net list currently held in main memory). A connectivity generation is automatically carried out after re-loading the layout, and then the layout is ready for processing by the **Bartels AutoEngineer**.

Chapter 5
IC / ASIC Design

NOTE
The software modules described in this chapter are only available in Bartels AutoEngineer IC Design.

This chapter describes how to use the **Chip Editor (IC Mask Editor)**, **Cell Placer**, and **Cell Router** program modules for the physical design of IC and/or ASIC mask layouts. The **GDS View** and **CIF View** program modules for importing and/or checking cell libraries and/or IC mask data in GDS and CIF format are also introduced. Unfortunately, we can't provide real design examples in this chapter because the IC manufacturing process parameters and the cell libraries of logical primitives are usually provided by the manufacturer. Although your manufacturer is unlikely to charge anything for the provision of such data in a format suitable for import into the **Bartels AutoEngineer** (such as GDS), the publication of this data is usually strictly prohibited by a non disclosure agreement (NDA).

.

Contents

Chapter 6 Neural Rule System

This chapter describes the **Bartels Neural Rule System**, i.e., how to define neural rules with the **Bartels Rule Specification Language**, how to compile rule specification source files using the **Bartels Rule System Compiler** and how to apply neural rules throughout the **Bartels AutoEngineer** design process.

Contents

6.1 General

A **Neural Rule System** is integrated to the **Bartels AutoEngineer**. This allows for the definition of rules and/or rule sets which can be assigned to individual **Bartels AutoEngineer** objects. It is possible to define attributes for controlling design processes such as preferences for the placement of certain part types (e.g., restrictions for part rotation and/or part mirroring), layer-specific clearances to be considered by the **Autorouter**, track layout and routing rules for net and/or net-groups (maximum and/or minimum trace length, maximum parallel routing of traces, etc.). The **Rule System** can also be used to apply more complex design processes for the solution of specific design problems such as special design rule checks for analogues circuitry, high-frequency technique, etc. or for setting up **Autorouter** passes with strategies and options adapted to certain routing problems.

Rules can be specified using a programming language similar to Prolog. The Bartels Rule Specification Language provides powerful operators for finding not only all possible, but rather optimum solutions to a given rule system query or output request. A Rule System Compiler is provided for the translation of **Bartels Rule System** source code. Compiled rules are either applied automatically by certain BAE system functions or can be activated using customer-defined **User Language** programs.

Rules which only set a single predicate value can be dynamically generated and assigned without without having to define and compile such rules through a `.rul` file. This simplifies rule system management procedures significantly. All system-supported rules can be conveniently set through menu-assigned **User Language** programs. In-depth knowledge of the **Neural Rule System** is not necessary for these rule system applications.

6.2 Rule Definition

6.2.1 Bartels Rule Specification Language

Rules and/or rule sets can be defined using a programming language similar to Prolog, however with special operators to find not only all possible, but rather the optimum solution to a given rule system query or output request. Rules can be specified for and assigned to individual items such as parts, nets, traces, etc. The **Rule System** also supports more complex tasks such as special design rule checks or autorouter passes with certain parameter setups according to the routing problem. The **Rule System** uses a neural net approach to focus the rule evaluation on the probably best path.

6.2.2 Bartels Rule System Compiler

The **RULECOMP** utility program is the compiler to be used for translating Bartels Rule Specification source code. See Bartels AutoEngineer User Manual - Chapter 7.14 for a more detailed description of the **RULECOMP** utility program.

Compiled rules are either applied automatically by certain in-built BAE system functions or can be activated with customer-defined **User Language** programs.

6.3 Rule System Applications

A series of advanced **Bartels AutoEngineer** features are implemented through the integrated **Neural Rule System**. This chapter describes the rule system applications provided with the **Schematic Editor** and the PCB layout system.

6.3.1 Circuit Design Rule System Applications

Antenna Highlight Mode

Assigning the **`scm_pin_drc`** rule on SCM sheet or project file level causes pins to be highlighted which only have a short single-segment connection. Usually such pins are highlighted to provide a function for fading out pins which are not to be connected. The **`scm_pin_drc`** rule is defined in the **`scm.rul`** rule definition file from the **User Language** directory (**`baeulc`**). The **SCMRULE User Language** program supports rule assignments to SCM paln elements and/or project files. **SCMRULE** can be called through Run User Script or through the Rule Attachment function from the Settings menu if **UIFSETUP** is activated.

Warning

The antenna highlight mode can only be activated if the **`scm.rul`** rule definition file (and thus the **`scm_pin_drc`** rule) has been compiled with the Rule System Compiler **RULECOMP**. **RULECOMP** stores the compiled rules to the **`brules.vdb`** file in the BAE programs directory. I.e., the rules are not saved with the design. When transferring the design onto a different computer, the **`scm_pin_drc`** rule must be transferred to this computer as well; otherwise antenna highlight mode assignments won't work.

Bus Display Mode

Assigning the **`scm_bus_fill`** rule on SCM sheet level causes busses to be displayed and plotted in fill mode. The **`scm_bus_fill`** rule is defined in the **`scm.rul`** rule definition file from the **User Language** directory (**`baeulc`**). The **SCMRULE User Language** program can be used for assigning rules to SCM elements. The **SCMCON User Language** program can be used for setting the desired bus display mode through the Outline Display and/or Filled Display option. **SCMCON** can be called through Run User Script or through the Bus Display Mode function from the Connections menu if **UIFSETUP** is activated.

Warning

The bus display mode can only be set if the **`scm.rul`** rule definition file (and thus the **`scm_bus_fill`** rule) has been compiled with the Rule System Compiler **RULECOMP**. **RULECOMP** stores the compiled rules to the **`brules.vdb`** file in the BAE programs directory. I.e., the rules are not saved with the design. When transferring the design onto a different computer, the **`scm_bus_fill`** rule must be transferred to this computer as well; otherwise bus display mode assignments won't work.

Text Visibility Control Text Classes

The Text Class function with the Text Assignment, Symbol Mask and Sheet Mask options from the Other Functions submenu of the Text menu. can be used to assign text classes for display and/or plot visibility control. The Text Assignment option activates a dialog box for assigning text classes to mouse selectable texts. The Symbol Mask option can be used on schematic plan level to fade-out texts of a specific text class from mouse-selectable symbols. The Sheet Mask option can be used to fade-out all schematic plan level texts of a specific text class. Each text can be assigned to different text classes. A text is faded-out if one its assigned text classes is faded-out. Texts without text class assignment are always visible.

A total of up to 31 different text classes are supported. Text class names can be optionally specified in the **`bae.ini`** file. The system assignes default text class names such as **`Class 1`**, **`Class 2`** etc. if no text class names are defined in **`bae.ini`**.

Text classes can be used to increase the legibility of schematic plans by fading out less significant (symbol) attributes.

Text Visibility on Rotated Symbols

The `scm_rot_vis_0`, `scm_rot_vis_90`, `scm_rot_vis_180` and `scm_rot_vis_270` from the `scm.rul` rule definition file (see **User Language** directory `baeulc`). can be used to control text visibility on SCM symbol level depending on SCM symbol rotation on SCM sheet level. Texts with `scm_rot_vis_0` rule assignments are only displayed on symbols with zero degree placement rotation angle on SCM sheet level, etc. These rules can be used to place and/or display symbol name and attribute value texts depending on symbol rotation modes. The Disolve Rotations and Combine Rotations options from the **SCMRULE User Language** program can be used for easy text rotation visibility rule assignments on SCM symbol level. Disolve Rotations creates four copies of the currently loaded symbol at 0, 90, 180 and 270 degree rotation angles, allowing for rotation-specific text editing. Combine Rotations copies the texts back to symbol and automatically attaches the corresponding rules to the modified texts. **SCMRULE** can be called through Run User Script or through the Rule Attachment function from the Settings menu if **UIFSETUP** is activated.

Warning

Rotation-specific text visibility modes mode can only be set if the `scm.rul` rule definition file including the `scm_rot_vis_0`, `scm_rot_vis_90`, `scm_rot_vis_180` and `scm_rot_vis_270` rules have been compiled with the Rule System Compiler **RULECOMP**. **RULECOMP** stores the compiled rules to the `brules.vdb` file in the BAE programs directory. I.e., the rules are not saved with the design. When transferring the design onto a different computer, the `scm_rot_vis_*` rules must be transferred to this computer as well; otherwise text rotation visibility mode assignments won't work.

Pin Graphic Plot

On default, graphics (graphic lines, dotted lines, graphic areas) on pin marker level are not plotted. However, the `scm_pin_marker_plot` rule from the `scm.rul` rule definition file in the **User Language** directory (`baeulc`) can be assigned to the currently loaded SCM sheet to force plot output of pin marker graphics. The **SCMRULE User Language** program can be used for assigning rules to SCM elements.

Warning

The pin marker plot rule can only be set if the `scm.rul` rule definition file (and thus the `scm_pin_marker_plot` rule) has been compiled with the Rule System Compiler **RULECOMP**. **RULECOMP** stores the compiled rules to the `brules.vdb` file in the BAE programs directory. I.e., the rules are not saved with the design. When transferring the design onto a different computer, the `scm_pin_marker_plot` rule must be transferred to this computer as well; otherwise pin marker graphic plot requests won't work.

Plot Elements Visibility Control

Plot visibility option are available in the Rule Attachment submenu of the Set Group Rules and Select Elements functions from the Settings menu. On default, all SCM elements except for comment texts and tag symbols are plotted. The Plot visibility option can be used to assign the Disable Plotting attribute to other elements. Plot-disabled elements are displayed on screen using the Variant Attributes color.

Plot visibility modes are saved with design variants, thus allowing for variant-specific plot outputs.

6.3.2 PCB Design Rule System Applications

Part Type specific Graphic and Text Display

Part type specific graphics and texts can be defined by assigning rules containing `llnvis` predicate value settings (Logical Library Name Visibility; for examples see the `layout_llname_*` rules in the `layout.rul` definition file of the **User Language** programs directory `baeulc`) to documentary layer polygons and/or texts on layout part level. These part elements are only displayed on layout level if the `llnvis` predicate value matches the part `$llname` (Logical Library Name) attribute value. This, e.g., allows for the definition of footprints (e.g., SMD `s1206`) with different insertion plan graphics for capacitors (e.g., `$llname c`) and resistors (e.g., `$llname r`).

Layout Placement Preferences

The `rot0`, `rot90`, `rot180`, `rot270`, `mirroron` and/or `mirroroff` rules from the `partplc.rul` rule definition file in the **User Language** directory (`baeulc`) can be assigned to parts, padstacks or pads to define placement preferences such as rotation and/or mirror modes for these elements. These rules are automatically considered by the manual and autoplacement functions from the **Layout Editor** and the **Autorouter**. After compiling the `partplc.rul` file with the Rule System Compiler **RULECOMP**, the **LDEFMANG** **User Language** program can be used on part, padstack or pad level to set these rules.

Trace Edit Display Mode

The `lay_edit_wide_filled`, `lay_edit_wide_outline` and `lay_edit_wide_filldist` rules from the `layout.rul` rule definition file (see **User Language** programs directory `baeulc`) can be assigned to layouts to force unfilled or filled or filled wide trace segments display during manual routing. The Edit Display function from the **GEDTRACE** **User Language** program can be used for assigning one of the trace edit display modes Line Display (default trace segments line display), Filled Display (wide trace segments filled display), Outline Display (wide trace segments outline display), Fill & Distance (wide trace segments with minimum distance lines display) or Fill & DRC (wide trace segments with minimum distance lines display and DRC).

With Fill & DRC, traces are displayed as with Fill & Distance, however, a design rule check is activated during interactive routing, changing the distance line display color from trace color to white upon design rule violations. Elements of the currently processed net are excluded from the design rule check. The distance line indicates the DRC spacing between traces. I.e., it is possible for a distance line to cut a pad without a DRC error being displayed if the current trace to copper clearance setting is less than the trace to trace clearance setting.

GEDTRACE can be called through Run User Script or through Other Functions from the Traces menu if **UIFSETUP** is activated.

Warning

Trace edit display modes can only be set if the `layout.rul` rule definition file (and thus the `lay_edit_wide_filled` `lay_edit_wide_outline` rules) have been compiled with the Rule System Compiler **RULECOMP**. **RULECOMP** stores the compiled rules to the `brules.vdb` file in the BAE programs directory. I.e., the rules are not saved with the design. When transferring the design onto a different computer, the trace edit display mode rules must be transferred to this computer as well; otherwise trace edit display mode assignments won't work.

Via Check Range for Manual Routing

A **Layout Editor** design rule check function for displaying valid via positions during manual routing can be activated with the Via Check Range function from the Other Functions submenu of the The via check range is specified in multiples of the currently selected input grid, with valid via check ranges from 1 to 5. A via check range of 1 only checks the current grid point, a via check range of two includes all grid points adjacent to the current grid point, thus totalling the via check area to 9 grid points, etc. Valid via positions are indicated through a small white circle with a diameter equal to the width of the currently routed trace, but not exceeding 40 percent of the input grid width. The via check is deactivated if the input grid is disabled or if the No Via Check option from the Via Check Range function is selected.

Dashed Polygon Line Display

Assigning one of the **poly_dash1**, **poly_dash2** or **poly_dash3** rules from the **polygon.rul** rule definition file (see **User Language** directory **baeulc**) to a documentary line cause the documentary line to be displayed in dash mode. **poly_dash1** creates standard mode dash lines. **poly_dash2** creates dashed lines with short line segments and long gaps. **poly_dash3** creates alternating short and long lines separated by short gaps. The Dash Mode function from the **GEDPOLY User Language** program can be used for assigning dash modes to documentary lines or for resetting any previously assigned dash modes through the Straight Line option. **GEDGROUP** can be called through Run User Script or through the Other Functions function from the Areas menu if **UIFSETUP** is activated.

Warning

Documentary dash line modes can only be set if the **polygon.rul** rule definition file (and thus the **poly_dash1**, **poly_dash2** and **poly_dash3** rules) have been compiled with the Rule System Compiler **RULECOMP**. **RULECOMP** stores the compiled rules to the **brules.vdb** file in the BAE programs directory. I.e., the rules are not saved with the design. When transferring the design onto a different computer, the polygon dash rules must be transferred to this computer as well; otherwise documentary dash line mode assignments won't work.

Polygon Line Width

The Set Polygon Line Width function from the Other Functions submenu of the Areas **Layout Editor** menu can be used to assign individual line widths to selectable documentary lines and split power plane areas. Polygon-specific line widths are considered by the display and plot functions. The default line width of zero displays lines with one pixel width and uses the standard line width when plotting polygon lines.

Split power plane line width settings are considered by the connectivity. Any drill hole intersecting or touching a split power plane line is isolated from the corresponding power layer.

Text Line Width

The Set Text Plot Width option from the Other Functions submenu of the Text **Layout Editor** menu can be used to assign individual plot widths to selectable texts. Text-specific plot widths are considered by the display and plot functions. Default text plot width zero displays text with one pixel width and uses the standard line width when plotting texts.

Height DRC

The Height DRC function from the Other Functions submenu of the **Layout Editor** Parts menu can be used for part height design rule assignments. The Height Offset option is used to define part-specific DRC height offsets for selectable parts. The DRC height offset of a part is added to its keepout areas' height DRC specifications when performing part design rule checks. The vertical regions between the board surface and part-specific height DRC offsets is free for part placement, thus allowing to place parts underneath each other. The Check Exclude option can be used for selecting an alternate part to be excluded from the part height design rule check. This feature is useful for placing alternate parts at the same position for variable and/or exclusive insertion.

The Height DRC function for assigning polygon heights (option Height Specification) and polygon height limits (option Height Limit) to selectable documentary layer keepout areas is provided with the Other Functions submenu from the Areas **Layout Editor** menu. Height specifications are usually assigned to keepout areas on part level whilst keepout area height limits are usually specified on layout level. Intersecting keepout areas with height specifications are treated like documentary distance violations.

Height DRC errors are indicated using a rectangle with diagonal lines. Keepout areas with non-zero height limits are not checked against each other, and the **Height Distance Violations** entry for displaying the number of height design rule check violations has been added to the report generated by the Report function from the **Layout Editor** Utilities menu.

Copper Fill Net Assignments to Isolated Vias

The Set Fill Net function from the Via Functions submenu of the **Layout Editor** Traces menu can be used to assign copper fill default nets to group-selected vias. The automatic copper fill functions (see chapter 4.6.8) connect isolated vias with copper fill net assignments to corresponding net-specific copper fill areas. I.e., this feature can be used to force the generation of heat-traps in otherwise plain net-specific copper fill areas. Copper fill net assignments for vias with existing physical trace or copper connections to signal nets are ignored by the automatic copper fill routines.

Drill Hole Power Layer Assignments

The Other Functions submenu from the Text, Drill **Layout Editor** menu provides the Drill Power Layers and Drill Heat Traps functions for assigning drill hole to power layer connection preferences. The Drill Power Layers function can be used on padstack level for selecting the power layer(s) occupied by mouse-selectable drill holes. The Drill Heat Traps function can be used on padstack level for assigning power layer specific drill hole connection modes such as heat trap mode (default) or direct connection mode to mouse-selectable drill holes. Both functions allow for repeated drill hole selection and drill hole power layer settings assignment in a dialog box. User interfaces without dialog box support allow for power layer bit mask input.

Drill hole power layer and heat trap rule assignments are important for blind and buried via definitions and are considered by the facilities for connectivity generation, power layer display and CAM output.

Chapter 7 Utilities

This chapter describes the utility programs of the **Bartels AutoEngineer** software. These utilities are additional programs that are run outside **Bartels AutoEngineer**, i.e., at the operating system prompt or in a batch file. With these programs useful add-on tools are supplied such as library management, DDB file contents listing, BAE software setup and configuration, ASCII net list import, etc.

Contents

7.1 BAEHELP

Name

baehelp - BAE Windows Online Documentation Access

Synopsis

```
baehelp [file]
```

Description

The **BAEHELP** utility program activates the Windows default web browsers and loads the optionally specified (HTML) file (or URL). **BAEHELP** automatically loads the BAE User Manual from the **../baedoc** directory relative to the BAE programs directory if the argument is omitted.

Warnings

BAEHELP can only be used under Windows.

7.2 BAESETUP, BSETUP

Name

baesetup - Bartels AutoEngineer Setup Module
bsetup - Bartels AutoEngineer Setup Utility

Synposis

```
bsetup -encode <code>

bsetup setupfile
```

Description

Releasing BAE Software Authorizations

The **BSETUP** command format

```
bsetup -encode <code>
```

is used for releasing BAE software updates and/or authorizations on previously delivered hardlock keys. The **-encode** option requires one argument specifying the authorization code for the BAE software configuration to be released. BAE authorization codes are provided by Bartels System GmbH on demand. When using the **-encode** option, **BSETUP** must be called from the BAE programs directory on the machine where the hardlock key to be released is currently mounted. One BAE call is required immediately after running **BSETUP** with the **-encode** option (and a *valid* authorization code) to release the new software authorization, i.e., to transfer the new authorization code from the BAE setup file **bsetup.dat** (where it has been stored to with **BSETUP**) to the hardlock key (note message **New Options : <sw-config>**). Correct BAE authorization check is then ensured on *subsequent* BAE calls, i.e., BAE must be exited immediately after being called for releasing a new BAE authorization.

Defining and Modifying BAE Setup Data

The **BSETUP** command format

```
bsetup setupfile
```

is used for defining user specified BAE configuration parameters and defaults, such as BAE menu color setup, layer menus, documentary layer definitions, standard library access paths, etc. **BSETUP** accepts the setup file name **setupfile** as argument. This file must have an extension of **.def** but this extension must not be included with the command line. **BSETUP** translates the setup file and stores the defined setup parameters to a file named **bsetup.dat** in the current directory. When starting the **Bartels AutoEngineer** or activating one of BAE's program modules the corresponding setup parameters are loaded from the **bsetup.dat** file in the BAE programs directory.

The Windows, Linux and Unix versions of the BAE software also provide the **BAESETUP** module for modifying the BAE system parameters. **BAESETUP** can be called using the Setup function from the BAE main menu. **BAESETUP** uses a graphical interface with dialog boxes, making it much easier to be operated than the **BSETUP** utility which usually requires a DEF file containing the complete BAE parameter data set to be created and/or modified. **BAESETUP** also provides a function for exporting the setup data to a **BSETUP** compatible DEF file.

Input File Format

Start Data, End Data, Comments

The setup file format must start with the keyword **SETUP** and must end with the keyword **END.**. Commentary text can be placed between **/*** and ***/**.

LAYMENUTEXT Command

The **LAYMENUTEXT** command is used to set the names of the most used signal layers in the BAE layout system menus. The formal syntax of the **LAYMENUTEXT** command is

```
LAYMENUTEXT LINE <line> ("<text>",<layer>);
```

where **<line>** is the menu line number (in range 1 to 12), **<text>** is the text to be displayed in the layer selection menus, and **<layer>** is the signal layer to be used (range 1 to 100). A special **LAYMENUTEXT** command format is given by

```
LAYMENUTEXT TOPLAYER ("<text>");
```

The above command above defines the menu text for the top signal layer option of the layer selection menus.

LAYPADLAYER Command

The **LAYPADLAYER** command is used to enable or disable layer assignments on BAE layout library pad hierarchy level. The formal syntax of the **LAYPADLAYER** command is

```
LAYPADLAYER (ENABLE) ;
```

and/or

```
LAYPADLAYER (DISABLE) ;
```

The **LAYPADLAYER** command is historic and is used to support earlier versions of the BAE software. It is strongly recommended to use the **DISABLE** option when creating new layout elements. The pad layer assignment should only be enabled for updating old job files.

LAYPLTMARKLAY Command

The **LAYPLTMARKLAY** command defines a documentary layer to be used by the **CAM Processor** for film registration marks. This layer is always plotted with the All Layers mode. The formal syntax of the **LAYPLTMARKLAY** command is

```
LAYPLTMARKLAY (<layer>) ;
```

where **<layer>** is the documentary layer number (we recommend the "Plot Markers" layer).

LAYGRPDISPLAY Command

The **LAYGRPDISPLAY** command is applied for defining a documentary layer to be used for displaying layout groups at movement with the group Display Mode set to Display Layer Only. The formal syntax of the **LAYGRPDISPLAY** command is

```
LAYGRPDISPLAY (<layer>) ;
```

where **<layer>** is the documentary layer number.

LAYDOCLAYER Command

The **LAYDOCLAYER** command is used to define up to 100 documentary layers for the BAE layout system. Each documentary layer can have information relevant to either or both sides of the PCB. I.e., each documentary layer consists of three sides (sub-layers) named Side 1 (solder side, bottom layer), Side 2 (component side, top layer) and Both Sides (both component and solder side). This layer structure enables mirroring of SMT parts with all documentary text and graphic from the component side to the solder side and vice versa. When plotting Side 1 or Side 2 of a documentary layer with the All Layer mode, the Both Sides elements automatically are added to the output, i.e., they are plotted together with Side 1 and/or Side 2. The formal syntax of the **LAYDOCLAYER** command is

```
LAYDOCLAYER <layer> ("<text>",<side>,<rotate>[<,index>]) ;
```

where **<layer>** is the number of the documentary layer in a range of 1 to 100. The **<text>** entry is used for specifying an up to 18 character long name for the documentary layer (e.g., **Silkscreen**, **Insertion Plan**, etc.). The documentary layer name is displayed in the documentary layer selection menus of the BAE layout system. The **<side>** entry is used for specifying the query mode to be used with side selections when defining objects on the corresponding documentary layer. The choices for **<side>** are

SIDE1	enables interactive input on Side 1 of the documentary layer only
SIDE2	enables interactive input on Side 2 of the documentary layer only
BOTH	enables interactive input on Both Sides of the documentary layer only
NONE	causes the BAE layer menu functions to offer a submenu choice for selecting the documentary layer side on interactive input

The **<rotate>** parameter specifies the mode of text to be created on the corresponding documentary layer. The choices for **<rotate>** are:

LOGICAL	text can be rotated and mirrored but remains readable (e.g., for silkscreen)
PHYSICAL	text can be fully mirrored and rotated but cannot be moved with the Move Name and/or Move Attribute functions (e.g., for insertion plan)
NOROTATE	text with fixed mirror and rotate mode, i.e., the part can be rotated and mirrored but the text position remains fixed (e.g., for drill plan)

The **<index>** parameter is optional and specifies the color palette and layer menu output index for the documentary layer. Documentary layer output index numbers start at 1. Documentary layers without output index assignments are automatically assigned to free index output positions. This feature allows for frequently used documentary layers to be placed at the top of the documentary layer selection menus and/or to group documentary layer definitions within layer menus according to their functionality.

DOCMENU Command

The **DOCMENU** command can be used to assign frequently used docmentary layers to the top levels of the layer selection menus in the layout system. The formal syntax of the **DOCMENU** command is as follows:

```
DOCMENU <menuline> (<layer>) ;
```

<menuline> specifies the line and/or position for displaying the documentary layer **<layer>** in the top level layer menus.

USERUNITS Command

The **`USERUNITS`** command is used for setting the default units for the BAE layout system. Once set, this default is used whenever the user is prompted for size or coordinate input. The formal syntax of the **`USERUNITS`** command is

```
USERUNITS (METRIC) ;
```

and/or

```
USERUNITS (IMPERIAL) ;
```

where **`METRIC`** applies for mm units and **`INCH`** applies for inch units. Alternate units can always be used by adding either **`"`** (for inches) or **`mm`** (for mm) to the input value.

SCMDEFLIBRARY Command

The **`SCMDEFLIBRARY`** command is used to set the default SCM library path and the name of the SCM library file containing all the standard SCM symbol definitions (such as bustap, standard label, pin symbol, module port, etc.). The formal syntax of the **`SCMDEFLIBRARY`** command is

```
SCMDEFLIBRARY ("<libpath>/<stdlib>");
```

where **`<libpath>`** is the path to the directory which contains the SCM symbol library files and **`<stdlib>`** specifies the name **`<stdlib>.ddb`** of the SCM standard symbol library file in that directory. Please note the use of the slash **`/`** as directory and database hierarchy delimiter (instead of backslash **`\`** as in DOS). With the SCM library path properly set any SCM symbol can be accessed either by selecting the library file and symbol from the Add Symbol popup menu or by specifying the symbol at the corresponding Add Symbol function prompt as in

```
<scmlib>/<symbolname>
```

where **`<scmlib>.ddb`** is the name of a SCM library file in the SCM library directory (e.g., **`74ls/74ls90`** for symbol **`74ls90`** in library **`74ls.ddb`**, **`passiv/r`** for symbol **`r`** in library **`passiv.ddb`**, etc.). The SCM command Select Library (see menu Settings) can be used to redefine the standard SCM library path (**`-`** input resets the library, **`!`** and/or **`.`** input restores the library path defined through **`SCMDEFLIBRARY`**).

LAYDEFLIBRARY Command

The **`LAYDEFLIBRARY`** command is used to set the default layout library file containing the layout symbols. The formal syntax of **`LAYDEFLIBRARY`** command is

```
LAYDEFLIBRARY ("<libpath>");
```

where **`<libpath>`** is the path name of the library file **`<libpath>.ddb`**. Please note the use of the slash **`/`** directory and database hierarchies delimiting (instead of backslash **`\`** as in DOS). With the layout library path properly set any layout symbol can be accessed by specifying the symbol with the Add Part function as in

```
<symbolname>
```

where **`<symbolname>`** is the name of the part symbol (e.g., **`dil16`**, **`sot23`**, etc.). The **Layout Editor**r command Select Library (see menu Settings) can be used to redefine the standard layout library path (**`-`** input resets the library, **`!`** and/or **`.`**> input restores the library path defined through **`LAYDEFLIBRARY`**).

The **`LAYDEFLIBRARY`** also sets the standard library file name to be used for logical library definition queries with the Show Symbol Logic function of the **Schematic Editor**.

LAYDEFELEMENT Command

The **LAYDEFELEMENT** command is used to set the default element name used when the enter key only is pressed in response to the layout board **Element Name ?** system prompt. The formal syntax of the **LAYDEFELEMENT** command is

```
LAYDEFELEMENT ("<layout-elementname>");
```

where **<layout-elementname>** is the default layout board element name. This command considerably simplifies the work with the BAE layout modules since on corresponding user queries the default layout element name can be specified just by pressing the return key ↵.

PROJROOTDIR Command

The **PROJROOTDIR** command is used to set the directory root path for the optional directory selection menus of the BAE file access functions. The directory popup menus display all subdirectories of the root directory specified with the **PROJROOTDIR** command. The formal syntax of the **PROJROOTDIR** command is:

```
PROJROOTDIR ("<rootdir>");
```

where **<rootdir>** is the path name of the directory selection root directory. If there is no **PROJROOTDIR** command defined in the setup file, then the current directory is used on default. Examples for **<rootdir>** are **/** (root directory of the current drive), **d:** (root directory of PC drive D:), **c:/cad_data** (directory **cad_data** on PC drive C:), **/pcb/projects** (directory **pcb/projects** of the current drive), etc. The **<rootdir>** entry must not contain any special characters such as **.** or \.

WINMENUMODE Command

The Windows and Motif versions of the BAE software can be operated with either the BAE standard user interface with side menus or with the windows-like user interface with pull-down menus. The **WINMENUMODE** command is used to activate the desired user interface. Use the following command to activate the BAE standard user interface with side menus:

```
WINMENUMODE (SIDEMENU);
```

Use the following command to activate the BAE user interface with pull-down menus (context menus through left mouse button, function repetition through right mouse button) for Windows and/or Motif versions of the BAE software:

```
WINMENUMODE (PULLDOWN);
```

Use the following command to activate the BAE user interface with pull-down menus (context menus through right mouse button, function repetition through left mouse button) for Windows and/or Motif versions of the BAE software:

```
WINMENUMODE (PULLDOWN_RMB_CONTEXT);
```

The BAE standard user interface is activated with the DOS and/or X11 versions of the BAE software or if the **WINMENUMODE** command is omitted in the BAE setup file.

FRAMECOLOR Command

The **FRAMECOLOR** command is applied for defining the BAE menu color setup. The formal syntax of the **FRAMECOLOR** command is

```
FRAMECOLOR <screenarea> (<colornumber>);
```

where **<screenarea>** designates the workarea of the BAE user interface and **<colornumber>** defines the color for the corresponding workarea. The **<screenarea>** choices for the user interfaces of the BAE graphic program modules (BAE Shell, **Schematic Editor**, **Layout Editor**, **Autorouter**, **CAM Processor**, **CAM View**) are:

Identifier	**Workarea**
DIALAREA	status/input line
LISTAREA	text output/graphic workarea
MENUHEAD	menu header/info field
MENUHEAD BACK	menu header/info field background
MAINMENU	main menu
MAINMENU BACK	main menu background
SUBMENUA	menu/submenu
SUBMENUA BACK	menu/submenu background
EMENMARK	menu cursor enabled (system waits for input)
EFILMARK	menu bar enabled (system waits for input)
DMENMARK	menu cursor disabled (system is busy)
DFILMARK	menu bar disabled (system is busy)
POPMTEXT	popup menu text
POPMBUTT	popup menu button
POPMBACK	popup menu background
POPMFRAM	popup menu frame
POPMFILL	directory popup menu background

The `<colornumber>` value is an integer between 1 and 15. The following table lists the color to color number assignments.

Color Number	**Color**
1	blue
2	green
3	cyan
4	red
5	magenta
6	brown
7	light gray
8	dark gray
9	light blue
10	light green
11	light cyan
12	light red
13	light magenta
14	yellow
15	white

Examples

The following listing shows the contents of the BAE standard setup file **`stdset.def`**; this file resides in the BAE programs directory and is used as template for the user defaults:

```
SETUP



/* Bartels AutoEngineer Standard Setup */



/* Menu Layer Texts */

LAYMENUTEXT LINE 1   ("Layer &1 (Solds.)",1);

LAYMENUTEXT LINE 2   ("Layer &2",2);

LAYMENUTEXT LINE 3   ("Layer &3",3);

LAYMENUTEXT LINE 4   ("Layer &4",4);

LAYMENUTEXT TOPLAYER ("Layer n (Par&ts.)");



/* Documentary Layer Definitions */

LAYDOCLAYER 1 ("Insertion Plan",SIDE2,LOGICAL);

LAYDOCLAYER 2 ("Solder Mask",NONE,PHYSICAL);

LAYDOCLAYER 3 ("Drill Plan",BOTH,NOROTATE);

LAYDOCLAYER 4 ("Film Markers",BOTH,PHYSICAL);

LAYDOCLAYER 5 ("Floor Plan",BOTH,LOGICAL);

LAYDOCLAYER 6 ("Part DRC",SIDE2,LOGICAL);

LAYDOCLAYER 7 ("Pin Number",SIDE2,LOGICAL);

LAYDOCLAYER 8 ("Solder Paste (SMT)",SIDE2,LOGICAL);

LAYDOCLAYER 9 ("Measure/Notes",SIDE2,LOGICAL);



/* Pad Layer Query */

LAYPADLAYER (DISABLE);



/* Film Markers Doc. Level */

LAYPLTMARKLAY (4);
```

```
/* Group Load Display Doc. Level */
LAYGRPDISPLAY (5);

/* Standard Search Paths and Names */
SCMDEFLIBRARY ("/baelib/stdsym");
LAYDEFLIBRARY ("/baelib/laylib");
LAYDEFELEMENT ("s1");

/* Default User Measure Units */
USERUNITS (METRIC);

/* Windows/Motif Menu Type */
WINMENUMODE (PULLDOWN);

/* Standard Graphic Color Setup */
FRAMECOLOR DIALAREA (11);
FRAMECOLOR LISTAREA (14);

/* Side Menu Color Setup */
FRAMECOLOR MENUHEAD (10);
FRAMECOLOR MENUHEAD BACK (8);
FRAMECOLOR MAINMENU (12);
FRAMECOLOR MAINMENU BACK (8);
FRAMECOLOR SUBMENUA (10);
FRAMECOLOR SUBMENUA BACK (8);
```

```
/* Menu Cursor Color Setup */

FRAMECOLOR EMENMARK (2);

FRAMECOLOR DMENMARK (15);

FRAMECOLOR EFILMARK (8);

FRAMECOLOR DFILMARK (4);



/* Popup Menu Color Setup */

FRAMECOLOR POPMTEXT (3);

FRAMECOLOR POPMBUTT (14);

FRAMECOLOR POPMBACK (8);

FRAMECOLOR POPMFRAM (15);

FRAMECOLOR POPMFILL (1);



/* Text Programs Color Setup */

FRAMECOLOR DIALLINE (10);

FRAMECOLOR OUTLINES (14);

FRAMECOLOR HEADLINE (12);


END.
```

It is recommended to change the **`setup.def`** file to define required setup parameters such as library path settings, layer assignments, etc. Then apply the **BSETUP** utility program to convert the setup file **`stdset.def`** to the **`bsetup.dat`** file as in

```
>  bsetup stdset ↵
```

Files

`bsetup.dat` -- Setup parameter file (in BAE programs directory)
`stdset.def` -- Setup file template

See also

BAE Shell, **Schematic Editor**, **Layout Editor**, **Autorouter**, **CAM Processor**, **CAM View**, **Packager**.

Diagnosis

The error messages issued by **BSETUP** are intended to be self-explanatory.

7.3 BICSET (IC Design)

NOTE: The **BICSET** utility program is only available in **Bartels AutoEngineer IC Design**!

Name

bicset - Bartels AutoEngineer IC Design Setup Utility

Synopsis

```
bicset setupfile
```

Description

The **BICSET** utility program is used for the definition and/or configuration of **Bartels AutoEngineer IC Design** system parameters such as standard cell dimensions for automatic placement, layer assignments and layer menu definitions, DRC parameters, etc. **BICSET** accepts the setup file name **`setupfile`** as argument (this file must have the extension **`.def`**, however, this file name extension should be omitted in the command line). **BICSET** translates the setup file and stores the defined setup parameters to a file named **`bsetup.dat`** in the current directory. These parameters are loaded and/or activated from the **`bsetup.dat`** file in the BAE programs directory when starting any of the program modules of the **Bartels AutoEngineer IC Design** system.

Input File Format

Start Data, End Data, Comments

The setup file format must start with the keyword **`SETUP`** and must end with the keyword **`END.`**. Commentary text can be placed between **`/*`** and **`*/`**.

Sorry, this information is currently being updated.

Examples

The **Bartels AutoEngineer IC Design** system comes with the following setup file (`icset.def`; see BAE programs directory):

```
SETUP


/* Bartels AutoEngineer IC Design Setup */

```

```
Sorry, this information is currently being updated.
```

```


END.
```

You can adapt the settings in `icset.def` to your specific requirements and use the **BICSET** utility program to compile and activate these settings:

```
>  bicset icset ↵
```

Files

`bsetup.dat` -- Setup parameter file (in BAE programs directory)
`icset.def` -- Setup file template

See also

BAE Shell, **Chip Editor**, **Cell Placer**, **Cell Router**.

Diagnosis

The error messages issued by **BICSET** are intended to be self-explanatory.

7.4 BLDRING (IC Design)

NOTE: The **BLDRING** utility program is only available in **Bartels AutoEngineer IC Design**!

Name

bldring - Bartels AutoEngineer IC Design Build Ring Utility

Synopsis

```
bldring ringdescriptionfile
```

Description

The **BLDRING** utility program automatically creates a chip layout template from a description file. This file contains informationen about the cells used and sections for placing an outer (rectangular) ring with bonding pad cells and other arbitrary cell placements. Only the placement sequence of the bonding pad cells must be specified for each ring side. The actual bonding pad cell placement coordinates are automatically calculated from the cell macro descriptions and the ring dimensions. Ring sections which are not occupied by bonding pads are automatically filled with metal structures for carrying two power supply signals through. Fixed cell placement coordinate specifications can be relative to the ring outline, thus allowing for pre-defined structures to be used on chip layouts with different dimensions. **BLDRING** expects the ring description file `ringdescriptionfile` as argument (this file must have the extension `.rig`, however, this file name extension should be omitted in the command line). The name of the output project file and the element name of the chip layout to be created are specified in the ring description file.

Input File Format

Start Data, End Data, Comments

The setup file format must start with the keyword `ring` and ends when the end of the file is reached. Commentary text can be placed between `/*` and `*/`.

Sorry, this information is currently being updated.

Examples

Sorry, this information is currently being updated.

Files

See also

Chip Editor.

Diagnosis

The error messages issued by **BLDRING** are intended to be self-explanatory.

7.5 CONCONV

Name

conconv - Connections Conversion Utility

Synopsis

```
conconv projectname libraryfile
```

Description

The **CONCONV** utility program is used for transferring ASCII net list data to the **Bartels AutoEngineer**. Supported net list formats are BARTELS, CALAY, MARCONI, and RACAL. Net lists from other systems can often be written in one of these formats providing a convenient link between other schematic systems and the powerful BAE layout facilities.

CONCONV accepts the net list file name **`projectname`** as first argument. This file must have an extension of **`.con`** but this extension must not be included with the command line.

CONCONV accepts the layout library file name **`libraryfile`** as second argument. This file must have an extension of **`.ddb`** but this extension must not be included with the command line. The **`libraryfile`** is expected to be in BAE DDB (Design DataBase) format and must contain the layout part definitions referenced from the net list.

CONCONV reads the ASCII net list **`<projectname>.con`** and checks all net list layout symbols with the corresponding entries in the layout library **`<libraryfile>.ddb`**. **CONCONV** generates the design file **`<projectname>.ddb`** and a free list named **`<projectname>.fre`**. The free list contains an unconnected pins report. After successful processing, an internal physical net list will exist in the design file named **`<projectname>.ddb`**. A layout element can then be created in that design file, parts placed and traces routed.

Input File Format

Start Data, End Data, Comments

The net list file format must start with the **`LAYOUT`** command and must end with the keyword **`END.`**. Commentary text can be placed between **`/*`** and **`*/`**. The formal syntax of the **`LAYOUT`** command is:

```
LAYOUT <elementname> ;
```

where **`<elementname>`** is the name of the net list and/or layout element to be created in the destination file.

Part List

The part list section is expected after the **`LAYOUT`** command and must start with the keyword **`PARTS`**. Each part is defined by a command in the form

```
<part> : <plname> ;
```

where **`<part>`** is the part name (e.g., **`C1`**, **`R1`**, **`IC15`**, etc.), and <plname> is the physical library name of the package (e.g., **`DIL14`**, **`SO20`**, etc.).

Net List

The net list section is expected after the parts list. Different net list formats are supported. A keyword preceding the net list information is used to designate the net list format type (`CONNECT` for **Bartels AutoEngineer** format, `CALAY` for CALAY format, `RACAL` for RACAL format, `MARCONI` for MARCONI format). Each net in the **Bartels AutoEngineer** format is defined by a command in the form

```
<part>.<pin>=<part>.<pin>=...=<part>.<pin>
```

and/or

```
/<net>/ <part>.<pin>=<part>.<pin>=...=<part>.<pin>
```

where `<part>` is the part name as defined in the part list, `<pin>` is the pin name of that part, and `<net>` is the net name. The BAE net list format supports optional net attribute definitions to be specified after the `"/<net>/"` command in the following form:

```
PRIORITY(<prior>) MINDIST(<dist>) ROUTWIDTH(<width>)
```

The net attributes are used for controlling the Autorouting process. `<prior>` is the net-specific routing priority, `<dist>` is the net-specific minimum distance, and `<width>` is the net-specific routing width. With each pin a pin-specific routing width can be defined optionally; this value must be enclosed in parentheses behind the `<pin>` specification. `<prior>` must be in positive integer units (the greater the value the higher the routing priority), all other net attribute values are assumed to be in positive mm units.

Each net in the CALAY format is defined by a command in the form

```
<part>(<pin>),<part>(<pin>),...,<part>(<pin>)
```

and/or

```
/<net> <part>(<pin>),<part>(<pin>),...,<part>(<pin>)
```

The CALAY format supports optional pin-specific routing widths (in mm units). The pin routing width specification must follow the `<pin>` definition and must be separated from `<pin>` by a comma character (`,`).

Each net in the RACAL format is defined by a command in the form

```
.ADD_TER        <part> <pin> <net>

.TER            <part> <pin>

                <part> <pin>

                :
                <part> <pin>
```

The RACAL net list must end with the keyword `.END`.

Each net in the Marconi format is defined by a command in the form

```
<part> <pin> <part> <pin> ... <part> <pin> ; <net> /
```

Examples

Net list (`design.con`) in **Bartels AutoEngineer** format:

```
LAYOUT board;

PARTS

        c1 : cap50;

        c2 : cap75;

        r1 : res;

        t1 : tebc;

CONNECT

        /net1/ c2.2=t1.3;

        /net2/ c1.2(0.4)=t1.2=r1.2;

        /gnd/ PRIORITY(2) MINDIST(0.4) t1.1=c1.1(0.4);

        /vcc/ PRIORITY(1) ROUTWIDTH(0.5) c2.1=r1.1;
END.
```

Net list (`design.con`) in CALAY format:

```
LAYOUT board;

PARTS

        c1 : cap50;

        c2 : cap75;

        r1 : res;

        t1 : tebc;

CALAY

        /net1 c2(2),t1(3);

        /net2 c1(2,0.4),t1(2),r1(2);

        /gnd t1(1),c1(1,0.4);

        /vcc c2(1,0.5),r1(1,0.5);
END.
```

Net list (`design.con`) in RACAL format:

```
LAYOUT board;

PARTS

        c1 : cap50;

        c2 : cap75;

        r1 : res;

        t1 : tebc;

RACAL

        .ADD_TER        c2 2 net1

        .TER            t1 3

        .ADD_TER        c1 2 net2

        .TER            t1 2

                        r1 2

        .ADD_TER        t1 1 gnd

        .TER            c1 1

        .ADD_TER        c2 1 vcc

        .TER            r1 1

.END
END.
```

Net list (**design.con**) in MARCONI format:

```
LAYOUT board;

PARTS

        c1 : cap50;

        c2 : cap75;

        r1 : res;

        t1 : tebc;

MARCONI

        c2 2 t1 3 ; net1 /

        c1 2 t1 2 r1 2 ; net2 /

        t1 1 c1 1 ; gnd /

        c2 1 r1 1 ; vcc /
END.
```

The net lists above can be transferred to BAE by applying the **CONCONV** program as in the command

```
>  conconv design laylib ↵
```

which causes **CONCONV** to read the ASCII net list **design.con**, check the part list entries against the layout library definitions in **laylib.ddb**, and store the net list named **board** to the job file **design.ddb**. After successful processing, the **Layout Editor** can be started, a layout named **board** (same name as net list for correct connectivity access) can be createdw and the net list parts can be loaded and placed using the Place Next Part function from the Parts menu. For pin assignment correctness the same layout library as specified with the **CONCONV** call must be used for loading the parts (apply the Select Library function from the Settings menu to select the correct library file).

See also

NETCONV

Diagnosis

The error messages issued by **CONCONV** are intended to be self-explanatory.

Warnings

Input file identifiers for part names, pin names or net names containing special characters (**-**, **+**, **/**, **(**, **=**, ...) must be enclosed in single-quotes or double-quotes.

7.6 COPYDDB

Name

copyddb - Copy Design Database Utility

Synopsis

```
copyddb srcfile dstfile {-ms|-md|-mr}

        {-a|-as|-al|-ac|-sp|-ss|-sl|-sm|-ll|-lp|-ls|-ld|-cl|-cc|-cp|

         -drc|-llib|-gtab|-fnt|-sct|-lct|-ict|-ulp|-ull|-rule|-recover} [pattern]
```

Description

The **COPYDDB** utility program copies selected database class entries and all their references from one DDB (Design DataBase) file to another. **COPYDDB** can be used as a batch mode driven tool for merging libraries or updating job design files.

COPYDDB accepts two filenames as arguments. **`srcfile`** and **`dstfile`** are the names of the DDB source file and the DDB destination file (these files must be available with extension **`.ddb`** unless file name extensions are explicitly included with the file name specifications). **COPYDDB** copies elements from the source file to the destination file. The merge switch is used to control whether existing destination file entries should be replaced overwritten (source file is master) or not (destination file is master) or whether only existing destination file elements should be overwritten (replace).

COPYDDB optionally accepts a key **`pattern`** string as argument. The pattern denotes the name(s) of the elements to be copied. Wildcards are allowed with the pattern specification. All elements of the selected class are copied if no pattern is specified. The object class of the element(s) to be copied is selected with the class switch.

Options

Merge Switch (required):

`-ms`	merge source (source file is master)
`-md`	merge destination (destination file is master)
`-mr`	merge replace (source file is master)

Class Switch (required):

`-a`	all classes
`-as`	all SCM classes (same as all `-s?` switches)
`-al`	all layout classes (same as all `-l?` switches)
`-ac`	all chip/IC design classes (same as all `-c?` switches)
`-sp`	SCM plans (with part list and logical net list)
`-ss`	SCM symbols (with logical library)
`-sl`	SCM labels
`-sm`	SCM marker
`-ll`	layout plans (with physical net list and paths)
`-lp`	layout parts
`-ls`	layout padstacks
`-ld`	layout pads
`-cl`	chip/IC design layouts (with physical net list and paths)
`-cc`	chip/IC design cells
`-cp`	chip/IC design pins
`-drc`	layout design rule check parameter blocks
`-llib`	logical library entries
`-gtab`	Gerber aperture tables
`-fnt`	BAE font data
`-sct`	SCM color tables
`-lct`	layout color tables
`-ict`	chip/IC design color tables
`-ulp`	User Language programs
`-ull`	User Language libraries
`-rule`	Rule system definitions
`-recover`	all classes (recovery mode for corrupt DDB files)

Examples

To copy all layout symbols with element names matching **`so1*`** (e.g., **`so14`**, **`so16`**, ...) from **`newlib.ddb`** to **`laylib.ddb`** with **`laylib.ddb`** supposed to be the master file:

```
>  copyddb newlib laylib -md -lp so1* ↵
```

To copy all schematic design data from **`design.ddb`** to **`redesign.ddb`** with **`design.ddb`** supposed to be the master file:

```
>  copyddb design redesign -ms -as ↵
```

To update the padstack definitions matching **`finger*`** in **`design.ddb`** according to the definitions in **`laylib.ddb`**:

```
>  copyddb laylib design -ms -ls finger* ↵
```

To copy all **User Language** libraries from **`ulcprog.vdb`** to **`ullibs.sav`** with **`ulcprog.vdb`** supposed to be the master file:

```
>  copyddb ulcprog.vdb ullibs.sav -ms -ulp ↵
```

Replace layout parts in **`design.ddb`** with corresponding parts from **`library.ddb`**, i.e., update job-specific layout library in **`design.ddb`**:

```
>  copyddb library design -mr -lp ↵
```

Files

`bae.log` -- logfile (written to current directory)

See also

LISTDDB

The functionality for copying DDB file elements is also implemented through the **ddbcopyelem** **User Language** system function.

Diagnosis

The error messages issued by **COPYDDB** are intended to be self-explanatory.

Warnings

COPYDDB is a powerful tool for manipulating DDB file contents. Be aware of *WHAT* you are doing with **COPYDDB**. Conflicts can occur when merging SCM and/or layout plans from different design files since the part lists, net lists, part attributes, etc. are merged, too (i.e., you might run into serious **Packager** and/or **Backannotation** problems, when using **COPYDDB** inappropriately). We strongly recommend to check the destination file consistency after applying **COPYDDB**!

7.7 FONTCONV

Name

fontconv - Font Conversion Utility

Synopsis

```
fontconv fontfile libraryfile
```

Description

The **FONTCONV** utility program compiles standard ASCII format vector font data into a **Bartels AutoEngineer** font file.

FONTCONV accepts the font description file `fontfile` as first argument. This file must have an extension of `.fon` but this extension must not be included with the command line.

FONTCONV accepts the layout library file name `libraryfile` as second argument. This file must have an extension of `.fnt` but this extension must not be included with the command line. **FONTCONV** stores the translated font data to `libraryfile`.

The **Bartels AutoEngineer** standard font library file is named `ged.fnt`. This file is stored to the BAE programs directory and contains the font(s) used by the **AutoEngineer**.

Input File Format

The font description file must start with the `FONT` command (for defining the name of the font) and must end with the keyword `END.`. Commentary text can be placed between `/*` and `*/`. The font description file is structured according to

```
FONT <fontname>;

CHAR <ord>;

        POLY (0, 0), (10, 10), (10, 0) ;

        :

:
END.
```

where `<fontname>` is the name of the font to be created, and where `<ord>` is the ordinal ASCII number of the character (e.g., character `A` would be number 65). The valid value range for `<ord>` is 0..255. If the font description file contains multiple character definitions for the same `<ord>` value, then **FONTCONV** stores only the last of these definitions. Each character (`CHAR`) is defined by a list of polygon lines (`POLY`). Each polygon definition consists of a list of polygon corner point coordinates. The coordinates must be specified in positive integer units in a 32x48 point grid area. The [0,0] coordinate refers to the left bottom corner of the grid area. The valid value range for X coordinates is 0..31, and the valid value range for Y coordinates is 0..47. Each character definition can have a maximum of up to 32 corner points.

Examples

Font description file `test.fon` containing definitions for `!` and `"` (with 4 corner points and 2 polygons each):

```
/* Library font name */

FONT test;

/* ASCII code 33 for '!' */

CHAR 33;

        /* Lower vertical line */

        POLY (16,5),(16,9);

        /* Upper vertical line */

        POLY (16,13),(16,45);

/* ASCII code 34 for '"' */

CHAR 34;

        /* Left line */

        POLY (12,40),(4,32);

        /* Right line */

        POLY (16,32),(24,40);
END.
```

The font description file listed above can be transferred to the font library file `ged.fnt` by applying **FONTCONV** as in

```
>  fontconv test ged ↵
```

Files

`ged.fnt` -- BAE font library file (in BAE programs directory)

See also

FONTEXTR

Diagnosis

The error messages issued by **FONTCONV** are intended to be self-explanatory.

Warnings

FONTCONV overwrites existing fonts in the destination file without any comment.

7.8 FONTEXTR

Name

fontextr - Font Extraction Utility

Synopsis

```
fontextr fontname libraryfile
```

Description

The **FONTEXTR** utility program analyzes BAE internal vector font data and generates an editable ASCII font description file.

FONTEXTR accepts the font name **`fontname`** as first argument. This is the name of the font to be extracted from the font file. **FONTEXTR** writes the ASCII font data to a file named **`<fontname>.fon`**.

FONTEXTR accepts the font library file name **`libraryfile`** as second argument. This file must have an extension of **`.fnt`** but this extension must not be included with the command line. The font library file is the file from which the user wants to extract font data.

The **Bartels AutoEngineer** standard font library file is named **`ged.fnt`**. This file is stored to the BAE programs directory and contains the font(s) used by the **AutoEngineer**.

To find out which fonts are stored in a font library file apply the **FONTEXTR** call with a **`<fontname>`** that does not exist in the font file and **FONTEXTR** lists the fonts in the file.

Output File Format

The font description file starts with the **`FONT`** command (defining the name of the font) and ends with the keyword **`END.`**. Commentary text can be placed between **`/*`** and **`*/`**. The font description file is structured according to

```
FONT <fontname>;

CHAR <ord>;    /* 'ASCII character' */

        POLY (0, 0), (10, 10), (10, 0) ;

        :

:
END.
```

where **`<fontname>`** is the name of the font and **`<ord>`** is the ordinal ASCII number of the character (e.g., character **`A`** would be number 65). The valid value range for <ord> is 0..255. Each character (**`CHAR`**) is defined by a list of polygon lines (**`POLY`**). Each polygon definition consists of a list of polygon corner point coordinates. The coordinates are specified in positive integer units in a 32x48 point grid area. The [0,0] coordinate refers to the left bottom corner of the grid area. Hence, the valid value range for X coordinates is 0..31, and the valid value range for Y coordinates is 0..47. Each character definition has a maximum of up to 32 corner points. At the end of each character definition header line a comment is printed for showing the ASCII representation of the character.

Examples

To extract the data of font **`standard`** from the font library file **`ged.fnt`** (font data output is directed to ASCII file **`standard.fon`**):

```
>  fontextr standard ged ↵
```

Files

`ged.fnt` -- BAE font library file (in BAE programs directory)

See also

FONTCONV

Diagnosis

The error messages issued by **FONTEXTR** are intended to be self-explanatory.

7.9 INSTALL

Name

install - Bartels AutoEngineer Installation Utility

Synopsis

```
install

install [-c] srcfile[pattern] dstfile[directory\*]
```

Description

The **INSTALL** utility program is applied for installing the **Bartels AutoEngineer** PC software (or parts of it) to the PC hard disk. **INSTALL** can also be used for packing (compressing) and/or unpacking (decompressing) selectable files.

The **INSTALL** utility program is the one and only tool feasible for performing a correct PC installation since the BAE PC software files are stored in compressed format on the disks of the BAE install media.

Modes of Operation

BAE Software Installation

The BAE PC software installation process can be started with the

```
install
```

command, where the current directory must contain the contents of the BAE install media, i.e., either floppy disk 1 of the BAE install kit or the BAE CD-ROM must be mounted, and the working directory must be set to the corresponding drive. The **INSTALL** program comes up with a self-explanatory dialogue, where the user (amongst other parameters) has to specify the install mode. The install mode is used to select the file set to be installed. Standard install modes copies all software files. Update install modes does not replace certain system files ending on `.dat`, `.def` and `.fnt`. I.e., update install mode should be selected to prevent the **INSTALL** program from overwriting existing user-defined setup data, color tables, font definitions, aperture tables, etc. The **INSTALL** program also prompts for the destination directories. The directory default names can be accepted by pressing the return key ↵. Each destination directory path name can be edited or can even be deleted to suppress installation of the corresponding component(s) of the software. Any of the destination directories not yet existing is automatically created with user verification.

Compress/Decompress Files

The formal **INSTALL** syntax for compressing and/or decompressing selectable files is

```
install [-c] srcfile[pattern] dstfile[directory\*]
```

where option `-c` activates compression and decompressing is applied by omitting that option. Wildcards (`pattern`) can optionally specified with the `srcfile` source file argument. The `\*` string is required at the end of the destination directory specification when selecting multiple files with wildcards.

Examples

Decompress all **.ddb** files from floppy/drive A and store the decompressed files to directory **baelib** on hard disk drive C:

```
>  install a:\*.ddb c:\baelib\* ↵
```

Decompress the file **ged.fnt** from floppy drive B and store the decompressed file to directory **bae** on hard disk drive D:

```
>  install b:\ged.fnt d:\bae ↵
```

Compress the file **design.ddb** and store the compressed file to **design.cmp**:

```
>  install -c design.ddb design.cmp ↵
```

Diagnosis

The error messages issued by **INSTALL** are intended to be self-explanatory.

7.10 LISTDDB

Name

listddb - List Design Database Utility

Synopsis

```
listddb ddbfile listfile
```

Description

The **LISTDDB** utility program lists the contents of a DDB (Design DataBase) file to an ASCII file.

LISTDDB accepts the DDB file name `ddbfile` as first argument. This file must be available with extension `.ddb` unless a file name extension is explicitly included with the file name specification.

LISTDDB accepts the listing file name `listfile` as second argument. This is the name of the ASCII destination file for the listing.

Examples

To list the contents of DDB file `laylib.ddb` to ASCII file `laylib.lst`:

```
>  listddb laylib laylib.lst ↵
```

See also

COPYDDB

Diagnosis

The error messages issued by **LISTDDB** are intended to be self-explanatory.

Warnings

The **listfile** argument is the name of the ASCII file to be created by **LISTDDB**. **LISTDDB** does not perform any file existence check, and the destination file is overwritten without any comment. It is strongly recommended to refrain from specifying existing file names unless the destination file is not needed any more.

7.11 LOGLIB

Name

loglib - Logical Library Maintenance

Synopsis

```
loglib loglibfile libraryfile
```

Description

The **LOGLIB** utility program compiles an ASCII text file containing the relationship between logical symbols and physical parts into a Design DataBase (DDB) file. The information transferred by **LOGLIB** includes assignment of SCM symbols to layout parts, logical to physical pin mapping, pin/gate swap definitions, predefined power supply pins, fixed part attributes, etc. This information is required by the **Packager** for translating logical net lists (generated by SCM) into physical net lists (processed by the layout system), and it is also required by Backannotation for transferring net list changes (pin/gate swap, part name changes) from the layout to the schematic circuit.

LOGLIB accepts the loglib file name `loglibfile` as first argument. `loglibfile` is an ASCII file containing logical library part definitions. This file must have an extension of `.def` but this extension must not be included with the command line.

LOGLIB accepts the layout library file name `libraryfile` as second argument. This file must have an extension of `.ddb` but this extension must not be included with the command line. The `libraryfile` is expected to be in BAE DDB (Design DataBase) format and should contain the layout part definitions.

Usage

The **LOGLIB** utility program must be used to update the logical library whenever a new SCM symbol is added to the library or when an SCM symbol and/or a layout part definition has been modified in a way that changes the relationship between them (e.g., change of a pin name). Usually the SCM symbol is the first to be designed (or edited). Then the layout part will be defined (if not yet existing). Finally, the loglib file is created and the logical library definitions herein are translated to the layout library using the **LOGLIB** program.

Input File Format

Start Data, End Data, Comments

The **LOGLIB** input file format must start with the keyword **`LOGLIB`** and must end with the keyword **`END.`**. Commentary text can be placed between **`/*`** and **`*/`**.

part Command

The **`part`** command is applied for assigning an SCM symbol to a layout symbol. The formal syntax of the **`part`** command is:

```
part <llname> : <plname>
```

where **`<llname>`** is the logical library name (i.e., the name of the SCM symbol) and **`<plname>`** is the physical library name (i.e., the layout package type). Keyword **`default`** can optionally be set before **`<plname>`** to enable different (non-default) layout package assignment by setting a value for the **`$plname`** attribute at the corresponding SCM symbol:

```
part <llname> : default <plname>
```

The **`part`** also supports alternate package type definitions which can be assigned during part placement in the Layout system. The **`part`** command syntax for defining alternate package types is:

```
part <llname> : [default] <plname>[,<plname>,...,<plname>]
```

where **`<llname>`** specifies the name of the SCM symbol, and the **`<plname>`** entries specify the list of valid part package types. The first **`<plname>`** entry is used as default package type. The subsequent **`<plname>`** entries define the list of alternate package types for the corresponding part. Alternate part package types can be assigned using the Alternate Part function during part placement in the **Layout Editor**. The sequence of the alternate package type menu entries provided by the Alternate Part function corresponds with the sequence of **`<plname>`** entries specified with the **`part`** command. Note however that any package type assignment defined with **`$plname`** attribute settings override Alternate Part assignments.

The **`class`** keyword can be used to assign the part to a part class:

```
part <llname> : class <partclassname> [default] <plname>
```

Part class assignments are used by the **Packager** to check whether alternate part definition assignments through the **`$rlname`** (Requested Logical Library Name) attribute are permitted.

The following **`part`** command format is used for deleting part assignments from the logical library:

```
delete part <llname> ;
```

The **`part`** command provides special format

```
part <llname> : virtual ;
```

which does not reference any physical part. This is applied to avoid errors when symbols have been used in the schematic that don't relate to parts in the PCB (e.g., company logos, drawing borders, etc.). This format is also used for setting net attributes (see below).

The **`part`** command also allows for the definition of pure logical parts by using the keyword **`logical`** as in

```
part <llname> : logical ...
```

Logical parts do not have a physical package assignment and can be used for generating logical (e.g., EDIF) net lists for PLD and/or LCA design.

Some parts have elements that require different symbols, for example relays. The symbol for the relay's coil is very different from that of the contacts and springsets and these may well need to appear on different sheets of the schematics. One symbol is chosen to be the main symbol and given a logical library name whilst the other symbols are given other logical library names. The relationship between the symbols is defined using the keywords **`mainpart`** and **`subpart`** as in

```
part <mainllname> : mainpart <plname>

:
part <subllname> : subpart <mainllname>
```

Note that the **`subpart`** definition provides a link to the logical library name of the **`mainpart`** (**`<mainllname>`**) instead of a **`<plname>`**. This feature can also be applied for providing power supply symbols.

The relationship between the pin names in the SCM symbol and the pin names in the physical part definition could be 1:1. I.e., the pin names of a resistor could be **`1`** and **`2`** and these could translate to **`1`** and **`2`** in the physical part. In this case only the relationship between the names of the SCM symbol and the physical part need to be defined as in

```
part <llname> : <plname> ;
```

where the semicolon is used to complete the **`part`** command. In most cases however additional information is required (such as pin/gate swap definitions, fixed part attributes, power supply pins, etc.) with must be provided with part **`<commands>`** as in

```
part <llname> : <plname> { <commands> }
```

net Command

The **`net`** command is used to define pins that don't appear in the symbol but need to be connected to particular signals or nets (e.g., power supply pins). The **`net`** command syntax is

```
net "<netname>" : ( <pinlist> ) ;
```

where **`<netname>`** is the signal or net name to connect and **`<pinlist>`** is a list of the physical pins (separated by commas) connected to the net. Each pin name containing special characters must be enclosed with quotes.

By preceding the net name with a dollar sign (**`$`**), the **`net`** command allows for the definition of net name attributes as in

```
net "$<netname>" : ( <pinlist> ) ;
```

With this feature it is possible to assign a part-specific power supply by assigning a variable net name attribute value (such as **`vcc`** or **`+5v`**) for the net name attribute (e.g., **`$powernet`**) to the desired SCM symbol/part of the SCM sheet.

In some cases more than one pin is connected to a particular signal but only one connection is desired in the symbol. The syntax of the **`net`** command for defining internal pin connections is:

```
net internal : ( <pinlist> ) ;
```

where **`<pinlist>`** is a list of physical pin names (separated by commas) that are to be linked together but only one of these pins needs to appear in the SCM symbol.

bus Command

The **bus** command allows busses to be defined directly on a symbol making multiple connections a single operation. The formal syntax of the **bus** command is:

```
bus ( <buspinlist> ) ;
```

where **<buspinlist>** defines a list of bus connection pins **<buspin>** (separated by commas) in the SCM symbol. Subsequently **pin** and/or **xlat** commands (see below) can be applied to define bus signals using special pin name specifications as in

```
"<buspin>.<bussignal>"
```

where **<buspin>** is the name of a pin in the **<buspinlist>** and **<bussignal>** is one of the signal connections on that bus pin.

pin Command

The **pin** command is used to define SCM symbol pin names. This command is not normally required since the pin list can be defined with the **xlat** command as well. It is mainly used to provide additional information in cases where 1:1 pin mapping applies, i.e., where the pin names otherwise can not be shown in the loglib file. The formal syntax of the **pin** command:

```
pin ( <pinlist> ) ;
```

where **<pinlist>** contains a list of pin names defined on the SCM symbol.

The

```
<startpin>-<endpin>[:<step>]
```

pin name range pattern can be used for specifying pin lists. This allows for definitions such as **pin(a1-a4);** for **pin(a1,a2,a3,a4)** or **pin(c2-c10:2)** for **pin(c2,c4,c6,c8,c10)**. It is also possible to include multiple pin name ranges such as **pin(a1-a32,b1-b2,c1-c32)** within a single command. Pin name range patterns are only pin list aliases, the system still saves and displays (function Show Symbol Logic) the complete pin name lists.

The

```
pin none ;
```

suppresses automatic 1:1 assignments of symbol to layout pins for missing **pin** commands to allows for, e.g., the definition of **mainpart** symbols without pins for general use.

xlat Command

The `xlat` command is used to define the relationship between the logical pin names and the physical pin names. It can be used to translate one set of logical pin names to one or more sets of physical pin names, i.e., the `xlat` command is used to define the logical gates of a physical part. The formal syntax of the `xlat` command is:

```
xlat ( <lplist> ) to ( <pplist> )
  or ( <pplist> ) or ... or ( <pplist> ) ;
```

where `<lplist>` is a list of the logical pin names on the SCM symbol and `<pplist>` are the corresponding physical pin lists. The logical pin name set can translate to more than one physical pin name set to provide gate definition features. The number and sequence of the pin name definitions must match.

`xlat` commands with alternations (i.e. `xlat`s with `or` options for gate specifications) automatically introduce cross-part gate `swap` definitions if no explicit `swap internal` command is defined with the corresponding part.

swap Command

The `swap` command defines the way in which pins and gates can be swapped when working in the layout system. I.e., the definitions stored with the `swap` command is used by the BAE layout system to check whether particular pin/gate swaps are allowed or not. The formal syntax of the `swap` command is:

```
swap <swapdefinition> ;
```

where `<swapdefinition>` defines the relationship between pins and gates. These definitions use brackets to identify the swap hierarchy as in

`(`	`Part Swap`	`)`
`([`	`Pin Group Swap`	`])`
`([(`	`Gate Swap`	`)])`
`([((`	`Pin Swap`	`))])`

where the square brackets can be omitted if no pin group exists. Optionally, the `internal` keyword can follow the `swap` keyword to forbid swaps between different parts.

newattr Command

The `newattr` command enables information to be included in the physical net list data that can subsequently be extracted with **Bartels User Language** or the **USERLIST** utility program. It sets up user-definable attributes associated with the physical part. These fixed library attributes could be for such information as cost, company part number, internal stock number, etc. The formal syntax of the `newattr` command is:

```
newattr "$<attname>" = "<attvalue>" ;
```

where `<attname>` is a user-defined attribute name and `<attvalue>` is the attribute value assigned to the attribute of the corresponding part. Attribute value assignments can be shown in the BAE layout system by defining the `"$<attname>"` text on the corresponding layout part (e.g., on a documentary layer).

The `newattr` command optionally allows for the definition and assignment of pin-specific attributes as in

```
newattr "$<attname>" = "<attvalue>" to ( <pplist> ) ;
```

This feature can be used for specifying arbitrary pin-specific attributes such as pin types or pin fanouts for electronic rule checks (ERC) or for generating net list interfaces to simulators such as PSpice.

The **Packager** evaluates **`$pintype`** pin attribute settings to perform electrical rule checks (ERC). It is recommended to assign fixed ERC pin types through the logical symbol/part definitions. The following pin type attribute value settings are supported:

`$pintype`	**Pin Type**
`in`	Input Pin
`out`	Output Pin
`bidi`	Bi-directional Pin
`anl`	Analog Pin
`sup`	Power Supply Pin

The ERC issues a warning message such as **`Net 'netname' has only inputs!`** if a net with one or more input pins has no (normal, bi-directional or power supply) output pin. A warning message such as **`Driver collision on net 'netname'!`** is issued if a normal output pin is connected to another output pin, a bi-directional pin or a power supply pin.

The syntax of the **`newattr`** command also allows for the assignment of variant-specific attributes by specifying a comma-separated variant number after the attribute name quotes. This allows for the assignment of different fixed attributes to different predefined project variants such as **`110 Volt`** and **`230 Volt`** or **`deutsch`** and **`english`**. **`newattr`** attribute values without variant number specification are assigned to the default/base variant.

The **`newattr`** command can be used to trigger automatic ID attribute value generation by the **Packager** through the assignment of **`?id?`**, **`?symid?extension`** and **`?partid?extension`** values. **`?id?`** creates consecutive ID values (**`id1`**, **`id2`** etc.). The **`?symid?extension`** and **`?partid?extension`** values append the specified extension with underscore to the schematic symbol name and/or layout part name (**`?partid?diffpair1`** results in **`ic1_diffpair1`**, **`ic2_diffpair1`**, etc.). Automatic ID generation is useful if a **`newattr`** command refers to multiple pins, as this allows to create a reference between pins as required for differential pair indication.

The **`newattr`** command accepts special **`!unique!`** attribute value settings which prompt the **Packager** to assign gates to layout parts with matching **`!unique!`** attribute values only. The **`swap`** commands adhere to such assignments and swap gates only between layout parts with matching **`!unique!`** attribute values. I.e., the **`!unique!`** setting can be used to control the packaging of gates with different attribute values without using **`$rpname`** attributes. This is useful for certain part types such as resistor arrays:

```
part ra_so16r : so16r {

    newattr "$val" = "!unique!";

    pin (1-16);

    swap internal (

        (( 1,16)),(( 2,15)),(( 3,14)),(( 4,13)),

        (( 5,12)),(( 6,11)),(( 7,10)),(( 8, 9))

        );
    }
```

The following example illustrates the application of the **`!unique!`** value in the definition of an opamp with power supply assigenments through attribute values:

```
part op_lm324 : dil14 {

    pin (/i,i,o);

    net "$vplus" : (4);

    net "$vminus" : (11);

    newattr "$vplus" = "!unique!";

    newattr "$vminus" = "!unique!";

    xlat (/i, i, o)

      to ( 2, 3, 1)

      or ( 6, 5, 7)

      or ( 9,10, 8)

      or (13,12,14);

    swap internal ((2,3,1),(6,5,7),(9,10,8),(13,12,14));
    }
```

netattr Command

The **`netattr`** command can be used for the design of special SCM symbols for setting net attributes. The formal syntax of the **`netattr`** command is:

```
netattr <netatt> "$<attname>" : ( <pinlist> ) ;
```

where **`<netatt>`** is the name of the net attribute, **`<attname>`** is the name of the part attribute to be mapped to the net attribute, and **`<pinlist>`** is the list of logical pin names. Arbitrary net attribute names can be set with **`<netatt>`**, but the following keywords have special meaning for the control of the Autorouting process:

`routwidth`	net-specific routing width (in mm units)
`powwidth`	pin-specific routing width (in mm units) for the signals and/or pins defined with the **`net`** command (see above)
`mindist`	net-specific minimum clearance (in mm units)
`priority`	net-specific routing priority (in positive integer units; the greater the value the higher the priority)

The `route.ddb` SCM library file is provided with the **AutoEngineer**. `route.ddb` contains virtual SCM symbols for setting net attribute values. The corresponding loglib file (`route.def`) includes the following definitions:

```
loglib

part att_rw : virtual

{

    pin (x);

    netattr routwidth "$val" : (x);

}

part att_pw : virtual

{

    pin (x);

    netattr powwidth "$val" : (x);

}

part att_md : virtual

{

    pin (x);

    netattr mindist "$val" : (x);

}

part att_pr : virtual

{

    pin (x);

    netattr priority "$val" : (x);

}

end.
```

With each of the above listed net attribute symbols a pin named `x` and a net attribute is defined. Net attribute setting then is applied by connecting the desired net to the net attribute symbol's pin and setting the corresponding attribute value.

With arbitrary user-specific net attribute definitions additional net information can be processed for special purposes. Such net attributes can be used for controlling the layout process (maximum/minimum allowed trace length, parallel routing restrictions, layer assignments, etc.) or they can be evaluated for subsequent simulator processes, run-time analysis, checking ECL/EMC rules, etc. Tools for accessing and evaluating user-specific net attributes can be provided with **Bartels User Language** programs.

The **netattr** command can be used to trigger automatic ID attribute value generation by the **Packager** through the assignment of **?id?**, **?symid?extension** and **?partid?extension** values. **?id?** creates consecutive ID values (**id1**, **id2** etc.). The **?symid?extension** and **?partid?extension** values append the specified extension with underscore to the schematic symbol name and/or layout part name (**?partid?diffpair1** results in **ic1_diffpair1**, **ic2_diffpair1**, etc.). Automatic ID generation is useful if a **netattr** command refers to multiple nets, as this allows to create a reference between nets as required for differential pair indication.

call Command

The **call** command is used for hierarchical SCM design in order to assign SCM blocks to SCM symbols for later reference on SCM sheet level. The formal syntax of the **call** command is:

```
call <blockname> ;
```

where **<blockname>** is the name of the SCM block, and the corresponding part must be defined **virtual**. The block symbol pins are assigned to the name-corresponding module ports of the SCM block.

architecture Command

The **architecture** command can be used to define **virtual** logical parts consisting of different arbitrarily connected SCM symbols and/or layout parts. The formal syntax of the **architecture** command is:

```
architecture { <partlist> }
```

<partlist> contains the list of used symbols with comma-separated pin lists in parenthesis, where each pin specification has the following format:

```
<pinname:connection>
```

<connection> can be the name of a pin of the **<architecture>** symbol. A connection to a global net can be established with **<net netname>**. **<net & intnetname>** or **<& intnetname>** specifications can be used to refer to a local net of the **<architecture>** symbol.

Examples

Loglib file `example.def` containing two part definitions:

```
loglib

/* Example Loglib File */

part 74ls00 : dil14, so14

{

    newattr "$partnumber" = "A-NAND-X11B82";

    newattr "$pintype" = "in" to (1,2,4,5,9,10,12,13);

    newattr "$pintype" = "out" to (3,6,8,11);

    newattr "$pintype" = "sup" to (7,14);

    pin (a,b,y);

    net "vcc" : (14);

    net "$groundnetname" : (7);

    xlat ( a, b, y)

      to ( 1, 2, 3)

      or ( 4, 5, 6)

      or (13,12,11)

      or (10, 9, 8);

    swap (((1,2),3),((4,5),6),((13,12),11),((10,9),8));

}

part tx27 : default sot23;

part tr_bc547 : class "npn-transistor" default to92

{

    pin (e,b,c);

    xlat (e,b,c)

      to (1,2,3);

}

end.
```

The example above shows the definition of the gate `74ls00` which can be packed up to 4 times into a single `dil14` package. The logical pins (a,b,y) can be assigned to either of the pin sets (1,2,3), (4,5,6), (13,12,11) or (10,9,8). Within the layout system, this package type assignment can be changed to `so14`. A fixed attribute named `partnumber` has been assigned with attribute value `A-NAND-X11B82`. A fixed pin-specific attribute named `pintype` is assigned to each pin, thus designating the pin type to be either `in` (input pin), `out` (output pin) or `sup` (supply pin, i.e., power or ground). Pin 14 of the `dil14` package is a pre-defined power supply pin on signal `vcc`, whilst pin 7 can be connected to a signal net by assigning the desired net name (such as `vss`) to the `$groundnetname` attribute of the `74ls00` symbol on SCM level. The `swap` command is applied to define the following pin/gate swaps:

```
Pin Swap: (1,2) and/or (4,5) and/or (13,12) and/or (10,9)

Gate Swap: (1,2,3) with (4,5,6) or (13,12,11) or (10,9,8)
Part Swap: (1,2,3,4,5,6,13,12,11,10,9,8)
```

The example loglib file above also contains the `tx27` and `tr_bc547` part definitions. `tx27` is assigned to default `sot23` package with 1:1 pin mapping. The `tr_bc547` definition includes an assignment to part class `npn-transistor`.

To compile loglib file `example.def` and store the result to DDB file `mylib.ddb`:

```
>  loglib example mylib ↵
```

The following example illustrates the use of `mainpart` and `subpart` for assigning different SCM symbols (relay part with two contacts and one coil) to the same package:

```
loglib

/* Relays part */

part rel2 : mainpart dilrel

{

    xlat (a,b) to (1,7) or (8,14);

    swap ((1,7),(8,14));

}

part rel2sub : subpart rel2

{

    xlat (p,m) to (2,6);

}

end.
```

The following example shows the definition of part **`buspart`** with busses **`b1`** and **`b2`** including the bus signals **`0`**, **`1`**, **`2`** and **`3`** (the corresponding SCM symbol must have pins **`b1`** and **`b2`** defined for connecting and tapping the bus):

```
loglib

/* Bus part definition */

part buspart : sot8

{

    bus (b1,b2);

    xlat (b1.0,b1.1,b1.2,b1.3) to (1,2,3,4);

    xlat (b2.0,b2.1,b2.2,b2.3) to (5,6,7,8);

}

end.
```

The following example illustrates how to define a virtual part for hierarchical circuit design. This applies for the assignment of SCM symbol **`dff`** to SCM block **`dff`** with pin names **`s`**, **`r`**, **`q`** and **`/q`** referring to the module ports defined on the corresponding SCM hierarchy block element/sheet:

```
loglib

/* Hierarchical D-Flip-Flop */

part dff : virtual

{

    pins (s,r,q,/q);

    call dff;

}

end.
```

The following example shows a synthetically generated symbol definition. The `delay` SCM symbol consists of four internally connected `74ls04` inverters.

```
loglib

/* Synthetically generated Inverter/Delay Circuit */

part delay : virtual

{

    pin (in,out);

    architecture

    {

        part "74ls04" (a:in,y:&connect1);

        part "74ls04" (a:&connect1,y:&connect2);

        part "74ls04" (a:&connect2,y:&connect3);

        part "74ls04" (a:&connect3,y:out);

    }

}

end.
```

Files

The symbol and part library directory installed with the **Bartels AutoEngineer** software contains the **LOGLIB**(`*.def`) files for alle SCM library files. All **LOGLIB** files of the software are already compiled into the `laylib.ddb` layout library file.

See also

Packager.

The functionality for compiling Logical Library definitions is also implemented through the **con_compileloglib** **User Language** system function.

Diagnosis

The error messages issued by **LOGLIB** are intended to be self-explanatory.

Warnings

Input file identifiers such as part names, pin names or net names containing special characters (`-`, `+`, `/`, `(`, `=`, ...) must be enclosed in single-quotes or double-quotes.

The **Packager** copies logical library definitions from the specified layout library to the project file when first referenced. Certain SCM or layout symbol modifications from the project file such as pin name changes must be reflected by a logical library definition update. I.e., **LOGLIB** must be applied to the project file if the original logical library definition has already been transferred to the project file during an earlier **Packager** run. The **Packager** issues error messages such as `Part not found in library!`, `Part not defined!` or `Pin not found!` loglib definitions are wrong or missing.

7.12 NETCONV

Name

netconv - Logical Netlist Conversion Utility

Synopsis

```
netconv projectfile
```

Description

The **NETCONV** utility program is used for transferring logical (i.e., unpacked) net list data from BAE ASCII format to internal **Bartels AutoEngineer** format ready for processing with the BAE **Packager**.

NETCONV accepts the net list file name **`projectfile`** as first argument. This file must have an extension of **`.net`** but this extension must not be included with the command line. **NETCONV** converts logical ASCII net list data from input file **`<projectfile>.net`** to a BAE logical net list with the name **`netlist`** in a DDB (Design DataBase) file named **`<projectfile>.ddb`**. This internal logical net list can be converted to a physical net list with the **Packager**, in the same way as information from the BAE schematic. It will, as a result, be possible to perform pin and gate swaps in the subsequent layout process.

Input File Format

Start Data, End Data, Comments

The net list file format must start with the keyword **`NETLIST;`** and must end with the keyword **`END.`**. Commentary text can be placed between **`/*`** and **`*/`**.

Part list

The part list section is expected after the **`NETLIST`** command and must start with the keyword **`PARTS`**. Each part is defined by a command in the form

```
<part> : <llname> ;
```

where **`<part>`** is the part name (e.g., **`C1`**, **`R1`**, **`IC15`**, etc.), and <llname> is the logical library name of the part (e.g., **`74LS08`**, **`OP_LM211`**, **`i80386sx`**, etc.).

Net List

The net list section is expected after the parts list and must start with the keyword **`CONNECT`**. Each net is defined by a command in the form

```
<part>.<pin>=<part>.<pin>=...=<part>.<pin>
```

and/or

```
/<net>/ <part>.<pin>=<part>.<pin>=...=<part>.<pin>
```

where **`<part>`** is the part name as defined in the part list, **`<pin>`** is the pin name of that part, and **`<net>`** is the net name.

Examples

ASCII net list (**`design.net`**):

```
NETLIST;

/* Part list */

PARTS

   ic1 : 74ls00;

   ic2 : 74ls00;

   c1 : c;

/* Net list */

CONNECT

   ic1.a = ic2.y;

   ic2.a = ic1.y;

   /vcc/ c1.1;

   /vss/ c1.2;
END.
```

To convert the logical ASCII net list **`design.net`** to an internal logical net list named **`netlist`** in BAE DDB file **`design.ddb`**:

```
>  netconv design ↵
```

The command above causes **NETCONV** to read the ASCII net list **`design.net`**, and store the net list named **`netlist`** to the job file **`design.ddb`**. After successful processing, the **Packager** can be used to convert this net list to a physical net list ready to be processed by the BAE **Layout Editor**.

See also

CONCONV, **LOGLIB**, **Packager**.

Diagnosis

The error messages issued by **NETCONV** are intended to be self-explanatory.

Warnings

NETCONV converts logical net lists to the **Bartels AutoEngineer**. In cases where pin/gate swap or part name change is applied in the corresponding layout the physical net list data must be backannotated to the schematics, i.e., special tools might be required for transferring the assignments file information generated by BAE Backannotation back to the original SCM system.

Input file identifiers for part names, pin names or net names containing special characters (**`-`**, **`+`**, **`/`**, **`(`**, **`=`**, ...) must be enclosed in single-quotes or double-quotes.

7.13 REDASC

Name

redasc - REDAC ASCII Input Interface

Synopsis

```
redasc projectname [libraryfile]
```

Description

The **REDASC** utility program converts layout data from Redac MAXI systems. **REDASC** transforms layout symbols, part lists, net lists, placement information, etc. from CDI format to internal BAE DDB (Design DataBase) format.

REDASC accepts the CDI file name **`projectname`** as first argument. This file must have an extension of **`.cdi`** but this extension must not be included with the command line.

REDASC optionally accepts the layout library file name **`libraryfile`** as second argument. This file must have an extension of **`.ddb`** but this extension must not be included with the command line. The **`libraryfile`** is expected to be in BAE DDB (Design DataBase) format and must contain the layout part definitions referenced from the net list. The required layout library symbols are expected to be defined in the CDI file if the **`libraryfile`** argument is omitted.

REDASC converts the CDI file **`<projectname>.cdi`** to a BAE DDB design file with the name **`<projectname>.ddb`**. The generated layout element is named **`<projectname>`** and contains the placed parts and the net list connectivity. **REDASC** also generates a free list named **`<projectname>.fre`** which contains an unconnected pins report.

See also

CONCONV

Diagnosis

The error messages issued by **REDASC** are intended to be self-explanatory.

Warnings

REDASC does not convert traces from the CDI format. Use the **CAM View** facility to read in routing data via Gerber format to reconstruct the PCB.

7.14 RULECOMP

Name

rulecomp - Bartels Rule System Compiler

Synopsis

```
rulecomp srcfile [-l]
```

Description

The **RULECOMP** compiler is used for translating Bartels Rule Specification source code.

RULECOMP accepts the rule specification source file name **`srcfile`** as first argument. This file must have an extension of **`.rul`**, but this extension must not be included with the command line. **RULECOMP** translates the rule specification source file and stores the defined rules and/or rule sets to a file named **`brules.vdb`** in the BAE programs directory. These rules can be applied by certain in-built BAE system functions and/or customer-defined **User Language** programs.

Options

Command line options of the Rule System Compiler consist of the dash (**`-`**) start character followed by the option specification. Single-character option specifications optionally followed by a mode or toggle number are often known as switches or flags.

Listing Option [-l]

The **`-l`** option is used to control the listing file output. Omitting this option suppresses listing output. The **`-l`** option generates a rule compilation listing file. The listing output file name is derived from the corresponding source code file name, with the original file name extension replaced by **`.lst`**. The listing file is for user information purposes only, i.e., it is not required by system.

Examples

Compilation of the rules defined in **`routstd.rul`**; the compiled rules are stored with the name **`routstd`** to the **`brules.vdb`** file of the BAE programs directory:

```
>  rulecomp routstd ↵
```

Compilation of the rules defined in **`routstd.rul`** with listing file output (to **`routstd.lst`**); the compiled rules are stored with the name **`routstd`** to the **`brules.vdb`** file of the BAE programs directory:

```
>  rulecomp routstd -l ↵
```

Files

`brules.vdb` -- BAE rules data file (in BAE programs directory)

See also

Bartels AutoEngineer User Manual - Chapter 6

Diagnosis

The error messages issued by **RULECOMP** are intended to be self-explanatory.

Warnings

RULECOMP is a powerful software tool for implementing rules for automatically generating and/or manipulating **Bartels AutoEngineer** design data. It is strongly recommended to test each new rule in a non-critical environment (test software installation, test jobs, backup of real jobs, etc.) until confidence in the rule is established for unrestricted use on real designs.

7.15 ULC - User Language Compiler

Name

ulc - Bartels User Language Compiler

Synopsis

```
ulc [-wcon|-wcoff] [[-S[ource]] srcfile...]

    [-lib libname...] [-dll libname...]

    [{-cp|-cl} [dstname...]]

    [-I[nclude] includepath...] [-D[efine] macroid...]

    [-O[0|1]] [-e[0|1]] [-w[0|1|2|3|4]] [-t[0|1]]

    [-l[0|1|2|3|4|5]] [-ld listingdirectory name]

    [-dp prgname...] [-dl libname...]

    [-ulp prgfilename] [-ull libfilename]

    [-log logfilename]
```

Description

The **Bartels User Language Compiler** **ULC** translates **User Language** source code into **User Language** machine programs and/or **User Language** libraries. User Language machine programs can be executed by the **User Language Interpreter**. **User Language** libraries usually are generated from frequently used source code. **User Language** library machine code can be linked to other **User Language** machine code (programs or libraries). Library linking can be done either statically (at compile time by the **User Language Compiler**) or dynamically (at runtime by the **User Language Interpreter**). As an advantage of the **User Language** library concept, frequently used source code needs to be compiled *just once* and can subsequently be referenced through linking, which is much faster than compiling.

Options

Command line options of the **User Language Compiler** consist of the dash (`-`) or slash (`/`) start character followed by the option specification. Single-character option specifications optionally followed by a mode or toggle number are often known as switches or flags. Such special options can be grouped as in `/l2Ow3` or `-O1w3l2`, which both select listing mode 2, activate the optimizer and set the warning severity level to 3.

Wildcard Option [-wcon|-wcoff]

The wildcard option is used to activate or deactivate wildcard processing at the specification of file and/or element names. On default wildcard processing is activated, i.e., omitting the wildcard option leaves wildcard processing activated. Option `-wcon` can be used for explicitly activating wildcard processing. With wildcard recognition activated, the character `?` can be used for matching any arbitrary character, and the character `*` can be used for matching an arbitrary number of arbitrary characters. Option `-wcoff` can be used to turn off wildcard processing. Wildcard recognition must be deactivated for explicitly processing names containing wildcard characters such as **SCM_?** or **GED_***.

Source File Option [[-S[ource]] srcfile...]

This option is used for specifying the file name(s) containing the source code to be compiled. File name specifications can contain a directory path, i.e., the source file names need not reside in the current directory. Wildcards are supported with the source file name specification if wildcard recognition is activated (see option `-wcon` above). Source file names can be specified with or without file name extension. On source file name specifications without extension the Compiler automatically assumes and/or appends file name extension `.ulc`. I.e., source file names with non-default extension must be specified with their extension, respectively. It is possible to, e.g., generate **User Language** libraries from include files usually named with extension `.ulh`. The type of **User Language** machine code to be generated is designated with either option `-cp` (**User Language** programs; see below) or option `-cl` (**User Language** libraries; see below). The name of the machine code element to be generated is derived from the source file name by stripping the directory path and the file name extension from the source file name. Non-default destination program and/or library element names can be specified with options `-cp` and `-cl` (see below). The `-Source` and/or `-S` option keywords are not required with source file specifications where file names cannot be intermixed with other name specifications, e.g., if source file names are the first names specified on the ULC command line. However, the `-Source` (and/or `-S`) option can be used for explicit source file specification to avoid ambiguities in case where source file names are not the first name specifications on the ULC command line. At least one source file specification is required if neither option `-dp` nor option `-dl` (see below) is specified.

Static Link Option [-lib libname...]

The static link option `-lib` requires one or more library name specifications and at least one valid source file specification (see option `-Source` above). The machine code of the libraries specified with the `-lib` option must be available in the `ulcprog.vdb` file of the BAE programs directory, i.e., the required libraries must be compiled before they can be linked. The built-in linker of the **User Language Compiler** binds the machine code of these libraries to the machine code currently to be translated.

Dynamic Link Option [-dll libname...]

The dynamic link option `-dll` requires one or more library name specifications and at least one valid source file specification (see option `-Source` above). The machine code of the libraries specified with the `-dll` option must be available in the `ulcprog.vdb` file of the BAE programs directory, i.e., the required libraries must be compiled before they can be linked. The built-in linker of the **User Language Compiler** stores dynamic link request information with the machine code for the **User Language Interpreter** to perform dynamic linking of the requested libraries at runtime.

Create Program/Library Option [{-cp|-cl} [dstname...]]

The create option can be used to designate the type of machine code to be generated. The **User Language Compiler** can create either **User Language** programs or **User Language** libraries. On default program generation request is assumed, i.e., omitting both the `-cp` and the `-cl` option defaults to **User Language** program creation. Option `-cp` explicitly selects program generation whilst option `-cl` selects library generation; both options must not be used together. On default, the destination element name is derived from the corresponding source file name; both the directory path and the source file name extension are stripped from the source file name to generate destination element name. The `-cp` and `-cl` options allow for the specification of non-default destination program and/or library names. Only *one* source file specification (see option `-Source`) is allowed when explicitly specifying destination element name(s) with options `-cp` and `-cl`. The machine code generated by the Compiler is stored with the specified destination element name to `ulcprog.vdb` file of the BAE programs directory. Wildcards are not supported with destination element name specifications. Multiple destination element name specifications are supported in order to store the machine code of a single source under different names, e.g., to generate programs **scm_st**, **ged_st**, etc. from a single source file named **bae_st.ulh**.

Include Path Option [-I[nclude] includepath...]

The `-Include` (and/or `-I`) option is used for specifying multiple alternate include paths for include file name search. At least one include path argument is required. When encountering an `#include` preprocessor statement the Compiler first checks the current directory for include file access and then searches the include paths in the sequence as specified with the `-Include` option until the requested include file is found.

Define Option [-D[efine] macroid...]

The `-Define` (and/or `-D`) option is used for defining macros at the **User Language Compiler** call. At least one macro identifier is required. This option corresponds with the `#define` preprocessor statement, i.e., macros defined with the `-Define` option can be checked with the `#ifdef` or `#ifndef` preprocessor statements, thus giving more control on conditional compilation to the Compiler.

Optimizer Option [-O[0|1]]

The `-O` option is used to activate or deactivate the optimizer of the **User Language Compiler**. On default the optimizer is deactivated, i.e., omitting this option leaves the optimizer deactivated. Option `-O` or `-O1` activates the optimizer. Option `-O0` explicitly deactivates the optimizer. The optimizer frees the machine code from redundancies, and modifies it to make it more efficient. Optimizing machine code significantly reduces disk space and main memory requirements, and the resulting machine code can be loaded and executed much faster. It is strongly recommended to activate the optimizer.

Error Severity Option [-e[0|1]]

The `-e` option is used for setting the error severity level. On default the error severity level is set to 1, i.e., omitting this option selects error severity level 1. Option `-e0` sets error severity level 0. Option `-e` or `-e1` explicitly sets error severity level 1. Error severity level 1 compiles all specified sources; error severity level 0 causes the Compiler to stop the compilation process on any errors occurred during a single source compilation.

Warning Severity Option [-w[0|1|2|3|4]]

The `-w` option is used for setting the warning severity level in the range 0 to 4. On default, the warning severity level is set to 0, i.e., omitting this option selects warning severity level 0. Omitting explicit level specification with this option as with `-w` defaults to warning severity level 3. Each type of warning defined with the Compiler is assigned to a certain warning severity level. The Compiler only prints warnings with a warning severity level less than or equal the level selected with the `-w` option since higher warning severity levels denote less importance. With this option, it is possible to suppress less important warning messages.

Top Level Warnings Only Option [-t[0|1]]

The `-t` option controls whether warning messages related to the compilation of include files are omitted or not. On default (i.e., if this option is not specified or if its value is set to 0), warning messages related to the compilation of both top level source code files and include files are issued. Setting this option value to 1 prevents the **User Language Compiler** from issuing warning messages related to the compilation of include files, thus simplifying the analysis of warning messages when working with standard include files containing functions and variables which are not used by every program.

Listing Option [-l[0|1|2|3|4|5]]

The `-l` option is used to control the listing file output. Listing modes 0 to 5 can be specified. Mode 0 won't produce any listing output and mode 5 produces the most detailed listing. On default, listing mode 0 is selected, i.e., no listing is generated if this option is omitted. Omitting explicit listing mode specification with this option as with `-l` defaults to listing mode 5. The listing output file name is derived from the corresponding source code file name, where the original file name extension is replaced with extension `.lst`. The listing file is for user information purposes only, i.e., it is not required by system.

Listing Directory [-ld listingdirectoryname]

The `-ld` option allows for the specification of an alternative output directory for the listing files created with the `-l` option. This option is useful when applying `make` utilities for automatically compiling modified **User Language** programs as it allows to keep the source directories clean. With the BAE software, a `makefile` is provided in the `baeulc` directory. This `makefile` defines the dependencies between **User Language** programs and include files and works with listing files in a subdirectory (`lst`).

Delete Program Option [-dp prgname...]

The `-dp` option is used for deleting previously compiled programs from the `ulcprog.vdb` file in the BAE programs directory. At least one program element name is required. Wildcards are supported with the program element name specification if wildcard recognition is activated (see option `-wcon` above). Warnings are issued when trying to delete non-existent programs. Program deletion is always processed before any source code compilation in order to avoid compilation process conflicts such as deleting a program immediately after it has been compiled with the same ULC call.

Delete Library Option [-dl libname...]

The `-dl` option is used for deleting previously compiled libraries from the `ulcprog.vdb` file in the BAE programs directory. At least one library element name is required. Wildcards are supported with the library element name specification if wildcard recognition is activated (see option `-wcon` above). Warnings are issued when trying to delete non-existent libraries. Library deletion is always processed before any source code compilation in order to avoid compilation process conflicts such as deleting a library immediately after it has been compiled with the same ULC call.

Program Database File Name Option [-ulp prgfilename]

On default, the **User Language Compiler** stores **User Language** programs to a file named `ulcprog.vdb` in the **Bartels AutoEngineer** programs directory. The `-ulp` option can be used to select a different **User Language** program database file.

Library Database File Name Option [-ull libfilename]

On default, the **User Language Compiler** stores **User Language** libraries to a file named `ulcprog.vdb` in the **Bartels AutoEngineer** programs directory. The `-ull` option can be used to select a different **User Language** library database file.

Log File Option [-log logfilename]

The **User Language Compiler** prints all messages to standard output and to a log file. Log file output is generated to save long message lists which could be generated at the compilation of different sources. On default the log file name is set to `ulc.log` (in the current directory). The `-log` option can be used to specify a non-default log file name.

Examples

Compilation of the **User Language** program contained in **ulcprog.ulc** with optimization and warning message output; the produced machine program is stored with the name **ulcprog** to the **ulcprog.vdb** file of the BAE programs directory:

```
>  ulc ulprog -Ow ↵
```

Compilation of the **User Language** program contained in **ulcprog.ulc** with listing file output (to **ulcprog.lst**); the produced machine program is stored with the name **newprog** to the **ulcprog.vdb** file of the BAE programs directory:

```
>  ulc ulprog -l -cp newprog ↵
```

Deleting the **User Language** programs named **ulcprog** and **newprog** and all **User Language** libraries with names starting with **test** and ending on **lib** from the **ulcprog.vdb** file of the BAE programs directory:

```
>  ulc -dp ulprog newprog -dl test*lib ↵
```

Generate **User Language** library **libsll** from source file **libbae.ulh** (optimizer is activated; listing output is directed to file **libbae.lst**):

```
>  ulc libbae.ulh -cl libsll -l2O ↵
```

Compile all current directory files with extension **.ulc** and statically link the generated program machine codes with library **libsll** (macro **USELIB** is defined for controlling conditional compilation; optimizer is activated):

```
>  ulc *.ulc -Define USELIB -lib libsll -O ↵
```

Generate libraries **libstd** and **stdlib** from source file **std.ulh** (optimizer is activated, warning severity level is set to 2):

```
>  ulc -w2 -O -cl libstd stdlib -Source std.ulh ↵
```

Generate library **liblay** from source file **\baeulc\lay.ulh** with library **libstd** dynamically linked (optimizer is activated, warning severity level is set to 3, Compiler messages are directed to log file **genlib.rep** instead of **ulc.log**):

```
>  ulc /wO -cl liblay -S \baeulc\lay.ulh -dll libstd -log genlib.rep ↵
```

Generate programs **laypcr** and **tracerep** from source files **laypcr.old** and **tracerep.ulc** with library **liblay** dynamically linked (optimizer is activated):

```
>  ulc laypcr.old /dll liblay /cp -O /S tracerep ↵
```

Files

`ulcprog.vdb` -- BAE **User Language** database (in BAE programs directory)

See also

USERLIST, **User Language Interpreter**, Bartels User Language Programmer's Guide

Diagnose

The error messages issued by **ULC** are intended to be self-explanatory.

Warnings

ULC is a powerful software tool for implementing programs for the manipulation of DDB file contents and for generating CAM data. Even the BAE user interface can be considerably changed and/or extended with **User Language** programs. It is advisable to test each new **User Language** program in a non-critical environment (test software installation, test jobs, backup of real jobs, etc.) until confidence in the program is established for unrestricted use on real jobs. It is also strongly recommended to ensure security, e.g., to prevent foreign persons from implanting destructive **User Language** programs to `ulcprog.vdb`.

7.16 User Language Interpreter

Description

The **Bartels User Language Interpreter** is used for executing **Bartels User Language** programs which have been compiled with the **Bartels User Language Compiler**. The **Bartels User Language Interpreter** is integrated in most of **Bartels AutoEngineer** modules. I.e., **Bartels User Language** programs can be called from each of these BAE modules.

Starting User Language Programs

When calling a **User Language** program, the name of the program to be executed must be specified, and this program must be available in the `ulcprog.vdb` file of the BAE programs directory. The program name can be specified either explicitly or implicitly. An explicit **User Language** program call is applied with the BAE menu item Run User Script from the File menu of one of the interpreter environments **Schematic Editor**, **Layout Editor**, **Autorouter**, **CAM Processor**, **CAM View** or **Chip Editor**. After activating the Run User Script function the name of the required program must be typed in; on empty string or question mark (**?**) input or on mouse-click to the program name query a popup menu with the available **User Language** programs is displayed for selecting the required program with mouse-pick.

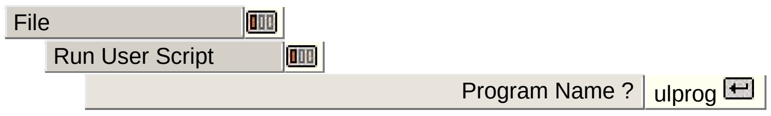
User Language programs can also be called by keystrokes, i.e., by pressing a standard key (0, 1, ..., 9, a, b, c, ...) or a function key (F1, F2, ..., F12). This implicit program call facility is available with the BAE function menu active, i.e., this type of program call is possible at any time unless another interactive keyboard input currently is requested. The keystroke program name is build from the currently active **User Language Interpreter** environment (`scm` for **Schematic Editor**, `ged` for **Layout Editor**, `ar` for **Autorouter**, `cam` for **CAM Processor**, `cv` for **CAM View**, `ced` for **Chip Editor**), an underscore and the name of the pressed key. E.g., the program named **scm_f4** is called when pressing function key F4 in the **Schematic Editor**, **cam_8** is called when pressing digit key 8 in the **CAM Processor**, **ged_r** is called when pressing standard key r in the **Layout Editor**, **ar_#** is called when pressing standard key # in the **Autorouter**. The **User Language Interpreter** only performs an automatic keystroke program calls if the corresponding program is available in `ulcprog.vdb`.

A special method of implicit **User Language** program call is provided with the startup sequence of the interpreter environment. This feature automatically executes a **User Language** program with a predefined name when starting an interpreter environment (**scm_st** in **Schematic Editor**, **ged_st** in **Layout Editor**, **ar_st** in **Autorouter**, **cam_st** in **CAM Processor**, **cv_st** in **CAM View**), thus, e.g., allowing for automatic system parameter setup.

Bartels User Language also provides system functions for performing key programming and menu assignments. Using these functions (e.g., in **User Language** startup programs) provides a convenient way of dynamically changing the **Bartels AutoEngineer** user interface. Another special **User Language** system function is provided for calling **User Language** programs from other **User Language** programs.

Examples

Call the **User Language** program named `ulprog`:


File
Run User Script
Program Name ? ulprog

Files

`ulcprog.vdb` -- BAE **User Language** database in BAE programs directory

See also

User Language Compiler, Schematic Editor, Layout Editor, Autorouter, CAM Processor, CAM View, **USERLIST**, Bartels User Language - Programmer's Guide

Diagnosis

The error messages issued by the **User Language Interpreter** are intended to be self-explanatory.

Warnings

The **Bartels User Language Interpreter** is a powerful tool for starting programs for the manipulation of DDB file contents and for generating CAM data. Even the BAE user interface can be considerably changed and/or extended with **User Language** programs. It is advisable to test each new **User Language** program in a non-critical environment (test software installation, test jobs, backup of real jobs, etc.) until confidence in the program is established for unrestricted use on real jobs. It is also strongly recommended to ensure security, e.g., to prevent foreign persons from implanting destructive **User Language** programs to `ulcprog.vdb`.

7.17 USERLIST

Name

userlist - User Programmable List Generator

Synopsis

```
userlist scriptfile projectfile jobname
```

Description

The **USERLIST** utility program is a user-programmable ASCII list generator. **USERLIST** interprets the user-defined **USERLIST** script **`<scriptfile>.usf`**, analyzes the net list named **`<jobname>`** from the requested job file **`<projectfile>.ddb`**, and produces a text output listing file named **`<projectfile>.<ext>`**. The listing file extension **`<ext>`**, the output format and the type of net list information to be extracted are defined in the **`<scriptfile>.usf`** userlist script.

Input File Format

Start Data, End Data, Comments

The **USERLIST** script file must start with the definition of the output file name extension which is defined with the command

```
EXTENSION = "<ext>";
```

where <ext> is the file name extension (a maximum length of up to three characters is allowed). The **USERLIST** script must end with the **`ENDSPEC`** keyword. Commentary text can be placed between **`/*`** and **`*/`**.

FOR Command

The **`FOR`** command is used for selecting elements of a certain class. The formal syntax of the **`FOR`** command is:

```
FOR (ALL <class> ) { <commands> }
```

where **`<class>`** specifies the class of the objects to be scanned. Valid classes can be selected with the keywords **`NETS`**, **`PARTS`**, **`PINS`**, **`ATTRIBUTES`** or **`<attname>`**. **`NETS`** iterates the net list object class, **`PARTS`** iterates the part list object class, **`PINS`** iterates pin lists and **`ATTRIBUTES`** iterates attribute lists. **`<attname>`** is used for scanning the attribute value list of the attributes named **`<attname>`**. The command list **`<commands>`** is processed once for each element of the specified object class. **`FOR`** commands can be nested to give more control. The nested **`FOR`** loop in

```
FOR (ALL NETS) { FOR (ALL PINS) {...}}
```

would for example find the first net, and then for all pins of that net perform **`<commands>`**. It would then repeat the operation on the second net and so on until it has completed processing of all nets.

Output Commands

The output commands are **PRINT** and **PRINTFOR**. The formal syntax of the **PRINT** command is:

```
PRINT(<parameters>);
```

where **<parameters>** specify the list of output items separated by commas. Output items enclosed in quotation marks are printed as literal text. **QUOTES** keyword can be used to print quotation marks. The **TAB** keyword prints a tab character. The **CR** keyword prints a newline. Other output items can be specified with attribute names (as listed below) followed by

```
:<length>:<decimals>
```

where **<length>** is the output length and **<decimals>** is the output precision for the corresponding number and/or string value. Negative **<length>** values apply for left-aligned output, and

```
%<length>:<decimals>
```

will include leading zeros. Default values are 3 for **<decimals>** and output item length for the **<length>** value. The **<length>** value is adjusted to the output item length if necessary. Distance values can be converted to mm or inch units by appending **" MM"** or **" INCH"**, respectively. The **PRINT** command can be applied as in

```
PRINT(QUOTES,PINWIDTH:7:3,QUOTES);   /* output: "  3.756" */

PRINT(QUOTES,PINWIDTH%7:3,QUOTES);   /* output: "003.756" */

PRINT(QUOTES,PINWIDTH:-7:3,QUOTES);     /* output: "3.756  " */
```

The **PRINT** command syntax allows for uppercase and/or lowercase name and/or attribute value outputs by adding blank-separated **UPPER** or **LOWER** keywords after name or attribute specifications.

The **PRINTFOR** command is used to scan through a particular object class and print a list of elements with a defined separator. The formal syntax of the **PRINTFOR** command is:

```
PRINTFOR (ALL <class>) SEPERATOR(<sep>),ELEMENTS(<elements>);
```

where **<class>** is the object class to be scanned (as defined in the **FOR** command), **<sep>** is the separator to be used (e.g., **","**, **"CR"**, etc.), and **<elements>** are the elements to be listed. The syntax of the parameter lists in **<sep>** and **<elements>** is the same as for the **PRINT** parameter list. If **PRINTFOR** is nested in a **FOR** loop, then the output is automatically restricted to the current element of the **FOR** loop.

IF Command

The **`IF`** command allows commands to operate conditionally. The formal syntax of the **`IF`** command is:

```
IF (<expr>) { <commands> }
```

and/or

```
IF (<expr>) { <commands> } ELSE { <commands> }
```

The formal syntax of the **`IF`** expression <expr> is given either by

```
? <attr>
```

or by the comparison expression

```
<attr> <operator> <attr|constant>
```

The **`? <attr>`** expression is used to check whether the attribute specified by <attr> is available. Available operators for comparison expression are **=** (equal), **<>** (not equal), **<** (less than), **>** (greater than), **<=** (less than or equal), **>=** (greater than or equal).

Counter

The commands

```
CLEARCOUNTER;
```

and

```
COUNTUP;
```

are used for controlling an internal counter. **`CLEARCOUNTER`** sets the counter value to zero. **`COUNTUP`** increments the counter value. The current counter value can be accessed via the **`COUNTVALUE`** attribute (see also below).

Attributes

The following attributes can be accessed:

Net Data	`NETNAME`	Net Name
	`NETPINCOUNT`	Number of Pins connected to Net
	`PRIORITY`	Net Priority (for the Router)
	`MINDIST`	Net Minimum Clearance (for the Router)
	`NETNUMBER`	Net Number
Part Data	`PARTNAME`	Part Name
	`PINCOUNT`	Number of Pins defined on Part
	`FREEPINS`	Number of unconnected Pins in the Part
	`PARTATTRIBCOUNT`	Number of Attributes in the Part
	`$attributname`	Value of selected Attribute in the Part
Pin Data	`PINNAME`	Pin Name
	`PINWIDTH`	Pin Routing Width
General Data	`PROJECTNAME`	Project and/or Design Name
	`ATTRIBCOUNT`	Number of Attributes matching current Name/Value Combination
	`ATTRIBNAME`	Name of selected Attribute
	`ATTRIBVALUE`	Value of selected Attribute
	`COUNTVALUE`	Current Counter Value

Examples

Net list generator **`conconv.usf`**:

```
/* Connection List Generator */

EXTENSION = ".con";

PRINT ("LAYOUT ", PROJECTNAME, ";", CR);

PRINT ("PARTS",CR);

FOR (ALL PARTS)

{

    PRINT ("  ",PARTNAME," : ",$plname,";",CR);

}

PRINT ("CONNECT",CR);

FOR (ALL NETS)

{

    PRINT ("  /", NETNAME,"/ ");

    PRINTFOR (ALL PINS)

        SEPERATOR ("="), ELEMENTS (PARTNAME,".",PINNAME);

    PRINT (";",CR);

}

PRINT ("END.",CR);

ENDSPEC
```

The **USERLIST** script **`conconv.usf`** can be applied as in

```
>  userlist conconv design board ↵
```

where net list **`board`** of the job file **`design.ddb`** is analyzed, and the output net list file **`design.con`** with the following contents is produced:

```
LAYOUT board;

PARTS

        ic1 : dil14;

        ic2 : dil14;

        ic3 : dil16;

CONNECT

        /gnd/ ic1.1=ic2.2=ic3.9;

        /vcc/ ic1.11=ic2.5=ic3.7;

END.
```

The part list generator **`partlist.usf`**

```
EXTENSION = ".ptl";

FOR (ALL $plname)

{

    PRINT (ATTRIBCOUNT," ",ATTRIBVALUE,CR);

}

ENDSPEC
```

would create output file **`<job>.ptl`** with the following contents:

```
3 cap50

4 dil14

2 dil16

1 r75
```

Diagnosis

The error messages issued by **USERLIST** are intended to be self-explanatory.

7.18 VALCONV

Name

valconv - VALID to Bartels Conversion

Synopsis

```
valconv projectname
```

Description

The **VALCONV** utility program converts part lists and net lists from VALID format (i.e., VALID schematic output) to a BAE DDB (Design DataBase) file.

VALCONV accepts the BAE DDB destination file name **`projectname`** as first argument. This file is created with extension **`.ddb`**, but this extension must not be included with the command line.

VALCONV reads the VALID part list file **`pstxprt`** and the VALID net list file **`pstxnet`**. These files are generated by the VALID system and must reside in the current directory when calling **VALCONV**.

After successful processing, a logical (unpacked) net list will exist in the BAE DDB file named **`<projectname>.ddb`**. This internal logical net list can be converted to a physical net list with the **Packager**, in the same way as information from the BAE schematic. A layout element can then be created in that design file, parts placed and traces routed, and it will also be possible to perform pin and gate swaps.

Files

`pstxprt` -- VALID part list
`pstxnet` -- VALID net list

See also

NETCONV, **Packager**, **LOGLIB**

Diagnosis

The error messages issued by **VALCONV** are intended to be self-explanatory.

Warnings

Input file identifiers for part names, pin names or net names containing special characters (**`-`**, **`+`**, **`/`**, **`(`**, **`=`**, ...) must be enclosed in single-quotes or double-quotes. Please contact our technical support for guidance if you want to transfer a net list format different from the description herein .