CR-5000

For System Administrator

Contents

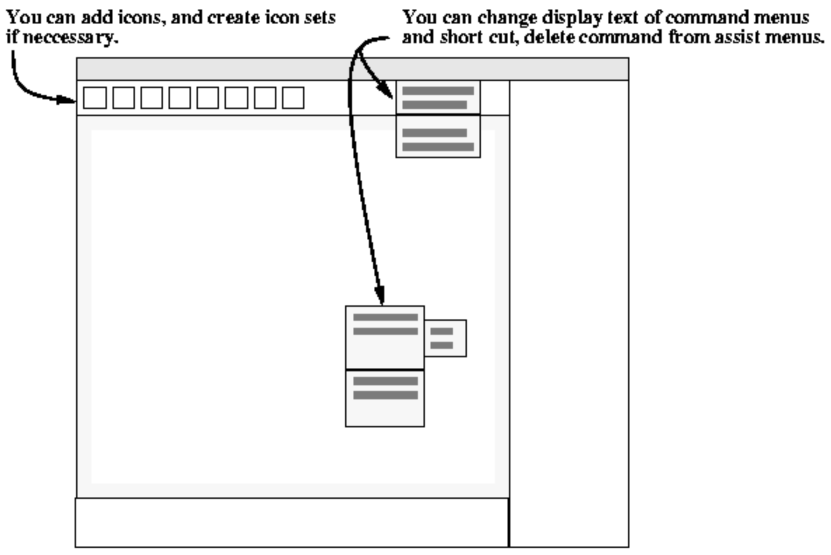
Chapter 1 SCHEMATIC EDITOR MENU CUSTOMIZATION

System Designer schematic editor has a menu customizing function to support exible use to meet designers' needs. The former half of this chapter explains simple methods of changing the menu.
The latter half of this chapter explains the methods of changing menus and changing icon sets depending on selection status.
The contents of each section are roughly as follows:

1.1 Overview of Customization	Explains brie y what you can do with the menu customizing function.
1.2 Menu Definition File	Explains a format of the menu def-inition file.
1.3 Menu Customization 1	Expalins how to customize basic menus.
1.4 Menu Customization 2	Explains the methods of changing menus and changing icon sets depending on selection status.
1.5 Keyword of Menu Denition File	Explains keywords of the menu definition file.

1.1 Overview of Customization

The menu customizing functionality of the schematic editor customizes editor menus, convinient for designers to make the environment more useful. For example, you can change display text of command menus and short cut, delete commands from assist menus, change commands, or add commands to assist menus as well as add icons, and create icon sets if necessary.


You can add icons, and create icon sets if neccessary.
You can change display text of command menus and short cut, delete command from assist menus.

The schematic editor consists of GUI in the figure above. Items possible to be customized are as follows:

- Command Menu
  Adding/Changing/Deleting commands. Assigning short cut keys and mnemomics.
- Assist Menu
  Adding/Changing Deleting commands. Structuring assist menus corresponding to selection status of object.
- Icon Bar
  Arranging/Adding/Changing/Deleting icons.

"Command" here refers to command text strings of the schematic editor. For the details of syntax, refer to "Schematic Editor Command Reference".

1.2 Menu Definition File

GUI definition of the schematic editor is written in the list structuring text file (which is similar to lisp language).[1]

1.2.1 GUI Definition File

The files defining GUI of the schematic editor are explained here. The files in the list below structure GUI of the schematic editor.

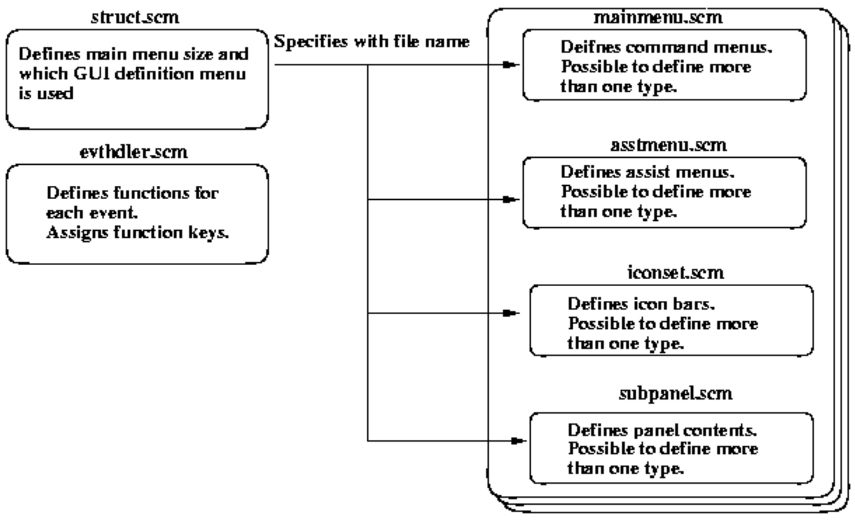
File Name	Contents to be Defined
struct.scm	Whole GUI Structure
mainmenu.scm	Command Menu Structure
asstmenu.scm	Assist Menu Structure
iconset.scm	Icon Set for Icon Bar
evthdler.scm	Functin Allocation for Each Event
subpanel.scm	Panel Structure used for Sub-panel

The GUI definition files are installed in "$ZDSROOT/scm/editor/(eng or eng)" To edit them, copy them under "$HOME/cr5000/ds/editor". When using them after finishing custo mizing, copy them to "$ZDSROOT/scm/editor/(jpn or eng)".

1.In this revision, to edit this text file, use a general text editor.

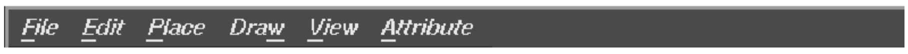
1.2.2 Relations among GUI Definition Files

The chart below indicates relations among the GUI definition files.

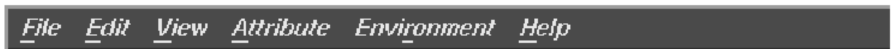

struct.scm
Defines main menu size and which GUI definition menu is used
evthdler.scm
Defines functions for each event. Assigns function keys.
Specifies with file name
mainmenu.scm
Deifnes command menus. Possible to define more than one type.
asstmenu.scm
Defines assist menus. Possible to define more than one type.
iconset.scm
Defines icon bars. Possible to define more than one type.
subpanel.scm
Defines panel contents. Possible to define more than one type.

struct.scm, mainmenu.scm, asstmenu.scm, and iconset.scm can control multiple GUI definitions with keywords. struct.scm enables you to select GUI for designers or GUI for environment managers by specifying a command menu (mainmenu.scm) or an assist menu (asstmenu.scm) with its keyword.

- Example of GUI for Designers
  Example of limiting use by displaying command menus for only designing.



- Example of GUI for Environment Managers
  Example of removing Draw/Place/Hierarchy menus so that command menus to change environment remain.



Preparation (Copy GUI definition file in local environment)

(1) Copy "struct.scm" in local environment.

- Copy GUI definition file by using (PC) explorer.
  Copy "Install directory\zds\scm\editor\eng\struct.scm " in "$HOME\cr5000\ds\editor" folder.

- Copy GUI definition file by using (PC) explorer.
  Copy "$ZDSROOT\scm\editor\eng\struct.scm" in
  "$HOME\cr5000\ds\editor"

(2) Copy each menu definition file in local environment.

- Copy GUI definition file by using (UNIX) UNIX shell command "cp".
  Copy "$ZDSROOT/scm/editor/eng/pcb-std/*.scm" in "$HOME/cr5000/ds/editor".
- Copy GUI definition file by using (PC) explorer.
  Copy "Install directory\zds\scm\editor\eng\pcb-std\*.* " in
  "$HOME\cr5000\ds\editor" folder.

(3) Change stored place of each menu definition file defined in "struct.scm".
For UNIX, open "struct.scm" with "vi" or "Screen Editor".
For PC, open "struct.scm" with "Notepad".
Stored place of each menu definition file is described as follows.

```
(define sd:editor-struct-list
   (
     ("PCB-std"
     "Standard UI"
          (SD:MAINCNVS  500 500)
          (SD:ICONSET   "$ZDSROOT/scm/editor/eng/pcb-std/iconset.scm")
          (SD:MAINMENU  "$ZDSROOT/scm/editor/eng/pcb-std/mainmenu.scm")
          (SD:ASSTMENU  "$ZDSROOT/scm/editor/engeng/pcb-std/asstmenu.scm")
          (SD:EVTHDLER  "$ZDSROOT/scm/editor/eng/pcb-std/evthdler.scm")
          (SD:ICONBAR   "$ZDSROOT/scm/editor/eng/pcb-std/iconbar.scm")
          (SD:SUBPANEL  "$ZDSROOT/scm/editor/eng/pcb-std/subpanel.scm")
          (SD:USERAREA  "$ZDSROOT/scm/editor/eng/pcb-std/userarea.scm"))
     ("PCB-next"
     "Sample UI for designer"
          (SD:MAINCNVS  500 500)
          (SD:ICONSET   "$ZDSROOT/scm/editor/eng/pcb-next/iconset.scm" )
          (SD:MAINMENU  "$ZDSROOT/scm/editor/eng/pcb-next/mainmenu.scm")
          (SD:ASSTMENU  "$ZDSROOT/scm/editor/eng/pcb-next/asstmenu.scm")
          (SD:EVTHDLER  "$ZDSROOT/scm/editor/eng/pcb-next/evthdler.scm")
          (SD:ICONBAR   "$ZDSROOT/scm/editor/eng/pcb-next/iconbar.scm" )
          (SD:SUBPANEL  "$ZDSROOT/scm/editor/eng/pcb-next/subpanel.scm")
          (SD:USERAREA  "$ZDSROOT/scm/editor/eng/pcb-next/userarea.scm"))
   ))
(define sd:editor-struct-current-struct "PCB-std")
```

Change stored place of menu definition files that are described from the line "pcb-std" to the line "pcb-next" as follows.

```
(define sd:editor-struct-list
    (
      ("PCB-std"
      "Standard UI"
           (SD:MAINCNVS  500 500)
           (SD:ICONSET   "$ZDSROOT/scm/editor/eng/pcb-std/iconset.scm") ;;change
           (SD:MAINMENU  "$ZDSROOT/scm/editor/eng/pcb-std/mainmenu.scm");;change
           (SD:ASSTMENU  "$ZDSROOT/scm/editor/engeng/pcb-std/asstmenu.scm");;change
           (SD:EVTHDLER  "$ZDSROOT/scm/editor/eng/pcb-std/evthdler.scm");;change
           (SD:ICONBAR   "$ZDSROOT/scm/editor/eng/pcb-std/iconbar.scm");;change
           (SD:SUBPANEL  "$ZDSROOT/scm/editor/eng/pcb-std/subpanel.scm");;change
           (SD:USERAREA  "$ZDSROOT/scm/editor/eng/pcb-std/userarea.scm"));;change
      ("PCB-next"
      "Sample UI for designer"
           (SD:MAINCNVS  500 500)
           (SD:ICONSET   "$ZDSROOT/scm/editor/eng/pcb-next/iconset.scm" )
           (SD:MAINMENU  "$ZDSROOT/scm/editor/eng/pcb-next/mainmenu.scm")
           (SD:ASSTMENU  "$ZDSROOT/scm/editor/eng/pcb-next/asstmenu.scm")
           (SD:EVTHDLER  "$ZDSROOT/scm/editor/eng/pcb-next/evthdler.scm")
           (SD:ICONBAR   "$ZDSROOT/scm/editor/eng/pcb-next/iconbar.scm" )
           (SD:SUBPANEL  "$ZDSROOT/scm/editor/eng/pcb-next/subpanel.scm")
           (SD:USERAREA  "$ZDSROOT/scm/editor/eng/pcb-next/userarea.scm"))
    ))
(define sd:editor-struct-current-struct "PCB-std")
```

(4) After modification, start Schematic Sheet Editor of System Designer. If it starts properly, GUI definition file has already been prepared in local environment.

1.3 Menu Customization1

1.3.1 Introduction

This section explains a basic method of customizing menus. This is a method of customizing command menus and assist menus used as GUI when designing a schematic using System Designer.

1.3.2 Format

The mainmenu.scm format and asstmenu.scm format are simply explained here.
The mainmenu.scm and asstmenu.scm are in almost the same format.
The format is a list type text file starting with ’(’ and endin* *g with ’)’. The visible image of a menu is directly written in the text file. The examp* *le is as follows:

```
;;
;; This is a definition of mainmenu.scm.
;;
   ( "Open..."                                                  ;;  1
      (SD:KEY_NO_USE  "Ctrl+o")                                 ;;  2
          ((! SD:STAT_C_IDLE))                                  ;;  3
          (SD:ACTION_SCHEME   "(sd:edt-menu-open %c
’update)")                                                      ;;  4
   )                                                            ;;  5
   SD:MENU_SEPARATOR                                            ;;  6
   ( "Save"                                                     ;;  7
        (SD:KEY_NO_USE  "Ctrl+s")                               ;;  8
        ((! SD:STAT_C_IDLE))                                    ;;  9
        (SD:ACTION_SCHEME   "(sd:edt-menu-save %c)")            ;; 10
   )                                                            ;; 11
```

This is an example that "Open..." and "Save" are defined. The explanation of each line is as follows: (A comment line starts from ’;’ and ends at the end of the line.)

(1) 1st Line: The starting text of menu item ’(’, and the defi* *nition of the title text of "Open...".

(2) 2nd Line: The definition of a mnemomic (Alt+key) and a short cut (Ctrl+key).
A mnemonic and short cut are defined in the line enclosed with ’(’ and ’)’. In the example here, the mnemonic is defined as "SD:KEY_NO_USE", which means "No Definition", while the short cut is defined as "Ctrl+o", which means that the command "Open..." is assigned to a short cu* *t "Ctrl+o".

(3) 3rd Line: The definition of conditions for executing "Open..." command by the schematic editor.
The line enclosed with ’(’ and ’)’ defines a condition, i.e. the schematic editor status. In the example here, you can see
"! SD:STAT_C_IDLE", which means that the command "Open..." can be effective only when the schematic editor is in idling status.
In the third line, multiple conditions can be defined. For example, "Cannot be available when inputting a part." and "Can be available only when selecting a part", etc., can also be defined here. The keywords to describe the schematic editor status can be refered to in section 1.5.

(4) 4th Line: The definition of commands to be activated when "Open..." is selected.
The line enclosed with ’(’ and ’)’ defines commands to be activated. The first field describes the type of command. There are 2 types of commands.

- SD:ACTION_SCHEME(Scheme Command Execution)
  Each dialog in the schematic editor is written in Scheme language. In this example, a dialog, File Opener, is displayed and prompts you to open a schematic sheet.
  The scheme command which can be translated into "SD:ACTION_SCHEME" has not been made public in this revision.
- SD:ACTION_COMMAND(Editor Command Execution)
  Executes editor commands for the schematic editor. For the details of the command syntax, refer to "Schematic Editor Command Reference".
- SD:ACTION_COMMAND_F(Editor Command Forced Execution)
  If other editor command is being executed, cancels that command and executes the defined editor command compulsorily. For the details of the command syntax, refer to "Schematic Editor Command Reference".

(5) 5th Line: ')' indicating the end of the definition of "Open..." menu.

(6) 6th Line: Definition of separtor.
To divide a menu, write "SD:MENU_SEPARATOR"

1.3.3 Customizing Command Menu

The method of customizing command menus is explained here. A command menu refers to a menu bar displayed at the top of the main window.

To customize the command menu, edit mainmenu.scm .

The initial-installed menu bar looks like the Figure below. It has some categories such as "File" or "Edit".

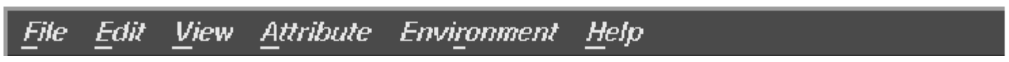


Figure 1.1 Command Menus

Removing one of the categories from the command menu.

The method of removing a category "Environment" from the command menu is explained here as an example.

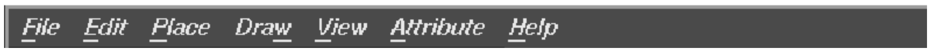
(1) Open "mainmenu.scm" by a text editor.

(2) Search ("Environment" line, and remove from ("Environment" to before ("Utilities" line, or make those lines comment lines.

```
;;--------
( “Enviromnent” “R” (SD:STAT_F_IDLE)
;;--------
   to
)
   before
;;--------
( “Utilities” “U” (SD:STAT_F_IDLE)
;;--------
```

Remove the line before ("Utilities", i.e. through ')', or make those lines comment lines.

(3) Restart the schematic editor.
The category "Environment" is removed from the command menu like the following figure.



Adding "Search Resister" to a category "Edit".

The method of adding [Search Resister] to in between [Redo] and Select Mode in a category "Edit" is explained here as an example.

(1) Open "mainmenu.scm" by a text editor.

(2) Search ("Edit" line, and add the following command definition to after the "Redo" definition within that "Edit" definition.

```
( "Search Resister"                                                        ;; 1
    (SD:KEY_NO_USE    SD:KEY_NO_USE)                                       ;; 2
    (SD:STAT_F_IDLE (! SD:STAT_C_IDLE))                                    ;; 3
    (SD:ACTION_COMMAND  "(select focus:component reference:R*)")           ;; 4
)                                                                          ;; 5
```

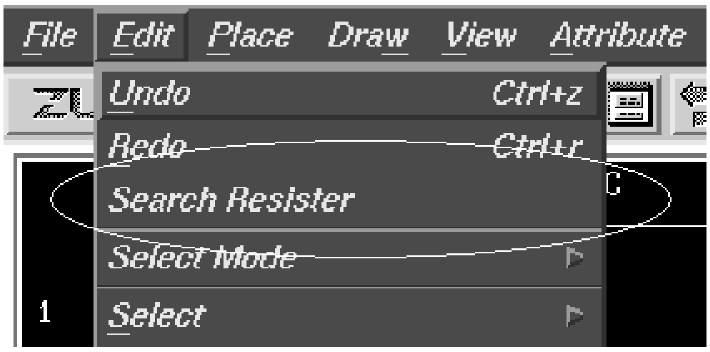
a. 1st Line: Title.

b. 2nd Line: Definition of mnemomic and short cut.
Write "SD:KEY_NO_USE" meaning "no definition" here

c. 3rd Line: Definition of execution conditions. Define "when schematic sheet is open (SD:STAT_F_IDLE) and no command is being executed (!SD:STAT_C_IDLE), the "Search Resister" command can be executed."here.

d. 4th Line: Command definition. This line means to select parts of which initial reference letter is "R" using a schematic editor command "Select Command".
For the details of "Select Command", refer to "Select Command" chapter in "Schematic Editor Command Reference".

[Edit] -> [Search Resister]is created like the following figure.



Adding "Reference Duplication Check" to a category "Utilities".

The method of adding [Reference Duplication Check] to the top of a category "Utilities" is explained here as an example.

(1) Open "mainmenu.scm" by a text editor.

(2) Search ("Utilities" line, and add the following command definition to before the "Icon Bar" definition within that "Utilities" definition.

```
( "Reference Duplication Check"                                             ;; 1
   (SD:KEY_NO_USE    SD:KEY_NO_USE)                                          ;; 2
   (SD:STAT_F_IDLE (! SD:STAT_C_IDLE))                                       ;; 3
   (SD:ACTION_COMMAND                                                        ;; 4
     "(rulecheck referenceDupl )( mark color:1 objectList:dupref )")         ;; 5
)                                                                            ;; 6
```

a. 1st Line: Title.

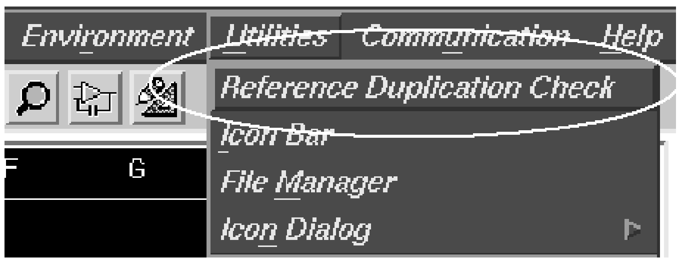
b. 2nd Line: Definition of mnemomic and short cut. Write "SD:KEY_NO_USE" meaning "no definition" here.

c. 3rd Line: Definition of execution conditions. Define "when schematic sheet is open (SD:STAT_F_IDLE) and no command is being executed (!SD:STAT_C_IDLE), the "Search Resister" command can be executed." here.

d. 4th and 5th Lines: Command definition. These lines mean to mark parts that "Mark Command" issued error to after checking reference duplication using a schematic editor command "Rule Check".
For the details of "Mark Command" and "Rule Check", refer to Chapter

"mark" and Chapter "ruleCheck" in "Schematic Editor Command Reference" respectively.

[Utilities]->[Reference Duplication Check] is created like the following figure.



1.3.4 Customizing Assist Menu

The method of customizing assisit menus is explained here. An assist menu is a menu that is displayed by pressing the right button of mouse when a command is being executed.

To customize the assisst menu, edit asstmenu.scm.

Command options in an initial-installed assist menu can be selected depending on an executed command.

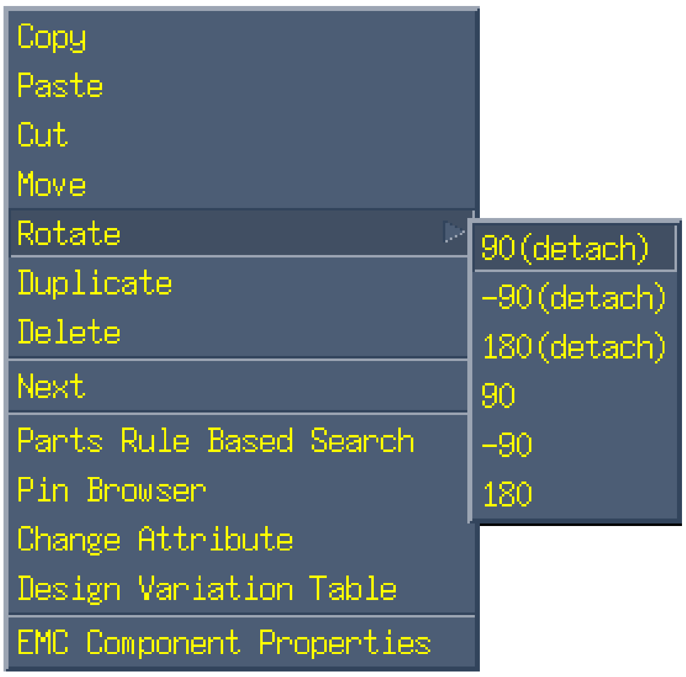


Figure 1.2 Assist Menu

Enter the angle in the submenu that appears when the [Rotate] AssistMenu is selected.

You can add angles to the [Rotate] AssistMenu by customizing the "asstmenu.scm".
As an example, we will describe how to add [45 (Detach)] to the submenu of the [Rotate] AssistMenu.

[Rotate] - [45 (Detach)] command rotates the selected component and detaches it from other components.

(1) Open "asstmenu.scm" using a general-purpose editor for text editing.
(2) Find the line with "Rotate" and add the definition of "45 (Detach)" in the same way as the already defined angles.
Enter the [Rotate] angle using integers or real numbers that are larger than 0.
Enter "#f" after the angle for components that are not to be detached.

```
( "Rotate"
(SD:KEY_NO_USE                              SD:KEY_NO_USE)
(SD:STAT_F_INSSCH)

( "45 (detach)"
(SD:KEY_NO_USE                              SD:KEY_NO_USE)
()
(SD:ACTION_SCHEME "( sd:editor-menu-rotate-comp %c 45)"))

( "90 (detach)"
(SD:KEY_NO_USE                              SD:KEY_NO_USE)
()
(SD:ACTION_SCHEME "( sd:editor-menu-rotate-comp %c 90)"))

( "-90 (detach)"
(SD:KEY_NO_USE                              SD:KEY_NO_USE)
()
(SD:ACTION_SCHEME "( sd:editor-menu-rotate-comp %c -90)"))

( "180 (detach)"
(SD:KEY_NO_USE                              SD:KEY_NO_USE)
()
(SD:ACTION_SCHEME "( sd:editor-menu-rotate-comp %c 180)"))
)
( "90"
(SD:KEY_NO_USE                              SD:KEY_NO_USE)
()
(SD:ACTION_SCHEME "( sd:editor-menu-rotate-comp %c 90 #f)"))

( "-90"
(SD:KEY_NO_USE                              SD:KEY_NO_USE)
()
(SD:ACTION_SCHEME "( sd:editor-menu-rotate-comp %c -90 #f)"))

( "180"
(SD:KEY_NO_USE                              SD:KEY_NO_USE)
()
(SD:ACTION_SCHEME "( sd:editor-menu-rotate-comp %c 180 #f)"))
)
```

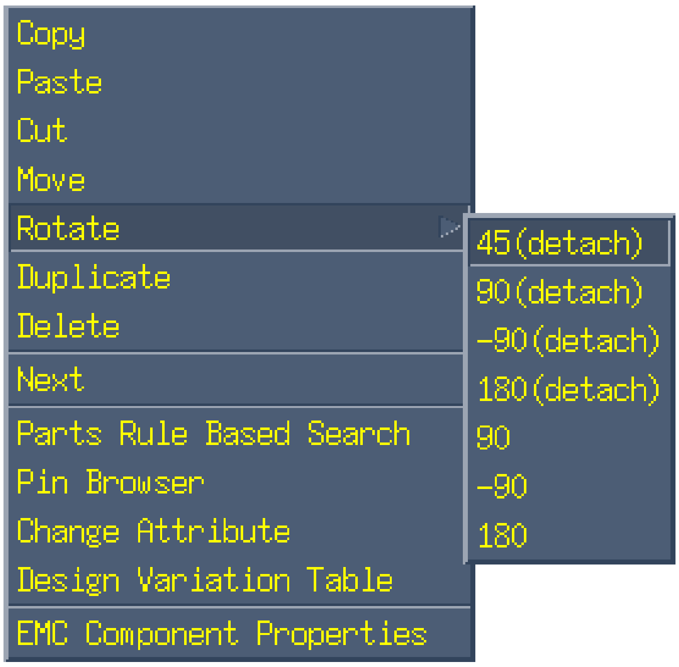
a. 1st Line: Title.

b. 2nd Line: Definition of mnemomic and short cut. Write "SD:KEY_NO_USE" meaning "no definition" here.

c. Define the condition of execution on the third line. Here this condition can be defined only when the schematic sheet is open (SD:STAT_F_INSSCH).

d. The angle of rotation is defined on the 5th line.
Title, mnemomic/short cut and execution conditions are defined here in a nest format with the same methods as 1st, 2nd, and 3rd lines. The syntax is the same as one in 1st, 2nd, and 3rd lines.

Add [Rotate] - [45 (Detach)] as shown below.



Adding Line Width Specification to the assist menu of Draw command.

The method of adding Line Width Specification to the assist menu of polyline input is explained here.

(1) Open "asstmenu.scm" by a text editor.

(2) Search ("DrLine" line, and add the following command definition to in between "Input Mode" and "Lead Angle" both defined within that "DrLine" definition.

```
( "Line Width Specification"                                   ;;  1
(SD:KEY_NO_USE    SD:KEY_NO_USE)                               ;;  2
  ((! SD:STAT_C_LINE))                                         ;;  3
                                                               ;;  4
( "Line Width 0"                                               ;;  5
   (SD:KEY_NO_USE    SD:KEY_NO_USE)                            ;;  6
   ()                                                          ;;  7
   (SD:ACTION_COMMAND  "width:0"))                             ;;  8
( "Line Width 1"                                               ;;  9
   (SD:KEY_NO_USE    SD:KEY_NO_USE)                            ;;10
   ()                                                          ;;11
   (SD:ACTION_COMMAND  "width:1"))                             ;;12
( "Line Width 2"                                               ;;13
     (SD:KEY_NO_USE    SD:KEY_NO_USE)                          ;;14
     ()                                                        ;;15
     (SD:ACTION_COMMAND  "width:2"))                           ;;16
( "Line Width 3"                                               ;;17
     (SD:KEY_NO_USE    SD:KEY_NO_USE)                          ;;18
     ()                                                        ;;19
     (SD:ACTION_COMMAND  "width:3"))                           ;;20
( "Line Width 4"                                               ;;21
     (SD:KEY_NO_USE    SD:KEY_NO_USE)                          ;;22
     ()                                                        ;;23
     (SD:ACTION_COMMAND  "width:4"))                           ;;24
)                                                              ;;25
```

a. 1st Line: Title.

b. 2nd Line: Definition of mnemomic and short cut. Write "SD:KEY_NO_USE" meaning "no definition" here.

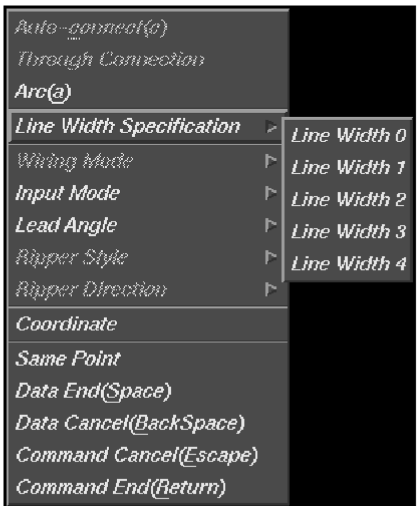
c. 3rd Line: Definition of execution conditions. The condition defined here is "only when "Line Command" is being executed, the command can be effective." (! SD:STAT_C_LINE).

d. 5th Line through 7th Line: Definition of specifying 0 in width. Title, mnemomic/short cut and execution conditions are defined here in a nest format with the same methods as 1st, 2nd, and 3rd lines. The syntax is the same as one in 1st, 2nd, and 3rd lines.

e. 8th Line: Command definition when the line width is 0. Designate 0 for "width:" which is an option of a schematic editor command "Line Command". For the details of "Line Command", refer to Section "Line" in Chapter "Draw Commands" in "Schematic Editor Command Reference".

f. 9th Line through 24th Line: These lines are repeat of 5th - 8th lines but different values of line width.

[Line Width Specification] -> [Width 0 - 4] is created like the following figure.



1.4 Menu Customization 2

This section explains the method of adding icons.

1.4.1 Adding Icons

The method of adding icons to the icon bar in the main window is explained here.
To customize the icon bar, edit iconset.scm .

Adding the icon to input line/polyline to the icon bar.

The method of adding "Line/Polyline" command defined in [Draw] -> [Line/Polyline] and in Drawing Icon dialog to the icon bar is explained here as an example.

(1) Open "iconset.scm" by a text editor.

(2) (;; 1st line in the fourth line is the_icon_bar_definition. There is only one type of icon bar definition initially. [Line/Polyline] icon will be added to after this definition. ’)’ in the fifty-ninth line corresponds to ’(’ of (;; 1st line in the fourth line. Define the following command before the fifty-ninth line.

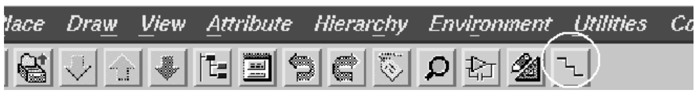
```
    ((SD:PATH_ZUE_PIX_24x24 "polyline.pm")            ;;1
    (SD:STAT_F_IDLE)                                  ;;2
    (SD:ACTION_COMMAND "line "))                      ;;3
```

a. 1st Line: Icon file specification.
   SD:PATH_ZUE_PIX_24x24 is a directory where the icon file is stored in and is equivalent to "$ZSYSROOT/pix/24x24".
   To refer to icon files, use the vueicon program for System Designer on HP(UNIX), or use "Paint" for System Designer on PC.

b. 2nd Line: Definition of execution conditions.
   The condition,"File is open" (SD:STAT_F_IDLE), is defined here to make this icon menu effective.

c. 3rd Line: Command definition. This is for executing "Line/Polyline" command like the command menu or assist menu.
   For the details of schematic editor command text strings, refer to Section "line" in "Schematic Editor Command Reference".

The "Line/Polyline" input icon is added to the menu bar like the following figure
.



1.5 Keyword

This section explains the format of GUI definition file and explains the execution conditions described in Section 1.3 and Section 1.4.

1.5.1 Format of GUI Definition File

struct.scm

```
(define sd:editor-struct-list
’(
    "PCB-std"
    "Standard UI"
    (SD:MAINCNVS  500 500)
    (SD:ICONSET   "$ZDSROOT/scm/editor/eng/pcb-std/iconset.scm")
    (SD:MAINMENU  "$ZDSROOT/scm/editor/eng/pcb-std/mainmenu.scm")
    (SD:ASSTMENU  "$ZDSROOT/scm/editor/eng/pcb-std/asstmenu.scm")
    (SD:EVTHDLER  "$ZDSROOT/scm/editor/eng/pcb-std/evthdler.scm")
    (SD:ICONBAR   "$ZDSROOT/scm/editor/eng/pcb-std/iconbar.scm")
    (SD:SUBPANEL  "$ZDSROOT/scm/editor/eng/pcb-std/subpanel.scm")
    (SD:USERAREA  "$ZDSROOT/scm/editor/eng/pcb-std/userarea.scm"))

    ("PCB-next"
    "Sample UI for designer"
    (SD:MAINCNVS  500 500)
    (SD:ICONSET   "$ZDSROOT/scm/editor/eng/pcb-next/iconset.scm" )
    (SD:MAINMENU  "$ZDSROOT/scm/editor/eng/pcb-next/
    mainmenu.scm")
    (SD:ASSTMENU  "$ZDSROOT/scm/editor/eng/pcb-next/asstmenu.scm")
    (SD:EVTHDLER  "$ZDSROOT/scm/editor/eng/pcb-next/evthdler.scm")
    (SD:ICONBAR   "$ZDSROOT/scm/editor/eng/pcb-next/iconbar.scm" )
    (SD:SUBPANEL  "$ZDSROOT/scm/editor/eng/pcb-next/subpanel.scm")
    (SD:USERAREA  "$ZDSROOT/scm/editor/eng/pcb-next/userarea.scm"))

  ))
(define sd:editor-struct-current-struct "PCB-std")
```

mainmenu.scm

```
;;
;;===========================================================
;;  mainmenu.scm format explanation
;;===========================================================
;;mainmenu.scm -> menuSet[N]
;;
;; menuSet         -> ’(’ menuSetName menuDef[N] ’)’
;;
;; menuDef         -> ’(’ menuName nimonicKey prhtedStat menuItemDef[N] ’)’
;;
;; menuItemDef  -> ’(’ menuItemName keyAssign prhtedStat action menuType ’)’|
;;                        ’(’ menuItemName keyAssign prhtedStat menuItemDef[N] ’)’  |
;;                        SD:MENU_SEPARATOR               # Menu Separator
;;
;; keyAssign    -> ’(’ nimonicKey shortCutKey ’)’
;;
;; prhtedStat   -> ’(’ statusDef[N] ’)’
;;
;; action      -> ’(’ actionKind <string> ’)’
;;
;; statusDef    -> ’(’ status[N] ’)’    | # Impossible to execute if enumerated.
;;                        status            | # Impossible to execute if specified.
;;                      ’(’ ! status[N] ’)’ | # Possible to execute if enumerated.
;;
;; status          -> commandStatus | fileStatus | selectStatus|
;;  # For the choices of each status, refer to "Execution Conditions".
;;
;; menuSetName  -> <string>
;; menuName       -> <string>
;; menuItemName -> <string>
;; nimonicKey       -> SD:KEY_NO_USE | <string>
;; shortCutKey      -> SD:KEY_NO_USE | <string>
;; actionKind        -> SD:ACTION_SCHEME _ SD:ACTION_COMMAND
;; menuType       -> ’’ | SD:MENU_NORMAL | SD:MENU_RADIO|SD:MENU_CHECK
;;              # Same as SD:MENU_NORMAL if omitted.
;;
;;
;;==============================================================
(define sd:mainmenu-def’(
;;==============================================================
( "File" "f" ((! SD:STAT_C_IDLE))
( "Open..."
(SD:KEY_NO_USE  "Ctrl+o")
((! SD:STAT_C_IDLE))
(SD:ACTION_SCHEME   "(sd:edt-menu-open %c ’update)"))

( "Save"
(SD:KEY_NO_USE  "Ctrl+s")
(SD:STAT_F_IDLE (! SD:STAT_C_IDLE))
(SD:ACTION_SCHEME   "(sd:edt-menu-save %c)"))
( "Save As..."
(SD:KEY_NO_USE  "Ctrl+a")
(SD:STAT_F_IDLE (! SD:STAT_C_IDLE))
(SD:ACTION_SCHEME   "(sd:edt-menu-saveas %c)"))

( "Sheet Frame..."
("s"           SD:KEY_NO_USE)
(SD:STAT_F_IDLE (! SD:STAT_C_IDLE))
(SD:ACTION_SCHEME" sd:dialog-map %c SD:D_SHEET_FRAME ’reset)"))

SD:MENU_SEPARATOR

( "Next Sheet"
("n"           SD:KEY_NO_USE)
((! SD:STAT_F_DEFSCH)   (! SD:STAT_C_IDLE))
(SD:ACTION_SCHEME   "(sd:edt-menu-nextsheet %c ’next)"))

( "Previous Sheet"
("o"           SD:KEY_NO_USE)
((! SD:STAT_F_DEFSCH)   (! SD:STAT_C_IDLE))
(SD:ACTION_SCHEME   "(sd:edt-menu-nextsheet %c ’prev)"))

( "Any Sheet..."
("t"           SD:KEY_NO_USE)
((! SD:STAT_F_DEFSCH)   (! SD:STAT_C_IDLE))
(SD:ACTION_SCHEME   "(sd:edt-menu-othersheet %c)"))

SD:MENU_SEPARATOR

( "Print..."
(SD:KEY_NO_USE  "Ctrl+p")
(SD:STAT_F_IDLE (! SD:STAT_C_IDLE))
(SD:ACTION_SCHEME   "(sd:edt-menu-print %c)"))

SD:MENU_SEPARATOR

( "Quit"
(SD:KEY_NO_USE  "Ctrl+q")
((! SD:STAT_C_IDLE))
(SD:ACTION_SCHEME   "(sd:edt-menu-exit %c)"))
)
```

asstmenu.scm

```
;;
;;=========================================================
;;  asstmenu.scm format explanation
;;=========================================================
;; asstmenu.scm -> asstmenuSet[N]
;;
;; asstmenuSet  -> '(' asstMenuName asstMenuDef[N] ')'
;;
;; asstMenuDef  -> '(' menuTitle prhbtedStat menuItemDef[N] ')'
;;
;; menuItemDef  -> '(' menuItemName keyAssign prhbtedStat action menuType ')'|
;;                 '(' menuItemName keyAssign prhbtedStat menuItemDef[N] ')' |
;;                  SD:MENU_SEPARATOR           # Menu Separator
;;
;; keyAssign    -> '(' nimonicKey shortCutKey ')'
;;
;; prhbtedStat  -> '(' statusDef[N] ')'
;;
;; action      -> '(' actionKind <string> ')'
;;
;; statusDef    -> '(' status[N] ')'    | # Impossible to execute if enumerated.
;;                  status              | # Impossible to execute if specified.
;;                 '(' ! status[N] ')'  | # Possible to execute if enumerated.
;;
;; status      -> commandStatus | fileStatus | selectStatus
;;  # For the choices of each status, refer to "Execution Conditions".
;;
;; asstMenuName -> <string>
;; menuTitle    -> <string>
;; menuItemName -> <string>
;; nimonicKey   -> SD:KEY | NO | USE | <string>
;; shortCutKey  -> SD:KEY | NO | USE | <string>
;; actionKind   -> SD:ACTION_SCHEME | SD:ACTION_COMMAND
;; menuType     -> '' | SD:MENU_NORMAL | SD:MENU_RADIO |
SD:MENU_CHECK
;;                  # Same as SD:MENU_NORMAL if omitted.
;;
;;=========================================================
(define sd:asstmenu-def'(
;;=========================================================
;;-------------------------------------------------------------------
( "ComponentSelect"
;;-------------------------------------------------------------------
```

```
(SD:STAT_F_IDLE (! SD:STAT_C_IDLE)
(! SD:STAT_S_COMPONENT SD:STAT_S_P_COMPONENT))

( "Copy"
(SD:KEY_NO_USE SD:KEY_NO_USE)
()
(SD:ACTION_SCHEME "(sd:editor-menu-copy-to-buffer %c 1)"))

( "Paste"
(SD:KEY_NO_USE SD:KEY_NO_USE)
()
(SD:ACTION_COMMAND "( Paste cBuf:1 "))
( "Cut"
(SD:KEY_NO_USE SD:KEY_NO_USE)
()
(SD:ACTION_SCHEME "(sd:editor-menu-cut-to-buffer %c 1)"))

( "Move"
(SD:KEY_NO_USE SD:KEY_NO_USE)
()
(SD:ACTION_COMMAND "( Move autoLift "))

( "Duplicate"
(SD:KEY_NO_USE SD:KEY_NO_USE)
()
(SD:ACTION_COMMAND "( Copy autoLift "))

( "Delete"
(SD:KEY_NO_USE SD:KEY_NO_USE)
()
(SD:ACTION_COMMAND "( Cut )"))

SD:MENU_SEPARATOR

( "Next"
(SD:KEY_NO_USE SD:KEY_NO_USE)
()
(SD:ACTION_COMMAND "( Select next )"))

SD:MENU_SEPARATOR

( "Parts Rule Based Search"
(SD:KEY_NO_USE SD:KEY_NO_USE)
()
(SD:ACTION_SCHEME "(sd:editor-srch-parts-dialog-map %c)"))
( "Attribute"
(SD:KEY_NO_USE SD:KEY_NO_USE)
()
(SD:ACTION_SCHEME "(sd:editor-property-dialog-map %c)"))
)
```

iconset.scm

```
;;
;;=========================================================
;;  iconset.scm format explanation
;;=========================================================
;; iconset.scm  -> iconset[N];;;; iconset     -> '(' iconsetName prhbtedStat lineDef[n] ')'
;;
;; lineDef     -> '(' iconDef[N] ')'
;;
;; iconDef     -> '(' filePath prhbtedStat action ')'
;;
;; filePath    -> '(' pathName fileName ')'
;;
;; action     -> '(' actionKind <string> ')'
;;
;; prhbtedStat -> '(' statusDef[N] ')'
;;
;; statusDef   ->  '(' status[N] ')'    # Impossible to execute if enumerated.
;;                  status              # Impossible to execute if specified.
;;                 '(' ! status[N] ')'  # Possible to execute if enumerated.
;;
;; iconsetName -> <string>
;; fileName    -> <string>
;; actionKind  -> SD:ACTION_SCHEME _ SD:ACTION_COMMAND
;;
;; pathName    -> SD:PATH_ZUE_PIX_16x16    |  # $ZSYSROOT/pix/16x16
;;                SD:PATH_ZUE_PIX_20x16    |  # $ZSYSROOT/pix/20x16
;;                SD:PATH_ZUE_PIX_24x24    |  # $ZSYSROOT/pix/24x24
;;                SD:PATH_ZUE_PIX_40x16    |  # $ZSYSROOT/pix/40x16
;;                SD:PATH_ZUE_PIX_32x32    |  # $ZSYSROOT/pix/32x32
;;                SD:PATH_ZUE_PIX_OTHER  | # $ZSYSROOT/pix/other
;;                SD:PATH_ZDS_PIX_16x16    |   # $ZDSROOT/pix/16x16
;;                SD:PATH_ZDS_PIX_20x16    |   # $ZDSROOT/pix/20x16
;;                SD:PATH_ZDS_PIX_24x24    |   # $ZDSROOT/pix/24x24
;;                SD:PATH_ZDS_PIX_40x16    |   # $ZDSROOT/pix/40x16
;;                SD:PATH_ZDS_PIX_32x32    |   # $ZDSROOT/pix/32x32
;;                SD:PATH_ZDS_PIX_OTHER  | # $ZDSROOT/pix/other
;;                <string>                    # Environment variable with its path.

;;
;;
;;=========================================================
(define sd:iconset-pcb-iconbar '( "pcb-iconbar" ()
;;=========================================================
( ;; 1st line
   ((SD:PATH_ZUE_PIX_OTHER "zlogo")
   ()
   (SD:ACTION_COMMAND "cCancel")
   "Cancel")
```

```
((SD:PATH_ZUE_PIX_24x24 "dataLoad")
()
(SD:ACTION_SCHEME "(sd:editor-menu-open %c)")
"Open")

((SD:PATH_ZDS_PIX_24x24 "push")
((! SD:STAT_S_COMPONENT))
(SD:ACTION_COMMAND_F "( Push )")
"Push Instance")

((SD:PATH_ZUE_PIX_24x24 "pop")
((! SD:STAT_F_INSSCH))
(SD:ACTION_COMMAND_F "( Pop )")
"Pop Block")

((SD:PATH_ZDS_PIX_24x24 "another")
((! SD:STAT_S_COMPONENT))
(SD:ACTION_SCHEME "(sd:editor-menu-definition-sheet %c)")
"Push Definition")

((SD:PATH_ZDS_PIX_24x24 "hmgr")
(SD:STAT_F_SYMBOL)
(SD:ACTION_SCHEME "((sd:id->subpanel %c) ’hcymgr)")
"Hierarchy Design Maneger")

((SD:PATH_ZDS_PIX_24x24 "brows")
((SD:STAT_F_SYMBOL SD:STAT_F_IDLE) (! SD:STAT_C_IDLE))
(SD:ACTION_SCHEME "(sd:editor-menu-exec-component-browser %c)")
"Component Browser")

((SD:PATH_ZUE_PIX_24x24 "setProp")
(SD:STAT_S_IDLE)
(SD:ACTION_SCHEME "(sd:editor-property-dialog-map %c)")
"Change Attribute")

((SD:PATH_ZUE_PIX_24x24 "view")
()
(SD:ACTION_SCHEME "(sd:editor-view-icon-dialog-map %c)")
"View Command Icon")

((SD:PATH_ZDS_PIX_24x24 "schm")
()
(SD:ACTION_SCHEME "(sd:editor-schematic-icon-dialog-map %c)")
"Schematic Capture Icon")

((SD:PATH_ZUE_PIX_24x24 "drwDlg")
()
(SD:ACTION_SCHEME "(sd:editor-draw-icon-dialog-map %c)")
"Drawing Command Icon")
```

```
    ((SD:PATH_ZUE_PIX_24x24 "undo")
    (SD:STAT_F_IDLE
    (! SD:STAT_C_IDLE   SD:STAT_C_COMPONENT SD:STAT_C_CPIN
    SD:STAT_C_CIRCLE SD:STAT_C_LINE     SD:STAT_C_SNET
    SD:STAT_C_BNET   SD:STAT_C_RECTANGLE SD:STAT_C_ARC))
    (SD:ACTION_SCHEME "(sd:editor-menu-undo %c)")
    "Undo")

    ((SD:PATH_ZUE_PIX_24x24 "redo")
    (SD:STAT_F_IDLE
    (! SD:STAT_C_IDLE   SD:STAT_C_COMPONENT SD:STAT_C_CPIN
    SD:STAT_C_CIRCLE SD:STAT_C_LINE     SD:STAT_C_SNET
    SD:STAT_C_BNET   SD:STAT_C_RECTANGLE SD:STAT_C_ARC))
    (SD:ACTION_SCHEME "(sd:editor-menu-redo %c)")
    "Redo")
    )
))
```

1.5.2 Execution Conditions

The conditions, i.e. status of the schematic editor, for executing a command written in the GUI definition file are defined with text strings explained here.

The conditions for executing a command can be defined by enumerating those text strings in the field for execution conditions (Refer to 1.3) in the GUI definition file.

Command Status

This is a text string definition to show which command is being executed in the schematic editor.
Write text strings in the execution condition field in mainmenu.scm, asstmenu.scm or iconset.scm in the follwing format.

- ’(’ Status Text String Status Text String....’)’
  Impossible to execute if enumerate like this.
- Status Text String
  Impossible to execute if specify.

- '(' ! Status Text String Status Text String....')'
  Possible to execute if enumerate like this.

Status Text String	Contents
SD:STAT_C_IDLE	No command is executed.
SD:STAT_C_ALIGN	Alignment Command
SD:STAT_C_BNET	Inputting Bus
SD:STAT_C_BREAKBLOCK	Breaking Circuit Block
SD:STAT_C_CHGSHEET	Changing Sheet
SD:STAT_C_CIRCLE	Inputting Circle
SD:STAT_C_COMPONENT	Inputting Component
SD:STAT_C_EXPORTCOMPONENT	Inputting Hierarchical Connector
SD:STAT_C_MULTICOMPONENT	Collective Gate Placement
SD:STAT_C_SWAPMULTICOMPONENT	Swap Components
SD:STAT_C_COPYVARIATION	Copying properties of design variation among different components.
SD:STAT_C_MOVEVARIATION	Moving properties of design variation among different components.
SD:STAT_C_REMOVEVARIATION	Delete Design Variation Property
SD:STAT_C_COPY	Copy
SD:STAT_C_CPIN	Inputting Pin
SD:STAT_C_CUT	Cut
SD:STAT_C_DEFINE	Define
SD:STAT_C_FRAME	Inputting Frame
SD:STAT_C_GENEBLOCK	Generating Partial Block
SD:STAT_C_JUMP	Moving to Other Layer
SD:STAT_C_LINE	Inputting Line
SD:STAT_C_MACRO	Executing Macro
SD:STAT_C_MARK	Mark
SD:STAT_C_MOVE	Move

Status Text String	Contents
SD:STAT_C_PASTE	Paste
SD:STAT_C_PATTERN	Paint
SD:STAT_C_POP	Moving to Upper Layer
SD:STAT_C_PUSH	Moving to Lower Layer
SD:STAT_C_PVIEWER	Inputting Property Viewer
SD:STAT_C_REDO	Redo
SD:STAT_C_RESHAPE	Reshape
SD:STAT_C_SELECT	Select
SD:STAT_C_SET	Setting Property
SD:STAT_C_SNET	Inputting Net
SD:STAT_C_STRETCH	Stretch
SD:STAT_C_TEXT	Inputting Text
SD:STAT_C_TEXTEDIT	Editing Text
SD:STAT_C_UNCONNECT	Unconnect
SD:STAT_C_UNDO	Undo
SD:STAT_C_ZOOM	Zoom

Selection Status

This is a text string definition to show which object is being selected. Write text strings in the execution condition field in mainmenu.scm, asstmenu.scm or iconset.scm in the follwing format.

- ’(’ Status Text String Status Text String....’)’
  Impossible to execute if enumerate like this.
- Status Text String
  Impossible to execute if specify.
- ’(’ ! Status Text String Status Text String....’)’
  Possible to execute if enumerate like this.

Status Text String	Contents
SD:STAT_S_IDLE	Nothing is selected.
SD:STAT_S_MIX	Multiple Objects

Status Text String	Contents
SD:STAT_S_BNETSEG	Bus Segment
SD:STAT_S_SNETSEG	Net Segment
SD:STAT_S_CIRCLE	Circle
SD:STAT_S_COMPONENT	Component Cell
SD:STAT_S_CPIN	Pin
SD:STAT_S_FRAME	Frame
SD:STAT_S_LINE	Line
SD:STAT_S_PATTERN	Paint
SD:STAT_S_PVIEWER	Property Viewer
SD:STAT_S_SPIN	Symbol Pin
SD:STAT_S_TEXT	Text
SD:STAT_S_P_NETSEG	Multiple Bus Segments
SD:STAT_S_P_CIRCLE	Multiple Circles
SD:STAT_S_P_COMPONENT	Multiple Component Cells
SD:STAT_S_P_CPIN	Multiple Pins
SD:STAT_S_P_FRAME	Multiple Frames
SD:STAT_S_P_LINE	Multiple Lines
SD:STAT_S_P_PATTERN	Multiple Paints
SD:STAT_S_P_PVIEWER	Multiple Property Viewers
SD:STAT_S_P_SNETSEG	Multiple Net Segments
SD:STAT_S_P_SPIN	Multiple Symbol Pins
SD:STAT_S_P_TEXT	Multiple Texts

Editor Status

This is a text string de_nition to show which _le is being edited in the schemtaic editor.
Write text strings in the execution condition _eld in mainmenu.scm, asstmenu.scm or iconset.scm in the follwing format.

- '(' Status Text String Status Text String....')'
  Impossible to execute if enumerate like this.
- Status Text String
  Impossible to execute if specify.
- '(' ! Status Text String Status Text String....')'
  Possible to execute if enumerate like this.

Status Text String	Contents
SD:STAT F IDLE	Nothing is edited.
SD:STAT F DEFSCH	Sheet Edit
SD:STAT F INSSCH	Instance Hierarchy
SD:STAT F SYMBOL	Symbol Edit

Chapter 2 Restrictions Applicable Between UNIX and DOS File Systems

This appendix explains differences in restrictions between the UNIX file system and DOS FAT file system.

2.1 Using the Windows System Designer to store data files in the DOS file system

This section explains the restrictions on storing symbols and schematic data in the DOS FAT file system using the Windows version of System Designer.

This section also explains the restrictions that must be observed when an FTP or commercially available network software is used to make copies between UNIX file systems.

2.1.1 File name restrictions (Storing data in the DOS file system)

This subsection explains the restrictions that must be observed when storing data files in the DOS file system.

The names of all files which System Designer handles are dependent on the OS file system. For this reason, if System Designer operates on computers whose file systems are different, you must name files with care.

See Table (2.1).

File system	File naming specifications
UNIX file system	A maximum of 256 characters including an extension Uppercase and lowercase letters are differe ntiated (case sensitive). ’.’ is allowed for filenames.
DOS FAT file system	A maximum of 256 characters 3 file extension characters (’.’ not included). All must be uppercase letters.(’.’ is not allowed for filenames.

Table 2.1 Filename restrictions

When creating circuit data in a UNIX file system and copying (not mounting) this circuit data to the DOS file system, name the design data file according to the specifications of the DOS file system, which are more strict.

2.1.2 Restrictions that must be observed when copying design data from a UNIX file system to the DOS file system

When design data created in the UNIX file system is copied[1] to the DOS FAT file system, the file name may be converted into uppercase.

See the table(2.6).

Data name	Example filename before moving	Filename after moving
Schematic directory	sample.cir	SAMPLE.CIR
Schematic sheet data	001.sht	001.SHT
Symbol sheet	abc.smb	ABC.SMB

Table 2.2 Filename after moving from UNIX file system to DOS FAT file system

2.1.3 Restrictions that must be observed when copying design data from the DOS file system to a UNIX file system and how to handle them

When design data is created in the DOS file system and copied to a UNIX file system, file names will be copied without any changes.

1.when an ftp command is used for copying

See Table (2.3).

Data name	Example filename before moving	Filename after moving
Schematic directory	SAMPLE.CIR	SAMPLE.CIR
Schematic sheet data	001.SHT	001.SHT
Symbol sheet	ABC.SMB	ABC.SMB
Data resource file	LANDATA.RSC	LANDATA.RSC
Environment resource file	LANENV.RSC	LANENV.RSC

Table 2.3 Filename after moving from DOS FAT file system to UNIX file system

In this case, the system-reserved filename (the resource file in the example in Table 2.3) and the extension will be converted to all uppercase letters. For this reason, the data cannot be opened in the UNIX file system. To solve this problem, the filename conversion tool needs to be started after moving the file.

This conversion tool converts only system-reserved file names and keeps design data filenames unchanged. For information about files which need to be converted, see Table (2.4).

Data name	before conversion	after conversion	Remarks
Schematic directory	sample.cir	SAMPLE.cir	Named by the designer. Only the extension is converted.
Block file	sample.blk	SAMPLE.blk	Named by the designer. Only the extension is converted.
Schematic sheet data	001.sht	001.sht	System reserved. Only the extension is converted.
Symbol sheet	abc.smb	ABC.smb	Named by the designer. Only the extension is converted.

Data name	before conversion	after conversion	Remarks
Resource pointer file	RCPATH	rcpath	System reserved. The filename is converted.
Data resource file	LANDATA.RSC	landata.rsc	System reserved. The filename is converted.
Environment resource file	LANENV.RSC	lanenv.rsc	System reserved. The filename is converted.
Log storage directory	LOG	log	System reserved.The directory name is converted.
Various tool output directory	EXT	ext	System reserved. The directory name is converted.
Frame data storage directory	FRAME	frame	System reserved. The directory name is converted.
DRC log storage directory	DRC	drc	System reserved. The directory name is converted.

Table 2.4 Files for conversion by the conversion tool

Convert according to the following steps:

Converting design data moved from DOS FAT file system to a UNIX filename

(1) Start Design File Manager.

(2) Select the directory you wish to convert. If you wish to convert only one circuit, select the schematic directory. To convert everything under a particular directory, select that directory.

(3) Select [Tool] -> [Action] -> [Convert to UNIX filename].

(4) All files listed for conversion below the selected directory will be converted.

2.1.4 Renaming Files

When schematics are copied, data resources and environment resources may not be referenced in some cases.

The schematic directory (*.cir) with stored schematics contains a file named "rc-path". In this file, the name of the directories containing the data resource and environment resource to be referenced are described.

If you rename or copy a schematic under a different name, do as follows:

Updating the path for referencing the data resource and environment resource:

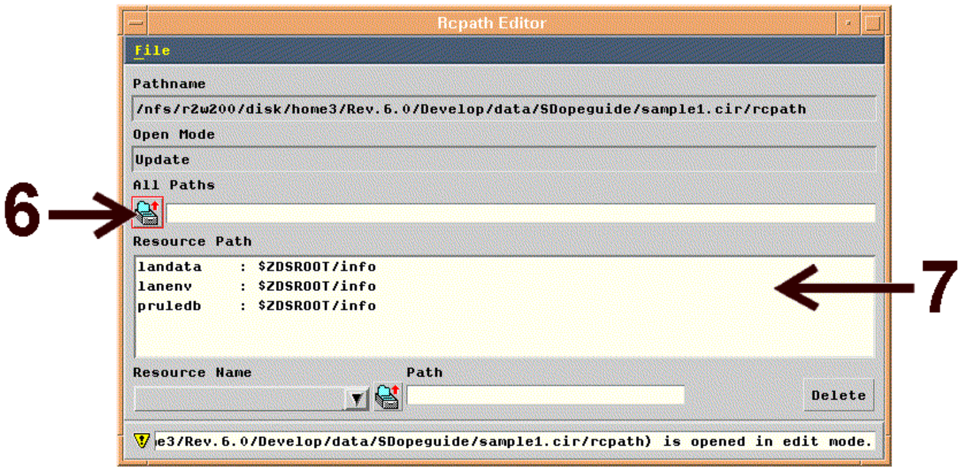
(1) Start Design File Manager.

(2) Go to the schemaic directory (.cir).

(3) Select "rcpath".

(4) Either double-click the filename field with the Left mouse button or select [Tool] -> [Action] -> [Rcpath Editor].

(5) This will start the resource editor for editing "rcpath".



```
/nfs/r2w200/disk/home3/Rev.6.0/Develop/data/SDopeguide/sample1.cir/rcpath
Update
landata    :  $ZDSROOT/info
lanenv     :  $ZDSROOT/info
pruledb    :  $ZDSROOT/info
e3/Rev.6.0/Develop/data/SDopeguide/sample1.cir/rcpath) is opened in edit mode.
```

Figure 2.1 rcpath editor

(6) Designate the name of the directories where the data resource and environment resource are stored.

(7) Be sure that the directory names have been updated to the designated ones.

(8) Designate [File] -> [Save] to save the files.

2.2 Using the Windows version of System Designer to store data files in the UNIX files system and mount them on the NFS

This section explains the restrictions that must be observed when using the Windows version of System Designer to store symbols and schematic data in the UNIX file system and to mount then on the NFS.

2.2.1 File name restriction (Storing data files in a UNIX file system and mounting them on the NFS)

This subsection explains the restrictions on the names of data files to be stored in the UNIX file system and mounted on the NFS by the Windows version of System Designer.

See the table (2.5).

File system	File naming specifications
UNIX file system	Up to 256 characters including the extension. Case-sensitive. ’.’ can be used for a file name.
DOS FAT file system	Up to 256 characters. Three characters for the extension (excluding ’.’). All lowercase letters.’.’ cannot be used for a file name.

Table 2.5 File name restrictions during mounting

Operation varies depending on the network software used.

2.3 Restrictions that must be observed when copying design data from a UNIX file system to the DOS file system

When design data created in the UNIX file system is copied[2] to the DOS FAT file system, the file name may be converted into uppercase.

See the table(2.6).

Data name	Filename example before move	Filename after move
Circuit directory	sample.cir	SAMPLE.CIR
Circuit sheet data	001.sht	001.SHT
Symbol sheet	abc.smb	ABC.SMB

Table 2.6 File names after move from a UNIX file system to the DOS file system

When this move is made during a mount application, lowercase letters that were being used for symbols and circuit data will all be converted to uppercase letters,resulting in mismatches with the symbol file names.

It is better not to mix mount applications with applications in which files arecopied to the DOS file system.

2.mounted and copied using the file manager

Chapter 3 The Conversion Tables Used for EDIF-200 Connectivity ViewOutput

This chapter explains the conversion tables used for EDIF-200 Connectivity Viewoutput.

3.1 Element name/pin name conversion table

This is a user-defined text file.

This file is used for converting System Designer symbol names and pin names to the names in the library of the EDIF-200 Connectivity Viewdestination system.

The specified file will be used only if it possesses ".elf" as its suffix.

Any symbols or pins not described in this table will be used as is.

(1) Syntax

```
partname{
    (symbolName     output cell name)
    (pinLabel       pinLabel            outputpinLabel)
    (pinLabel       pinLabel            outputpinLabel)
    (pinLabel       pinLabel            outputpinLabel)
    }
```

- Output cell name
  Cell name outputted to netlist
- Pin name
  Pin name on schematic sheets of System Designer.
- Output pin name
  Pin name outputted to netlist.

(2) Example
For example, if the entries shown below are made when the pin names are "A0", "B0" and "Y" in the System Designer symbol "74LS00.smb", the cell name of the output result will be "LS00", and the pin names will be "A", "B" and "Y".

```
Example{
   74LS00.smb{
   (symbolName     LS00)
   (pinLabel       A0              A)
   (pinLabel       B0              B)
   (pinLabel       Y               Y)
   }
```

Because creating this file takes some work, a template creation function is provided.

Template creation is registered in the data converter. Select "Create Template for conversion table for EDIF-200-Netlist" from the data converter, set parameters and execute. Template files will be created for the symbols used in the specified circuit.

Since the source and the destination of the conversion will be **shown as being identical, the text editor must be used to convert the symbol a**nd pin names at the destination.

3.2 Property conversion table

This table is used for converting the property names used in System Designer to the names used by the EDIF-200 Connectivity Viewdestination system.

(1) Syntax)

One line of this file corresponds to one property name, and each line consists of 3 fields..

```
properties{
(conversion-result-property-name  property-name property-viewer-number)
}
```

(2) Example)

```
properties {
        (COMP        partName      0 )
        (REFN        reference     1 )
        (VALUE      value          2 )
}
```

Those properties not described in the table will be output using their original property names.

Chapter 4 Format Definition File Creationfor The Netlist Processor

Format definition files ($ZDSROOT/etc/format-name.frm) allow the Netlist Processor to create component tables and net lists in desired formats. This chapter explains how to create a format definition file.

4.1 Reference list output definition

4.1.1 Component table file name specification

Specifies a file of a component table to be created.

```
outputFile : ( $circuitName ".rlf" )
```

In this case, the output will be sent to a file called "circuit-name+.rlf".

output file name specification

Syntax: outputFile : ($circuitName suffix) •$circuitName This is the keyword that specifies a circuit.
•suffix Specifies a suffix to be added to a circuit name, etc.

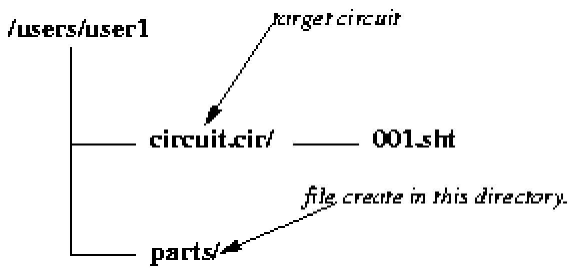
4.1.2 Specifying a directory in which to create a component table

Specifies the directory in which to store a component table.

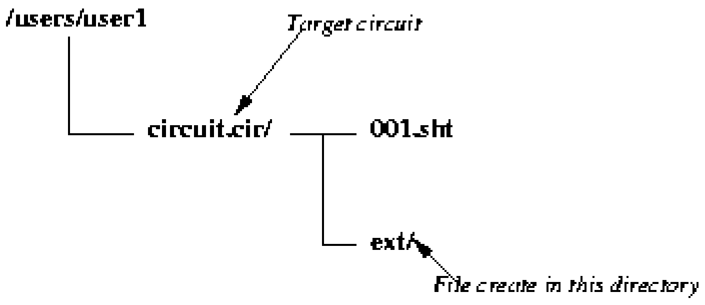
```
outputDir: ( ./parts )
```

In this case, the file will be created in the following directory

.



This specification can be omitted.
If the specification is omitted, the file will be created under "ext" under the circuit directory named (circuit-name.cir)
.



output directory specification

Syntax:
outputDir: (directory-name)

- Directory name

  Specifies the directory in which a component table file is to be created.
  Specify either an absolute path name, or a relative path name from the circuit directory.

4.1.3 Specifying the output of components information

Declares that the components in the schematic should be output in groups based on the reference and pin numbers of each component.

```
netFormat: "parts"
```

With this specification, the information on each component can be output.

output mode specification

```
Syntax:
  netFormat : output-mode
     • Output mode
       parts     :part-net
       gate      :component-net
```

(1) Part net

Specify this item when it is necessary to output information on each component.

Components possessing identical references are treated as a single component.

Pins possessing identical pin numbers are also treated as a single pin.

(2) Component net

Specify this item when information on each component is not to be output.

Components in the circuit will be processed as a single unit for output.

Multiple components and multiple pins will be expanded uniformly at output.

Select this specification if the information on each component is not required, since this specification results in faster processing speed.

4.1.4 Header output specification

Specifies that the circuit name and the author name be output first, as part of the header.

```
outputFormat {
    (text       "Circuit-name:"    1)
    (property   $circuitName      12 0 left " ")
    (endline)
    (text       "Author:"          1)
    (property   author            12 0 left "-")
}
```

In this case, the output will appear as shown below.

```
Circuit name:  sample.cir
Author:    T.Yamada
```

Here, it is possible to specify that properties, arbitrary text, and line feed of the sheet be output.

Specification of arbitrary (user-defined) text output

To output a fixed text, specify as follows.

```
Syntax:
(text "text" output-column)
  • Text
    Text to be output.
    • Output column
    Specifies the number of columns in which the text is to be output.
    If 0 is specified, the text will be output continuously.
    The default value is 0.

    number :  Absolute position specification
    +number: Relative position specification
    Offset from the output column count of the previous data will be used.
```

- Example

```
outputFormat {
    (text "TEXT TEST" 5)
}
```

- Output result

```
. .TEXT . TEST
```

- Example

```
outputFormat {
    (text "TEXT1" 5)
    (text "TEXT2" +10)
}
```

- Output result

```
. . TEXT1 . . . TEXT2
      <-         10          ->
```

Line feed specification

If the specified number of columns are exceeded during output, line feed will automatically occur. To specify the line feed position, specify as follows.

```
Syntax:
  (endline)
```

- Example

```
    outputFormat {
        (text "TEXT1" 5)
        (endline)
        (text "TEXT2" 10)
    }
```

- Output result

```
         TEXT1
              TEXT2
```

Property output specification

Syntax:

(property property-name output-column maximum-byte-count shift-specification default-text)

- Property name

  Specifies the name of the property to be output.

- Output column (optional)

  Specify the start column if it is necessary to align the property output start positions.
  If 0 is specified, properties will be output contiguously.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Maximum byte count

  Specify a maximum byte count if properties should not be output in more than a certain number of bytes.
  When this count is specified, only the specified number of bytes will be output.
  If 0 is specified, there is no restriction on the number of bytes.
  The default value is 0.

- Shift specification (optional)

  Specify this item if it is necessary to right-justify the output of properties.
  This item is valid if the maximum byte count is 1 or larger.

  left :left-justify
  right :right-justify
  The default setting is left-justification.

- Default text (optional)

  Specifies the text to be output when the specified property has not been input.
  In the default mode, nothing will be output.

- Example

```
outputFormat {
    (property author 5 0  left " ")
    (property author 5 5  left " ")
    (property author 5 10 right " ")
}
```

- Output result

```
T.Yamada
  |
 5
T.Yama
<-  5  ->
  T.Yamada
  <-  10   ->
```

4.1.5 Component reference name list output specification

Specifiesthatreferencenamesofcomponentsintheschematicbeoutput.

```
netFormat : "parts"
outputFile : ( $circuitName ".rlf" )
outputDir :  ( ./ )
outputFormat {
    (text        "Circuit name:"   1)
    (property    $circuitName     12 0 left " ")
    (text        "Author:"         1)
    (property    author           12 0 left "-")
    (endline)
# Component output specification
   (parts       $format           1 "no parts" ",")
#   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    (endline)
}
# Component output format specification
$format  {
    (property    reference         0  0 left "-")
}
```

In this case, the output will appear as follows.

```
Circuit name:  sample.cir
Author:        T.Yamada
-,-,C4,R6,R1,C2,Q1,SY1,R2,-,-,-,-,IC1,-,SY2,EX1,-,D1,-,C1,R3,R5, X3,X4,-,-
,-,R4,X1,X2,C3
```

Object output format specification

Specifies the output of an object inside a circuit and its output format.
Outputs items such as the properties of the object according to the specified format definition.
Within this format, it is also possible to specify the output for other objects.

Syntax:
(object-name format-name output-column default-text delimiter-text)

- Object name
  Specifies the name of the object to be output.
- Format name
  Specifies the name of the output format for the object.
  This name must be unique within the file, and must begin with "$".
- Output column (optional)
  Specify the start column if it is necessary to align the object output start positions.
  If 0 is specified, objects will be output contiguously.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.
- Default text (optional)
  Specifies the text to be output when there is no object to be output.
  In the default mode, nothing will be output.
- Delimiter text (optional)
  Specifies the text to be used for delimiting output objects.
  In the default mode, nothing will be output.

Object output format specification

```
Syntax:
  Format-name{
     (if output-target-specification)
           :
     (sort output-sequence-specification)
           :
     (output-target-keyword output-format)
           :
  }
```

- Output target specification
  Specify this item when you want to limit the output of objects to those possessing a certain property.
- Output sequence specification
  Specify this item when you want to output objects in ascending or descending order of properties.
- Output target keyword
  Specifies keywords that specify items such as properties and text of an objects that should be output.

Keyword	Meaning
text	Output of arbitrary text
endline	Output of line feed
property	Output of object property
set	Value setting for a variable
append	Value appending to a variable
variable	Output of a variable
include	Reading of a file
Object name	Output of an object
Object group name	Output of object group

- Output format
  Specifies a format, such as an output column, for output.
  Syntax to be specified will differ depending on the output target.

- Objects to be output and their keywords
  Outputs other objects possessed by the object targeted for output.
  For example, specify the output from the pins of a component or nets that are connected to the pins.

Keyword	Output target
circuit	Circuit
parts	Parts
partsPin	Parts pin
component	Component
componentPin	Component pin
net	Net
netSegment	Net segment
netCons	Net construct point

- Object groups to be output and their keywords
  Outputs other objects possessed by the object targeted for output, after rouping them according to certain properties.
  This type of output will be explained later.

Keyword	Grouping target
partsGroup	Parts
partsPinGroup	Parts pin
componentGroup	Component
componentPinGroup	Component pin
netGroup	Net
netConsGroup	Net construct point

- Example

```
outputFormat {
        (component    $format1        1 " "  " ")
}
$format1 {
        (if componentType == "gate | parts")
        (sort reference up reference)
        (property     reference       1 20 left "-")
        (componentPin $format2       22 " "  " ")
        (endline)
}
$format2 {
        (sort $objectIdNumber up)
        (net         $format3         0  "0" " ")
}
$format3{
        (property     netLabel        0 20 left "?")
}
```

4.1.6 Maximum column count and continuation line output specification

In the output in the previous section, all references are output in one line. Here, line feed is set to occur after 40 columns, and the continued lines are set to begin with ">".

```
# Maximum column count specification
width : 40
#^^^^^^^^^^
 # Continued line specification
continue : ">"
#^^^^^^^^^^^^^
netFormat : "parts"
outputFile : ( $circuitName ".rlf" )
outputDir :  ( ./ )
outputFormat
   (text        " name:"        1)
   (property    $circuitName   12 0 left "")
   (text        "Author:"       1)
   (property    author          12 0 left "-")
   (endline)
   (parts       $format         5 "no parts" ",")
   (endline)
}
$format {
   (property    reference       0  0 left "-")
}
```

In this case, the output will appear as follows.

```
Circuit name:  sample.cir
 Author:    T.Yamada
    -,-,C4,R6,R1,C2,Q1,SY1,R2,-,-,-,-,
>   IC1,-,SY2,EX1,-,D1,-,C1,R3,R5,X3,X4,
>   -,-,-,R4,X1,X2,C3
```

Maximum column specification

Specifies the maximum column count for the output file.
If this column count is exceeded during output, a line feed will occur and output will continue.

If 0 is specified, there will be no column count limit, and no line feed will occur.
The default value is 0.

```
Syntax:
   width : Maximum column count
```

Continued line start character specification

Specifies the character to be used to signify the continuation of output on a new line when the maximum column count is exceeded.
In the default mode, nothing will be output.

```
Syntax:
   continue : Continued line character
```

4.1.7 Limiting the parts to be output

Based on the specifications made so far, the format definition file to be used for outputting a component table will appear as follows.

```
width : 80
continue : ">"
netFormat : "parts"
outputFile : ( $circuitName ".rlf" )
outputDir :  ( ./ )
outputFormat
    (text       "Circuit name:"   1)
    (property    $circuitName     12 0 left " ")
    (text       "Author:"        1)
    (property    author          12 0 left "-")
    (endline)
    (endline)
    (text       "Reference"       1)
    (text       "Part name"      20)
    (endline)
    (parts      $format          1 " "  " ")
    (endline)
}
 $format {
   (property    reference        0  8 left "-")
   (property    partName         20 20 left "-")
   (endline)
}
```

The component table that is output using this definition file will appear as follows.

```
Circuit name:  sample.cir
Author:    T.Yamada

Reference name     Part name
-                  GND
-                  GND
C4                 CE0J100Z
R6                 RD1/8W1H473J
R1                 RD1/8W1H152J
C2                 CKF1H104M
Q1                 2SA429G
SY1                2SC1815GR
R2                 RD1/8W1H103J
-                  GND
-                  GND
-                 GND
-                 GND
IC1               74HC74S
-                 GND
SY2               TP-AL400
 EX1               CN-MOL20510
 -                 GND
 D1                1N4148
 -                 GND
 C1                CE1A470Z
 R3                EVN10DC102B
 R5                RD1/8W1H103J
 X3                TL074D
 X4                TL074D
 -                 GND
 -                 GND
 -                 GND
 R4                RD1/8W1H103J
 X1                TL074D
 X2                TL074D
 C3               CKB1H222K
```

Grounding components are also output in this component table.
The examplebelow limits these to the "parts" and "gate" component types.

```
width : 80
continue : ">"
netFormat : "parts"
outputFile : ( $circuitName ".rlf" )
outputDir :  ( ./ )
outputFormat {
   (text        "Circuit name:"   1)
   (property    $circuitName     12 0 left " ")
   (text        "Author:"         1)
   (property    author           12 0 left "-")
   (endline)
   (endline)
   (text        "Reference"       1)
   (text        "Part name"      20)
   (endline)
   (parts       $format           1 " "  " ")
}
$format {
#  Output condition specification
    (if componentType == "gate | parts")
#   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
   (property    reference        0  8 left "-")
   (property    partName         20 20 left "-")
   (endline)
}
```

With this specification, the output will appear as follows.

```
Circuit name:  sample.cir
Author:    T.Yamada

Reference name    Part name
C4                CE0J100Z
R1                RD1/8W1H152J
R6                RD1/8W1H473J
Q1                2SA429G
C2                CKF1H104M
R2                RD1/8W1H103J
SY1               2SC1815GR
IC1               74HC74S
SY2               TP-AL400
D1                1N4148
R3                EVN10DC102B
C1                CE1A470Z
R5                RD1/8W1H103J
X3                TL074D
X4                TL074D
R4                RD1/8W1H103J
X1                TL074D
X2                TL074D
C3                CKB1H222K
```

Output condition specification

Specifies the condition(s) for object output.

```
Syntax:
  (if property-name comparison-operator comparison-target)
```

- Property name
  Specifies the name of the property to be compared.
- Comparison operators
  "==", "_", ">", "<", ">=", "<="
- Comparison target
  Specify a text or a number.

When this item is specified, only those objects that meet the specified condition will be output.
If this item is specified more than once, only those objects that meet all of the specified conditions will be output.

Conditions that can be specified will differ depending on the data type defined in the property definition file.

(1) Component type

(if NAME == "xxx") : Matches

(if NAME != "xxx") : Does not match

Conditions that can be used as comparison targets

Comparison target	Meaning
fig	Figure
frame	Frame
parts	Parts
gate	Gate
block	Block
vcc	Power supply
gnd	Ground
hieConnector	Hierarchical connector
shtConnector	connector
powerBox	Power Box

When these are connected using "|", the result will be a match (or no match) between an object and the comparison target(s).

(2) "text" type

Comparisons are made for text based on ASCII character code values.

(if NAME == "xxx") : Matches

(if NAME != "xxx") : Does not match

(if NAME <= "xxx") : Equal or smaller

(if NAME >= "xxx") : Equal or greater

(if NAME < "xxx") : Smaller

(if NAME > "xxx") : Greater

"*" can be used for comparison for "==" and "!=".

Comparison target	Meaning
"*"	Property has been input
"^xxx*"	Property begins with "xxx"
".*xxx$"	Property ends with "xxx"
".*xxx.*	Property contains "xxx"

(3) "float" type

(if NAME == xxx)	: Matches
(if NAME != xxx)	: Does not match
(if NAME <= xxx)	: Equal or smaller
(if NAME >= xxx)	: Equal or greater
(if NAME < xxx)	: Smaller
(if NAME > xxx)	: Greater

(4) "int" type

(if NAME == xxx)	: Matches
(if NAME != xxx)	: Does not match
(if NAME <= xxx)	: Equal or smaller
(if NAME >= xxx)	: Equal or greater
(if NAME < xxx)	: Smaller
(if NAME > xxx)	: Greater

4.1.8 Part output sequence specification

Although the sequence of part output is normally unspecified, output can be specified to be sorted in ascending order of reference names.

```
  width : 80
  continue : ">"
  netFormat : "parts"
  outputFile : ( $circuitName ".rlf" )
  outputDir :  ( ./ )
  outputFormat {
     (text       "Circuit name:"   1)
     (property    $circuitName     12 0 left " ")
     (text       "Author:"        1)
     (property    author          12 0 left "-")
     (endline)
     (endline)
     (text       "Reference"       1)
     (text       "Part name"      20)
     (endline)
     (parts      $format          1 " "  " ")
     (endline)
  }
  $format {
     (if componentType == "gate | parts")
  # Output sequence specification
     (sort reference up reference)
  #   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
     (property    reference        0  8 left "-")
     (property    partName        20 20 left "-")
     (endline)
}
```

```
Circuit name:  sample.cir
Author:    T.Yamada

Reference name    Part name
C1                CE1A470Z
C2                CKF1H104M
C3                CKB1H222K
C4                CE0J100Z
D1                1N4148
IC1               74HC74S
Q1                2SA429G
R1                RD1/8W1H152J
R2                RD1/8W1H103J
R3                EVN10DC102B
R4                RD1/8W1H103J
R5                RD1/8W1H103J
R6                RD1/8W1H473J
SY1               2SC1815GR
SY2               TP-AL400
X1                TL074D
X2                TL074D
X3                TL074D
X4                TL074D
```

Output sequence specification

Specifies_an output sequence, sorted by property.

Syntax:
(sort property-name ascending/descending text-comparison-method)

- Property name

  Specifies the name of the property to be sorted.

- Ascending/descending Specifies the sorting order.

  up : ascending

  down : descending

- Text comparison method

  Specifies that text-type properties will be sorted.

  ascii : ASCII sort

  reference : reference sort

  Interprets the number portion at the end of a text as a numerical value.

The following example results would appear as follows.
ascii : R1 R10 R2 R20 R3 R30
reference : R1 R2 R3 R10 R20 R30

4.1.9 Component information library for circuit design referencing specification

When "parts" is specified as the output mode, the information inside an Component information library for circuit design can be output.

- Outputs the properties defined inside the Component information library for circuit design.
- Outputs the parts pins (power pins, grounding pins, etc.) that are not registered for the symbol.

Syntax:
target : Component type

- Component type

  Specifies the component type that refers to Component information library for circuit design .

Component type	Meaning
fig	Figure
frame	Frame
parts	Parts
gate	Gate
block	Block
vcc	Power supply
gnd	Ground
hieConnector	Hierarchical component
shtConnector	Sheet connector
powerBox	Power box

To specify more than one type, connect them using "|".

4.2 Part name list output definition

4.2.1 Part grouping specification

Specifies that a part name list of the parts inside a circuit be output.

```
width : 80
continue : ">"
netFormat : "parts"
outputFile : ( $circuitName ".plf" )
outputDir :  ( ./ )
outputFormat
   (text          "Circuit name:" 1)
   (property    $circuitName   12 0 left " ")
   (text          "Author:"      1)
   (property    author         12 0 left "-")
   (endline)
   (endline)
   (text          "Part name"     1)
   (endline)
# Part group output specification
   (partsGroup  $format1        1 "" "")
#   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
}
# Specification of output format for parts grouping
$format1{
       (if componentType == "gate | parts")
# Parts grouping condition specification
       (group partName)
       (property    partName        1 20 left "-")
       (endline)
}
```

With this specification, the output will appear as follows.

```
Circuit name:  sample.cir
Author:    T.Yamada

Part name
1N4148
2SA429G
2SC1815GR
74HC74S
CE0J100Z
CE1A470Z
CKB1H222K
CKF1H104M
EVN10DC102B
RD1/8W1H103J
RD1/8W1H152J
RD1/8W1H473J
TL074D
TP-AL400
```

Grouping target object specification

Syntax:

(group-name format-name output-column default-text delimiting-text)

- Group name

  Specifies the grouping target.

- Format name

  Specifies the name of the output format to be used for the group.

  This name must be unique within the file, and must begin with $".

- Output column (optional)

  Specify the start column if it is necessary to align the property output start positions.

  If 0 is specified, properties will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no group to be output.

  In the default mode, nothing will be output.

- Delimiting text (optional)

  Specifies the text to be used for delimiting the group output.

  In the default mode, nothing will be output.

Grouping result output format specification

Syntax:
```
        format-name {
              (group grouping-condition-specification)
                        :
              (if output-target-specification)
                       :
              (sort sorting-sequence-specification)
                       :
              (output-target-keyword output-format)
                       :
        }
```

- Grouping condition specification
  Specifies the property to be used as the grouping condition.
- Output target specification
  Specify this item when it is necessary to limit the output_of objects to those possessing a certain property.
- Output sequence specification
  Specify this item when it is necessary to output objects in an ascending or descending order.
- Output target
  Specifies keywords that specify items such as properties and text of an object that should be output.

Keyword	Meaning
text	Output of arbitrary text
endline	Output of line feed
property	Output of object property
set	Value setting for a variable
append	Value appending for a variable
variable	Output of a variable
include	Reading of a file
Object name	Output of an object
Object group name	Output of object group

- Output format
  Specifies a format, such as output column, for an output.
  Syntax to be specified will differ depending on the output target.

- Object groups to be output and their keywords
  Keywords to be specified will differ depending on the grouping target.
  Grouping results can be further grouped according to a different condition, and other objects possessed by the grouped objects can be grouped.

Keyword	Grouping target
partsGroup	Parts
partsPinGroup	Parts pin
componentGroup	Component
componentPinGroup	Component pin
netGroup	Net
netSegmentGroup	Net segment
netConsGroup	Net construct point

- Objects to be output and their keywords
  Specifies that individual objects and other objects possessed by those objects be output from the grouped objects.

Keyword	Output target
circuit	Circuit
parts	Parts
partsPin	Parts pin
component	Component
componentPin	Component pin
net	Net
netSegment	Net segment
netCons	Net construct point

Grouping condition specification

Specifies a property to be used as the grouping condition.
Objects possessing the same property value will be grouped together.

```
Syntax:
   (group property-name)
     • Property name
       Specifies the name of the property to be used as the grouping
       condition.
```

Only one property name can be specified.

4.2.2 Parts count output specification

Specifies that parts be grouped by part name.

```
width : 80
continue : ">"
netFormat : "parts"
outputFile : ( $circuitName ".plf" )
outputDir :  ( ./ )
outputFormat
    (text       "Circuit name:" 1)
    (property   $circuitName   12 0 left " ")
    (text       "Author:"       1)
    (property   author         12 0 left "-")
    (endline)
    (endline)
    (text       "Part name"     1)
    (text       "Parts count"  22)
    (endline)
    (partsGroup  $format1       1 "" "")
}
$format1 {
    (if componentType == "gate | parts")
    (group partName)
    (property   partName       1 20 left "-")
# Parts count output specification
    (count      count         22)
#   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    (endline)
 }
```

With this specification, the output will appear as follows.

```
Circuit name:  sample.cir
Author:    T.Yamada

Part name        Parts count
1N4148                1
2SA429G               1
2SC1815GR             1
74HC74S               1
CE0J100Z              1
CE1A470Z              1
CKB1H222K             1
CKF1H104M             1
EVN10DC102B           1
RD1/8W1H103J          3
RD1/8W1H152J          1
RD1/8W1H473J          1
TL074D                4
TP-AL400              1
```

Object count grouping specification

Specifies that the objects inside the group be counted and output.

Syntax:
(count count output-column)

- Output column (optional)

  Specifies the column count in which to output the grouping result.

  If 0 is specified, the result will be output continuously.

  The default value is 0.

  number : absolute position specification

  +number: relative position specification

  Offset from the output column count of the previous data will be used.

4.2.3 Property grouping result specification

In the example below, prices are grouped.

```
width : 80
continue : ">"
netFormat : "parts"
outputFile : ( $circuitName ".plf" )
outputDir :  ( ./ )
outputFormat
  (text       "Circuit name:" 1)
  (property   $circuitName   12 0 left " ")
  (text       "Author:"      1)
  (property   author         12 0 left "-")
  (endline)
  (endline)
  (text       "Part name"     1)
  (text       "Part count"   22)
  (text       "Unit price"   32)
  (text       "Subtotal"     42)
  (endline)
  (partsGroup  $format1       1 "" "")
#   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
}
$format1 {
   (if componentType == "gate | parts")
   (group partName)
   (property   partName       1 20 left "-")
   (count      count          22)
   (property   price          32 20 left "-")
#   Property grouping specification
     (sum        price          42)
#   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
   (endline)
}
```

With this specification, the output will appear as follows.

```
Circuit name:  sample.cir
Author:    T.Yamada
```

Part name	Part count	Unit price	Subtotal
1N4148	1	5	5
2SA429G	1	3	3
2SC1815GR	1	3	3
74HC74S	1	3	3
CE0J100Z	1	3	3
CE1A470Z	1	3	3
CKB1H222K	1	3	3
CKF1H104M	1	3	3
EVN10DC102B	1	3	3
RD1/8W1H103J	3	3	9
RD1/8W1H152J	1	2	2
RD1/8W1H473J	1	1	1
TL074D	4	5	20
TP-AL400	1	10	10

Property grouping specification

Outputs the result of adding the properties of the objects inside the group.

Syntax:
(sum property-name output-column)

- Property name

  Specifies the name of the property to be grouped. Specify either a "float" or "int" type property.

- Output column (optional)

  Specifies the column count in which to output the grouping result.
  If 0 is specified, the result will be output continuously to the pre-vious output.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

4.2.4 Property total output specification

Specifies that the total of the properties of all parts be output.

```
 width : 80
 continue : ">"
 netFormat : "parts"
 outputFile : ( $circuitName ".plf" )
 outputDir :  ( ./ )
 outputFormat
    (text       "Circuit name:" 1)
    (property   $circuitName   12 0 left " ")
    (text       "Author:"      1)
    (property   author         12 0 left "-")
    (endline)
    (endline)
    (text       "Part name"     1)
    (text       "Parts count"  22)
    (text       "Unit price"   32)
    (text       "Subtotal"     42)
    (endline)
    (partsGroup  $format0       1 "" "")
}
$format0{
    (if componentType == "gate | parts")
    (partsGroup  $format1       1 "" "")
    (text "-----------------------------------------" 1)
    (endline)
    (text       "Total"        1)
# Total output specification
    (count      count         22)
#   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    (sum        price         42)
#   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    (endline)
}
$format1{
    (group partName)
    (property   partName       1 20 left "-")
    (count      count          22)
    (property   price          32 20 left "-")
    (sum        price          42)
    (endline)
 }
```

If no group condition is specified, all parts will be grouped into one group. Thiscan be used to output subtotals and the total.

```
Circuit name:  sample.cir
Author:    T.Yamada

Part name                     Parts count    Unit price    Subtotal
1N4148                          1              5             5
2SA429G                         1              3             3
2SC1815GR                       1              3             3
74HC74S                         1              3             3
CE0J100Z                        1              3             3
CE1A470Z                        1              3             3
CKB1H222K                       1              3             3
CKF1H104M                       1              3             3
EVN10DC102B                     1              3             3
RD1/8W1H103J                    3              3             9
RD1/8W1H152J                    1              2             2
RD1/8W1H473J                    1              1             1
TL074D                          4              5            20
TP-AL400                        1             10            10

----------------------------------------------------------------------------------------------------

Total                          19             50            71
```

4.2.5 Grouped parts output

Specifies that a reference be output for the grouped parts.

```
  width : 80
  continue : ">"
  netFormat : "parts"
  outputFile : ( $circuitName ".plf" )
  outputDir :  ( ./ )
  outputFormat
      (text       "Circuit name:" 1)
      (property    $circuitName   12 0 left " ")
      (text       "Author:"      1)
      (property    author         12 0 left "-")
      (endline)
      (endline)
```

```
    (text       "Part name"     1)
    (text       "Parts count"  22)
    (text       "Unit price"   30)
    (text       "Subtotal"     38)
    (text       "Reference"    44)
    (endline)
    (partsGroup  $format0       1 "" "")
}
$format0 {
    (if componentType == "gate | parts")
    (partsGroup  $format1       1 "" "")
    (text "----------------------------------------------------" 1)
    (endline)
    (text       "Total"         1)
    (count      count          22)
    (sum        price          42)
    (endline)
}
$format1{
    (group partName)
    (property    partName       1 20 left "-")
    (count      count          22)
    (property    price         30 20 left "-")
    (sum        price          38)
#   Output specification of parts inside the group
    (parts      $format2       44 "" "")
#   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    (endline)

 }
 $format2 {
    (sort reference up reference)
    (property    reference       0  0 left "-")
 }
```

Output can be specified for the grouped objects.

```
Circuit name:  sample.cir
Author:    T.Yamada

Part name         Parts count   Unit price   Subtotal   Reference
1N4148            1             5            5          D1
2SA429G           1             3            3          Q1
2SC1815GR         1             3            3          SY1
74HC74S           1             3            3          IC1
CE0J100Z          1             3            3          C4
CE1A470Z          1             3            3          C1
CKB1H222K         1             3            3          C3
CKF1H104M         1             3            3          C2
EVN10DC102B       1             3            3          R3
RD1/8W1H103J      3             3            9          R2,R4,R5
RD1/8W1H152J      1             2            2          R1
RD1/8W1H473J      1             1            1          R6
TL074D            4             5            20         X1,X2,X3,X4
TP-AL400          1             10           10         SY2

------------------------------------------------------------------------------------------------------

Total             19            50           71
```

Appendix A Format Definition File Reference for The Netlist Processor

The format definition file is a text file that defines the output format for the Netlist Processor.

A.1 Comment specification

Comment specification

Entries between "#" and the end of the line are treated as* * comments.
This line is ignored by the system.

```
Syntax:
# comment
```

Special character specification

The following characters possess special meanings inside the format definition file.

```
{ } \ : " # %
```

When using these characters, they must be specified before a back slash ("\").

A.2 Maximum column count specification

Maximum column count specification

Specifies the maximum column count for the output file.
If this column count is exceeded during output, line feed will occur and the output will continue.
If 0 is specified, there will be no column count limit, and no line feed will occur.
The default value is 0.

```
Syntax:
width : maximum-column-count
```

Continued line start text specification

Specifies the text to be used for the start of a new line when the maximum column count is exceeded.
In the default mode, nothing will be output.

```
Syntax:
continue : Continued line character
```

A.3 Output file name specification

Output directory specification

Specifies the directory in which the output file is to be created.
Either an absolute or relative path name can be specified for the directory.

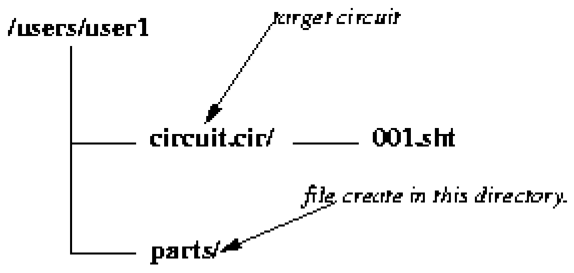
```
Syntax:
outputDir : ( directory-name )
```

If a relative path name is specified, it will be a relative path from the directory in which the circuit to be output resides.

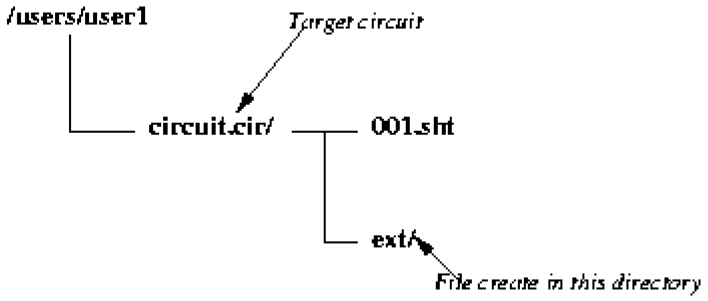
Example

```
outputDir: ( "./parts" )
```

In this case, the file will be created in the following directory.



This specification is optional. If it is omitted, the file will be created under "ext" under the circuit directory (circuit-name.cir).



Specification of file name to be created

Specifies a file name to be created.

(1) When outputting to a fixed file
To always create the same file regardless of the circuit to be output, use the specification shown below.

```
Syntax:
outputFile : ( file-name )
```

(2) When outputting to a file having the circuit name with a ceratin suffix

```
Syntax:
outputFile : ( $circuitName suffix )
```

Example

```
outputFile : ( $circuitName ".prt" )
```

If the circuit to be output is "example.cir", a file named "example.prt" will be created.
Note that the following keywords will be converted to certain strings.

Keyword name	Meaning
$circuitName	circuit name.

(3) When using a circuit sheet property as the file name

The properties of circuit sheets can be used as file names.

```
Syntax:
  outputFile {
    (property property-name default-text suffix-operation-specification)
    (text     Text)
  }
```

- Property name

  Specifies the name of a property possessed by the circuit sheet.
- Default text

  Specifies the text to be used when the specified property has not been input.
- Suffix operation specification

  Specifies a suffix operation when a property is used as the file name.

  cutSuffix: Deletes the suffix of the property value.

  No specification: Uses the property value as is.
- Text

  Specifies an arbitrary text.

The syntaxes are sequentially interpreted from the top, and the text resulting from the sum of these are used as the file name.

Example

```
  outputFile {
    (property partName  "noname" cutSuffix )
    (text ""_")
    (property function  "noname" cutSuffix )
    (text ".prt")
  }
```

- Part name (partName) : "PARTNAME"
- Function (function) has : Not been input.

In this case, the file name will be "PARTNAME_noname.prt".

A.4 Output mode specification

Output mode specification

```
Syntax:
  netFormat : Output mode
   • Output mode
      parts      :Parts net
      gate       :Component net
```

(1) Parts net
Specify this item to output the information for each component.
Components possessing identical references are treated as one component.
Pins possessing identical pin numbers are also treated as one pin.

(2) Component net
Specify this item when the information on each component is not to be output.
Components in the circuit will be treated as a single unit in the output.
Multiple components and multiple pins will be expanded into single items before being output.
Select this specification if the information on each component is not required, since this specification results in faster processing speed.
Example

```
    netFormat : "gate"
```

Output suppression of hierarchical connector

When activating a net output, you can specify whether the hierarchical connector and sheet connector are to be ignored during the operation.
For normal net output, you operate the system with this mode set to "no", since the hierarchical connector and sheet connector are not required.

Syntax
  conOut : Flag

- yes
  The sheet connector and hierarchical connector are included in the object of the output.
- no
  The sheet connector and hierarchical connector are excluded from the object of the output.

Output setting for SI units

When outputting the property of float type, specify if you are going to use SI units such as T, G, or M. The default is to use SI units
.

Syntax
  useSI: Flag

- yes
  Use SI units such as T, G, or M when outputting the property of float type.
- no
  Do not use SI units.

Component information library for circuit design reference mode specification

If "parts" is specified as the output mode, the information inside Component information library for circuit design can be output

Syntax:
target : component-type

- Component type

  Specifies a component type that references Component information library for circuit design.

Component type	Meaning
fig	Figure
frame	Frame
parts	Parts
gate	Gate
block	Block
vcc	Power supply
gnd	Ground
hieConnector	Hierarchical connector
shtConnector	Sheet connector
powerBox	Power Box

To specify more than one type, connect them using " | ".

Use this specification in the following cases.

- To output the properties defined in Component information library for circuit design
- To output the parts pins (power pins, grounding pins, etc.) that are not registered in the symbol.

  Example

```
target : "parts | gate"
```

A.5 Output format specification

Specifies the format for outputting the objects and properties inside the circuit.

Syntax:
```
outputFormat
   (Output-target output-format)
          :
          :
}
```

- Output target

  Specifies the keywords that specify the properties of a circuit or arbitrary texts that are to be output.

Keyword	Meaning
text	Output of arbitrary text
endline	Output of line feed
property	Output of object property
set	Value setting for a variable
append	Value appending for a variable
variable	Output of a variable
include	Reading of a file
Object name	Output of an object
Object group name	Output of object group

- Output format

  Specifies a format, such as output column, for an output.
  Syntax to be specified will differ depending on the output target.

Example

```
outputFormat {
       # Outputting of fixed text
            (text     "Author:"  1)
       #Outputting of circuit property
            (property author    12 0 left "-")
       # Outputting of line feed
            (endline)
            (text     "Reference" 1)
            (endline)
       # Outputting of parts inside the circuit
            (parts    $format    1 "" "")
            (endline)
}
# Parts output format specification
$format {
        # Outputting of parts property
            (property reference  0  8 left "-")
            (endline)
}
```

A.5.1 Object output specification

Specifies in outputFormat the output format of an object inside a circuit.
Objects inside the circuit will be output according to the definition of the specified format name.
Within the specification of the output format, it is also possible to specify outputting of other objects.

(1) Object output specification

Syntax:
(object-name format-name output-column default-text delimiter-text)

- Object name
  Specifies the name of the object to be output.
- Format name
  Specifies the name of the output format to be used for an object.
  This name must be unique within the file, and must begin with "$".
- Output column (optional)
  Specify the start column if it is necessary to align the object information output start positions.
  If 0 is specified, the object information will be output continuously to the previous output.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.
- Default text (optional)
  Specifies the text to be output when there is no object to be output.
  In the default mode, nothing will be output.
- Delimiter text (optional)
  Specifies the text to be used as the delimiter for outputting objects.
  In the default mode, nothing will be output.

(2) Object output format specification

```
Syntax:
   Format-name{
        (if output-target-specification)
              :
        (sort output-sequence-specification)
              :
        (output-target-keyword output-format)
              :
   }
```

- Output target specification
  Specify this item to limit objects to be output to those possessing a certain property.
- Output sequence specification
  Specify this item when it is necessary to output objects in an ascending or descending order of properties.
- Output target keyword
  Specifies keywords that specify items such as properties and text of an object that should be output.

Keyword	Meaning
text	Output of arbitrary text
endline	Output of line feed
property	Output of object property
set	Value setting for a variable
append	Value appending for a variable
variable	Output of a variable
include	Reading of a file
Object name	Output of an object
Object group name	Output of object group

- Output format
  Specifies a format, such as output column, for an output.
  Syntax to be specified will differ depending on the output target.

Example

```
 outputFormat {
          (component    $format1        1 "" "")
 }
 $format1 {
          (if componentType == "gate | parts")
          (sort reference up reference)
          (property     reference       1 20 left "-")
          (componentPin $format2       22 "" "")
          (endline)
}
 $format2 {
          (sort $objectIdNumber up)
          (net         $format3         0  "0" " ")
}
 $format3 {
          (property     netLabel        0 20 left "?")
```

Circuit output specification

Within outputFormat, it is possible to output the properties of a circuit. However, to output a circuit while other objects are being output, the following specification must be used.

(1) Circuit output specification

Syntax:

```
(circuit format-name output-column default-text delimiter-text)
```

- Format name

  Specifies the name of the output format for the circuit.
  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the circuit information output start positions.
  If 0 is specified, the circuit information will be output continuously to the previous output.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no circuit to be output.
  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the circuit output.
  In the default mode, nothing will be output.

(2) Circuit system property

Circuits possess the following system properties, in addition to the user-defined properties.

Property name	property type	meaning
$circuitName	text	circuit name
$circuitName.base	text	circuit name without suffix
$updateDate	text	circuit update time (Tue Feb 21 13:00:36 1995)

Property name	property type	meaning
$updateDate.sec	text	seconds of circuit update time (00-59)
$updateDate.min	text	minutes of circuit update time (00-59)
$updateDate.hour	text	hours of circuit update time (00-23)
$updateDate.day	text	circuit update date (01-31)
$updateDate.month	text	circuit update month (01-12)
$updateDate.year	text	circuit update year (00-99)
$updateTime.sec	int	seconds of circuit update time (0-59)
$updateTime.min	int	minutes of circuit update time (0-59)
$updateTime.hour	int	hours of circuit update time (0-23)
$updateTime.day	int	circuit update date (1-31)
$updateTime.month	int	circuit update month (1-12)
$updateTime.year	int	circuit update year (1900-9999)
$date	text	current time (Tue Feb 21 13:00:36 1995)
$date.sec	text	seconds of current time (00-59)
$date.min	text	minutes of current time (00-59)
$date.hour	text	hours of current time (00-23)
$date.day	text	current date (01-31)
$date.month	text	current month (01-12)
$date.year	text	current year (00-99)
$time.sec	int	seconds of current time (0-59)
$time.min	int	minutes of current time (0-59)
$time.hour	int	hours of current time (0-23)
$time.day	int	current date (1-31)
$time.month	int	current month (1-12)

Property name	property type	meaning
$time.year	int	current year (1900-9999)
$destination	text	destination name
$destination.keyword	text	destination keyword

(3) Objects that can be output from a circuit

Keyword	Output target
parts	All parts inside circuit
partsPin	All parts pins inside circuit
component	All components inside circuit
componentPin	All component pins inside circuit
net	All nets inside circuit
netSegment	All net segments inside circuit
netCons	All net construct points inside circuit

(4) Objects that can be grouped from a circuit

Keyword	Grouping target
partsGroup	All parts inside circuit
partsPinGroup	All parts pins inside circuit
componentGroup	All components inside circuit
componentPinGroup	All component pins inside circuit
netGroup	All nets inside circuit
netSegmentGroup	All net segments inside circuit
netConsGroup	All net construct points inside circuit

Parts output specification

Specify parts output in outputFormat or inside the output format of another object, as follows.
Output is repeated in the specified format for each part.

(1) Parts output specification

Syntax:
(part format-name output-column default-text delimiter-text)

- Format name
  Specifies the name of the output format for the part.
  This name must be unique within the file, and must begin with "$".
- Output column (optional)
  Specify the start column if it is necessary to align the parts information output start positions.
  If 0 is specified, parts information will be output continuously to the previous output.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.
- Default text (optional)
  Specifies the text to be output when there is no part to be output.
  In the default mode, nothing will be output.
- Delimiter text (optional)
  Specifies the text to be used for delimiting the part output.
  In the default mode, nothing will be output.

(2) Parts system property

Parts possess the following system properties, in addition to the userdefined properties.

Property name	Property type	Meaning
componentKind	int	Component type
componentType	Special type	Component type
symbName	text	Symbol name
symbPath	int	Symbol path number
$compPinCount	int	Component pin count
$partsPinCount	int	Parts pin count

(3) Objects that can be output from parts

Keyword	Output target
circuit	Circuit
partsPin	Parts pin
component	Component
componentPin	Component pin

(4) Objects that can be grouped from parts

Keyword	Grouping target
partsPinGroup	Parts pin
componentGroup	Components
componentPinGroup	Component pin

Parts pin output specification

Specify parts pin output in outputFormat or inside the output format of another object, as follows.
Output is repeated in the specified format for each parts pin.

(1) Parts pin output specification

Syntax:

```
(partsPin format-name output-column default-text delimiter-text)
```

- Format name

  Specifies the name of the output format for the parts pin.
  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the parts pin information output start positions.
  If 0 is specified, parts pin information will be output continuously to the previous output.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no parts pin to be output.
  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the parts pin out put.
  In the default mode, nothing will be output.

(2) Parts pin system property

Parts pins possess the following system properties, in addition to the user-defined properties.

Property name	Property type	Meaning
$isConnected	text	Connected? (TRUE/FALSE)
$compPinCount	int	Component pin count

(3) Objects that can be output from parts pins_

Keyword	Output target
circuit	Circuit
parts	Parts
component	Component that possesses the Component pin
componentPin	Component pin

Keyword	Output target
net	Net connection
netSegment	All net segments
netCons	All net construct points

(4) Objects that can be grouped from parts pins

Keyword	Grouping target
partsGroup	Parts
componentGroup	Component that possesses Component pins
componentPinGroup	Component pin
netGroup	Net connection
netSegmentGroup	All net segments
netConsGroup	All net construct points

Component output specification

Specify component output in outputFormat or inside the output format of another object, as follows.
Output is repeated in the specified format for each component.
Multiple components will be expanded to single components before being output.

(1) Component output specification

Syntax:
(component format-name output-column default-text delimiter text)

- Format name
  Specifies the name of the output format for the component.
  This name must be unique within the file, and must begin with "$".
- Output column (optional)
  Specify the start column if it is necessary to align the com ponent information output start positions.
  If 0 is specified, component information will be output continuously to the previous output.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.
- Default text (optional)
  Specifies the text to be output when there is no component to be output.
  In the default mode, nothing will be output.
- Delimiter text (optional)
  Specifies the text to be used for delimiting the component output.
  In the default mode, nothing will be output.

(2) Component system property

Components possess the following system properties, in addition to the user-defined properties.

Property name	Property type	Meaning
objectId	text	Object ID
$objectIdNumber	int	Object ID number
$sheetIdNumber	int	Sheet ID number
componentKind	int	Function types
componentType	Special type	Component type
symbName	text	Symbol name
symbPath	int	Symbol path number
$symbPathName	text	Symbol path name
$circuitPathName	text	Internal circuit path number name

Property name	Property type	Meaning
$compPinCount	int	Component pin count

(3) Objects that can be output from components

Keyword	Output target
circuit	Circuit
parts	Parts
componentPin	Component pin

(4) Objects that can grouped from components

Keyword	Grouping target
partsGroup	Parts
componentPinGroup	Component pin

Component pin output specification

Specify component pin output in outputFormat or inside the output format of another object, as follows.
Output is repeated in the specified format for each component pin.
Multiple pins will be expanded to single pins before being output.

(1) Component pin output specification

Syntax:
(componentPin format-name output-column default-text delimiter-text)

- Format name
  Specifies the name of the output format for the component pin.
  This name must be unique within the file, and must begin with "$".
- Output column (optional)
  Specify the start column if it is necessary to align the component pin information output start positions.
  If 0 is specified, component pin information will be output continuously to the previous output.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.
- Default text (optional)
  Specifies the text to be output when there is no component pin to be output.
  In the default mode, nothing will be output.
- Delimiter text (optional)
  Specifies the text to be used for delimiting the component pin output.
  In the default mode, nothing will be output.

(2) Component pin system property
Component pins possess the following system properties, in addition to the user-defined properties.

Property name	Property type	Meaning
objectId	text	Object ID
$objectIdNumber	int	Object ID number
$sheetIdNumber	int	Sheet ID number
$isConnected	text	Connected? (TRUE/FALSE)

(3) Objects that can be output from component pins

Keyword	Output target
circuit	Circuit

Keyword	Output target
parts	Parts
partsPin	Parts pin
component	Component
net	Net connection
netSegment	All net segments
netCons	All net construct points

(4) Objects that can be grouped from component pins

Keyword	Grouping target
partsGroup	Parts
partsPinGroup	Parts pin
componentGroup	Component
netGroup	Net connection
netSegmentGroup	All net segments
netConsGroup	All net construct points

Net output specification

Specify net output in outputFormat or inside the output format of another object, as follows.
Output is repeated in the specified format for each net.
Of the results of connection state interpretation, those possessing the same potential are output as one net.
All paths are separated into single nets before being output.

(1) Net output specification

Syntax:
(net format-name output-column default-text delimiter-text)

- Format name
  Specifies the name of the output format for the net.
  This name must be unique within the file, and must begin with "$".
- Output column (optional)
  Specify the start column if it is necessary to align the net information output start positions.
  If 0 is specified, net information will be output continuously to the previous output.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.
- Default text (optional)
  Specifies the text to be output when there is no net.
  In the default mode, nothing will be output.
- Delimiter text (optional)
  Specifies the text to be used for delimiting the net output.
  In the default mode, nothing will be output.

(2) Net system property

Nets possess the following system properties, in addition to the user-defined properties.

Property name	Property type	Meaning
$netType	text	Net type (ground/power)

(3) Objects that can be output from nets

Keyword	Output target
circuit	Circuit
partsPin	Parts pin to be connected
componentPin	Component pin to be connected
netSegment	Net segment owned
netCons	Net construct points owned

(4) Objects that can be grouped from nets

Keyword	Grouping target
partsPinGroup	Parts pin to be connected
componentPinGroup	Component pin to be connected
netSegmentGroup	Net segment owned
netConsGroup	Net construct points owned

Net segment output specification

Specify net segment output in outputFormat or inside the output format of another object, as follows.
Output is repeated in the specified format for each net segment.
All path segments are separated into single net segments before being output.

(1) Net segment output specification

Syntax:

```
(netSegment format-name output-column default-text delimiter text)
```

- Format name

  Specifies the name of the output format for the net segment.

  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the net segment information output start positions.

  If 0 is specified, net segment information will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no net segment.

  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the net segment output.

  In the default mode, nothing will be output.

(2) Net segment system property

Net segments possess the following system properties, in addition to the user-defined properties.

Property name	Property type	Meaning
$netObjectId	text	Net object ID
$netObjectIdNumber	int	Net object ID number
$objectIdNumber	int	Object ID number
$sheetIdNumber	int	Sheet ID number of the sheet to which the net segment belongs
$netType	text	Net type (ground/power)

(3) Objects that can be output from net segments

Keyword	Output target
circuit	Circuit
partsPin	Parts pin connected to the net to which the net segment belongs
componentPin	Component pin connected to the net to which the net segment belongs
net	Net to which the net segment belongs

(4) Objects that can be grouped from net segments

Keyword	Grouping target
partsPinGroup	Parts pin connected to the net to which the net segment belongs
componentPinGroup	Component pin connected to the net to which the net segment belongs
netGroup	Net to which the net segment belongs

Net construct point output specification

Specify net construct point output in outputFormat or inside the output format of another object, as follows.
Output is repeated in the specified format for each net construct point.
All path segments are separated into single net construct points before being output.

(1) Net construct point output specification

Syntax:
(netCons format-name output-column default-text delimiter-text)

- Format name
  Specifies the name of the output format for the net construct point.
  This name must be unique within the file, and must begin with "$".
- Output column (optional)
  Specify the start column if it is necessary to align the net construct point information output start positions.
  If 0 is specified, net construct point information will be output continuously to the previous output.}
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.
- Default text (optional)
  Specifies the text to be output when there is no net construct point.
  In the default mode, nothing will be output.
- Delimiter text (optional) Specifies the text to beused for delimiting the net construct point output.
  In the default mode, nothing will be output.

(2) Net construct point system property
Net construct points possess the following system properties, in addition to the user-defined properties.

Property name	Property type	Meaning
$netObjectId	text	Net object ID
$netObjectIdNumber	int	Net object ID number
$objectIdNumber	int	Object ID number
$sheetIdNumber	int	Sheet ID number of the sheet to which the net corner belongs
$netType	int	Net type (ground/power)

(3) Objects that can be output from net construct points

Keyword	Output target
circuit	Circuit
partsPin	Parts pin connected to the net to which the net construct point belongs
componentPin	Component pin connected to the net to which the net construct point belongs
net	Net to which the net construct point belongs

(4) Objects that can be grouped from net construct points

Keyword	Grouping target
partsPinGroup	Parts pin connected to the net to which the net construct point belongs
componentPinGroup	Component pin connected to the net to which the net construct point belongs
netSegmentGGroup	Net to which the net construct segment belongs
netGroup	Net to which the net construct point belongs

A.5.2 Object grouping output specification

Groups the specified objects and outputs them in the specified format.
Specify in outputFormat or inside the output format of each object.

(1) Object grouping output specification

Syntax:

```
(group-name format-name output-column default-text delimiter-text)
```

- Group name

  Specifies the grouping target.

- Format name

  Specifies the name of the output format for the group.

  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the group information output start positions.

  If 0 is specified, group information will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no group.

  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the group output.

  In the default mode, nothing will be output.

(2) Object grouping output format specification

```
Syntax:
  Format-name{
     (group grouping-condition-specification)
            :
     (if output-target-specification)
            :
     (sort output-sequence-specification)
            :
     (output-target-keyword output-format)
            :
  }
```

- Grouping condition specification
  Specifies the property to be used as the grouping condition.
- Output target specification
  Specify this item to limit objects to be output to those possessing a certain property.
- Output sequence specification
  Specify this item when it is necessary to output objects in an ascending or descending order of properties.
- Output target
  Specifies keywords that specify items such as properties and text of an object that should be output.

Keyword	Meaning
text	Output of arbitrary text
endline	Output of line feed
property	Output of object property
set	Value setting for a variable
append	Value appending for a variable
variable	Output of a variable
include	Reading of a file
Object name	Output of an object
Object group name	Output of object group

- Output format
  Specifies a format, such as output column, for an output.
  Syntax to be specified will differ depending on the output target.

(3) Example

```
outputFormat {
   (text       "Part name"     1)
   (text       "Reference"    22)
   (text       "Parts count"  42)
   (endline)
   (partsGroup  $format1       1 "" "")
   (endline)
}
$format1{
    (if componentType == "gate | parts")
    (partsGroup  $format2       1 "" "")
    (endline)
    (text       "total"        1)
    (count      count          42)
    (endline)
}
$format2{
    (group partName)
    (property    partName       1 20 left "-")
    (parts       $format3       22 " " ",")
    (count       count          42)
    (endline)
}
$format3{
    (sort reference up reference)
    (property    reference       0  8 left "-")
}
```

Parts grouping output specification

Syntax:
(partsGroup format-name output-column default-text delimiter-text)

- Format name

  Specifies the name of the output format for the grouping result.

  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the output start positions.

  If 0 is specified, the grouping result will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no group.

  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the group output.

  In the default mode, nothing will be output.

Parts pin grouping output specification

Syntax:
(partsPinGroup format-name output-column default-text delimiter-text)

- Format name

  Specifies the name of the output format for the grouping result.

  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the output start positions.

  If 0 is specified, the grouping result will be output continuously to the previous output.

  The default value is 0.

number :	absolute position specification
+number:	relative position specification Offset from the output column count of the previous data will be used.

- Default text(optional)

  Specifies the text to be output when there is no group.

  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the group output.

  In the default mode, nothing will be output.

Component grouping output specification

Syntax:
(componentGroup format-name output-column default-text delimiter-text)

- Format name

  Specifies the name of the output format for the grouping result.

  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the output start positions.

  If 0 is specified, the grouping result will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no group.

  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the group output.

  In the default mode, nothing will be output.

Component pin grouping output specification

Syntax:
(componentPinGroup   format-name   output-column   default-text delimiter-text)

- Format name

  Specifies the name of the output format for the grouping result.

  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the output start positions.

  If 0 is specified, the grouping result will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no group.

  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the group output.In the default mode, nothing will be output.

Net grouping output specification

Syntax:
(netGroup format-name output-column default-text delimiter-text)

- Format name

  Specifies the name of the output format for the grouping result.

  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the output start positions.

  If 0 is specified, the grouping result will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no group.

  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the group output.

  In the default mode, nothing will be output.

Net segment grouping output specification

Syntax:
(netSegmentGroup format-name output-column default-text delimiter-text)

- Format name

  Specifies the name of the output format for the grouping result.

  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the output start positions.

  If 0 is specified, the grouping result will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no group.

  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the group output.

  In the default mode, nothing will be output.

Net construct point grouping output specification

Syntax:
(netConsGroup format-name output-column default-text delimiter-text)

- Format name

  Specifies the name of the output format for the grouping result.

  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the output start positions.

  If 0 is specified, the grouping result will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no group.

  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the group output.

  In the default mode, nothing will be output.

A.5.3 Text output specification

Arbitrary text output specification

Syntax:
(text text output-column)

- Text

  Text to be output.

- Output column

  Specify the column count at which to start text output.

  If 0 is specified, the text will be output continuouslyto the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

Line feed specification

During output, line feed will automatically occur when the specified number of columns have been exceeded. To cause line feed at a desired position, use the following specification.

Syntax:
(endline)

A.5.4 Property output specification

Property output specification

Syntax:

(property property-name output-column maximum-byte-count shift-specification default-text)

- Property name

  Specifies the name of the property to be output.

- Output column (optional)

  Specify the start column if it is necessary to align the property output start positions.

  If 0 is specified, properties will be output continuously to the previous output.

  The default value is 0.

number :	absolute position specification
+number:	relative position specification Offset from the output column count of the previous data will be used.

- Maximum byte count (optional)

  To limit the output of the property to a certain byte count, specify a maximum byte count.

  Only the specified byte count will be output.

  If 0 is specified, there will be no limit.

  The default value is 0.

- Shift specification (optional)

  Specify this item when it is necessary to right-justify the property output.

  This item is valid only when the maximum byte count is 1 or greater.

left	:left-justify
right	:right-justify

  The default mode is left justification.

- Default text (optional)

  Specifies the text to be output when the specified property has not been input.

  In the default mode, nothing will be output.

Specification of property output with format specification

When it is necessary to output properties with property names and arbitrary texts added, specify the output after separately defining a format.

(1) Property output format specification

Syntax:

(property property-name format-name output-column default-text)

- Property name

  Specifies the name of the property to be output.

- Format name

  Specifies the name of the output format for the property.
  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the property information output start positions.
  If 0 is specified, the property information will be output continuously to the previous output.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when no property has been input.
  In the default mode, nothing will be output.

(2) General property output format specification

Syntax:

```
Format-name{
    (output-target-keyword output-format)
    (output-target-keyword output-format)
                   :
                   :
    (output-target-keyword output-format)
}
```

- Output target
- Specifies keywords that specify items such as property information and arbitrary text that should be output.

Keyword	Meaning
name	Output of property name
value	Output of property value
text	Output of arbitrary text
endline	Output of line feed
set	Value setting for a variable
append	Value appending for a variable
variable	Output of variable
include	Reading of a file

- Output format

  Specifies a format, such as output column, for an output. Syntax to be specified will differ depending on the output target.

(3) Hide-type property output format specification

Syntax:

```
Format-name {
   (output-target-keyword output-format)
   (output-target-keyword output-format)
           :
           :
   (output-target-keyword output-format)
}
```

- Output target

  Specifies keywords that specify items such as property information and arbitrary text that should be output.

Keyword	Meaning
name	Output of property name
value	Output of property value
text	Output of arbitrary text
endline	Output of line feed
groupName	Output of group name
property	Output of internal property
set	Value setting for a variable
append	Value appending for a variable
variable	Output of variable
include	Reading of a file

- Output format

  Specifies a format, such as output column, for an output. Syntax to be specified will differ depending on the output target.

Property name output specification

Syntax:
(name output-column maximum-byte-count shift-specification)

- Output column (optional)

  Specifies the column count for starting the output.

  If 0 is specified, the property name will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Maximum byte count (optional)

  To limit the output of property names to a certain byte count, specify a maximum byte count.

  }Only the specified byte count will be output.

  If 0 is specified, there will be no limit.

  The default value is 0.

- Shift specification (optional)

  Specify this item when it is necessary to right-justify the property name output.

  This item is valid only when the maximum byte count is 1 or greater.

  left :left-justify
  right :right-justify

  The default mode is left justification.

Property value output specification

Syntax:
(value output-column maximum-byte-count shift-specification default-text)

- Output column (optional)

  Specifies the column count for starting the output.

  If 0 is specified, the property value will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Maximum byte count (optional)

  To limit the output of property values to a certain byte count, specify a maximum byte count.

  Only the specified byte count will be output.

  If 0 is specified, there will be no limit.

  The default value is 0.,

- Shift specification (optional)

  Specify this item when it is necessary to right-justify the property value output. This item is valid only when the maximum byte count is 1 or greater.

  left :left-justify
  right :right-justify

  The default mode is left justification.

- Default text (optional)

  Specifies the text to be output when no property has been input.

  In the default mode, nothing will be output.

Group name output specification

Syntax:

(groupName Index output-column maximum-byte-count shift-specification Default text)

- Index (optional)

  Specify the index for group names.

  If "0" is specified, an entire group will be output.

  The default value is 0.

- Output column (optional)

  Specifies the column count for starting the output.

  If 0 is specified, group names will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Maximum byte count (optional)

  To limit the output of group names to a certain byte count,specify a maximum byte count.

  Only the specified byte count will be output.

  If 0 is specified, there will be no limit.

  The default value is 0.

- Shift specification (optional)

  Specify this item when it is necessary to right-justify the group name output. This item is valid only when the maximum byte count is 1 or greater.

  left :left-justify
  right :right-justify
  The default mode is left justification.

- Default text (optional)

  Specifies the text to be output when the specified property has not been input.

  In the default mode, nothing will be output.

A.5.5 Variable setting and output specification

Text-type variables can be used in the format definition file.

These variables are global variables which are valid throughout the entire format definition file.

Setting of a text in a variable

Sets the specified text in a variable.

```
Syntax:
   (set variable-name text text)
   • Variable name
     Specifies the variable to be set.
   • Text
     Specifies the text to be set.
```

Appending of a text to a variable

Appends the specified text to a variable.

```
Syntax:
   (append variable-name text text)
   • Variable name
     Specifies the variable to be set.
   • Text
     Specifies the text to be appended.
```

Addition of variable and text

The value of the variable and a text are added together as numeric values and set as the variable.

Syntax:

```
(add variable-name text text)
```

- Variable-name

  Specify the name of the variable to be set.

- Text

  Specify the character string to be added.

Subtraction of variable and text

A text is subtracted from the value of the variable as a numeric value and set as the variable.

Syntax:
(sub variable-name text text)
- Variable-name
  Specify the name of the variable to be set.
- Text
  Specify the text to be subtracted.

Setting of a property in a variable

Sets the specified property value in a variable as a text.

Syntax:
(set variable-name property property-name)
- Variable name
  Specifies the variable to be set.
- Property name
  Specifies the property name to be set.
  If no property has been input, a text of 0 length will be set.

Appending of a property to a variable

Appends the specified property value to a variable as a text.

Syntax:
(append variable-name property property-name)
- Variable name
  Specifies the variable to be set.
- Property name
  Specifies the property name to be appended.
  If no property has been input, the variable will not change.

Addition of variable and property

The value of the variable and the value of a property are added together as numeric values and set as a separate variable.

Syntax:
(add variable-name property property-name)

- Variable name
  Specify the name of the variable to be set.
- Property name
  Specify the name of the property to be added.

Subtraction of variable and property

The value of a property is subtracted from the value of the variable as a numeric value and set as the variable.

Syntax:
(sub variable-name property property-name)

- Variable name
  Specify the name of the variable to be set.
- Property name
  Specify the the name of the property to be subtracted.

Setting of a variable into another variable

Copies the value of one variable to another variable.

Syntax:
(set variable-name-1 variable variable-name-2)

- Variable name 1
  Specifies the variable to be set.
- Variable name 2
  Specifies the variable to be copied to variable name 1.
  If no variable has been input, a text of 0 length will be set.

Appending of a variable to another variable

Appends the value of one variable to another variable.

Syntax:
(append variable-name-1 variable variable-name-2)

- Variable name 1
  Specifies the variable to be set.
- Variable name 2
  Specifies the variable name to be appended to variable name 1.
  If no variable has been input, the variable will not change.

Addition of variable and another variable

The value of the variable and that of another variable are added together as numeric values and set as a separate variable.

Syntax:
(add variable-name-1 variable variable-name-2)

- Variable name 1
  Specify the name of the variable to be set.
- Variable name 2
  Specify the name of the variable to be added.

Subtraction of variable and another variable

The value of another variable is subtracted from the value of the variable as a numeric value and set as a separate variable.

Syntax:
(sub variable-name-1 variable variable-name-2)

- Variable name 1
  Specify the name of the variable to be set.
- Variable name 2
  Specify the name of the variable to be subtracted.

Variable output

Syntax:
(variable variable-name output-column maximum-byte-count shift-specification default-text)

- Variable name

  Specifies the name of the variable to be output.

- Output column (optional)

  Specify the start column if it is necessary to align the variable output start positions.

  If 0 is specified, variables will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Maximum byte count (optional)

  To limit the output of variables to a certain byte count,specify a maximum byte count.

  Only the specified byte count will be output.

  If 0 is specified, there will be no limit.

  The default value is 0.

- Shift specification (optional)

  Specify this item when it is necessary to right-justify the variable output.

  This item is valid only when the maximum byte count is 1 or greater.

  left :left-justify
  right :right-justify
  The default mode is left justification.

- Default text (optional)

  Specifies the text to be output when the specified variable has not been input.

  In the default mode, nothing will be output.

A.5.6 Special character escape

Escape specification on character basis

When characters requiring special handling are included in the property values in the output format, escape characters can be inserted immediately before and after those special characters.
To accomplish this, those property values must first be set in variables.

Syntax:
(escapeChar variable-name comparison-condition escape-target escape-start-character escape-end-character)

- Variable name
  Specifies the name of the variable in which the escape character string is stored.
- Comparison condition
  "==" or "!="
  "==" : Lets one of the specified characters escape.
  "!=" : Lets characters other than those specified escape.Use this when specifying characters other than alphanumeric characters.
- Escape condition
  Specifies the character to escape (not to escape).

- Example

```
(escapeChar string == "\\ :" "\" " " ")
```

With the above specification, the following conversion will be performed.

```
comment: ZZZ\YYY -> comment": ZZZ"YYY
```

Escape specification on character string basis

When characters requiring special handling are included in the property values in the output format, the entire character string can be surrounded by specified characters.
To accomplish this, those property values must first be set in variables.

Syntax:
(escapeString variable-name comparison-condition escape-target escape-start-character escape-end-character)

- Variable name
  Specifies the name of the variable in which the escape character string is stored.
- Comparison condition "==" or "!="
  "==" : Lets one of the specified characters escape.
  "!=" : Lets characters other than those specified escape. Use this when specifying characters other than alphanumeric characters.
- Escape condition
  Specifies the character to escape (not to escape).

- Example

```
    (escapeString string == "\\ :" "\" " "\" ")
```

With the above specification, the following conversion will be performed.

```
    comment: ZZZ\YYY -> "comment: ZZZ\YYY"
```

A.5.7 Output condition specification

Limits the targets by specifying the condition in the output format of each object.

Specification of output target by property

An object is output only when its property satisfies the condition.

Syntax:
(if property-name comparison-operator comparison-target)

- Property name
  Specifies the name of the property to compare.
- Comparison operator
  "==", "!=", ">", "<", ">=" or "<="
- Comparison target
  Specifies a character string or a number.

If there are multiple descriptions, those objects matching all of the conditions will be output.
The conditions that can be specified will differ depending on the data type defined in the property definition file.

(1) Component type
(if NAME == "xxx")   : Matches
(if NAME != "xxx")   : Does not match

Conditions that can be used for comparison

Comparison condition	Meaning
fig	Figure
frame	Frame
parts	Parts
gate	Gate
block	Block
vcc	Power supply
gnd	Ground
hieConnector	Hierarchical connector
shtConnector	Sheet connector
powerBox	Power Box

When these conditions are connected using "|", an object will (will not) match one of those conditions.

(2) "text" type

Comparison is made with the text type based on ASCII sequence.

(if NAME == "xxx")   : Matches
(if NAME != "xxx")   : Does not match
(if NAME <= "xxx")   : Equal or smaller
(if NAME >= "xxx")   : Equal or greater
(if NAME < "xxx")   : Smaller
(if NAME > "xxx")   : Greater

"" can be used for comparison for "==" and "!=".

Comparison target	Meaning
"*"	Property has been input
"^xxx.*"	Property begins with "xxx"
".*xxx$"	Property ends with "xxx"
".*xxx.*	Property contains "xxx"

(3) "float" type

(if NAME == xxx)   : Matches
(if NAME != xxx)   : Does not match
(if NAME <= xxx)   : Equal or smaller
(if NAME >= xxx)   : Equal or greater
(if NAME < xxx)   : Smaller
(if NAME > xxx)   : Greater

(4) "int" type

(if NAME == xxx)   : Matches
(if NAME != xxx)   : Does not match
(if NAME <= xxx)   : Equal or smaller
(if NAME >= xxx)   : Equal or greater
(if NAME < xxx)   : Smaller
(if NAME > xxx)   : Greater

Specification of output target by variable

The object is output only when the variable specified satisfies the condition.

Syntax:
(ifv variable-name comparison-operator comparison-target)

- Variable name
  Specifies the name of the variable to be compared.
- Comparison operator
  "==", "!=", ">", "<", ">=" or "<="
- Comparison target
  Specifies a text.

If there are multiple descriptions, those objects matching all of the conditions will be output.
Comparison is made with the text type based on ASCII sequence.

```
(ifv NAME == "xxx")   : Matches
(ifv NAME != "xxx")   : Does not match
(ifv NAME <= "xxx")   : Equal or smaller
(ifv NAME >= "xxx")   : Equal or greater
(ifv NAME <  "xxx")   : Smaller
(ifv NAME  >  "xxx")      : Greater
```

"" can be used for comparison for "==" and "!=".

Components target	Meaning
"*"	Property has been input
"^xxx.*"	Property begins with "xxx"
".*xxx$"	Property ends with "xxx"
".*xxx.*	Property contains "xxx"

A.5.8 Output sequence specification

Output sequence specification

The sequence of part output is normally unspecified.
To use a property key to rearrange the output, use the following specification.

Syntax:
(sort property-name ascending/descending text-comparison-method)

- Property name

  Specifies the name of the property to be sorted.

- Ascending/descending

  up   : Ascending

  down : Descending

- Text comparison method

  A comparison method for sorting text-type properties.

  ascii   : ASCII sort
  reference : Reference sort
  The numbers at the end of the text are interpreted as a numerical value.

  For example, the result may appear as follows.

  ascii   : R1 R10 R2 R20 R3 R30
  reference : R1 R2 R3 R10 R20 R30

A.5.9 Grouping condition specification

Grouping condition specification

Specifies the property to be used as the grouping condition.
Objects possessing an identical value for this property will be grouped into one group.

Syntax:
(group property-name)

- Property name

  Specifies the name of the property to be used as the grouping condition.

Only one name can be specified.

A.5.10 Property grouping result output specification

Object count grouping specification

Specifies that the number of objects within the group be c* *ounted and the result be output.

```
Syntax:
(count count output-column)
  • Output column (optional)
    Specifies the column count in which to output the grouping result.
    If 0 is specified, objects will be output continuously to the previous
    output.
    The default value is 0.

    number :  absolute position specification
    +number: relative position specification
             Offset from the output column count of the previous data will
             be used.
```

Property grouping specification

Properties of the objects within the group are added and output.

```
Syntax:
(sum property-name output-column)
  • Property name
    Specifies the name of the property to be grouped. Specify either a "float"
    or "int" type property.
  • Output column (optional)
    Specifies the column count in which to output the grouping result.
    If 0 is specified, the result will be output continuously to the pre vious
    output.
    The default value is 0.

    number :  absolute position specification
    +number: relative position specification
             Offset from the output column count of the previous data will
             be used.
```

A.5.11 Error/warning output specification

An error or a warning can be output if the specified property has not been input.

Error condition specification

If the specified property has not been input, an error message will be created in the message file ("circuit-directory/log/dsnetprc.err"), and no net list will be created.

```
Syntax:
  (error property property-name)
   • Property name
    Specifies the property name that should result in an error if it has not
    been input.
```

Warning condition specification

If the specified property has not been input, a warning message will be created in the message file ("circuit-directory/log/dsnetprc.wrn").

```
Syntax:
  (warning property property-name)
  • Property name
    Specifies the property name that should result in a warning if it has not
    been input.
```

A.5.12 External file reading specification

External file reading specification

```
Syntax:
  (include variable-name)
  • Variable name
    Specifies the variable name in which the file name  to be read has been
    set.
```

Specify the file name with an absolute path.

A.5.13 Macro output specification

This item is used for describing an operation that is repeated in multiple locations, or for changing output text according to the property status.

Macro output specification

(1) Macro output specification

```
Syntax:
  (macro format-name output-column)
  • Format name
    Specifies the name of the output format for a macro.
    This name must be unique within the file, and must begin with "$".
  • Output column (optional)
    Specify the start column if it is necessary to align the macro output
    start positions.
    If 0 is specified, the macro will be output continuously to the
    previous output.
    The default value is 0.

    number :  absolute position specification
    +number: relative position specification
             Offset from the output column count of theprevious data
             will be used.
```

(2) Macro output format specification

```
Syntax:
    Format-name {
       (if output-target-specification)
            :
       (output-target-keyword output-format)
            :
    }
```

• Output target specification
To output only those objects possessing a certain property,specify this item.

• Output target keyword
Specifies keywords that specify items such as object property and arbitrary text that should be output.

Keyword	Meaning
text	Output of arbitrary text
endline	Output of line feed

property	Output of object property
set	Value setting for a variable
append	Value appending for a variable
variable	Output of a variable
include	Reading of a file

- Output format

  Specifies a format, such as output column, for an output. Syntax to be specified will differ depending on the output target.

A.6 Conversion program name specification

After net list creation, the program described here will be activated.
Use this specification when it is necessary to use shell or awk to further convert net lists created by the Netlist Processor.

Syntax:

```
execute : name-of-program-to-activate
```

- Name of program to activate

  Specifies the name of the program to activate.
  Specify either an absolute path or a file name located in thecommand search path.
  Activation syntax:

  ```
  name-of-program-to-activate output-file-name
  ```

A.7 Hierarchical expansion specification

The Netlist Processor can expand the blocks inside a circuit and output them.

Expansion level specification

Specifies the level to expand.

Syntax:
breakBlockLevel : Expansion level

- Expansion level
  Specifies the component type at which the expansion is to be stopped.
  Component types possess the following relationship.
  block > parts > gate
  - Higher hierarchy lower hierarchy

Expansion level	Meaning
all	Expand all.
no	No expansion.
block	No expansion.
parts	Expand blocks. Do not expand parts and gates.
gate	Expand blocks and parts. Do not expand gates.

Reference name expansion specification

Specifies how to handle the reference name of the components inside the internal circuits of the expanded block.

Syntax:
breakReferenceMode : (expansion-mode delimiter-character)

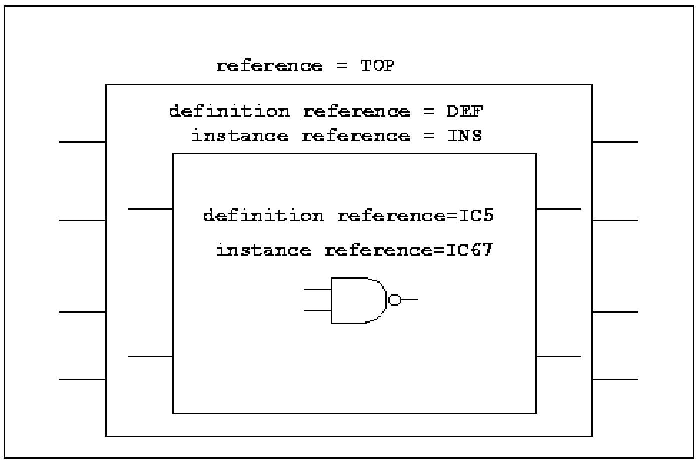
- Expansion mode

Expansion level	Meaning
instance	Instance hierarchy reference
definition	Definition hierarchy reference
insert	Output with block reference as prefix
append	Output with block reference as suffix

- Delimiter character (optional)
  Specifies the delimiter character for adding a block reference.
  In the default mode, no character will be output.

- Instance hierarchy reference name
  Outputs an instance hierarchy reference name.
- Definition hierarchy reference name
  Outputs a definition hierarchy reference name.
  If the same block is used more than once, the reference name will overlap.
- Output with block reference name as prefix
  The reference name of the expanded block is added to the definition hierarchy reference name as a prefix before the blocks are output.
  block-reference name + delimiter-character + ... + delimiter-character + reference name
  Example) BLOCK1/BLOCK2/BLOCK3/R1
- Output with block reference name as suffix
  The reference name of the expanded block is added to the definition hierarchy reference name as a suffix before the blocks are output.
  The sequence will be reversed from that used in insert.
  reference name + delimiter-character + ... + delimiter-character + block reference name

Example) R1/BLOCK3/BLOCK2/BLOCK1



When expansion level =instcnce : IC67
When expansion level =definition : IC5
When expansion level =insert : TOP.DEF.IC5
When expansion level =append : IC5.DEF.TOP

A.8 Block information output specification

A.8.1 Block information output specification

The block information inside a circuit can be output regardless of expansion specification.

Block information output

The blocks located immediately below the object will be output. In other words, if this specification is made in the circuit output format, only the highest-order blocks will be output.

```
Syntax:
  (block format-name output-column default-text delimiter-text)
```

- Format name

  Specifies the name of the output format for the block.

  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the block information output start positions.

  If 0 is specified, the block information will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no block.

  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the block output.

  In the default mode, nothing will be output.

Block pin information output

Specifies block pin output**.**

Syntax:
(blockPin format-name output-column default-text delimiter-text)

- Format name

  Specifies the name of the output format for the block pin.

  This name must be unique within the file, and must begin with "$".

- Output column (optional)\

  Specify the start column if it is necessary to align the block pin information output start positions.

  If 0 is specified, the block pin information will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no block pin.

  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the block pin output.

  In the default mode, nothing will be output.

All block information output

Specifies that all blocks within the circuit be output regardless of the expansion state.
This specification can be made only inside the circuit output format.

Syntax:
(allBlock format-name output-column default-text delimiter-text)

- Format name
  Specifies the name of the output format for the block.
  This name must be unique within the file, and must begin with "$".
- Output column (optional)
  Specify the start column if it is necessary to align the block information output start positions.
  If 0 is specified, the block information will be output continuously to the previous output.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.
- Default text (optional)
  Specifies the text to be output when there is no block.
  In the default mode, nothing will be output.
- Delimiter text (optional)
  Specifies the text to be used for delimiting the block output.
  In the default mode, nothing will be output.

Block information recursive output specification

The block tree inside the circuit is sequentially read beginning at the top and is recursively output.
This specification can be made only inside the circuit output format.

Syntax:
(recursive Block format-name output-column default-text delimiter-text)

- Format name
  Specifies the name of the output format for the block.
  This name must be unique within the file, and must begin with "$".
- Output column (optional)
  Specify the start column if it is necessary to align the block information output start positions.
  If 0 is specified, the block information will be output continuously to the previous output.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)
  Specifies the text to be output when there is no block.
  In the default mode, nothing will be output.
- Delimiter text (optional)
  Specifies the text to be used for delimiting the block output.
  In the default mode, nothing will be output.

The above specification has the same effect as the following specification.

```
outputFormat {
   (block $blockFomat
   (block $format    1 "" "")
}
$format {
   (block $format    1 "" "")
}
```

However, in reality the above specification cannot be made since the format definition file is prohibited from calling itself inside the format.

Block output format specification

Syntax:

```
 Format-name {
   (if output-target-specification)
        :
   (sort output-sequence-specification)
        :
   (output-target-keyword output-format)
        :
   }
```

- Output target specification
  To output only those objects possessing a certain property,specify this item.
- Output sequence specification
  Specify this item when it is necessary to output objects in an ascending or descending order of properties.
- Output target keyword
  Specifies keywords that specify items such as object property and arbitrary text that should be output.

Keyword	Meaning
text	Output of arbitrary text
endline	Output of line feed
property	Output of object property
set	Value setting for a variable
append	Value appending for a variable
variable	Output of a variable
include	Reading of a file
Object name	Output of an object
Object group name	Output of object group

- Output format

  Specifies a format, such as output column, for an output.

  Syntax to be specified will differ depending on the output target.

A.8.2 Block information grouping output specification

Block information inside the circuit can be grouped and output regardless of the expansion specification.

Block grouping output specification

The blocks located immediately below the object will be grouped. In other words, if this specification is made in the circuit output format,only the highest-order blocks will be grouped and output.

Syntax:
(blockGroup format-name output-column default-text delimiter-text)

- Format name

  Specifies the name of the output format for the grouping result output.

  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the output start positions.

  If 0 is specified, the grouping result will be output continuously to the previous output.

  The default value is 0.

number :	absolute position specification
+number:	relative position specification Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no group.

  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the group output.

  In the default mode, nothing will be output.

Block grouping output specification

Groups block pins.

Syntax:
(blockPinGroup format-name output-column default-text delimiter- text)

- Format name

  Specifies the name of the output format for the grouping result output.
  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the output start positions.
  If 0 is specified, the grouping result will be output continuously to the previous output.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no group.
  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the group output.
  In the default mode, nothing will be output.

All block grouping output specification

Specifies that all blocks within the circuit be grouped and output, regardless of the expansion state.
This specification can be made only inside the circuit output format.

Syntax:
(allBlockGroup format-name output-column default-text delimiter-text)

- Format name

  Specifies the name of the output format for the grouping result output.

  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the output start positions.

  If 0 is specified, the grouping result will be output continuously to the previous output.

  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no group.

  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the group output.

  In the default mode, nothing will be output.

Specification of recursive grouping output for blocks

The block tree inside the circuit is sequentially read and grouped beginning at the top and is recursively output.
This specification can be made only inside the circuit output format.

Syntax:
(recursiveBlockGroup format-name output-column default-text delimiter-text)

- Format name

  Specifies the name of the output format for the grouping result output.
  This name must be unique within the file, and must begin with "$".

- Output column (optional)

  Specify the start column if it is necessary to align the output start positions.
  If 0 is specified, the grouping result will be output continuously to the previous output.
  The default value is 0.

  number : absolute position specification
  +number: relative position specification
  Offset from the output column count of the previous data will be used.

- Default text (optional)

  Specifies the text to be output when there is no group.
  In the default mode, nothing will be output.

- Delimiter text (optional)

  Specifies the text to be used for delimiting the group output.
  In the default mode, nothing will be output.

Specification of grouping output format for blocks

Syntax:

```
Format-name {
    (group grouping-condition-specification)
              :
    (if output-target-specification)
              :
    (sort output-sequence-specification)
              :
    (output-target-keyword output-format)
              :
 }
```

- Grouping condition specification
  Specifies the property to be used as the grouping condition.
- Output target specification
  To output only those objects possessing a certain property,specify this item.
- Output sequence specification
  Specify this item when it is necessary to output objects in an ascending or descending order of properties.
- Output target keyword
  Specifies keywords that specify items such as object property and arbitrary text that should be output.

Keyword	Meaning
text	Output of arbitrary text
endline	Output of line feed
property	Output of object property
set	Value setting for a variable
append	Va lue appending for a variable
variable	Output of a variable
include	Reading of a file
Object name	Output of an object
Object group name	Output of object group

- Output format

  Specifies a format, such as output column, for an output.

  Syntax to be specified will differ depending on the output target.

A.9 Specifying message outputs

Specifying message outputs

- Specifying net-related message outputs
  You specify how the outputs of the following messages are to be handled:

(1) "Multiple different nets are connected to the part (Reference: pin of <Reference> (Pin No.: <Pin No.>). Check to see if there is reference duplication or gate duplication."

(2) "The same net name is located in another sheet. The same net name will be duplicated."

(3) "The same net name is located on another level. The same net name will be duplicated."

(4) "Corresponding sheet connector is not found."

(5) "Reference is duplicated. Please check."

```
Syntax:
    messMode : Output classification
Output classification:
    • error
      Output as an error.
    • warning
      Output as a warning.
    • notout
      No output is delivered.
```

- Specifying part-related message outputs
  You specify how the output of the following message is to be handled:

(1) "There is no CDB name entered."

```
Syntax:
   noLCDB : Output classification
Output classification:
   • yes
     No output is delivered.
   • no
     Output as a warning.
```

(2) "Package symbol for part (CDB name:xxx) is not registered in LCDB."

```
Syntax:
   noLCDBPackage : Output classification
Output classification:
   • yes
     No output is delivered.
   • no
     Output as a warning.
```

(3) "Part (CDB name: XX) is not registered in LCDB."
"CDB name XX : This part does not have component cell (component name: YY)."

```
Syntax:
   noLCDBParts : Output classification
Output classification:
   • error
      Output as an error.
   • warning
      Output as a warning.
```

- Specifying pin-related message outputs
  You specify how the output of the following message is to be handled:

(1) "There are multiple pins with the same name (Pin name: <Pin name>) in component (Component name: <Component name>)."

```
Syntax:
   dupPin : classification
classification:
```

- error
  Output as an error.
- warning
  Output as a warning.
- notout
  No output is delivered.

A.10 Specification of output in MS-DOS text file format

Normally, files are output in the binary mode in order to maintain compatibility with UNIX.
When this specification is used, files are output in the Windows text mode. Care must be taken since UNIX will treat these files as containing special codes.

```
Syntax:
    fileFormat: DosText
```

Appendix B Prohibited characters list and Properties required for outputting netlists in CR-5000/CDB, SD, BD

This appendix provides a list of prohibited characters and a list of properties passing between systems.

- Prohibited characters in CR-5000 (SD, CDB, BD, BP/Rev.7.0, PWS/ Rev.14.0)
- Properties required for outputting netlists (Rev.7.0)

B.1 Usable characters and count of characters in CR-5000

In CR-5000, identifiers (IDs) are used to organize various kinds of information together and to identify similar objects. For example, the IDs are a part name in CDB library and a reference of component.
The setting of IDs is restricted by certain rules in CR-5000. Some applications have further restrictions in addition to the rules on usable characters and count of characters.

This section shows a list of characters that cannot be used in IDs and usable maximum character counts for each system. Refer to this for help when creating CDB library, technology library, and other libraries, and when creating a schematic and a PC board.

How to reference the table

Among identifiers used in CR-5000/CDB, SD, BD, and BP, there are some properties predetermined by the system. These IDs are called system reserved properties.
System reserved properties for each system are as follows

System name	System reserved property
CDB	- Part name - Package name - Stock code - Footprint spec name - Pinassign name - Footprint name - Pin name - Footprint layer name - Pin number - Padstack group name - Function name - Padstack name - Pad name - Function pin name
SD	- Part name - Component name - Reference designator - CDB name - Pin number - Function name - Component group name - Pin name - Net group name - Net name - Default power net - Default ground net - Stock code - Split component - I/O properties - Common terminal - Pin type - Accept net - Global flag - Net kind

System name	System reserved property
BD,BP	- Technology name - User defined layer name - Design rule stack name - Component group name - Wiring width stack name - Design rule unit name - Reference designator - Symbol ID - Net group group name - Panel specification name - Net group name - Pinpair group name - Pinpair group group name - Net name - Visible layer group name - Photo format name - Drill machine name - Photo machine name - Drill format name - Grid name - Tool table name - Aperture table name

The followings are the meanings of notations used in the prohibited character list.

Notation	Meaning
▨	This indicates a prohibited character for each tool.
*1	Since this is a prohibited character in SD, it cannot be used in the property name converted to LCDB.
*2	Enclose a text string with double-quotation marks when using BD Ascii/IO.
*3	Put backslash "\" (the escape character) preceding this character when using BD AsciiI/O.
*4	This indicates case-insensitive. All of these are handled as uppercase characters.
*5	For notes on terminal number, see 5.4 "Notes On Terminals with Alphanumeric Characters" in PWS Basic Vol.1 "Introduction to PWS."
*6	"/*" and "*/" cannot be used since they are handled as comment in an ASCII file.
*7	This is replaced to "_" in SPECCTRA I/F.
*8	This is converted to ASCII code of hexadecimal notation.
*9	This is replaced to "_" in SPECCTRA QUEST I/F.
*10	This indicates case-insensitive in SPECCTRA QUEST I/F.
*11	This indicates that head of the string is deleted if the character count exceeds the maximum value in SPECCTRA QUEST I/F.
*12	A file name that begins with "." is prohibited.
*13	This character is not available in some tools on Windows version.

Notation	Meaning
*14	TPA(Version3.6) I/F: If a character count exceeds the maximum value, characters are removed from the end of the string.
*15	TPA(Version3.6) I/F: A text string that begins with "#" is replaced to "_".
*16	This is replaced to "_" in BD-ICX I/F.
*17	A property name that begins with "pvw_" is prohibited.
*18	To use this character, changing the data resource is required. However, the character is prohibited when it is used as a bit separator.
*19	This character is prohibited when only one character "*" is used.
*20	*Refer to A.5 "Limitation values in CDB and BD".
*21	For properties passed to CDB and BD, refer to A.5 "Limitation values in CDB and BD".
*22	Numbers only cannot be used.
*23	"#TEMPORARY" and "#UNCONNECT" cannot be used.
*24	When the name is automatically generated by the system, this character may be used as a prefix.
*25	Note that this character is converted to "Å}" when a component symbol is generated with the same text string as a reference designator or a device name. For details, see Appendix B "Characters Registered in User Font File" in PWS Basic Vol.1 "Introduction to PWS."
*26	For Windows version, the characters are case-insensitive.

From next page, the prohibited character list is shown.

Prohibited characters in CR-5000 (SD,CDB,BD,BP/Rev.7.0,PWS/Rev.14.0)

System Name	Identifier(ID)	Count	NL	Tab	sp	!	"	#	$	%	&	`	(	)	*	+	,	-	.	/	0-9
CDB	System reserved property value	*20	■	■	■		*1,3	*2					*2	*2			*1,2				
	User defined property name	*20	■	■	■	*1	*1,3	*1	*1	*1	*1	*1	*1,2	*1,2	*1	*1	*1,2	*1	*1	*1	
	User defined property name	*20	■	*2	*2		*1,3						*2	*2							
	File name	*20	■	■	■		■		■	*13	■	■			■		■		*12	■	
SD	System reserved property value	No limit(*21)	■	■	■		■								*19		*18				
	User defined property name	No limit(*21)	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	
	User defined property value	No limit(*21)					■								*19						
	File name	sys	■	■	■	■	■	■	■	■	■	■	■	■	■		■		■	■	
BD/BP	System reserved property value	*20	■	■	■		*3	*2					*2	*2			*2				
	Design rule name	*20	■	■	■		■		■		■	■	*2	*2	■		■		*12	■	
	Comment	*20	*2	*2	*2		■	*2					*2	*2			*2				
	File name	*20	■	■	■		■		■	*13	■	■			■		■		*12	■	
Plot tool	Page name	*20	■	■	■		■		■	*13	■	■			■		■		*12	■	
Photo tool	Film name	*20	■	■	■		■		■	*13	■	■			■		■		*12	■	
Drill tool	Step name	*20	■	■	■		■		■	*13	■	■			■		■		*12	■	
PWS	Device name	20	■	■	■	■	■	*24	■			■	■	■	*6		■			*6	
	Symbol name	20	■	■	■	■	■		■			■	■	■	*6		■			*6	
	Pin name	20	■	■	■	■	■		■			■	■	■	*6		■			*6	
	Terminal number/Pin number *5	20	■	■	■	■	■	■	■			■	■	■	■		■			■	
	Pin name *22	20	■	■	■	■	■		■			■	■	■	*6		■			*6	
	Signal name *23	20	■	■	■	■	■	*24	■			■	■	■	*6		■			*6	
	Symbol idendifier	20	■	■	■	■	■	*24	■			■	■	■	*6		■			*6	
	Reference designator	20	■	■	■	■	■	*24	■			■	■	■	*6		■			*6	
	File name	20	■	■	■	■	■		■			■	■	■	■		■			■	

- *1: Since this is a prohibited character in SD, it cannot be used in the property name converted to LCDB. *2: Enclose a text string with double-quotation marks when using BD Ascii/IO. *3: Put backslash "\" (the escape character) preceding this character when using BD Ascii/IO. *5: For notes on terminal number, see 5.4 "Notes On Terminals with Alphanumeric Characters" in PWS Basic Vol.1 "Introduction to PWS." *6: "/*" and "*/" cannot be used since they are handled as comment in an ASCII file.
  *12: A file name that begins with "." is prohibited. *13: This character is not available in some tools on Windows version. *18: To use this character, changing the data resource is required. However, the character is prohibited when it is used as a bit separator. *19: This character is prohibited when only one character "*" is used.
  *20: *Refer to A.5 "Limitation values in CDB and BD". *21: For properties passed to CDB and BD, refer to A.5 "Limitation values in CDB and BD". *22: Numbers only

cannot be used. *23: "#TEMPORARY" and "#UNCONNECT" cannot be used. *24: When the name is automatically generated by the system, this character may be used as a prefix.

- For "system reserved poperties" for each system, refer to the pageB-2.

System Name	Identifier(ID)	:	;	<	=	>	?	@	A-Z	[	¥	]	^	_	`	a-z	{	|	}	~	del
CDB	System reserved property value	*2	*2								*1,3										■
	User defined property name	*1,2	*1,2	*1	*1	*1	*1	*1		*1	*1,3	*1	*1	*1	*1		*1	*1	*1	*1	■
	User defined property name		*2								*1,3										■
	File name	■	■	■	*13	■	■				■		*13		■			■			■
SD	System reserved property value										■										■
	User defined property name	■	■	■	■	■	■	■		■	■	■	■	*17	■		■	■	■	■	■
	User defined property value										■										■
	File name	■	■	■	■	■	■	■		■	■	■	■		■		■	■	■	■	■
BD/BP	System reserved property value	*2	*2								*2										■
	Design rule name	■	■	■		■	■				■				■			■			■
	Comment	*2	*2								*2										■
	File name	■	■	■	*13	■	■				■		*13		■			■			■
Plot tool	Page name	■	■	■	*13	■	■				■		*13		■			■			■
Photo tool	Film name	■	■	■	*13	■	■				■		*13		■			■			■
Drill tool	Step name	■	■	■	*13	■	■				■		*13		■			■			■
PWS	Device name	■	■	■		■			*4		■		■	*25	■	*4	■	■	■	■	■
	Symbol name	■	■	■		■			*4		■		■		■	*4	■	■	■	■	■
	Pin name	■	■	■	■	■			*4		■		■		■	*4	■	■	■	■	■
	Terminal number/Pin number *5	■	■	■	■	■	■		*4		■		■		■	*4	■	■	■	■	■
	Terminal name *22	■	■	■		■			*4		■		■		■	*4	■	■	■	■	■
	Signal name *23	■	■	■		■			*4		■		■		■	*4	■	■	■	■	■
	Symbol idendifier	■	■	■		■			*4		■		■		■	*4	■	■	■	■	■
	Reference designator	■	■	■		■			*4		■		■	*25	■	*4	■	■	■	■	■
	File name	■	■	■		■	■		*26	■	■	■	■		■	*26		■			■

- *1: Since this is a prohibited character in SD, it cannot be used in the property name converted to LCDB. *2: Enclose a text string with double-quotation marks when using BD Ascii/IO. *3: Put backslash "\" (the escape character) preceding this character when using BD Ascii/IO. *4: This indicates case-insensitive. All of these are handled as uppercase characters. *5: For notes on terminal number, see 5.4 "Notes On Terminals with Alphanumeric Characters" in PWS Basic Vol.1 "Introduction to PWS."
  *13: This character is not available in some tools on Windows version. *17: A property name that begins with "pvw_" is prohibited. *22: Numbers only cannot be used.
  *23:   "#TEMPORARY" and "#UNCONNECT" cannot be used. *25: Note that this character is converted to "+-" when a component symbol is generated with the same text string as a reference designator or a device name. For details, see Appendix B "Characters Registered in User Font File" in PWS Basic Vol.1 "Introduction to PWS."
  *26: For Windows version, the characters are case-insensitive.
- For "system reserved poperties" for each system, refer to the page B-2.

System Name	Identifier(ID)	Count	NL	Tab	sp	!	"	#	$	%	&	`	(	)	*	+	,	-	.	/	0-9
CR-5000 SPECCTRA Autorouter	Reference designator (Reference designator)	No limit	■	■	■								■	■							
	Net name (Signal name)	No limit	■	■	■								*7	*7							
	Footprint name	No limit	■	■	■								■	■							
	Padstack name	No limit	■	■	■								■	■							
	Pin name	No limit	■	■	■								■	■							
	Pin number (Terminal number)	No limit	■	■	■								■	■				■			
CR-5000 FLEX-ART	Reference designator (Reference designator)	32	■	■	■																
	Net name (Signal name)	32	■	■	■								■	■							
	Part name (Device name)	32	■	■	■																
	Footprint name	32	■	■	■																
	Padstack name	32	■	■	■																
	Pad name	32	■	■	■																
	Pin number (Terminal number)	4	■	■	■																
CR-5000 QUAD XTK,QUIET	Reference designator (Reference designator)	No limit	■	■	■		■	■				■	■	■			■				
	Net name (Signal name)	No limit	■	■	■		■	■				■	■	■			■				
	Part name (Device name)	No limit	■	■	■		■	■				■	■	■			■				
	Padstack name	No limit	■	■	■		■	■				■	■	■			■				

- *7: This is replaced to "_" in SPECCTRA I/F.
- For "system reserved poperties" for each system, refer to the page B-2.

System Name	Identifier(ID)	:	;	<	=	>	?	@	A-Z	[	¥	]	^	_	`	a-z	{	|	}	~	del
CR-5000 SPECCTRA Autorouter	Reference designator (Reference designator)		■																		■
	Net name (Signal name)		*7																		■
	Footprint name		■																		■
	Padstack name		■																		■
	Pin name		■																		■
	Pin number (Terminal number)		■																		■
CR-5000 FLEX-ART	Reference designator (Reference designator)																				■
	Net name (Signal name)						■			■		■									■
	Part name (Device name)																				■
	Footprint name																				■
	Padstack name																				■
	Pad name																				■
	Pin number (Terminal number)																				■
CR-5000 QUAD XTK,QUIET	Reference designator (Reference designator)	■	■		■										■						■
	Net name (Signal name)	■	■		■										■						■
	Part name (Device name)	■	■		■										■						■
	Padstack name	■	■		■										■						■

- *7: This is replaced to "_" in SPECCTRA I/F.
- For "system reserved poperties" for each system, refer to the page B-2.

System Name	Identifier(ID)	Count	NL	Tab	sp	!	"	#	$	%	&	`	(	)	*	+	,	-	.	/	0-9
CR-5000 Apsim	Reference designator (Reference designator)	22	■	■	■																
	Net name (Signal name)	19	■	■	■																
	Part name (Device name)	22	■	■	■																
	Footprint name	14	■	■	■																
CR-5000 ICX	Reference designator	1000	■	■	■		*16	*16		*16	*16	*16									
	Net name	1000	■	■	■		*16	*16		*16	*16	*16									
	Part name	1000	■	■	■		*16	*16		*16	*16	*16									
	Padstack name	1000	■	■	■		*16	*16		*16	*16	*16									
	Pad name	1000	■	■	■		*16	*16		*16	*16	*16									
	Footprint name	1000	■	■	■		*16	*16		*16	*16	*16									
	Component group name	1000	■	■	■		*16	*16		*16	*16	*16									
	Comment	1000	■	■	■		*16	*16		*16	*16	*16									
CR-5000 SPECCTRA-Quest	Padstack name	No limit	■	■	■																
	Net name	30 *11	■	■	■	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	
	Part name	30 *11	■	■	■	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	
	Pin name	30 *11	■	■	■	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	
	Reference designator	30 *11	■	■	■	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	
	Design rule stack name	30 *11	■	■	■	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	
	Wiring width stack name	30 *11	■	■	■	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	
	Net group name	18 *11	■	■	■	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	*9	

- *9: This is replaced to "_" in SPECCTRA QUEST I/F. *11: This indicates that head of the string is deleted if the character count exceeds the maximum value in SPECCTRA QUEST I/F. *16: This is replaced to "_" in BD-ICX I/F.
- For "system reserved poperties" for each system, refer to the page B-2.

System Name	Identifier(ID)	:	;	<	=	>	?	@	A-Z	[	¥	]	^	_	`	a-z	{	|	}	~	del
CR-5000 Apsim	Reference designator (Reference designator)																				▨
	Net name (Signal name)																				▨
	Part name (Device name)																				▨
	Footprint name																				▨
CR-5000 ICX	Reference designator	*16	*16		*16		*16	*16			*16		*16		*16		*16	*16	*16		▨
	Net name	*16	*16		*16		*16	*16			*16		*16		*16		*16	*16	*16		▨
	Part name	*16	*16		*16		*16	*16			*16		*16		*16		*16	*16	*16		▨
	Padstack name	*16	*16		*16		*16	*16			*16		*16		*16		*16	*16	*16		▨
	Pad name	*16	*16		*16		*16	*16			*16		*16		*16		*16	*16	*16		▨
	Footprint name	*16	*16		*16		*16	*16			*16		*16		*16		*16	*16	*16		▨
	Component group name	*16	*16		*16		*16	*16			*16		*16		*16		*16	*16	*16		▨
	Comment	*16	*16		*16		*16	*16			*16		*16		*16		*16	*16	*16		▨
CR-5000 SPECCTRA-Quest	Padstack name																				▨
	Net name	*9	*9	*9	*9	*9	*9	*9	*10	*9	*9	*9	*9		*9	*10	*9	*9	*9	*9	▨
	Part name	*9	*9	*9	*9	*9	*9	*9	*10	*9	*9	*9	*9		*9	*10	*9	*9	*9	*9	▨
	Pin name	*9	*9	*9	*9	*9	*9	*9	*10	*9	*9	*9	*9		*9	*10	*9	*9	*9	*9	▨
	Reference designator	*9	*9	*9	*9	*9	*9	*9	*10	*9	*9	*9	*9		*9	*10	*9	*9	*9	*9	▨
	Design rule stack name	*9	*9	*9	*9	*9	*9	*9	*10	*9	*9	*9	*9		*9	*10	*9	*9	*9	*9	▨
	Wiring width stack name	*9	*9	*9	*9	*9	*9	*9	*10	*9	*9	*9	*9		*9	*10	*9	*9	*9	*9	▨
	Net group name	*9	*9	*9	*9	*9	*9	*9	*10	*9	*9	*9	*9		*9	*10	*9	*9	*9	*9	▨

- *9: This is replaced to "_" in SPECCTRA QUEST I/F. *10: This indicates case-insensitive in SPECCTRA QUEST I/F. *16: This is replaced to "_" in BD-ICX I/F.
- For "system reserved poperties" for each system, refer to the page B-2.

B.2 Properties required for outputting netlists (Rev.7.0)

	Property	Compo-nent	Gate	Power box	Sheet Connector	Hierarchical Connector	Block	Power	Ground
CR-5000/CCF	Part name	R	R	R	R			R	R
	Reference	R	R	R	N	N	N	N	N
	CDB name	L	L	L					
	Component name	L	L	L					
	Function name								
	Pin name								
	Pin number	R	R	R					
	IO property	P	P	R					

	Property	Compo-nent	Gate	Power box	Sheet Connector	Hierarchical Connector	Block	Power	Ground
CR-5000/GNF	Part name	R	R	R	R			R	R
CR-5000/ECF	Reference	R	R	R	N	N	N	N	N
	CDB name	L	L	L					
	Component name	L	L	L					
	Function name	R	R						
	Pin name	R	R						
	Pin number	R	R	R					
	IO property	P	P	R					

	Property	Compo-nent	Gate	Power box	Sheet Connector	Hierarchical Connector	Block	Power	Ground
CR-5000/ NDF,RUF LCDB is essential.	Part name	R	R	R	R			R	R
	Reference	R	R	R	N	N	N	N	N
	CDB name	R	R	R					
	Component name	R	R	R					
	Function name		R						
	Pin name	R	R						
	Pin number	R	R	R					
	IO property	P	P	R					

Meaning of the notations

R...Required   L...Required when referencing LCDB

P...Required when there are power, ground, and NC pins

N...Cannot overlap reference of component, gate, and power box