EEDRAW

User Guide

EEdraw Manual For Version 2.4,
Gershon Elber & Peter Cooper

EEDRAW - Electrical Engineering Drawings (ver 2.4)

BECAUSE EEDRAW/EED-PS/EEDEPSON ARE LICENSED FREE OF CHARGE, I PROVIDE ABSOLUTELY NO WARRANTY, TO THE EXTENT PERMITTED BY APPLICABLE STATE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING, I GERSHON ELBER PROVIDE EEDRAW/EED-PS/EEDEPSON PROGRAMS "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE EEDRAW/EED-PS/EEDEPSON PROGRAMS PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW WILL GERSHON ELBER, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY LOST PROFITS, LOST MONIES,OR OTHER SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR A FAILURE OF THE PROGRAM TO OPERATE WITH PROGRAMS NOT DISTRIBUTED BY GERSHON ELBER) THE PROGRAM, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

1. INTRODUCTION

EEDraw is a simple parametric drawing program, specifically designed for EE (Electrical Engineering) type of drawing. The program fully allows new parametric types, through library mechanisms, and in fact all the EE types are defined this way. This library mechanism is extremely useful in defining new types and can be used for other applications other than EE.

The following graphic devices are supported for display: Hercules, CGA, ATT, EGA, VGA and SVGA (through user provided BGI driver). The following printing devices are supported (via the printing drivers): EPSON compatible printers (8 pins), PostScript printers.

Usage: EEDraw [-z] *[FileName[.EED]]

Options: [-z] Print version number and usage.

Quick start:
The program comes with quite common setup. If you have a mouse and a VGA/EGA/Hercules/CGA then you can simply start eedraw. Typing 'eedraw 82720-1.eed 82720-2.eed' will restart eedraw and prompt you for two windows for the above two files. Click on left button (return on keyboard) to pick window corners. Click on right button (space bar on keyboard) if you want to default size (maximum size). Alt-Q, by default, will take you out.

From version 2.4 onwards a layer structure has been implemented, this will enable the user to display only the information on the screen they want, also in adding new layers of there own further detail and notes can be added to a drawing. Hard copy of drawings show only the layers that were turned on when the drawing was last saved, all other information is hidden from the printed output.

2. INSTALLATION

For correct execution of the main program (EEDraw), the following files must exist in one of the path directories:

1. EEDraw.exe
2. EEDraw.cfg
3. EEDraw.bnd
4. EEDraw.hlp
5. *.lib

For correct execution of the printer drivers (EED-PS, EED-Epsn), the following files must exists in one of the path directories (see also Appendix B):

1. EED-PS.exe
2. EED-PS.cfg
3. *.lib

1. EED-Epsn.exe
2. EED-Epsn.cfg
3. *.lib

The library files (*.lib) are the same for the printer drivers and eedraw itself and should exist probably only once. See SETUP below shows how to change the setup of this programs. The default values should be enough to start and learn this program.

3. SETUP

EEDraw set up is done through two files: EEDraw.cfg and EEDraw.bnd.

EEDraw.cfg

This files contain global parameters that should be selected and set only once, in final installation stage. Empty lines or anything after a semicolon (';') is ignored. This file is read once when eedraw is being started. Parameters that can be set are:

1. AutoPan - Boolean flag. If TRUE, any operation involves in the drawing space (such as move, copy or draw) will auto-pan if cursor exists screen. If FALSE, Panning option must be used manually to perform this operation. This Boolean flag can be set from the Status main menu.
   For example:
   AutoPan TRUE
2. WindowName - Boolean flag. If TRUE each EEDraw data file name is printed in the head of the window. If FALSE no name is printed and this makes some more space for the drawing itself. This Boolean flag can be set from the Window main menu.
   For example:
   WindowName TRUE
3. ATKeyboard - Boolean flag. If TRUE, it is assumed AT style keyboard bios calls (also known as Enhanced keyboard - if you have F11/F12 on your keyboard, good chance you can set this to

TRUE). This will enable using more complex scan codes, not supported by regular keyboard (such as F11/F12). Note however it may HANG your system if it is not so.

For example:

ATKeyboard FALSE

4. MouseSensitivity - Integer flag. Default is set to 10 which is a good start. Making this number bigger makes the mouse less sensitive.

For example

MouseSensitivity 10

5. HVLines - Boolean flag. If TRUE, lines are coerced to be vertical or horizontal only. This Boolean flag can be set from the Status main menu.

For example

HVLines TRUE

6. SnapDistance - Integer value. Sets the distance in drawing space of points to be snapped. You probably do not want to change this, especially not in the middle of a drawing. This Integer value can be set from Status main menu.

For example:

SnapDistance 16

7. Allow256Colors - Boolean flag. 256 colours BGI drivers will have a richer set of colours. If FALSE such a driver will be forced to use only 16 colours effectively having same colours are EGA or VGA default colours (EGAVGA.BGI).

For example:

Allow256Colors FALSE

8. AllowAsyncEvents - Boolean flag. The user interface supports async event handling. This means that one can pop up something while something else is being popped up already for example. This has the advantage of the ability to arbitrary "jump" between commands. However this may be VERY confusing as well. Leave it FALSE at least at the beginning!

For example:

AllowAsyncEvents FALSE

9. SaveBackMethod - Integer. Underlaying windows are being saved for short life time pop up items such as queries. This integer selects where to save this data.

Options are:

1 - Conventional memory. This obviously reduces the amount of memory for the EE drawings themselves.

2 - Expanded memory.

3 - XMS Extended memory (currently not supported).

4 - Disk file. For reasonable response this better be a ram disk. See also SaveBackPath below. The device, this data is saved into, should be able to hold ~100k for 16 colours BGI drivers and twice that for 256 colours BGI drivers.

For example:

SaveBackMethod 1

10. SaveBackPath - String. This string specifies where back windows should be saved. This full path should have a postfix '\' in it.

For example:

SaveBackPath "f:\temp\"

11. Mouse - Boolean flag. Inform the system if a Mouse exists or not. If this is commented

out, auto detection will be used.

For example:

Mouse TRUE

12. Joystick - Boolean flag. Inform the system if a joystick exists or not. Must be TRUE for a joystick to be used. Personally the joystick was not found accurate enough for this type of drawing.

For Example:

Joystick FALSE

13. BGIDriverPath - specifies where to look for the *.BGI drivers. Note the drivers are supplied with this program (type 'dir *.bgi') and you should update this path to the exact place you had moved them.

For example:

BGIDriverPath "C:/TC/BGI/ATI"

14. GraphDriver - Set what type of display device to use. The relative driver *.BGI are looked in the directory specified in BGIDriverPath above. The following values are valid (same as TC 2.0/TC++ 1.0 graphics.h file if that helps):

0 - Auto detect (default, so you can delete this one...)
1 - CGA (Hires two colours mode)
3 - EGA
4 - EGA64
5 - EGAMONO
7 - Hercules
8 - ATT (Not tested)
9 - VGA
999 - SVGA (super VGA - see SVGANameMode below)

For example:

GraphDriver 999

15. SVGANameMode - select a BGI driver to use. This driver is searched for only if GraphDriver above is set to 999. Thus driver is assumed to be 800 by 600 in resolution but any driver with 4:3 aspect ratio will probably be o.k.. This driver is searched in the directory specified by BGIDriverPath above. For example if the driver you want to use is ATI.BGI and in mode 2:

SVGANameMode "ATI.2"

16. Libraries - String value. A list of libraries (neither file type, nor full path is required - only the name) to automatically load on restart, separated with commas (','). Libraries can also be loaded using the libs main menu, and/or will be automatically loaded when a data file needs them.

For example:

Libraries "TTL4LS,EPROM,GENERAL,LINEAR"

17. LibrariesPath - String Value. Full path where libraries from LoadLibByName should be searched for. Note libraries loaded automatically will be searched in all path dirs and specified by PATH env. variable.

18. FrameWidth - Integer value. Sets the border width for all windows.

19. Colours control. Below are all colours values to select. Each can be one of the following 8 colours:

0 - WHITE

1 - BLACK
2 - RED
3 - GREEN
4 - BLUE
5 - YELLOW
6 - CYAN
7 - MAGENTA

The colours are self explanatory and grouped into the following groups: Root controls colours of the back ground root window. Actv sets the active drawing window colours. Psv sets the passive window(s) colours. PopUp controls all pop ups like queries colours. Highlight sets the highlight objects colour. Below are the colour groups themselves.

```
;Root window - the background.
RootFrameolor       7
RootForeColor       3
RootBackColor       5
RootXorColor        6

;Pop up menus/queries.
PopUpFrameColor  2
PopUpForeColor   5
PopUpBackColor   5
PopUpXorColor       6
```

```
;The currently active file/window if any.
ActvFrameColor      2
ActvForeColor               5
ActvBackColor               4

;All other passive files/windows if any.
PsvFrameColor               3
PsvForeColor                5
PsvBackColor                4

;High light colour - selected items.
HighLightColor              2
```

Following is the primary system layer colours,the program has upto 45 layers available to the user with several of them preset, The numeric representation for these layers is given at the end of this manual.

```
LayerWire                   0
LayerBus                    5
LayerGate                   3
LayerIEEE           2
LayerPinFun         4
LayerPinNum                 4
LayerPinNam                 4
LayerRefDes         2
LayerAttr                   6
LayerDevice         7
LayerNotes          6
LayerNetNam                 3
LayerPin                    3
```

If this configuration file is not found in any of the path directories, a message is printed to the screen, and internal defaults will be used instead. This is NOT recommended - fix that as soon as you can, as the internal defaults are not guaranteed to be anything specific.

EEDraw.bnd

This file allows you to bind any of the program internal functions (as can be selected from any of the menus) to a specific key. Note some keys are hard-bound and can not be used to rebind function (such as arrow keys - see Functionality below). The format of the file is following. Empty lines, or lines begin with a semicolon (';') are ignored. Every line consists of three items: "FunctionName KeyName ScanCode" separated by white spaces. FunctionName are the fixed names this program supports (see blow). KeyName is the name of the key ("AltK" for example) this FunctionName will be bound to.

Up to 4 characters are allowed for KeyName, and this name will appear in the main menu which contains this function. ScanCode is the scan code of the bounded key. If the key is regular ascii code, this ascii code is the scan code. If it is extended code (such as F? keys are), add 256 to this value. All numbers should be in decimal. Currently available FunctionName(s) and their associated main menu (see Functionality section below for their description):

File menu:
- LoadEEDFile
- NewEEDFile
- SaveAllEED
- SaveEEDFile
- SaveEEDAsOld
- SaveEEDAsNew
- SaveNetList
- CloseEEDFile
- ClearAll
- Directory
- ChangeDir
- ShellToDos
- ExitEEdraw

Window menu:
- PopWindow
- PushWindow
- MoveWindow
- ResizeWindow
- FullSize
- ShowName
- MakeActive
- NameActive
- PanWindowLeft
- PanWindowRight
- PanWindowUp
- PanWindowDown

Display menu:
- ZoomOut
- ZoomIn
- ZoomReset
- RedrawAll

Libs menu:
- LoadLibrary
- LoadLibByName

FreeLibrary
ViewLibrary
DirLibrary
ChDirLibrary

Draw menu:
DrawLine
DrawBus
DrawConnect
DrawText
DrawLibItem
ChangeLayer
TglOneColor
SetOneColor

Modify menu:
ModifyMove
ModifyCopy
EditLibItem
ModifyDelete
ModifyUnDel
ModifyCut
ModifyPaste
ModifyDrop

Status menu:
Help
GetMemoryFree
GetZoomFactor
SetDrawText
SetTextSize
SetHVLines
SetSnapFactor
SetLineWidth

Other (not in any menu):
SameAgain

4. Functionality

This section describes the currently available function set, and main modes. Although most of the keys may be bound to any function as described in the Setup section (EEDraw.bnd description), some of them are hard-bound and should not be used. Note that in order for enhanced Keyboard scan codes to be used, such as F11 and F12, 'ATKeyboard' must be set to TRUE in eedraw.cfg file. See 'Mouse' and 'Joystick' in eedraw.cfg for mouse and joystick setup. Below are the "hard-bound"

keys:

A. All the numeric keypad (arrow keys, home, PhUp, End, PgDn). These keys are used to move to cursor (even if mouse exists).
B. All the numeric keys but 0 (used by shift-arrows).
C. Enter (or Return) key. this is the SELECT key.
D. Space bar. This is the ABORT key.
E. Tab. This is equivalent to MIDDLE button key.

The arrow keys are used to move the cursor (co-exist with mouse), and if shifted, move cursor faster. In addition to the cursor movement, three important keys are defined as well:

A. SELECT key, hard-bound to the Enter (or Return), left button on mouse, if such exists, and button 1 on the joystick, if exists.
B. ABORT key, hard-bound to the Space bar, right button on mouse if such exists, and button 2 on the joystick, if exists.
C. MIDDLE key, hard-bound to the Tab, middle button on mouse if such exists (or pressing both mouse buttons), and both buttons on the joystick. All the eedraw functions described below uses these hard-bound keys.

This version supports multi file editing via multi window system (running on regular DOS). Each time a file is being open, a new window is created for it. If window header is to be shown (see ShowName function below) the file name the window is associated with is displayed, in addition to its current drawing layer, and the status information. The status is shown on the right hand side of the title bar, displaying the current windows drawing modes and attributes. Below is a representation of the status elements:

File Mode	Line Drawing	Snap	Line Width	Single Colour

File Mode:
O)riginal - this file is exactly the same as first loaded in.
M)odified - the current file state should be saved. Data has been modified.

Line Drawing:
O)rthoganal - Lines only either vertical or horizontal
A)ngular - Lines can be drawn in any direction

Snap:
This gives a numeric display of the current snap radius

Line Width:
This shows the current line drawing width in use

Single Colour:

S)ingle - All the layers which are turned on are displayed in a single colour.

N)ormal - All the layers which are turned on are displayed in the native colour.

There is no limit on the number of windows/files that may be open at once, aside from DOS and memory limits. The libraries are being shared between all files, so reduce memory usage. This means that even if two (or more) files needs LINEAR.LIB it will be loaded once into memory. However, there is no way for the program to detect which file uses which library when they are being saved and it is assumed ALL libraries currently loaded are being used by all files.

Few type of interactors are defined to query the user:

A. Yes/No question - prompt the question, with Yes and No buttons. The SELECT key will select the answer. ABORT key is ignored. One may use the Y/N keys as well to signal Yes/No answer.

B. Continue question - prompt a message, and wait for SELECT on the continue button. ABORT key is ignored. One may use the C key to continue as well.

C. String question - prompt a message, and waits for string input. This facilitates a full line editor as follows (independent of key bindings): * Right and Left arrows will move right and left. Insert toggles Insert/Overwrite modes (toggles cursor shape). Delete deletes current character. * Backspace deletes character to the back. Home and End will move to beginning/end of string respectively. Esc clears the current string. Return will terminate editing.

D. List question - prompt with a list of items (files names for example). An item will be picked by the SELECT key if applicative (when viewing a directory nothing needs to be picked). ABORT key will usually abort the operation, but may have some special operation as well or may be ignored.

E. Menu(s) question. Two types of menus are used - top down from the mainmenu, and pop up on special occasions. Menus behave very much like List questions. SELECT key selects menu item while ABORT key usually abort the menu with no selection at all.

A. File menu:

LoadEEDFile - Prompts for a file name (List question), and reads it from the current directory, if SELECT key, abort if ABORT key. Note file type must be '.eed'. A new window is created for it while the user is prompted to set the window size (see ResizeWindow).

NewEEDFile - Creates a new drawing window not assosicated with any file. This is the way to create new drawings. The user is prompted to set the window size (see ResizeWindow).

SaveAllEED - Saves all unsaved drawings. This is probably useful before quitting eedraw.

SaveEEDFile - Saves currently edited file. This routine can be used only after a new file was saved at list once, or it was loaded. In both cases, it will be saved on the same name, after renaming old one to have '.bak' extension. Ask for verification (Yes/No question).

SaveEEDAsOld - Same as SaveEEDFile, but prompts with list of all '.eed' files in current directory, and saved by the name selected if SELECT key, or abort if ABORT key.

SaveEEDAsNew - Same as SaveEEDFile, but prompts for a new file using String question.

SaveNetList - Saves a net list for the active window. Default name is same as '.eed' but with '.net' type. Net list is generated by scanning the line and bus object in the drawing. A drawing should be designed ahead of time so the net list will be correct in two respects:

1. The net list can not identify individual pins in a BUS and will list them all as one (BUS) connection. Therefore bus lines should be made individually if this program is to make a full net list for it.
2. Since only lines and buses are scanned, library items connected directly connection will not be detected. An easy solution is to put a dummy line connection there so that connection will be scanned correctly. When the net list is generated the scanned lines/buses are highlighted so the remaining non scanned ones can be easily viewed.

CloseEEDFile - Closes a drawing window associated with a file. Note that non of the above saving functions close the window, and this function is the only way to close a window. Ask for verification if data modified (Yes/No question).

ClearAll - Clears all eedrawing data in a picked window. The user is requested to "click" on the window to clear. Ask for verification (Yes/No question).

Directory - Display content of current directory as in List question. both SELECT key and ABORT key can be used to exit this mode. Unlike LoadEEDFile, all files are displayed.

ChangeDir - Prompts with String question for a new directory. Old directory is provided to begin with.

ShellToDos - "Fork"s (I wish this was true...) to the DOS O.S. . Type 'exit' at the DOS prompt to return to EEDraw.

ExitEEdraw - Exits the Program. Ask for verification (Yes/No question) and issue a warning if not all files were saved.

B. Window menu:

PopWindow - Pops up a window the user picks. The window is being picked by "clicking" on it. ABORT key aborts this operation.

PushWindow - Pushes down a window the user picks. The window is being picked by "clicking" on it. ABORT key aborts this operation.
MoveWindow - Moves a window the user picks. The window is being picked by "clicking" on it. ABORT key aborts this operation. The picked window frame is popped up so current position is all visible. The cursor prompt changes to a box in the window size that the user needs to place as the new position.
ResizeWindow - Same as MoveWindow but for resizing the window.
FullSize - Blows up a window the user picks. The window is being picked by "clicking" on it. ABORT key aborts this operation. This make the window as big as possible in the current display device. Calling this function again on the window will recover its original size. This function may be activated on drawing windows only.

ShowName - A toggle to show/hide window name. The window name is simply the filename the window is associated with. Also displayed is the current status of the drawing window, for the status codes, see earlier in this manual.
MakeActive - All drawing operations are performed on the active window. If only one drawing window exists it is the active one. If more, exactly one of them is the active one. This function selects the active window by a user pick. The window is being picked by "clicking" on it. ABORT key aborts this operation.
NameActive - All drawing operations are performed on the active window. If only one drawing window exists it is the active one. If more, exactly one of them is the active one. This function selects the active window by name. A list query of all drawing window (file) names is being popped up and the selected becomes the active one.
PanWindowLeft - Pan the ACTIVE window a half window to the left.
PanWindowRight - Pan the ACTIVE window a half window to the right.
PanWindowUp - Pan the ACTIVE window a half window up.
PanWindowDown - Pan the ACTIVE window a half window down.

C. Display menu:
ZoomOut - Zoom out (center of screen remains at center) by factor of 2. Note that if you zoom out from regular zoom factor, text will disappear as it can not be shrinked. You can enable text from the Status menu manually.
ZoomIn - Zoom in (center of screen remains at center) by factor of 2. ZoomReset - Reset Zoom value to the regular zoom factor.
RedrawAll - Redraw all screen.

D. Libs menu:

LoadLibrary - Prompts for a file name (List question), and reads it from the library directory, if SELECT key, abort if ABORT key. Note file type must be '.lib'.
LoadLibByName - Loads one or several libraries (separated by commas) by providing their names. To load EPROM.LIB and LINEAR.LIB "eprom,linear" should be typed. Libraries are been

search in all directories as specified by the PATH environment variable.
FreeLibrary - Prompts with all loaded libraries with List question, and frees the selected one if SELECT key, abort if ABORT.
ViewLibrary - Prompts with all loaded libraries with List question, If a library has been selected via SELECT key, prompt with all parts defined in this library again with a List question. Selection of a part with SELECT key will display it. ABORT will go back to library List question, which can be aborted again via the ABORT key.
DirLibrary - Displays content of library directory library files (those with '.lib' extension) as in List question. Both SELECT key and ABORT key can be used to exit this mode.

ChDirLibrary - Prompts with String question for a new directory for library search. Old directory is provided to begin with. This directory is used for Load/DirLibrary routines above only. LoadLibByName and auto library loading upon eedraw file loading will search all directories as specified by the PATH environment variable.

E. Draw menu:
DrawLine - Enters drawing mode. If HVLines is set (see SETUP, EEDraw.cfg section), only horizontal/vertical lines are allowed. In this mode, pressing SELECT key will insert a new point in the created polyline and pressing ABORT will delete the last entered point. If however, there are no points at all in the current polyline, ABORT key will abort this mode. Also, pressing SELECT twice (at the SAME place as the last point entered) will complete the drawn polyline. MIDDLE key will also complete the line drawing.
DrawWire - Same functionality as above, but forces the drawing layer to the 'Wire' layer, wether it is enabled or not. Once the line is entered, the current layer is returned.
DrawBus - Same as DrawLine, but draw thicker lines.
DrawConnect - Draw a connection junction, if SELECT key is pressed, or abort if ABORT key is pressed.
DrawText - Prompts for a string with String question, and then position the text horizontally or vertically. If the positioning is terminated with ABORT it is ignored, SELECT will update.

DrawLibItem - Prompts for a part name with String question. If this is not the first invocation to this mode, last part will be used as default string (remember you can always use Esc to clear to a fresh line). If part is found (name is exactly the same as shown in the ViewLibrary function above), a pop up menu will allow to rotate it 90 degree CW or CCW or mirror it. Successive SELECT will transform the library item to the desired orientation, while ABORT will abort this DrawLibItem mode. SELECT PlaceIt (top menu entry) will allow to position the part in the exact place. The user is also being prompted for Chip (i.e. 74LS00) and Part (i.e. IC7) names if they are required/allowed (see Appendix on library file definition) for this part. These are positioned similar to the way DrawText position its text, except that Chip name is already known and can not be modified.
ChangeLayer - Prompts the user to either change the mode, name or colour of an already defined layer, else add a new layer to the data base. The first menu presented to the user is a list of all the currently defined layers, also two extra entries 'Quit' and 'NewLayer', this selection list works like all other long lists in EEDraw, if there are more entries than screen space the list can be scrolled

up, by using the scroll bar on the right hand side of the list. If a currently defined layer is selected the user is presented with a sub menu to change the layer settings, These changes are: Display mode, Display Colour, Layer Name, and SetUsable, the last option is a quick method of setting a selected layer both usable for drawing, and active.

If 'NewLayer' is selected the user is lead through a series of system prompts to setup a new layer, this process is straight forward, at any point in either layer create, or modify the ABORT function will take the user back one menu level.

TglOneColor - This function will take all the displayable layers to one colour, which is defined with the next function. All primary layer control is retained, where only layers which are turned on or usable are displayed.

SetOneColor - Prompts the user to select the 'OneColor' display colour, care should be taken not to set this colour to any which is unusable!

F. Modify menu:

This menu is heavily based on Picking objects. a single object is being picked if the cursor is on an object. The cursor is considered on an object if one of the following:

A. Line and Bus items: the cursor is on one of the Line/Bus end points (any point that was set by SELECT when it was created, including internal points).
B. Connect item: the cursor is on the Connect center.
C. Text item: the cursor is inside the smallest axes parallel bounding box holding this text.
D. Library item: the cursor is on any polyline end point (including internal points).

The picked object will blink. SELECT key will select it, ABORT key will cause the next object, if such exists at this position, to blink until one is selected via the SELECT key, or no more objects can be picked.

If no object at all is found at the cursor point, the point is used as first corner of a box to pick a set of objects - a second SELECT key will pick the second corner of the box, ABORT key will abort this mode. All objects intersect with the defined box, will be considered picked, with no more verification.

ModifyMove - Picks object(s), and moves it (them) to a new position. Once the object(s) is (are) picked, SELECT key will set its(their) new position, ABORT will leave it unchanged.
ModifyCopy - Picks object(s), and copies it (them) to a new position. Once the object(s) is (are) picked, SELECT key will place it(them) in new position, ABORT will abort the copy operation.
EditLibItem - if the picked object(s) is a single library item, allows editing any of the following (via a pop up menu):

A. Editing the part name. This is the only way to add a part name to a library item if the library item has none by default.
B. Toggle to Draw/Undraw chip name (if this chip is allowed to have one).
C. Reposition Chip name (if this chip have one).
D. Reposition Part name (if this chip is allowed to have one).
E. Reorient the lib part.

Picking one of the above via SELECT will allow text positioning or text editing (via String question). ABORT or selecting Done (first menu entry) will abort this EditLibItem mode.

ModifyDelete - Deletes the picked object(s). The deleted object(s) is (are) saved in a special 10 places deep stack, and deleted items can be undeleted using the ModifyUnDel function. Each of the 10 placed stack can hold any result of a pick no matter how many objects are inside. This Delete/Undelete mechanism can be used to cut and paste even between different files/windows - this stack is never being cleared.
ModifyUnDel - Undelete the last delete operation (see ModifyDelete), from the delete stack (LIFO).

ModifyCut - Same as ModifyDelete, but original is NOT deleted. The object is being pushed onto the undelete stack and can then be ModifyUnDel, ModifyPaste etc.
ModifyPaste - Same as ModifyUnDel, but the undelete stack is NOT modified. This function enables pasting and placing the same object few times.

ModifyDrop - Drops the top of stack from the undelete stack. May be used to drop a cut/paste object sequence from stack.

G. Status menu:

Help - Pops up an help window and displays the content of eedraw.hlp. If the help window is already popped up, this function will close it (acts like a toggle.).

GetMemoryFree - Prompts with a continue question, on the current free memory. Getting close to zero is unsafe!

GetZoomFactor - Allows you to see the current zoom factor

SetDrawText - Sets drawing of text smaller than minimal size (8x8). This will draw text at 8x8 size which is too big, but may be sometimes useful.

SetTextSize - Sets the scaling (up only) of the text size. By default this is equal to 1, but scaling up by 7 is allowed.

SetAutoPan - If TRUE, auto panning will occur when placing objects that have been moved off the active WINDOW. The object will be place so it is centered in the window.

SetHVLines - Allows drawing of diagonal lines as well if set to FALSE. Uses Yes/No question.

SetSnapFactor - Sets the invisible grid the cursor position is snapped to. You probably do not want to change this, especially not in the middle of file editing. Uses String question, and display current Snap factor to begin with.

SetLineWidth - Sets the current line width, at the moment there is only widths '1' and '3' available, this should change in the next release.

H. Other (not in any menu):

SameAgain - Executes the last function again. This is hard bounded to MIDDLE key, although in may be bounded to another key as well.

5. Spacial Notes

Designing using A4 page size does not require much memory even when few libraries are loaded. My system with 640k (-70k of the O.S) has almost 300k free to use (Use the GetMemoryFree function to query free memory), when all the provided libraries are loaded. I guess you will need to work hard to get down to less than 100k. If you succeed and/or you system has less than 640k memory, make yourself a rule: never to go below 64k of free memory. The program need about 32k of temporary memory for the different operations, and getting close to that is dangerous! When you approach 32k of free memory, you will be issued a warning (Continue question): Free memory too small - dangerous to continue. Every 30 second (if you do any operation in that time). Continuing at this point may result with loss of all your work - the program will exit when no more memory will be available! This is especially true if 'SaveBackMethod' in eedraw.cfg is set to conventional memory. Yes, I know this is ugly, but you eat what you cook, right? I used this program quite a bit, and never succeeded to go below 200k, so why bather.

6. Acknowledgement

I would like to thank skh@eng.sun.com (Steve Howell) for his great comments during the last stages of bringing version 2.0 up.

Appendix A. Glossary

SELECT Key - The key used for the select operation. Currently hard-bound to Enter (or Return), the Left mouse button if mouse exists, and to Button 1 in a joystick.

ABORT Key - The key used for the abort operation. Currently hard-bound to Space bar, the Right mouse button if mouse exists and to Button 2 in a joystick.

MIDDLE key - Another general purpose key. Currently hard-bound to Tab key, the Middle mouse button if mouse exists and has middle button (or both right and left buttons pressed simultaneously) and to both Buttons of the joystick.

Screen Space - the space of the display device. Depends on the graphic device coordinates on the selected mode.

Drawing Space - the space in which the drawing objects exist. This space is bounded by a wide box (zoom out twice to see this box on an empty drawing) so it will fit into an A4 (8.5" by 11") page in both the EPSON and the PostScript printer drivers (see Appendix B).

Appendix B. Printer drivers and general support tools

The following printer drivers are available (see also installation section):

1. EPSON printer.

Usage: EEDEpson [-g] [-r] [-1] [-2] [-3] [-d] [-z] FileName[.EED]

Options: [-g] Output goes to standard output as 1 bit per pixel (B&W) GIF file.
[-r] Output goes to standard output as raw data.
[-1], [-2], [-3] Output goes to LPT1:, LPT2:, LPT3: respectively.
[-d] If output goes to Epson printer - print in double the density.
[-z] Print version number and usage.

* if neither [-g], not [-r] are specified, output is in Epson compatible format (default).
* If non of [-1], [-2], [-3] are specified, output goes to standard output.
* [-1], [-2], [-3], [-d] are ignored if [-g] or [-r] are specified.

2. PostScript Printer.

Usage: EED-PS [-f FontName] [-z] FileName[.EED]

Options: [-z] Print version number and usage.
[-f FontName] Specify the font name to use instead of the default name (Times-Roman).

PostScript output goes to stdout. If a PostScript Printer is connected to com2: serial port then 'eed-ps drawing.eed > com2:' will print drawing.eed on that printer.

3. Converter.

Usage: CONV [-l] FileName.EXT

Options: [-l] Convert the old style library files into version 2.4 files,

This program is used to convert old style data files into the version 2.4 format, adding layer information where needed. The layers are selected on a best guess basis, hence the user may need to edit the data files to change the default information. Unlike other EEDraw programs the entire filename plus extension needs to be passed into the convertor program, as it can take in both '.EED' files and '.LIB' files. The convertor is called automatically from the LoadEEDFile menu item when an old style file is called.

Appendix C. Library file format

This appendix describes the format of the library file. It contains all the information required to create your own libraries. The libraries are designed to be parametric only. I.e. no bitmaps are supported. You can look at the given libraries while reading this appendix for better understanding.

Empty lines and lines that starts with '#' are ignored. In order to identify a library as such the first line must start with 'EEDRAW-LIB'. Following in the same line is version number as 'Version X.X'. Currently 'X.X' equal '1.0' and is ignored. However it may be used in future versions, to ensure backward compatibility. Following are the parts themselves. Each part definition begins as follows:

DEF Name Prefix #Pins TextInside DrawNum #Units #PinsPerUnit

Were

1. Name is the Part name, i.e. "74LS00". If Name is prefixed with "~" (for example "~74LS00") then the name is not drawn, nor it will be accessible to editing by the user.
2. Prefix is the prefix to be used in chip name (IC, R, C etc.). If prefix is "~", then no prefix is drawn.
3. #Pins is the same as number of entries in PINS (see below), i.e 14 for 74LS00.
4. TextInside is 0 if pin text is to be put outside as well (as the pin numbers are), otherwise inside. You may put any positive integer there to scale the distance inside. 10 is a good start.
5. DrawNum is 1 if pin numbers are to be drawn, 0 otherwise.
6. #Units holds number of multiple entries. For example 74LS00 has 4 units (Nand gates). For no multiple units put 0 here.
7. #PinsPerUnit holds number of entries for one unit if more than one unit per part. If #Units is 0, it is ignored.

The part definition follows by the following blocks, in any order:

1. DRAW/ENDDRAW - defines how to draw the part. This block is optional, and a simple box with half the pins on each side is drawn if this block is not defined. The following commands are supported:

A Ly x y r t1 t2 (Define ARC at x, y, radius r, angles t1 to t2)
P Ly n x y x y x y... [F] (Define a POLYLINE of (any) length n)
C Ly x y r (Define a CIRCLE at x, y, radius r)
S Ly x1 y1 x2 y2 (Define a square (Not filled))
T Ly x y h str (Define a text string at x, y (see below))
L Ly x1 y1 x2 y2 [I] (Define terminal line to part - see PINS)

Notes:

A. a text string (T) with be drawn horizontally if h is 1, vertically if 0. Also '~' characters will be replaced by spaces, or this provides a mechanism to have spaces in the string.
B. If the polyline (P) is postfixed by a F, the polyline should define a closed shape that will be filled. The first point of the polyline must be identical to last one in this case.
C. If the line (L) is postfixed by an I, this line is assumed to define negative logic, and a small circle will be drawn to express that.
D. The number of L definitions should be exactly the same as #Pins in the DEFS line if #Units = 0, and equal #PinsPerUnit if #Units != 0. The order will match to the order in the PINS/ENDPINS block if #Units = 0 and to the order in the MULTI/ENDMULTI if #Units != 0.
E. Arcs (A) should not exceed 180 degrees, are CCW, and 0 degrees is at 3 o'clock.
F. The 'Ly' (Second field on each line) is the layer designation field, Later in the appendix is the description of what each layer does.

2. PINS/ENDPINS - defines the names (strings) associated with each pin. This block must always exists, and it is the minimum required to define a part. The block holds #Pins lines each hold the string defines that pin. This order must be the same as define in the DRAW/ENDDRAW as L commands (if DRAW/ENDDRAW block is defined), from pin 1 to #Pins. Empty lines are allowed and defined with only '~' in them. Not sign for X (vertical bar above it) is defined by ~X.

3. MULTI/ENDMULTI - This block must exists if #Unit != 0. Each line in this provides a mapping from L terminal definitions in the DRAW/ENDDRAW block to the pin numbers in a multi unit part. This is required as same L terminal will have different pin number in different Units. Each line in this block holds the pin numbers of a unit. Number of lines should be equal to number units in part.

Part definition terminates with the ENDDEF line.

General Notes:

1. All parts are scaled up to easy the access of a single pin (by 16, but you should not wonder about it too much - its internal mapping to drawing space).
2. Library file can hold any number of parts (at least one though...).
3. Library file must have '.lib' extension, to be usable by the EED* programs.

Appendix D Layers.

Following is a description of what each of the primary system layers is used for, In the current version of the library files layers like the IEEE layer are never used, it is a good idea for the user to stick to the basic layer conversion when creating new symbols for the libraries as some of the layers are called automatically by layer number in the program, hence there is no way for the user to change there position in the table. A good example of this is the 'Wire' layer, Layer '0', called by layer number in the 'DrawWire' menu entry. All of the layer table is saved in the '.EED' data file, so any new layers added, or the status of existing layers, is saved when the data file is saved.

Wire	-	Basic wire interconnections
Bus	-	Buses (Groups of wires)
Gate	-	Basic outline of a device
IEEE	-	IEEE device outlines, (Not yet in libraries)
PinFun	-	Function attached to a particular device pin
PinNum	-	Pin Number on a device
PinNam	-	Name attached to a pin on a device
RefDes	-	Reference designator for a component
Attr	-	Attribute to be attached to a component
Device	-	Device type/name
Notes	-	User notes etc
NetNam	-	Wiring net name layer, needs changes to 'SaveNetList'
Pin	-	Device Pins/Connection Points

The layers are numbered from zero (Wire) to Twelve (Pin), when described in the database files and library files.

Gershon Elber + Peter Cooper
Email: gershon@cs.utah.edu pcc@uk.ac.york.minster