ee

COLLABORATORS			
	TITLE : ee		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 7, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ee

1.1 main

EE V0.9.2 (beta)

Source Editor and
Development Environment
for the E Programming Language

1.2 license

Use EE at your own risk. If yer careful, it won't chew off your arm. :)
If you use EE, you have to promise not to hold me legally liable (I'll
commiserate with you, try to fix the problem, even abide a little verbal
abuse; but hey, let's be realistic: whaddya want for free? :)

EE may not be distributed for a profit. If you plan to get rich from EE, I want a piece of it; so you must negotiate with me for my blessing. Fred Fish has explicit permission to include EE in his archives. Any distributers who provide a similar service, for a fee comparable to Fred Fish's, hereby have permission to distribute EE under these conditions.

EE may be included on CDROM collections such as those produced by Fred Fish and Aminet.

1.3 credits

MANY, MANY THANKS TO THE FOLLOWING INDIVIDUALS WHOSE HELP AND/OR SOFTWARE CONTRIBUTED GREATLY TO THE DEVELOPMENT OF EE:

Wouter van Oortmerssen - Amiga E, technical support, multitudinous ideas, awesome E modules and sources, and beta-testing EE! (I'm still convinced there's more than one of him. :-) :-)

Dave Higginson - Multiple Windows info (especially his mweg.e, Multi-Windows Eggsample, posted to the E mailing list), GadTools and miscellaneous gadget help, various on-line info, beta-testing EE, and an endless, never-boring supply of sideways faces! ;-)

Lionel Vintenat - tons of excellent suggestions, beta-testing EE, and impressively complete bug reports!! :-)

Gregor Goldbach - suggestions, bug reports, spurring me on relentlessly to "get that ARexx port working!", and the highly prized German translation of EE.guide! :)

Mark Langston - help resolving GadTools and miscellaneous gadget problems!

Nico Francois - ReqTools library!

David Larsson - KingCON!

Michael G. Binz - AProf!

Martin Korndorfer - Magic Menus!

1.4 introduction

Source Editor and
Development Environment
for the E Programming Language

This release fixes several nasty bugs and oversights present in the original distribution found in the E v3.0 demo distribution. Also added are case-sensitive Intuition menu command hotkeys, correct handling of dead keys, keyboard hilighting made more CED-like, speeded up display updating and search-replace, added option to turn on/off smart indent,

now working ARexx interface, repaired unlimited macros, and a few added functions.

EE is a multi-tasking, multi-windowing, OS 2.0 compliant, folding source editor, designed with E, and E programmers in mind. It has robust features, equally mouse and keyboard driven, which allow for quick editing and traversal. Naturally, it has many features which enhance the development of E programs, like compiling and running from within the editor, EC error reporting, relocating the cursor at the erroneous line, keyword/phrase lookup and completion, and more.

Main features:

- Completely written in E :-)
- Multi-windows with hotkey switching and window selection listview;
- Robust ARexx interface;
- Nice configuration GUI (thanks to Wouter for his easygui module!!);
- Custom assignment of any function to any qualifier/key combo;
- Customizable case-sensitive menu command hotkeys (a la Intuition menus);
- Unlimited macro recording and playback; macros may be called from within other macros;
- Custom fixed-width font support;
- Smart indent option;
- Choice of free-form cursor (goes anywhere on screen) .vs. stream-oriented cursor (adheres to text, wrapping around at end and beginning of lines);
- Option to open your own public screen; Default and Shanghai modes; Open/Visit/Fallback options;
- Fold on PROCs, OBJECTs, recognized comment, and hilighted block; folds can be Cut-n-Pasted;
- Configurable/editable command slots, as well as the ability to pick up hilighted text from the source for execution;
- Compile with auto-save, auto-backup, asynchronous execution, and other options;
- Single- and multi-line comment and uncomment;
- Keyword and phrase completion--comes with dictionary file of E function templates--just type the function name and complete it; you'll now have the E function template in your source; two convenient methods of completion available; custom dictionaries supported, as well as an unlimited number of dictionaries available simultaneously;
- Collect garbage on demand (or memory crisis); and other low-mem options;
- Optional bottom scrollbar for those who like to have that extra line of text at the bottom;
- Many small things that add to comfort and productivity that won't be mentioned for the sake of space and attention span :-)

1.5 requirements

System Requirements and Recommendations

- Amiga E (if you wanna edit/compile E programs. :-)
- At least v37 (Kickstart 2.04) of the Amiga operating system.
- At least 512Kb free ram, 1Mb+ to edit and compile larger projects.
- At least gadtools.library v37.

- At least reqtools.library v38.693.
- AmigaDOS commands TYPE and COPY, in your path or made resident.
- Logical device T: mounted (preferably in RAM:).

1.6 installation

Installation

The following files must exist in the specified directories subordinate to the directory where the EE executable is located. OS2.0+ programmers may recognize this location as "PROGDIR:". No ASSIGN ADDs or PATH ADDs are necessary (although it may be desirable to place the EE and PROGDIR:bin directories in your path).

- Properly configured settings file (defaults provided) in directory PROGDIR:prefs; (location configurable; see Preferences Files);
- EE keyword completion file in PROGDIR:prefs; (location configurable Completion Files);
- EE PrefsEditor program in PROGDIR;

If you uncompress EE and preserve the archived directory structure, all you need to do is install reqtools.library in your LIBS: directory, and the fonts to your FONTS: directory. Customization is possible, even desirable (see Customization). The included fonts E/11 and blerk/11 and /9 are highly recommended.

1.7 customization

EE Customization

Preferences Files
Completion Files
PrefsEditor Program
Viewing Key Assignments

1.8 preferences files

EE Preferences Files

EE Preferences files store configurable information needed by EE at startup.

Without a preferences file EE would be dead in the water. Rather than store a hunk of default info in the frequently used EE executable, I chose to put it in the less frequently used PrefsEditor program. If you need to create a new Preferences file this can be done quite simply via the PrefsEditor Program .

The title of this section implies that you can have and use more than one Preferences file, and this is so. There are four methods for selecting

```
a Preferences file, two active and two passive:

  - (active) a PREFS argument provided on the command line overrides the
    passive methods;
  - (passive) the current directory is searched for a file named
    EE.prefs;
  - (passive) the directory PROGDIR:prefs is searched for a file named
    EE.prefs;
  - (active) The "Load Prefs" function can be accessed after startup via
    the menu, a key assignment, and/or EE's ARexx port;  NOTE: the
    following settings are frozen at startup and are not changed by
    loading new Preferences file:  1) presence/absence of the bottom
    scroller, 2) public screen preferences (except for Default and
    Shanghai modes), and 3) foreground and background colors;

SEE ALSO:
   Command Template
   Load Prefs Function
   Assigning Keys
   ARexx Overview
```

1.9 completion files

```
                        EE Completion Files

  The file CompletionFiles exists in path PROGDIR:prefs.  This file
  consists of linefeed terminated strings.  It was created with EE and
  can be modified by EE.  The contents of this file contains the path and
  file names of the files that should be searched by the @(" Complete Block "  ↩
     link "Tools"}
  function.

  The default CompletionFiles file list points to the files
  PROGDIR:prefs/completions/E-Builtin-NoRtnVal and
  PROGDIR:prefs/completions/E-Builtin.

  The default completion files, like the default CompletionFiles file
  list consist of linefeed terminated strings.  The text of a completion
  file may say anything (as long as it's printable ascii).  The max
  allowable file size is virtually unlimited (i.e., only by AmigaDOS and
  how long you're willing to wait for a search).  The files' contents are
  stored on disk during runtime, not in memory.

  If you have already read the  Complete Block  function description, you
  know that the Complete Block function searches the Completion file(s)
  for the first matching occurrence of the hilighted text, and if found,
  replaces the hilighted block with the entire line of text where the
  match is found.  You may edit the default files, or interchange your
  own files.  You must understand, however, that the search is case
  sensitive, and that when hilighting your text you much chose enough
  text to make the string unique, else you may suffer massive confusion
  when EE insists on replacing the selection with an unexpected string.

  If it helps your understanding of how to use this function, EE uses the
  dos.library function Fgets() to read in the lines of text, and the E
```

function InStr() to check for a match.

With some thought, you can easily set up your file entries to be unique and flexible. As a sample, the two default completion files listed above contain the E builtin function templates. The former file contains the function templates without return values, the latter function templates with return-values. Functions with similar names are arranged internally from shorter to longer (e.g., Dispose() before DisposeLink(), String() before StringF()). Now, because of the way I arranged the default completion file list, if I hilight the word "OpenS", it is replaced with:

```
OpenS(width, height, depth, sflags, title, taglist)
```

but if I hilight the word "=OpenS", it is replace with:

```
sptr:=OpenS(width, height, depth, sflags, title, taglist)
```

Another idea is to give "names" to lines of source code. For example, a completion file entry could be the EasyRequestArgs() line from the EReq.e example program, with a comment tag on the end of it:

```
EasyRequestArgs(0,[20,0,0,body,gadgets],0,args) ->EZREQ
```

then when you complete "EZREQ", you get the source in it's place, and the tag is just a comment.

1.10 prefseditor

EE PrefsEditor Program

1.11 pe general

EE PrefsEditor: In General

The PrefsEditor program can be found in the directory PROGDIR:.

The PrefsEditor command template is 'FILE,PUBSCREEN', where FILE is the optional name of the Preferences file to edit, and PUBSCREEN is the optional name of the public screen to visit.

If FILE is not provided, default Preferences settings are used. In this case, the file is saved to T:EE.prefs if the 'Save' button is used to exit the program. A requester is posted to notify you of the creation of the temporary file. It is up to you to manually copy it to a suitable

location for salvage.

If PUBSCREEN is not provided, the Workbench screen is used.

Three exit actions are available.

- Choose 'Cancel' or the Close Window gadget to quit and discard changes.
- Choose 'Save' to write the changes to the file specified on the command line (or T:EE.prefs if no file is specified on the command line).
- Choose 'Use' if you invoked the Preferences editor from within EE and only wish to use the new settings for the current session. Regardless of the command-line and where the PrefsEditor was launched from, the preferences will be written to file T:EE.prefs.

NOTE: PrefsEditor is a stand-alone utility, so you may run it from the CLI. You can run PrefsEditor from the CLI (or other program-launching utility) without affecting a currently running instance of EE.

NOTE: if you invoke the PrefsEditor from within EE and exit with the 'Use' button, you can edit the current session's settings by invoking the PrefsEditor again from within EE. The changes will be written back to the temporary file. If afterwards you decide to save the temporary changes, you must manually copy T:EE.prefs to a suitable location. Don't confuse this tip with normal operation...this is just contingent upon you choosing 'Use' when you really meant 'Save'.

NOTE: to create a virgin preferences file, run PrefsEditor from the CLI with no args (well, PUBSCREEN is okay). PrefsEditor will then be started up using default settings for all fields, and NO KEY ASSIGNMENTS. You should choose "Save" at this point, then copy the file "T:EE.prefs" to a more permanent location. Then restart PrefsEditor, this time supplying the new preferences filename you just created, and edit your key assignments and other settings according to the info in this guide.

1.12 pe commands

PrefsEditor: Edit Commands

EE provides 5 slots for commands which can be executed from within the editor. Since the commands are executed via the exec.library function SystemTagList(), you may set these to any valid string which will be recognized by your configured Console (e.g., IO redirection).

1.13 pe compiler

EE PrefsEditor: Edit Compiler

Command

Enter the command line you would use to compile your project, omitting the project's filename. The filename will be supplied by EE. The default is "EC". Another example is "EC SYM QUIET SOURCE=", assuming

EC is in your path, in the current directory, aliased, or made resident(?).

You may specify the full pathname for EC.

Command is limited to 80 characters.

Save Before Compiling

Check this option if you want EE to save a modified project before invoking the compiler.

Report Unreferenced

Check this option if you want EE to display EC's reported "UNREFERENCED" identifiers.

1.14 pe global settings

PrefsEditor: Edit Global Settings

Due to the size of the GUI, this function has been replaced by a bank of buttons labeled "Checked Settings...", "Public Screen...", "Miscellaneous...", and "Ok".

Checked Settings GUI

FG<->BG

Check this if you want to swap the default colors FG=0, BG=1.

Bottom Scroller

Check this if you want EE to attach a horizontal scroller to the bottom border of its windows.

Find Case Sensitive

Check this if you want EE to start up in case-sensitive find mode.

Insert Mode

Check this if you want EE to start up in insert mode.

Backup Project

Check this if you want EE to backup your project file before saving a modified project. Backed up files have the string '.bak' appended to their names and are stored under the same directory as the project. WARNING: 1) use foresight in naming your valuable backup

files (i.e., don't name project 'foo.e' backup to 'foo.e.bak'; EE will overwrite it if this option is turned on); 2) leave enough room in your project filenames for the extension; your projects should be 26 characters (including extension) long at most; if editing an EE backup file, it's backup file will be, for example, 'foo.e.bak.bak'; this should be obvious.

This feature requires the AmigaDOS Copy command to be in your path, aliased, or resident.

No Fold When Loading

Don't fold a project upon loading it, even if has fold info.

WARNING: saving a project after loading in this fashion will lose all of it's fold info. Your only hope in such a case is to be using the Backup option, or be fortunate enough to have an ARexx script that will intelligently fold your project back up.

Free-form Cursor

Check this if you want EE to start up in Free-form Cursor mode. See Free-form Cursor Mode.

Justify Newline

Determines whether you want EE to smart-indent newlines.

Public Screen Settings GUI

Public Screen Name

Enter the name of the public screen that EE will open and/or visit. The default name is 'Workbench'.

Default

Check this if you want EE to become the default public screen at startup. When other programs call LockPubScreen(NIL), the Default Public Screen is locked. This option causes other applications to that ask for the Default Public Screen in this fashion to lock the EE screen. Many applications use this feature to decide which screen their windows will open on.

Shanghai

Check this if you want EE to become a Shanghai screen at startup. When other applications expect to open on the Workbench screen, this mode causes them open on EE's screen instead.

New

Check this if you want EE to attempt to open a new public screen at startup. The screen name is the one you specify in the Public Screen Name string gadget.

Visit

Check this if you want EE to visit the public screen specified in the Public Screen Name string gadget. If New is checked and EE fails to open the new public screen because a screen of that name is already open (e.g., by another EE), EE will attempt to visit that screen. If New is not checked and the named screen isn't already opened, then Visit will fail.

Fallback

Check this if you want EE to attempt to Fallback to the Workbench screen if New and/or Visit public screen operations fail. Note: these three options obviously take the precedence New, Visit, Fallback. If one is missing, the next lower-precedence option is used. If Fallback is missing and New and Visit fail, EE will abort.

Miscellaneous Settings GUI

Completions

The name of the Completion Files file list. (The file containing the names of files to be searched by the @(" Complete Block " link "Tools"} function. See also: Completion Files .

Indent

Enter the number of spaces to insert by an Indent operation.

Tab

Enter the number of spaces to expand tabs when loading a file. The default is 8.

Fold Extra Lines

Enter the number of blank lines to fold after a fold terminator. Useful for readability, i.e., if Fold Extra Lines is 1, the following unfolded source:

```
PROC foo()
  doFoo()
ENDPROC

PROC bar()
  doBar()
```

```
  ENDPROC

  PROC hooHah() IS $F00D
```

...folds to:

```
  >>PROC foo()
  >>PROC bar()
  PROC anotherServing() IS $F00D
```

There are two special values which affect folding of functions and objects: -1 and -2.

-1 will fold all but the ENDPROC line, so the return values will remain visible:

```
  PROC wutsup()
    lotsa_stuff()
  ENDPROC this,that,theOther
```

...becomes:

```
  >>PROC wutsup()
  ENDPROC this,that,theOther
```

With a setting of -1, objects' ENDOBJECT keyword are folded, and no trailing blank lines are folded.

-2 will fold all but the ENDPROC line, the same as -1. The difference is that objects' ENDOBJECT keyword plus one blank line will be folded.

NOTE: comments are NOT blank lines, and therefore will disable the Fold Extra Lines feature.

Window: L, T, H, W

Window Left, Top, Height, and Width at startup.

Console Spec

The con specification string to use for opening a console as output. For example: 'KCON:20/20/100/600/Mega Output Console/WAIT/AUTO/SCREEN' NOTE: the SCREEN parameter MUST always be supplied, and MUST always be last. The screen name should be left out since EE will automatically use the public screen name specified in the Public Screen Name string gadget (or the Workbench screen if EE 'falls back').

Font Name, Size

Just like it says. The font must be a black and white, fixed width, non-scaled font, else it will screw up. Obviously, some fonts will be better than others. The supplied E/11 and blerk/11 and 9 fonts are provided for your reading bliss. :-)

1.15 pe key assignments

PrefsEditor: Edit Key Assignments

The "Assign Keys" gui is a pair of list views, the left containing all the assignable functions. Click on a function to select it, the current key combination assignments will be displayed in the list view on the right.

After selecting a function with the mouse, you're in edit mode. You can assign the function to keys simply by typing key combinations. The key combination names will be added to the right list view as they are typed. Assignments may be removed by clicking with the mouse on the key assignment item in the right list view.

Special: well, maybe not so special...you can assign the 'deadkeys' (Alt-F thru Alt-K) by selecting 'Write Char', then pressing those key combinations. This is all that's required to get 'deadkey' accents in EE.

The small utility ListKeys has be provided to allow you to view what functions are bound to your keys.

1.16 listkeys

The ListKeys Utility

This is a small and very simple CLI utility that will read an EE Prefs file and display a list of keys that have functions bound to them.

Keys are listed as one of the combinations:

```
      NONE keyname  functionname
     SHIFT keyname  functionname
      CTRL keyname  functionname
       ALT keyname  functionname
  RCOMMAND keyname  functionname
  LCOMMAND keyname  functionname
```

where keyname is the alphanumeric name of the Vanilla ASCII code, and functionname is the name of the function as they appear in the AssignKeys gui of the PrefsEditor.

The command template is 'FILE/A,HEX/S,SHOWALL/S' where:

```
  FILE     name of the Preferences file to examine;
  HEX      display the hexadecimal key value (default is alphanumeric);
  SHOWALL  force display of keys bound to function "Write Char" (default
           is NOT to show these bindings, which are usually numerous);
```

1.17 pe menu command hotkeys

PrefsEditor: Edit Menu Command Hotkeys

Very straightforward. Enter the character you want intuition to display in the function's menu item. This character is bound to the Right-Amiga Command key and invokes the function when that key combination is pressed. Since Intuition intercepts these key codes before they are passed as Rawkeys, a character entered here overrides any assignment of this key combination via the Key Assignments Editor .
Conflicts between commands assigned here and command assigned via the Key Assignments Editor are not prevented
by this interface.

Conflicts between Menu Items are, however, prevented. Which is to say, you can't enter an "N" for menu items "New Window" and "Next Window"; the latter attempted assignment will cause a requester to appear, and the assignment will be ignored (even though the value appears to remain in the gadget.)

EE's menus are CASE-SENSITIVE. =) For instance, I like RCommand-e for 'Compile', and Rcommand-E for 'Compile and Run'.

1.18 functions by name

EE Functions by Name

Backspace
Backup Project
Begin Macro
Beginning Of Line

Cancel Block
Cancel Macro
Clear
Clear Macros
Command 1
Command 2
Command 3
Command 4
Command 5
Comment
Compile
Compile and Run
Complete Block
Copy Block
Copy Block To Find String
Copy Block To Replace String
Cut Block

Delete
Delete Left
Delete Line
Delete Right
Delete Word Left
Delete Word Right

```
Down
Duplicate Line

Edit Command 1
Edit Command 2
Edit Command 3
Edit Command 4
Edit Command 5
Edit Compile String
End Macro
End Of Line
Execute Macro

Find
Find Backward
Find Case Sensitivity
Find Next
Find Previous
Find Selected
Fold/Unfold

Goto Last Error
Goto Line

Ignore Fold Info When Loading
Indent
Insert Mode

Join Line
Jump Left
Jump Right
Jump To Bottom
Jump To Top
Justify Newline

Left
Load Macros
Load Prefs

Make Default Public Screen
Make Shanghai Public Screen
Mark Block

New Window
Next Window

Open
Open Line
Open New Window

Page Down
Page Up
Paste Block
Previous Window

Quit
```

```
Replace
Replace All
Replace Backward
Replace Next
Replace Previous
Replace Selected
Right

Save
Save As
Save Before Compiling
Save Macros
Scroll Down
Scroll Up
Set Tab Width
Set Trailing Blank Lines to Fold
Show Editor Info
Split Line

Tall Window
To Lower
To Upper
Top Of View

Undo
Uncomment
Up

Word Left
Word Right
Write Char

Zip Window
```

1.19 functions by class

```
    EE Functions by Class

        Editor Operations:
     Command Line
     Project
     Traversal
     Find
     Edit
     Settings
     Tools
     Block Editing Functions
     Mouse Operations

        Customization
     Preferences Files
     Completion Files
     PrefsEditor Program
```

1.20 command line

EE Command Line

EE's Command Line has been tailored to provide adequate dynamic control at startup, but also to avoid command line argument overkill.

```
Template                FILE/M,L=LEFT/N,T=TOP/N,W=WIDTH/N,H=HEIGHT/N,
                        TAB/N,INDENT/N,SWAP=SWAPCOLORS/S,NOFOLD/S,
                        VISIT=VISITPUB/K, NEW=NEWPUB/K,FALLBACK/S,
                        FN=FONTNAME,FS=FONTSIZE/N,
                        PREFS/K,HORIZ=HORIZSCROLLER/S,?=HELP/S
```

Breakdown

FILE	Name of file to load (multiple allowed)
L=LEFT	Window leftedge (default 0)
T=TOP	Window topedge (default 0)
W=WIDTH	Window width (default screen width)
H=HEIGHT	Window height (default screen height)
TAB	Tab width (default 8)
INDENT	Indent width (default 2)
SWAP=SWAPCOLORS	Swap fg/bg colors (default 0/1)
NOFOLD	Ignore fold info when loading file
VISIT=VISITPUB	Visit named public screen, takes string argument PubScreenName (case sensitive)
NEW=NEWPUB	Open named public screen, takes string argument PubScreenName (case sensitive)
FALLBACK	Fall back to Workbench if Visit or New fail
FN=FONTNAME	Specify fontname (e.g., E.font)
FS=FONTSIZE	Specify fontsize (MUST accompany fontname!)
PREFS	Get Preferences from path/filename
HORIZ=HORIZSCROLLER	Attach horizontal scroller to bottom border
?=HELP	Display command template

1.21 project

EE Project Management Functions

Function Name	Description
Clear*	Abandon current window's text, but do not close window.
Open*	Abandon current window's text and request another file to load.
Open New*	Open a new window and request a file to load. See note 1 for a tip.
New Window*	Open a new window. See note 1 for a tip.
Save*	Save current window's text to the file named in the

	window's title bar.
SaveAs*	Save current window's text to a file different than the file named in the window's title bar.
Quit*	Abandon current window's text and close the window.

* Function accessible via menu.

Note 1: when a new window is opened, it inherits the current path from the window from which it was launched. You can take advantage of this feature by switching to a window whose path is nearest to the directory where you want to load or create your file. If you have monster paths on your harddrive like me, you'll appreciate not having to bash the file requester to get to distant places from the startup directory all the time. ;-)

1.22 traversal

EE Traversal Functions

Function Name	Description
Left	Move cursor left one character.
Right	Move cursor right one character.
Word Left	Move cursor left one word.
Word Right	Move cursor right one word.
Jump Left	Shift text right by one half screen width.
Jump Right	Shift text left by one half screen width.
Beginning Of Line	Move cursor to beginning of current line. Repeated invocation toggles cursor between leftmost non-blank character and column 1.
End Of Line	Move cursor immediately after last character in current line.
Down	Move cursor down one line.
Up	Move cursor up one line.
Scroll Down	Scroll text down one line.
Scroll Up	Scroll text up one line.
Page Down	Scroll text up one full page.

Page Up	Scroll text down one full page.
Jump To Bottom	Go to last page of text.
Jump To Top	Go to first page of text.
Goto Line*	Go to specific line number.
Goto Last Error*	Go to line number at which last compilation error occurred. (If the erroneous line is within a fold, this function positions the cursor at the fold header. By repeatedly unfolding and invoking this function you will eventually arrive at the erroneous line.)

* Function accessible via menu.

1.23 find

EE Find Functions

Function Name	Description
Find Selected*	Pick up text in currently hilighted block and search forward for next occurrence.
Find* **	Request text and search forward for next occurrence.
Find Backward* **	Request text and search backward for previous occurrence.
Find Next* **	Continue most recent forward search.
Find Prev* **	Continue most recent backward search.
Replace Selected*	Pick up text in currently hilighted block and request replacement text; replace currently hilighted text and search forward for next occurrence. Note: there is no function by this name. This action is inherent to 'Replace'.
Replace*	Request search-text and replacement-text, search forward for next occurrence of search-text, replace it, then search again. This function will pick up hilighted text and use it as search-text.
Replace Backwards*	Request search-text and replacement-text, search backwards for previous occurrence of search-text, replace it, then search again. This function will pick up hilighted text and use it as search-text.

Replace Next* ***	Replace search-text under cursor with replacement-text, then search for next occurrence.
Replace Prev* ***	Replace search-text under cursor with replacement-text, then search for previous occurrence.
Replace All*	From current cursor location to end of text, replace all occurrences of search-text with replacement-text. WARNING - This is potentially a VERY dangerous function to bind to a macro key. There is no verification, *BANG!* your whole file is chewed, quite irretrievably.
Copy Block To Find String	Pick up text in currently hilighted block and place it in the search-text buffer for use in the next search operation.
Copy Block To Replace String	Pick up text in currently hilighted block and place it in the replacement-text buffer for use in the next replace operation.
Case Sensitivity*	Toggle between search case-sensitive and case-insensitive.

* Function accessible via menu.

** Function will extend (or retract) a hilighted block.

*** A successful "Replace", "Replace Backwards", "Find", "Find Backwards", or "Find Selected" must precede this operation. You must have valid search-text in the buffer, else the find will fail and nothing will be replaced. Also, the cursor must be at the beginning of the search-text, else the function only performs a find (safety feature).

1.24 edit

EE Editing Functions

Function Name	Description
Write Char	Normally self-explanatory, but when a block of text is selected, the character typed is searched for (forward), and the block is extended up to and including the typed character.
Open Line	Move cursor down one line and insert a blank line. Justify cursor with leftmost character of previous line.

Delete Line	Delete current line.
Dupe Line*	Duplicate current line and insert it immediately after.
Split Line	Split current line at cursor location and insert the splinter immediately after current line; justify splinter with leftmost character of parent line, leaving cursor at that location.
Join Line	Join current line with next line, leaving cursor at location of the joint.
Indent	Similar in functionality to tabs, simply inserts number of spaces required to position cursor at next (multiple of IndentWidth)+1.
Back Space	Move cursor left and delete character under cursor. When cursor is in column 1, this function moves the cursor up one line and performs a "Join Line".
Comment*	Comment a hilighted block of text.
Uncomment*	Comment a hilighted block of text.
Delete	Delete character under cursor. When cursor is beyond end of current line, this function performs a "Join Line".
Delete Word Left	Deletes all characters left of cursor until a beginning of word is encountered. If the cursor encroaches the top 1/3 of the view, the view is automatically scrolled.
Delete Word Right	Deletes all characters from character under cursor until an end of word is encountered. If the cursor encroaches the bottom 1.3 of the view, the view is automatically scrolled.
Delete Left	Deletes all characters left of cursor to beginning of current line. If cursor is in column 1, the current line is joined to the previously line.
Delete Right	Deletes all characters from character under cursor to end of current line.
To Upper*	Changes case of a single character to upper. If the character being converted is the last character in the line, the cursor jumps to the next non-blank character. If a block of text is hilighted, the entire block of text is converted to uppercase.
To Lower*	Changes case of a single character to lower. If the character being converted is the last character in the line, the cursor jumps to the next non-blank character. If a block of text is hilighted, the entire block of text is converted to lowercase.

* Function accessible via menu.

1.25 settings

EE Settings Functions
(Configuring EE from Within the Editor)

Function Name	Description
Insert Mode*	Toggle INSERT/OVERSTRIKE mode. Checked menu item signifies INSERT mode.
Save Before Compiling*	Toggle SAVE/NOSAVE mode. Checked menu item signifies SAVE FILE (if modified) before invoking compiler.
Backup Project*	Toggle BACKUP/NOBACKUP mode. Checked menu item signifies BACKUP mode for <file> to <file>.bak before saving.
Report Unreferenced*	Toggle REPORT/SILENCE mode of EC warnings about unreferenced variables. Checked menu item signifies REPORT mode.
No Fold When Loading*	Toggle REGARD/DISREGARD fold info when loading. Checked menu item signifies REGARD mode.
Default Pub Screen*	Make EE screen default public screen. Checked menu item signifies that EE screen IS the default public screen. When other programs call LockPubScreen(NIL), the Default Public Screen is locked. This option causes other applications to that ask for the Default Public Screen in this fashion to lock the EE screen.
Shanghai Pub Screen*	Make EE screen a shanghai public screen. Checked menu item signifies that EE screen IS a shanghai public screen. When other applications expect to open on the Workbench screen, this mode causes them to open on EE's screen instead.
Free-form Cursor*	Toggle FREE-FORM/STREAM mode. Checked menu item signifies FREE-FORM (page) mode. This affects the behavior of the cursor. STREAM mode restricts the cursor to the region of extent text, and wraps around at beginning of line and end of line. When scrolling, if the cursor comes to rest beyond end of line, it is yanked to the end-of-line position for that line. FREE-FORM mode allows the cursor to roam wherever a

	character may exist, even out to position 1024, and even if there is no text out that far.
Fast File Loading*	Fast file loading is twice as fast as normal file loading. It is really only noticable when loading large files. Normal loading is very memory efficient, while fast loading requires memory more than twice the size of the file: storage for the file buffer, plus storage for the editor data structures and line buffers. The buffer is freed when loading is complete. NOTE: you cannot use fast loading for devices which store compressed data and decompress the data when it is read (XPK, EPU, CompressDisk) unless the device can report the "virtual" size of the file. Otherwise, a partial load will result.
Justify Newline	Toggles smart-indentation of new lines.
Fold Extra Lines*	Set number of blank lines to fold after keywords ENDPROC and ENDOBJECT. NOTE: comments are NOT blank lines, and therefore will disable this feature. For an explanation of the special values -1 and -2, see Global Settings .
Indent Width*	Set indent width.
Tab Width*	Set tab width (NOTE: Tab Width only matters when loading files).
Select Font*	Invoke fixed-font requester.
Load Prefs*	Load key bindings from file.
Save Prefs*	Save current key bindings to file.
Execute Macro*	Execute the macro that is bound to a particular key sequence.
Begin Macro*	Create a new macro. You will be prompted to Press the key sequence that will invoke playback of the macro; then perform the operations you want to record in the macro. Invoke End Macro to terminate macro recording. WARNING - A potentially VERY dangerous function to bind to a macro key is the "Replace All" function. There is no verification, *BANG!* your whole file will be chewed, quite irretrievably.
End Macro*	Terminate macro recording and make macro available for playback.
Cancel Macro*	Abort the macro recording that is in progress.
Save Macros*	Save all macros in the environment to a file.
Load Macros*	Load macros into environment from a file. The

	environment is cleared of any macros prior to loading.
Clear Macros*	Clear all macros from the environment.
Edit Command*	This menu item has five subitems, each of which invokes a string requester for its command. The assignable function names are numbered 1-5. The new values are used for the current editing session only. To permanently change the values, use the PrefsEditor .

* Function accessible via menu.

1.26 tools

EE Tools Functions

Function Name	Description
Fold/Unfold*	The folding function has two ways of determining the boundaries of a fold: 1) fold the currently hilighted lines of text; 2) fold the text existing between the current line and the next forward occurring end-of-fold keyword. Keywords MUST begin in column 1. Keyword recognition is as follows: 1) if current line begins with "PROC" or "EXPORT PROC", text is folded until next occurrence of "ENDPROC", or end of text; 2) if current line begins with "OBJECT" or "EXPORT OBJECT", text is folded until next occurrence of "ENDOBJECT", or end of text; 3) lastly, text is folded from current line to next occurrence of "->endfold", or end of text. This keyword string MUST appear EXACTLY as it appears here, or it won't be recognized. A fold is indicated by a "fold header", a line flagged by the prefix ">>". Enfolded text is hidden from searching, but fold headers can be searched for. A fold header(s) can be copied and pasted all or in part, but it is otherwise protected against being edited. A copied fold header does not take with it a copy of the folded text; the header is pasted with the prefix changed to "->". An entire fold(s) can be cut and pasted one time; all enfolded text is moved with the "fold header".

Subsequent pastings only paste the header with the prefix changed to "->"; the enfolded text is not duplicated.

Folds may be nested as deeply as desired.

Folds are saved as an ascii comment at the end of a source file so as not to clutter the source when viewed with other tools. Safeguards exist to handle damaged fold info, but results are unpredictable should this info become damaged. If damage occurs you can use the command line option NOFOLD to tell EE to ignore this info. This option is also available via menu, and assignable via the key binding utility.

Next Window* — Bring next EE window to front.

Previous Window* — Bring previous EE window to front.

Goto Window* — Popup listview to choose a window to bring to front.

Tall Window* — Toggle between normal window size and full height of visible screen. (This means if the screen is pulled down halfway, the window will stretch only half the screen's height.)

Complete Block* — Search the Completion file list for a match of the hilighted text, and replace the hilighted text with its completion. See Block Editing Functions and Completion Files .

Compile* — Use the compile command string to invoke the E compiler for the file in the current window.

Goto Last Error* — Go to the line containing the error from the last compiler invocation. If the line is enfolded, the cursor is placed at the fold header containing the line. You can unfold text and invoke this command repeatedly until the offending line is reached.

Execute* — Open a console window and execute one of the five command strings. If a block of text is hilighted, the hilighted text is picked up and used by the function. Commands are always displayed in a string requester so you can edit them before proceeding. Changes to a command are not retained unless they are changed via the Edit Commands function.

Show Editor Info* — Show some pertinent information: EE Arexx port name, EE public screen name.

* Function accessible via menu.

1.27 block editing functions

EE Block Editing Functions

Function Name	Description
Mark Block*	Begin hilighting a block of text. If a block is already being marked, the block is turned off. Blocks can be marked via clicking and dragging mouse.
Adjust Block	Any editor traversal function will adjust a block including Finds. The combination Shift-key+ Click-and-Drag mouse will also adjust a block. Mouse and keyboard manipulation maybe be inter-changed indefinitely. Typing a character will extend a block forward up to and including the next occurrence of the typed character (see function Write Char).
Cancel Block	Invoke function Mark Block a second time, or invoke the function Cancel Block, which is assignable via the key binding utility.
Cut Block*	Extract hilighted text and store temporarily in a "cut buffer".
Copy Block*	Copy hilighted text and store temporarily in a "cut buffer".
Paste Block*	Insert contents of "cut buffer" into text.
Copy Block To Find String	Pick up text in currently hilighted block and place it in the search-text buffer for use in the next search operation.
Copy Block To Replace String	Pick up text in currently hilighted block and place it in the replacement-text buffer for use in the next replace operation.
Auto Cut	Hilighted block of text is automatically cut when an editing function (Backspace, Delete, Paste, etc.) is invoked. Block can be retrieved using the Undo function. See note below.
Undo*	See note below.
Complete Block*	Complete Block Usage Completion Files

* Function accessible via menu.

Note: functions Auto Cut and Undo are related in the way of Cut Block and Paste Block. A major consideration is that separate storage is used for each of these two classes of clippings. Consequently, you can Region Cut a region and Auto Cut (via the backspace key, or whatever) another region, and then Region Paste and Undo as often as desired... very handy in cases requiring mildly complex duplication of text. In addition, complex deletions (i.e., Delete Line, Delete Word Left, Delete Word Right, Delete Left, and Delete Right) are placed in the Undo buffer, and can therefore be Undone as long as no other Auto Cut operations have been performed. NOTE: Auto Cut does not work for the function Write Char .

1.28 mouse operations

These are all fairly intuitive,
but just in case...

Function Name	Description
Relocate Cursor	Click.
Page Up/Down	Click in groove above/below vertical scroller.
Scroll Up/Down	Click and drag vertical scroller, or click scroller gadgets. Also click and hold on top or bottom border of window.
Shift Left/Right	Click in groove left/right of horizontal scroller.
Scroll Left/Right	Click and drag horizontal scroller. Also click and hold on left or right border of window.
Select Block	Click and drag.
Select Word	Double click (Not yet implemented).
Select Line	Triple click (Not yet implemented).
Select With Scroll	Click in window and drag onto any of the four window borders.
Adjust Region End	Click and drag while holding shift key.
Zip Window	Quickly resize window by clicking on the ZOOM gadget. This function is also assignable via the key binding utility in the PrefsEditor.

1.29 contacting the author

Send email to:

el269@cleveland.freenet.edu

Send snail mail to:

Barry Wills
5528D Pryor Dr.
SAFB, IL 62225 USA

1.30 arexx functions

(INCOMPLETE - Still need to update changes and describe arguments/return values ↩ .)

EE ARexx Functions

OVERVIEW

COMMAND NAME		ABBREV	
BackSpace		BACKS	
Backup Project		BACKU	(Toggle)
BeginMacro	2!	BEGINM	
BeginningOfLine		BEGINN	
CancelBlock		CANC	
Clear	1,2!	CLEA	
Cmd1	1!	CMD1	
Cmd2	1!	CMD2	
Cmd3	1!	CMD3	
Cmd4	1!	CMD4	
Cmd5	1!	CMD5	
Comment	1!	COMM	
Compile	1!	COMPILE	
CompileAndRun		COMPILEA	
CompleteBlock	2!	COMPLETE	
CopyBlock	2!	COPYBLOC	
CopyToFindString	2!	COPYTOFI	
CopyToReplaceString	2!	COPYTORE	
CursorDown		CURSORDO	
CursorLeft		CURSORLE	
CursorRight		CURSORRI	
CursorUp		CURSORUP	
CutBlock	2!	CUTB	
DeleteChar		DELETECH	
DeleteLeft		DELETELE	
DeleteLine		DELETELI	
DeleteRight		DELETERI	
DeleteWordLeft		DELETEWORDL	
DeleteWordRight		DELETEWORDR	

```
DuplicateLine                DUPE

EditPrefs                    EDIT
EndMacro                  2! ENDM
EndOfLine                    ENDO
ExecuteMacro              2! EXEC

Find                   1,2! FIND          ARG1=str 3!
FindBackward           1,2! FINDB         ARG1=str 3!
FindCaseSensitivity    1,2! FINDC         (Toggle)
FindNext               1,2! FINDN
FindPrevious           1,2! FINDP
FindSelected           1,2! FINDS
Fold                        FOLD
FoldExtraLines         1,3! FOLDE

GetChar                      GETC          RETURN [1]:CHAR
GetString                 4! GETS          ARG1=len, RETURN [len]:CHAR
GetWord                      GETW          RETURN [?]:CHAR
GotoColumn                4! GOTOCO        ARG1=col
GotoLastError                GOTOLA
GotoLine                  2! GOTOLI
GotoBottom                   GOTOBO
GotoTop                      GOTOTO

Indent                       INDE
InsertMode                   INSE          (Toggle)

JoinLine                     JOIN
JumpLeft                     JUMPL
JumpRight                    JUMPR
JustifyNewline               JUST          (Toggle)

LoadPrefs              1,2! LOAD          ARG1=name 3!
LockWindow                2! LOCK

MakeDefaultPublicScreen      MAKED         (Toggle)
MakeShanghaiPublicScreen     MAKES         (Toggle)
MarkBlock                    MARK
MoveWindow                3! MOVE

NewWindow                    NEWW
NextWindow                   NEXT
NoFoldWhenLoading         3! NOFO          (Toggle)

Open                      1! OPEN
OpenLine                     OPENL
OpenNew                   1! OPENN

PageDown                     PAGED
PageUp                       PAGEU
PasteBlock                   PAST
PreviousWindow               PREV
PutChar                   3! PUTC
PutLine                   3! PUTL
PutString                 3! PUTS
```

```
Quit                          QUIT
QuitAll                    3! QUITA

Replace                 1,2! REPLACE
ReplaceAll              1,2! REPLACEA
ReplaceBackward         1,2! REPLACEB
ReplaceNext             1,2! REPLACEN
ReplacePrevious         1,2! REPLACEP

Save                    1,2! SAVE
SaveAs                  1,2! SAVEA
SaveBeforeCompiling          SAVEB         (Toggle)
ScrollDown                   SCROLLDO
ScrollUp                     SCROLLUP
SetCmd1                    1! SETCMD1
SetCmd2                    1! SETCMD2
SetCmd3                    1! SETCMD3
SetCmd4                    1! SETCMD4
SetCmd5                    1! SETCMD5
SetCompileCmd              1! SETCOMP
SetIndentWidth             1! SETI
SetTabWidth                1! SETTA
ShowEditorInfo               SHOW
SizeWindow                 3! SIZE
SplitLine                    SPLI

TallWindow                   TALL
ToLower                      TOLO
ToUpper                      TOUP
TopOfView                    TOPO

Undo                         UNDO
Uncomment                  2! UNCO
UnlockWindow                 UNLO

WordLeft                     WORDL
WordRight                    WORDR

ZipWindow                    ZIPW

?Column                      ?COL          RETURN int
?DefaultPublicScreen         ?DEF          RETURN bool
?Filename                  3! ?FIL          RETURN str
?FindCase                    ?FIN          RETURN bool
?Folded                    3! ?FOLDED       RETURN bool
?FoldExtraLines              ?FOLDEX       RETURN bool
?IndentWidth                 ?IND          RETURN int
?InsertMode                  ?INS          RETURN bool
?JustifyNewline              ?JUS          RETURN bool
?Line                        ?LINE         RETURN int
?LineCount                   ?LINEC        RETURN int
?Modified                  3! ?MOD          RETURN bool
?NoFoldWhenLoading         3! ?NOF          RETURN bool
?PathAndFilename           3! ?PAT          RETURN str
?PubScreenName             3! ?PUB          RETURN str
?ShanghaiPublicScreen        ?SHA          RETURN bool
?TabWidth                    ?TAB          RETURN int
```

```
?TallWindow                    ?TAL          RETURN bool

Footnotes:
1!  Still need to circumvent requesters
2!  Need to add code to return warnings
3!  Not yet implemented
4!  Need bounds checking on ARGn
```

1.31 arexx overview

```
                        EE Arexx Overview


BRIEF

  A valid ARexx script skeleton is:

    /* ARexx script skeleton: */
    ADDRESS EE.0
    OPTIONS RESULTS
    'lockwindow'
    /* ARexx code and EE commands... */
    'unlockwindow'

  This sets up a rendezvous with instance 0 of EE, locks the active
  window, performs actions, then unlocks the window.  A virtually
  unlimited number of instances of EE may run concurrently, named EE.0,
  EE.1, EE.2, ...  You can find the port name of an instance of EE by
  selecting the 'Show Editor Info' menu item in the 'Tools' menu.

  EE ARexx commands are parsed using the dos.library function ReadArgs().
  The general command template is 'CMD/A,ARG1,ARG2,ARG3,REP=REPEAT/K/N'.
  Commands may be abbreviated, may take arguments (including a repetition
  specifier), and may return a Result String, all in accordance with
  their descriptions in section  ARexx Functions .


RENDEZVOUS WITH EE's AREXX PORT

  EE's ARexx port name follows the naming convention EE.x, where the name
  extension 'x' is a number starting with 0 and continuing to the maximum
  integer the computer will support.  Each instance of EE will take the
  lowest available digit as it's name extension.  You must establish
  contact with an EE ARexx port, by using the following line in your
  script:

    ADDRESS EE.x

  where, once again, 'x' is the number which corresponds to the instance
  of EE.  You can find EE's ARexx port name by selecting the 'Show Editor
  Info' menu item in the 'Tools' menu.


INITIATING AN AREXX COMMAND SESSION
```

An ARexx command session must be initiated by issuing the command 'lockwindow'. This tells EE to hold the active window, ignoring all events except those coming from it's ARexx port. If your script has locked the active window, then issues a window-related command ('newwindow', 'opennew', 'nextwindow', 'previouswindow', or 'closewindow') the lock is transferred to the succeeding window. (NOTE: AutoPoint and other SunMouse-type utilities automatically change EE's active window, and hence EE's active project. This is not surprising. Therefore, you must take care not to inadvertently activate another window with the mouse if switching to the shell to execute your ARexx script. Locking the window precludes any such problems during script execution, but it provides no protection prior to locking the window!)

(*** Maybe I should add a menu item to lock the current window.)

TERMINATING AN AREXX COMMAND SESSION

An ARexx command session must be terminated by issuing the command 'unlockwindow'. This tells EE to release the active window and return to normal event processing. NOTE: the 'unlockwindow' command does not necessarily have to be issued in the same script as the 'lockwindow' command. You could very well have one script to lock, one to unlock, and a collection of scripts that just perform actions on an already locked window.

ABOUT EE AREXX COMMANDS

Commands are not case sensitive. The maximum command length supported by EE is 512 bytes (512 characters).

The significant length of a command (it's abbreviation) is determined by the number of letters required to make a command name unique. This is a side-effect of the method used to parse the command name. It's your choice to take advantage of the abbreviations. But note well that in a future release if ARexx commands are added whose abbreviations clash with existing abbreviations, your scripts may become obsolete because of the need to lengthen an abbreviation. Something like this may happen if you use the full name, anyway, but it is not as likely.

Commands may or may not take arguments. The number and types are indicated for each command listed in the section ARexx Functions . EE's command parser is the dos.library function ReadArgs(), so it should be no surprise that commands take on the form of a CLI command-line.

The general command template is 'CMD/A,ARG1,ARG2,ARG3,REP=REPEAT/K/N', where: CMD is always required; ARG1, ARG2, and ARG3 are required only in the context of a particular command; and REPEAT is the number of times to repeat the command, e.g., 'cursorright REP=20'.

As in CLI command-lines, you may optionally specify the formal name of the argument, e.g., 'cmd=writestring rep=5 arg1="ha"'.

Note that the template dictates that the keyword REP or REPEAT must

always accompany the repetition argument. the REP argument is honoured by almost every command, but is obviously only useful with certain commands. For example, it makes sense to use REP with 'cursorleft', but not with 'gotoline' (how many times do you want to go to that line, ay? :-) REP is treated as 1 if omitted. REP less than 1 is usually not permitted, and is treated as 1. Some special commands allow REP=0. This is indicated in the section ARexx Functions .

Some commands return a Result String. This is indicated in the section ARexx Functions .