Table of Contents

Preface − Acknowledgements **1**

General Information **2**

- Editor Philosophy 3
- Features 3
- Cursor Positioning 4
  - Enter Key Behaviour 5
  - PF Display Line 5
  - Popup Command Line and Command Stack 5
- Installation 6
  - OS/2 Installation 6
  - Windows NT/95 Installation 7
  - DOS Installation 7
  - Linux Installation 8
  - Linux390 Installation 8
  - AIX Installation 9
  - Sun Solaris Installation 9
  - HP−UX Installation 10
- Invoking The Editor 10
  - File Specification 11
- Performance 12
  - Editor Performance Comparison − File Load 13
  - Quit 13
- Popup Windows 13
  - Ring Contents List 15
  - File Functions List 15
  - User Defined Popup Window 15
- File Margins 15
- Comment Formatting 16
  - Inline Comments 16
  - Block Comments 18
  - Comment Manipulation 18
- Highlighting 19
  - Comment Highlighting 19
  - Keyword Highlighting 19
  - Cursor Line Highlighting 20
- Splitting Text 20
  - Auto−Flow 20
- Compiler Errors 20
- Hidden Lines 21
- Saved File Information 22
- Marking Text 22
- Recorded Key Sequences 23
- Automatic Binary File Detection 23
- Editor Settings 23
  - BROWSE 24
  - EA 24
  - HEX 24

Table of Contents

INSMODE....24
LINEND....24
MSGMODE....24
SHADOW....25
SPAN....25
STATUS....25
SYNTAX....25
WRAP....25

Tutorial....26
Screen Areas....26
Sample Edit Session....28
Basic Navigation....28
Marking....29
The Command Line....30
Hidden Lines....30

Default Key Assignments....32
Unshifted Keys....32
Alphanumeric Keys....32
Function Keys....32
Special Character Keys....33
Special Keys....35
Shifted Keys....35
Alphanumeric Keys....35
Function Keys....35
Special Character Keys....36
Control Keys....37
Alphanumeric Keys....37
Function Keys....39
Special Character Keys....39
Alternate Keys....40
Alphanumeric Keys....40
Function Keys....43
Special Character Keys....43

User Profile....46
Creating The User Profile....47
Comments....48
Key Remapping....48
User Profile Key Remap....48
Colour Remapping....53
Colour Remapping....53
X−Windows Colour Remapping....56
Strings....57
Synonyms....58
Bracket Matching Characters....59
Initial Editor Settings....60

Table of Contents

Newline Character....60
Cursor Size....61
Saving Editor Information....61
Enter Key Behaviour....61
Insert Mode....62
Linend Setting....62
Popup Window Scrolling....62
Quick Bookmark Setting....62
Status Line....63
Automatic Bookmarks....63
Multiple Bookmarks....63
Command Line Location....64
Command Stack Window Size....64
Default Extension....64
Default List....65
Escape Character....65
Filename Completion Threshold....65
Linend Character....65
Null Character....66
OpenFile Paths....66
Quit Response When File Modified....66
Right Alt (AltGr) Key....66
Shell Prompt String....67
Beep Behaviour....67
X−Windows Font....67
Disk Specific Customisation....67
User Profile Disk Customisation....67
File Extension Specific Customisation....68
Default Extension....69
Inline Comment Formatting Control....69
Code Functions List....70
Syntax Expansion....74
Conditional Strings....75
Customising the OpenFile Function....76
Style Formatting....76
User Profile Extension Customisation....77

Commands and Macro Support....85
Macro Debugging....89
EXTRACT Command....89
Extract Options....90
Locate Text....94
Change Text....95
Popup Windows....97
List Box....97
Message Box....98
Prompt....99
Password Prompt....99

Table of Contents

Editor Commands....99
ACCENT....99
ADD....100
ALL....100
ALT....101
APPEND....101
ASCII....102
AUTOBOOKMARK....102
AUTOSAVE....102
BACKSPACE....103
BACKTAB....103
BACKWARD....104
BOOKMARK....104
BOTTOM....105
BOTTOMSCREEN....105
BROWSE....105
C, CHANGE....106
CASECHAR....106
CASEWORD....106
CD....107
CENTRELINE....107
CENTRETEXT....107
CHANGES....108
CLIP....108
CMDLINE....109
CMDTEXT....109
COMMAND....109
COMMENTLINE....110
COMMENT_STYLE....110
COMPARE....111
CONDITIONAL....111
COPYLINE....111
COPYTOCMD....112
COUNT....112
CURR_ALT_PFLINE....113
CURR_CTRL_PFLINE....113
CURR_PFLINE....113
CURR_SHIFT_PFLINE....114
CURSOR....114
DATE....115
DELCHAR....116
DELDUPES....116
DELETE....117
DELSYM....117
DELWORD....117
DIAG....118
DOWN....118
DUPLICATES....119

Table of Contents

E, EDIT, X....120
EA....120
EOF_TEXT....120
ERASEEOL....121
ERRORS....121
EXCLUDE....121
EXITRC....122
EXPAND....123
EXT....123
EXTRACT....123
FFILE....124
FIELDTEMPLATE....124
FILE....125
FIND_WORD....125
FORWARD....126
FT....126
FUNCWIN....126
GET....127
HELP....127
HEX....127
HIDEFILE....128
INPUT....128
INPUT_ERRORLINE....129
INSMODE....129
JOIN....129
KEY....130
KEYIN....130
KEYIN_NAME....131
KEYS_PLAY, PLAYBACK....131
KEYS_RECORD....131
KEYS_WRITE....132
L, LOCATE....132
LINECOLOUR....132
LINEFIELDS....133
LINEMACRO....133
LINEND....134
MA, MARGINS....135
MACRO....135
MARK....136
MATCH....137
MESSAGEBOX....138
MSG....138
MSGMODE....139
NAME....139
NEXT, NEXT_FILE....140
NEXT_ERROR....140
NEXT_FUNC....140
NEXT_PARA....141

Table of Contents

NEXT_SENTENCE....141
NEXT_SYM....141
NEXT_WORD....142
NOP....142
NUMFILES....143
OPENFILE....143
PAGEDOWN....143
PAGEUP....144
PASSWORD....144
PFLINE....144
PRESSKEY....145
PREVIOUS_FILE....145
PREVIOUS_FUNC....146
PREVIOUS_PARA....146
PREVIOUS_SYM....146
PREVIOUS_WORD....147
PROMPT....147
PUT....148
QQ, QQUIT....148
QUIT....148
REDO....149
REFORMAT....149
REFRESH....150
RENAME....150
REPEAT_FIND, REPFIND....151
REPLACE....151
RESOLVE_FN....152
RESTORE_FIND....152
REVERSE_FIND....152
RINGWIN....153
SAVE....153
SCROLL....154
SETRESULT....155
SHADOW....155
SHADOWTEXT....155
SHELL....156
SHOW....156
SHOWLINE....157
SORT....157
SPAN....158
SPLIT....158
SPLITJOIN....158
STATUS....159
STATUSTEXT....159
STYLE....160
SYNTAX....160
TAB....160
TABLINE....161

Table of Contents

TABS....161
TIMER....162
TITLE....162
TOFEOF....163
TOF_TEXT....163
TOP....163
TOPLINE....164
TOPSCREEN....164
UNDO....164
UNDO_BLOCK....165
UNDO_LIMIT....165
UP....166
WINDOW....166
WINLINE....167
WINSELECT....168
WINSORT....168
WINWAIT....168
WRAP....169
nnn....169
/text</< |/& /text2/>>....170
Command Summary....170
Command Summary (A−H)....170
Command Summary (I−P)....172
Command Summary (Q−Z)....174

Hexadecimal Mode Considerations....176

Editor Differences Between Operating Systems....178
Differences in the Windows NT/95 Version....178
Differences in the DOS Version....178
Differences in the Unix X−Windows Versions....179
Differences in the Linux Curses Version....179

Appendix A. Rexx Program to Measure Editor Load Times....181

Appendix B. Sample Macro to Create a Popup Window....183

Appendix C. Sample Profile for EOS2 Users....185

Preface − Acknowledgements

The X2 Editor was derived from **Tim Baldwin**'s XE sample editor. Extensive modifications have been made to the functionality and internal workings of the editor, but the author is indebted to Tim for sharing the original source code.

In functionality X2 derives from the E family of editors, E3 and EOS2 in particular. The authors of these editors are **Clark Maurer**, **Bryan Lewis**, **Jean Christophe Bandini**, **Richard Redpath**, **Davis Foulger**, and **Larry Margolis**. E in turn was influenced by Personal Editor, by **Jim Wyllie**.

X2 also includes some features from VM's XEDIT editor. XEDIT was originally written by **Xavier de Lamberterie**.

The Unix versions make use of X−Windows routines, most of which are taken from **Scott Schaffer**'s VE editor.

There have been dozens of people who have contributed ideas, macros, encouragement, and of course bug reports. I can't name you all, but I am very grateful for the interest you have shown in this project.

The author may be contacted through the Internet at bwt@interlog.com, through IBM's internal VM system on BLAIR at IBMCA, or on Lotus Notes at Blair Thompson/Markham/IBM@IBMCA.

General Information

- Editor Philosophy
- Features
- Cursor Positioning
  - Enter Key Behaviour
  - PF Display Line
  - Popup Command Line and Command Stack
- Installation
  - OS/2 Installation
  - Windows NT/95 Installation
  - DOS Installation
  - Linux Installation
  - Linux390 Installation
  - AIX Installation
  - Sun Solaris Installation
  - HP−UX Installation
- Invoking The Editor
  - File Specification
- Performance
- Popup Windows
  - Ring Contents List
  - File Functions List
  - User Defined Popup Window
- File Margins
- Comment Formatting
  - Inline Comments
  - Block Comments
  - Comment Manipulation
- Highlighting
  - Comment Highlighting
  - Keyword Highlighting
  - Cursor Line Highlighting
- Splitting Text
  - Auto−Flow
- Compiler Errors
- Hidden Lines
- Saved File Information
- Marking Text
- Recorded Key Sequences
- Automatic Binary File Detection
- Editor Settings
  - BROWSE
  - EA
  - HEX
  - INSMODE
  - LINEND
  - MSGMODE
  - SHADOW
  - SPAN

◊ STATUS
◊ SYNTAX
◊ WRAP

This section contains general information about the editor and the thoughts that went into building it. If you are new to the X2 Editor you may want to skip ahead to the Tutorial before reading this chapter.

Editor Philosophy

The X2 Editor was designed and built to enhance the process of producing source code. As a programmer's editor the most important considerations were performance and ease of use. Wherever possible, the editor has been written so that it will automatically do things when writing code. In order to do this, it must make certain assumptions about the format of the file being edited. Some of these assumptions can be tailored through the user profile, but others are imbedded in the editor. This may cut down on the flexibility of the interface, and some people may not agree with the shortcuts which the editor makes for the user. It is felt that this tradeoff is worthwhile for the productivity benefits that can be realised through such a design.

While the OS/2 Presentation Manager (PM) environment provides a useful graphical interface for most computer users, it does tend to use a large amount of the available computer cycles for doing nothing more than painting the screen. The author feels that productivity can be gained for serious programmers by utilising the full screen features of OS/2 and foregoing the nice windows and fonts available on the desktop. It is for the performance benefits that can be realised in such an environment that the X2 Editor was built. If you are not an editor "power user" or you like a full graphical environment, this editor is not your best choice.

The editor does **not** support interaction through a mouse interface, for several reasons:

◊ A mouse is best used in a graphical environment, where it can address individual pixels on the screen. The mouse resolution is very coarse in a VIO window.
◊ Mouse movement can be jerky and distracting in a text mode application.
◊ The mouse programming interfaces are not standardised between operating systems.
◊ It is felt that the impact to code size is not worth the possible benefits gained.

Features

The X2 Editor is an ASCII text editor suitable for editing any flat text file. There are versions available for the OS/2, DOS, Windows NT, Windows 95, AIX 4.1 & 4.2, Linux, and Solaris operating systems.

The OS/2, Windows 95, and Windows NT versions are 32 bit VIO (full screen) applications, which can be used either from a full screen session or from a VIO window. The Unix versions are X−Windows applications, although a fullscreen CURSES version is also available for Linux. In features the X2 Editor is most similar to the EOS2 editor, although it

has some important differences from that editor, including:

- ◊ The ability to selectively exclude and show file lines (folding)
- ◊ User definition of comment delimiter strings and keywords for highlighting on the display
- ◊ Rexx macro support
- ◊ The maximum line length has been increased to 50000 characters
- ◊ Display and edit (not insert/replace) of binary files
- ◊ X2 provides an unlimited Undo/Redo stack
- ◊ X2 provides several popup windows showing information about the edit session
- ◊ While EOS2 is limited to OS/2 systems, X2 provides portability between several popular operating systems
- ◊ X2 is easier to configure, through a single profile file
- ◊ Writing custom functions is done through a standard language (Rexx). This makes macros easier to write and understand, but more files must be copied to use the same editor on another PC.

For information on configuring the X2 Editor to behave similarly to the default EOS2 editor, consult Sample Profile for EOS2 Users.

This document describes the OS/2 version of the editor, except where noted. Details of the differences between the three versions may be found in Editor Differences Between Operating Systems.

Cursor Positioning

The cursor movement keys in the X2 Editor are defined to move the cursor where you are likely to want it to go, depending on the file contents. For example:

- ◊ The Enter key moves to the first non−blank character on the next line, unless the line is blank, when it lines up underneath the first non−blank character on the previous line. The exception to this rule is when the new line is blank, and the previous line begins or ends with a left brace ({) character. In this case, the cursor is indented by two spaces. The number of spaces is tailorable in the user profile, see User Profile Extension Customisation. If the beginning of the new line is identified as a comment, then the cursor will be moved to the first non−blank character of the comment itself (i.e. the comment identification string is bypassed).
- ◊ If the End key is pressed to move the cursor to the end of the line, and the cursor is already at the end of the line, it will be moved to the end of the next file line.
- ◊ Keys are defined to move to the beginning of the next and previous functions in the file. When moving the cursor to a function, the cursor is moved to the top of any comment block preceding the function. If screen scrolling is required or the new function is located in the bottom half of the screen, the top of the function will be positioned at the top of the screen. Otherwise, the cursor is just moved to the top of the function. Code Functions List details how a function is recognised.

Enter Key Behaviour

By default, the Enter key moves the cursor to the first non−blank character on the next file line. If the next line is blank, the cursor is positioned beneath the first non−blank of the current line. If the cursor is on the last visible line of the file, it will not be moved.

To insert a new line, press the *Ctrl−Enter* key. This will leave the current line unchanged, and a new line will be added beneath the current line. The cursor will be positioned on this new line, beneath the first non−blank character on the old current line. The contents of the new file line may vary. If the cursor starts on a line that looks like a block comment, then a new, empty block comment line will be inserted. Otherwise, the new line will be completely blank.

PF Display Line

The X2 Editor reserves the last line of the screen for the display of function key help. This information will dynamically change depending on user interactions − if the user presses and holds the Shift key, for example, the text will change to display the settings of the various shifted function keys. There are four possible lines that can be displayed, for Shifted, Alt, Ctrl, and unshifted states. These lines can be changed in the user profile.

Another feature of the PF display line is that it is used to indicate the CapsLock state of the keyboard. If CapsLock is ON, the PF line will contain only upper case letters. If CapsLock is OFF, the PF line reverts to the normal mixed case representation.

Popup Command Line and Command Stack

When the cursor is active in the file area, no command line is visible. When the cursor is moved to the command line with the Esc or Alt−Enter keys, the command line is written over the topmost visible file line on the screen. If commands have been issued previously, these are also displayed under the command line. These lines represent the *command stack*, which contains a maximum of 20 previous commands. They are saved in chronological order, the most recent command at the top of the stack. The default size of the command stack window is 10 lines, but all 20 commands can be scrolled with the cursor movement keys. As the commands are scrolled, the current (highlighted) command is copied to the command line. The command stack window size may be modified through the user profile. See Command Stack Window Size for details.

As a command is typed on the command line, the first few characters are compared against previous command stack entries. If a (case−insensitive) match is found, the current stack entry will be highlighted. This provides a convenient way to recall previous commands with only a few keystrokes.

The command stack is saved from session to session. It is saved on the XPATH (see Installation) as XCMDSTCK.DTA. Note that if the XPATH is undefined, the path to X.EXE will be used as the location for XCMDSTCK.DTA.

Stacked Commands

When entering commands on the command line, multiple commands may be issued by separating them with the *Linend* character. This character is defined as the caret (^). If the linend character is found on an input line, everything up to that character will be issued as a command, and subsequent text will be issued as another command when the first command has completed. The linend character may be turned on and off with the LINEND command, or modified through the profile (see Customising LINEND Character).

Installation

The X2 Editor is supported on various platforms: OS/2, Windows NT/95, DOS, Linux, Linux390, AIX 4.1, AIX 4.2, Sun Solaris SunOS 5.6, and HP−UX 10.2. The executables are delivered as **X2.ZIP**, **XWNT.ZIP**, **XDOS.ZIP**, **xlinux.tgz**, **xlin390.tgz**, **xaix41.tgz**, **xaix42.tgz**, **xsun_tar.Z**, and **xhp.tgz** respectively; you have to unpack the files yourself. The files are available from the World Wide Web (WWW) at address *http://blair.vsc.can.ibm.com* (internal to IBM), or the public site *http://www.interlog.com/~bwt*.

Useful macros are available on IBMPC in the X2 PROCS file, or in the xmacros.zip file on the download WWW page. Each macro has the file extension .X. Macros may be installed by copying selected files to a directory in your PATH. Macros must have file extension .X or the editor will not find them. Commands and Macro Support discusses how to write your own editor macros. Macro support is not available for the DOS version.

- ◊ OS/2 Installation
- ◊ Windows NT/95 Installation
- ◊ DOS Installation
- ◊ Linux Installation
- ◊ Linux390 Installation
- ◊ AIX Installation
- ◊ Sun Solaris Installation
- ◊ HP−UX Installation

OS/2 Installation

To use the X2 Editor on OS/2 follow these steps:

1. Download X2.ZIP to your hard disk in binary. Extract the editor files by executing **UNZIP X2.ZIP**, to generate *X.EXE*, *X2UTILS.DLL*, *X.INF*, *X.HLP*, *XAPIS.C*, *XPROFILE.EXE*, and *XPROFILE.DEF* in your current directory.
2. Copy *X.EXE* to a directory on your PATH.
3. Copy *X2UTILS.DLL* to a directory on your LIBPATH. This file contains useful utility functions, but is not necessary to run the editor.
4. **SET XPATH=path**. This statement can be added to your CONFIG.SYS.
5. If you wish to customise the editor, create a customisation file using *XPROFILE.DEF* as a guide.
6. Generate *X.PRO* by running **XPROFILE XPROFILE.DEF yourprof**, where *yourprof* may be omitted if you want the default configuration. Creating The User

Profile contains detailed profile customisation instructions.
7. Copy *X.PRO* to the XPATH.
8. Edit a file with **X filename**.
9. The online version of the help information may be displayed with **VIEW X.INF**.

Windows NT/95 Installation

The same executables can be used on either Windows NT or Windows 95. To use the X2 Editor on these systems follow these steps:

1. Download XWNT ZIPBIN to your hard disk as XWNT.ZIP, in binary. Extract the editor files by executing **UNZIP XWNT.ZIP**, to generate *XWNT.EXE*, *XWUTILS.DLL*, *X.HLP*, *XAPIS.C*, *XPROFWNT.EXE*, *XPROFILE.DEF*, and *XWNT.INF* in your current directory.
2. (Optionally) rename *XWNT.EXE* to X.EXE.
3. Copy *X.EXE* and *XWUTILS.DLL* to a directory on your PATH.
4. On Windows 95, **SET XPATH=path** in AUTOEXEC.BAT or from the command line. On Windows NT, select Settings from the Start menu, then Control Panel. Open the System control panel, and go to the Environment notebook page to set the XPATH variable.
5. If you wish to customise the editor, create a customisation file using *XPROFILE.DEF* as a guide.
6. Generate *XW32.PRO* by running **XPROFWNT XPROFILE.DEF yourprof**, where *yourprof* may be omitted if you want the default configuration. Creating The User Profile contains detailed profile customisation instructions.
7. Copy *XW32.PRO* to the XPATH.
8. Edit a file with **X filename**.
9. The online version of the help information may be displayed with **IVIEW XWNT.INF**. IVIEW comes with VisualAge C++ and probably some other IBM products, but cannot be distributed with X2 due to copyright restrictions.

If you want to run Rexx macros with the Windows NT version, you'll have to install Object Rexx, which is available as OBJREXXW PACKAGE on the PCTOOLS disk. It is also available externally, for a fee, from http://www2.hursley.ibm.com/orexx/orexx.htm. Alternatively, you may install the Regina Rexx interpreter, which is freely available from ftp://ftp.lightlink.com/pub/hessling/Regina/.

DOS Installation

To install the editor for DOS:

1. Unpack XDOS.ZIP with UNZIP: **UNZIP XDOS.ZIP**. This will generate *XDOS.EXE*, *XPROFDOS.EXE*, *XPROFILE.DEF*, and *X.HLP*.
2. Copy *XDOS.EXE* to a directory on your PATH
3. **SET XPATH=path**. This statement can be added to your AUTOEXEC.BAT.
4. If you wish to customise the editor, create a customisation file using *XPROFILE.DEF* as a guide.
5. Generate *XDOS.PRO* by running **XPROFDOS XPROFILE.DEF yourprof**, where *yourprof* may be omitted if you want the default configuration. Creating The User Profile contains detailed profile customisation instructions.

6. Copy *XDOS.PRO* to the XPATH
7. Edit a file with **XDOS filename**

Of course, *XDOS.EXE* may be renamed to anything you desire. Note that the profile may not be renamed; it must always be XDOS.PRO.

Linux Installation

To install the editor for Linux:

1. Copy xlinux.tgz to your Linux system in binary.
2. Unpack the files with gunzip and tar: **gunzip xlinux.tgz** followed by **tar −xvf xlinux**. This will generate the executable files *x*, *xx*, *xprofile*, and *xutils.so*, as well as *xprofile.def* and *xprofile.unx*, and some other utilities and help files.
3. Copy *x* and *xx* to a directory on your PATH
4. Set the **X2PATH** environment variable according to your shell.
5. Copy *xutils.so* to a directory on your **LD_LIBRARY_PATH**
6. If you wish to customise the editor, create a customisation file using *xprofile.def* as a guide.
7. Generate *XUNIX.PRO* by running **xprofile xprofile.def xprofile.unx yourprof**, where *yourprof* may be omitted if you want the default configuration. Creating The User Profile contains detailed profile customisation instructions.
8. Copy XUNIX.PRO to the X2PATH
9. Edit a file with **x filename**. If you wish to use the X−Windows version (recommended), use **xx filename** to edit a file.

The above installation may be simplified by use of the *x2install* utility that comes packaged with the rest of the files. If you decide to use x2install, make sure you read the code **before** running it so you understand what it will do.

The Linux version supports two versions of Rexx − Object Rexx and Regina. Rexx for Linux is available from http://service2.boulder.ibm.com/dl/rexx/orexxlinux−d or http://www2.hursley.ibm.com/orexx/orexx.htm. Alternatively, you may install the Regina Rexx interpreter, which is freely available from ftp://ftp.lightlink.com/pub/hessling/Regina/.

Linux390 Installation

To install the editor for Linux390:

1. Copy xlin390.tgz to your Linux390 system in binary.
2. Unpack the files with gunzip and tar: **gunzip xlin390.tgz** followed by **tar −xvf xlin390**. This will generate the executable files *xx*, *xprofile*, and *xutils.so*, as well as *xprofile.def* and *xprofile.unx*, and some other utilities and help files.
3. Copy *xx* to a directory on your PATH
4. Set the **X2PATH** environment variable according to your shell.
5. Copy *xutils.so* to a directory on your **LD_LIBRARY_PATH** and set symbolic links to it.
6. If you wish to customise the editor, create a customisation file using *xprofile.def* as a guide.
7. Generate *XUNIX.PRO* by running **xprofile xprofile.def xprofile.unx yourprof**,

where *yourprof* may be omitted if you want the default configuration. Creating The User Profile contains detailed profile customisation instructions.
8. Copy XUNIX.PRO to the X2PATH
9. Edit a file with **xx filename**.

The Linux390 version supports IBM Object Rexx.

AIX Installation

Note that the following instructions use the generic name *xaix.tgz* as a placeholder for either *xaix41.tgz* or *xaix42.tgz*. Please substitute the correct name depending on whether you are running AIX 4.1 or AIX 4.2 and above. To install the editor for AIX:

1. Copy xaix.tgz to your AIX system in binary.
2. Unpack the files with gunzip and tar: **gunzip xaix.tgz** followed by **tar −xvf xaix**. The 4.1 version will generate *xx*, *xxrexx*, *xaixutils.dll*, and *xprofile*, along with configuration files *xprofile.def* and *xprofile.unx*, a sample help file *xunix.hlp*, and a *README* file. The 4.2 version generates *xaixutils.so* instead of *xaixutils.dll*, and does not include a copy of *xxrexx* as Rexx support is dynamically loaded into the 4.2 version.
3. Copy *xx* and/or *xxrexx* to a directory on your PATH
4. Set the **X2PATH** environment variable according to your shell.
5. Copy *xaixutils.dll* or *xaixutils.so* to a directory on your LIBPATH
6. If you wish to customise the editor, create a customisation file using *xprofile.def* as a guide.
7. Generate *XUNIX.PRO* by running **xprofile xprofile.def xprofile.unx yourprof**, where *yourprof* may be omitted if you want the default configuration. Creating The User Profile contains detailed profile customisation instructions. You may want to specify a path to *xunix.hlp*, which is the help file specified in *xprofile.unx*.
8. Copy XUNIX.PRO to the X2PATH
9. Edit a file with **xx filename**. If you have Rexx installed on your 4.1 system, you may prefer to use **xxrexx filename** to edit files and run macros.

Sun Solaris Installation

To install the editor for Solaris SunOS 5.6:

1. Copy xsun_tar.Z to your Solaris system in binary.
2. Unpack the files with uncompress and tar: **uncompress xsun_tar.Z** followed by **tar −xvf xsun_tar**. This will generate *xx*, *xutils.so*, and *xprofile*, along with configuration files *xprofile.def* and *xprofile.unx*.
3. Copy *xx* to a directory on your PATH
4. Set the **X2PATH** environment variable according to your shell.
5. Copy *xutils.so* to a directory on your LD_LIBRARY_PATH
6. If you wish to customise the editor, create a customisation file using *xprofile.def* as a guide.
7. Generate *XUNIX.PRO* by running **xprofile xprofile.def xprofile.unx yourprof**, where *yourprof* may be omitted if you want the default configuration. Creating The User Profile contains detailed profile customisation instructions. You may want to specify a path to *xunix.hlp*, which is the help file specified in *xprofile.unx*.

8. Copy XUNIX.PRO to the X2PATH
9. Edit a file with **xx filename**. If you wish to use Rexx for macro support, you must install the Regina Rexx interpreter, which is available from ftp://ftp.lightlink.com/pub/hessling/Regina/.

HP−UX Installation

To install the editor for HP−UX:

1. Copy xhp.tgz to your HP−UX system in binary.
2. Unpack the files with gunzip and tar: **gunzip xhp.tgz** followed by **tar −xvf xhp**. This will generate the executable files *xx*, *xprofile*, and *libxutils.sl*, as well as *xprofile.def* and *xprofile.unx*, and some help files.
3. Copy *xx* to a directory on your PATH
4. Set the **X2PATH** environment variable according to your shell.
5. Copy *libxutils.sl* to a directory on your **SHLIB_PATH**.
6. If you wish to customise the editor, create a customisation file using *xprofile.def* as a guide.
7. Generate *XUNIX.PRO* by running **xprofile xprofile.def xprofile.unx yourprof**, where *yourprof* may be omitted if you want the default configuration. Creating The User Profile contains detailed profile customisation instructions.
8. Copy XUNIX.PRO to the X2PATH
9. Edit a file with **xx filename**. If you wish to use Rexx for macro support, you must install the Regina Rexx interpreter, which is available from ftp://ftp.lightlink.com/pub/hessling/Regina/.

Invoking The Editor

From the operating system command line, the editor is invoked with the following syntax:

```
<d:\path\>X <fn1 fn2... <−B> <−BIN> <−Ccmd>> <−ERR>
                        <−NOPROF> <−NOTABS> <−NOUNDO> <−Pprofname>
                        <−Q> <−S> <−T> <−TABS> <−TOP>
```

where:

d:\path
: The path to X.EXE. Not required if it is in the current directory or on the PATH.

fn1 fn2
: The file(s) to be loaded. If no files are specified, X2 loads a new, empty file with no name. If any are found, they are used for the current invocation; otherwise, a blank file is created. A status message is displayed for each file as it is loaded. If a file is greater than 500 kilobytes in size, the status message will be updated for every 10 percent of the file that has been processed. See File Specification for details on specifying a file name.

−B
: Load the file(s) in Browse mode. No changes to the file are permitted when it is being browsed. Browse mode is indicated by the text "Brw" on the status line instead of "Rep" when the cursor is in replace mode. Browse mode is turned on automatically when a file is open for read/write by another process, or when the

Read Only attribute bit is set for the file.

−*BIN*

Load the file in Hexadecimal view.

−*Ccmd*

Execute the given *cmd* after the file(s) have been loaded. This can be any command which can be issued from the command line, including macros. If parameters are required, the string may be enclosed in quotes, e.g. "−Ccmd parm1 parm2...".

−*ERR*

Load and parse a compiler error file *fn.ERR*.

−*NOPROF*

Load the file without checking for and loading *X.PRO*.

−*NOTABS*

Expand all embedded tab characters into blanks when the file is loaded.

−*NOUNDO*

Suppress the undo stack − no file changes will be saved for possible restoration.

−*Pprofname*

Override the default profile name with the specified profile. This switch overrides both the system default profile name and any profile name specified with the XPRO environment variable.

−*Q*

Quiet mode. All editor screen updates are withheld until the file(s) are loaded and any initial commands have been executed. If initial commands result in no file being loaded, the screen is not cleared upon exit.

−*S*

Search sub−directories. Any files from the supplied directory and all its sub−directories that match the supplied specification will be loaded into the editor. This option is only available in the OS/2 version.

−*T*

View tabs. No tab expansion will be performed for this file.

−*TABS*

View tabs. No tab expansion will be performed for this file.

−*TOP*

Initialise the cursor to the top line of the file.

When editing a file with the internal EDIT and X2 commands, the same options may be used, with the following differences and exceptions:

- ◊ If a file specified is already in the ring, a second copy will not be added to the ring. Instead, the original copy will be made the current file.
- ◊ If no filename is specified, the next file in the ring will be made the current file.
- ◊ The −NOPROF and −P options will be ignored.

Options begin with a dash (−) for Unix compatibility. On non−Unix systems, a slash (/) may be used in place of the dash.

File Specification

Files are loaded either as parameters to X.EXE, or as parameters on the EDIT command. Several shortcut keys are available when specifying file names:

◊ Any occurrence of an asterisk (*) wildcard character in the filename or extension will be expanded to match any sequence of characters in that position.
◊ Any occurrence of a question mark (?) wildcard character in the filename or extension will match any single character in that position.
◊ If the path is replaced with an equals sign (=), the path is assumed to be the same path as the currently edited file.
◊ If the filename is replaced with an equals sign, the filename is taken from the currently edited file.
◊ If the file extension is replaced with an equals sign, the extension is taken from the currently edited file.
◊ A check is made for the dollar sign ($), and if found a check for an environment variable with the same name is made. If the environment variable is found it will be substituted within the filename; otherwise, it will remain unmodified.

The following table contains examples of file name resolution. The currently edited file is **C:\MYDIR\MYFILE.TST**. An environment variable named X2PATH is set to **C:\TOOLS\X2**.

Input	Resolved Name
=OTHER.FIL	C:\MYDIR\OTHER.FIL
=.OUT	C:\MYDIR\MYFILE.OUT
=MYNAME.=	C:\MYDIR\MYNAME.TST
C:\OTHER\=	C:\OTHER\MYFILE.TST
C:\OTHER\MYTEST.=	C:\OTHER\MYTEST.TST
C:\OTHER\=.OUT	C:\OTHER\MYFILE.OUT
$X2PATH\XPROFILE.DEF	C:\TOOLS\X2\XPROFILE.DEF

Performance

A great deal of effort was invested in making file load times as fast as possible in the X2 Editor. *Editor Performance Comparison − File Load* and *Editor Performance Comparison − File Load & Quit* show the results of performance comparisons between the X, EOS2, and T2 editors. *Editor Performance Comparison − File Load* shows the time required to load the editor and a file, while *Editor Performance Comparison − File Load & Quit* shows the time required to load the editor and a file, and to quit back to a fullscreen prompt. The first results were obtained with a stopwatch, while the second results were obtained with the program in Rexx Program to Measure Editor Load Times. The second set of results are more reliable.

These tables show the results of the tuning efforts combined with the performance benefits of 32 bit compilation. While X2 is comparable in speed against T2 and EOS2 (both 16 bit editors) for small files, it is much faster when loading larger files.

Editor Performance Comparison − File Load

The table compares file load times for 3 editors against 3 files: T1 was 1000 lines of 10 characters each, T2 was 10000 lines of 10 characters each, and T3 was 100000 lines of 10 characters each. Times are in seconds as measured by a stopwatch, starting from an OS/2 fullscreen command line. The load time measures the time taken from hitting Enter to seeing the file data on the screen. The test machine was a PS/2 Model 77 running a 486SX chip at 33MHz. Times are the average of 3 trials for each editor on each file.

Editor	T1	T2	T3
EOS2 4.13A	0.78	1.38	6.46
T2 2.20	0.47	1.01	8.45
X2 1.26	0.61	0.73	2.94

Quit

The table compares times to load a file and then quit for 3 editors against 3 files: T1 was 1000 lines of 10 characters each, T2 was 10000 lines of 10 characters each, and T3 was 100000 lines of 10 characters each. Times are in seconds as measured by a Rexx program (see Appendix A) starting from an OS/2 fullscreen command line. The test machine was a PS/2 Model 77 running a 486SX chip at 33MHz. Times are the average of 3 trials for each editor on each file.

Editor	T1	T2	T3
EOS2 4.13A	0.67	0.94	5.51
T2 2.20	0.26	0.89	10.60
X2 1.26	0.43	0.52	2.39

Popup Windows

X2 provides several windows to help navigate between files and inside a single file. Each of these windows is displayed on the screen in response to user keys. When a popup window is active, several default keys have slightly different functionality:

Backspace
: Removes the last character from the filter string, and re−displays the window with all entries that match the new filter string.

Ctrl−End, Ctrl−PgDn
: Moves to the last entry in the popup window.

Ctrl−Home, Ctrl−PgUp
: Moves to the first entry in the popup window.

Cursor Down
: Moves the selected entry down one position. If the cursor is already at the end of the

list, it may wrap back to the first entry, depending on the value of the *popup_wrap* setting.

Cursor Up

Moves the selected entry up one position. If the cursor is already at the top of the list, it may wrap down to the last entry, depending on the value of the *popup_wrap* setting.

Cursor Left

Scroll the entire popup window one character to the left. If the left edge of the window is being displayed, this key has no effect.

Cursor Right

Scroll the entire popup window one character to the right. If the right edge of the longest window line is visible, this key has no effect.

End

Scroll the popup window maximum right. The last character of the longest line will be visible.

Enter

Selects the highlighted entry. The select action is dependent on the type of window being displayed.

Ctrl−Enter

Copies the highlighted entry text into the current file, at the current cursor position.

Alt−Enter

Inserts the highlighted entry text into the current file, as a new line after the current line.

Escape

Removes the window from the screen.

Home

Scroll the popup window maximum left. The first character of each line will be visible.

Page Down

Page down, from the current cursor position. If the cursor is on the first row the window is scrolled a full page. If the cursor is somewhere other than the first row, the cursor row is moved to be the top row of the window.

Page Up

Page up, from the current cursor position. If the cursor is on the last row the window is scrolled a full page. If the cursor is somewhere other than the last row, the cursor row is moved to be the last row of the window.

Character keys

"Filters" the window entries according to the letter(s) typed. The entered text will be matched against the beginning of each line or the beginning of the sort sequence, if available, such that only lines beginning with the filter text will be displayed. The filter text is displayed on the top left corner of the window. Note that the highlighted entry always moves to the top window item when a character key is pressed.

All other keys will remove the window, and then perform their normal function. Note that the functions defined to these default keys may be moved to other keys, in which case the new key will have the stated function. While the popup window is active, the PF line will change to provide information about the action of the Enter, Ctrl−Enter, and Alt−Enter keys.

Ring Contents List

Navigation between files when there are many files in the edit ring is sometimes difficult. A *Ring Window* can be displayed by pressing Ctrl−F12 on the default keyboard layout. This key will display a popup window containing a list of the names of the files in the ring, sorted alphabetically, with the current file selected. Any files which have been modified are displayed with the filename highlighted in the *window_emphasis* colour. The width defaults to 40 characters, but will dynamically re−size itself to display as much of the longest file name as possible.

If the Enter key is pressed while this window is active, the file at the cursor position will be made the new current file. No change is made to the order of files in the ring.

File Functions List

Code Functions List describes how to set up the user profile so that the X2 Editor can recognise functions in various programming languages. This information is used by the FUNCWIN command to display a window containing all the functions that are defined within the current file. Selecting a line and pressing Enter will move the cursor to the beginning of the specified function definition.

User Defined Popup Window

An interface exists to allow a user macro to create and manage a popup window. The commands WINDOW, WINLINE, WINSORT, and WINWAIT are used to manage a popup window, as follows:

- *WINDOW* creates the window and defines its dimensions and the maximum number of lines to be inserted.
- *WINLINE* is used to add a line to the window. The lines are added in sequential order.
- *WINSORT* is used to sort the lines in the window.
- *WINWAIT* may be used to wait while the window is displayed. Control is returned to the macro (i.e. *WINWAIT* terminates) when the window is dismissed.

Sample Macro to Create a Popup Window contains a sample macro which uses the *WINDOW* and *WINLINE* commands to create and respond to a popup window.

File Margins

X2 defines four margins for text formatting purposes. These are:

1. The left auto−flow margin
2. The right auto−flow margin
3. The left comment margin
4. The right comment margin

When entering text into a document it is useful for the text of a line to automatically split when it reaches a certain length. This allows you to continue typing and still see the entire

text that you have entered. Every time you enter a character, the length of the current line is checked against the right auto−flow margin. If the line is too long, the line will be split at the first blank to the left of this margin. If the cursor is positioned at the end of the line it will be moved to the end of the newly inserted line; otherwise, it remains in the same location on the screen.

The text that is split from the end of the current line will be placed on a new line if one of the following conditions is met:

1. INSERT mode is OFF
2. The indentation of the next line is different from the current line
3. The next line looks like a list, i.e. it begins with a dash and a space (−)
4. The next line is read only
5. The next line begins with either a period (.) or colon (:)

In all other cases, the split text is inserted into the beginning of the following line. An attempt is made to align the new text with the current line. The same indentation will be used unless the current line looks like a list, in which case the new line will be padded to align with the text following the dash on the current line.

The file margins may be queried or changed with the **MARGINS** command. Default margins are set in the user profile; however, once changed, they are saved with the file on disk.

Similar to file margins, comment margins are used to identify text that is being entered into a comment block. If the beginning of the line and end of the line match the comment identifiers that are in effect for the current file, and the last entered character will move the cursor onto the trailing comment text, a new block comment line will be inserted at the current position, and the cursor positioned for further typing.

The comment margins are mainly used when formatting block and inline comments. They are discussed in detail in the following section.

Comment Formatting

The capability exists to define two strings for any file extension, which will be recognised by the editor as indicating a comment. User Profile Extension Customisation discusses setting up these strings.

Inline Comments

Comments are very important when writing code, but care must be taken that they do not confuse the code by making it appear cluttered. Aligning comments is one way to make the code appear neater and easier to read, but manually aligning comments is an unnecessary chore which can be handled quite readily by the editor, if it is provided with sufficient formatting rules. The X2 Editor contains five settings for inline comment formatting:

◊ **Right aligned.** For the Right aligned setting, the comment is pushed to the right of the screen, so the rightmost edge of the comment is located in the right comment formatting column. Any extraneous whitespace within the comment is removed.

- ◊ **Left aligned.** For this setting, an attempt is made to push the beginning of the comment into column 40 of the file by padding the comment on the right with blanks. If the code text or comment is too long, the comment will be tabbed by the second *comment_column* amount until it fits. To change the leftmost comment column and the tab amount from the default of 40,8, see User Profile Extension Customisation.
- ◊ **C and C++ specific.** This is a special modification of the above rules for the C and C++ languages. For C code, the comment is left aligned if one of the following conditions is met:
  - · The code begins with the '#' character
  - · More than one word is found before any of the following characters: ;=<>](/,|&+−*"^.

  For C++ code, the comment is left aligned if one of the following conditions are met:
  - · The code begins with the '#' character
  - · The code begins with the word "class"

  In all other cases, the comment will be right aligned.
- ◊ **None.** Do not format comments at all. Comments are left the way they were entered into the file. This is the default value.

If a block mark exists over the comment, and it is only a single line deep, it will be moved with the comment.

Inline Comment Conversion

Inline comments are defined as comments which are located on the same line as some text. A shortcut method of entering these comments is defined by the editor. Simply indicate the comment with the *quick_comment* string defined for the file extension. The editor will detect this shorthand when you move off the current line with the cursor down or Enter keys. It will be converted by adding the comment prefix and suffix strings to the body of the comment, and aligning it to the desired margins. See the following picture for an example of how this works.

By default, the *quick_comment* string is "", which means no comment conversion will be performed. Comments are also not converted if a regular comment string is found on the same line. Quick comment conversion may be turned on in the user profile by defining a *comment_prefix* and a *quick_comment* string (see User Profile Extension Customisation).

```
if (a > b)
  max = a; // New maximum value

if (a > b)
  max = a;                                          /* New maximum value*/
```

Automatic Comment Conversion

Block Comments

Block comments are defined as comments which are on their own without any code on the line. These may be formatted with the Alt−P key or the REFORMAT command. X2 contains special logic for formatting these comments. First, the line is scanned for occurrences of the prefix and suffix comment strings. If found, these strings are removed. Any multiple blanks in the line are removed. Two spaces are inserted after every sentence, where a sentence is defined as follows:

- ◊ A sentence must end with either a period, question mark, or exclamation mark.
- ◊ If it ends with a period, the character two positions before it must not also be a period. This accounts for abbreviations like I.B.M.
- ◊ The word at the end of the sentence must not contain a colon or equals sign. The presence of either of these characters is assumed to indicate a GML markup tag.

The formatting will continue until one of the following conditions is reached:

- ◊ The end of file is reached
- ◊ A blank line (after removal of comment prefix and suffix) is reached
- ◊ A line beginning with the character ":" or "." is reached
- ◊ A line beginning with a *highlight_tags* prefix string is reached
- ◊ A line is reached which is longer than the comment formatting length, and which contains no blanks

Finally, the text is formatted to fit within the defined reflow margins, the comment strings are added to the beginning and end of every line, and the line is written to the file. The cursor will be placed at the next file line which was NOT processed. Note that comment strings are not always desired in the reformatted output. They may be excluded with the user profile. See User Profile Extension Customisation for details.

For editing purposes block comment markets are treated as close to invisible as possible. For example, when the cursor is moved to a new line it will normally move to the first non−blank text on that new line. If, however, the new line is a block comment, the cursor is moved to the first text **after** the comment start. Similarly, inserting text at the end of the block comment will automatically insert a new block comment and position the cursor appropriately to continue typing without interruption. Finally, deleting characters within a block comment will not adjust the right comment marker, so the block comment will remain intact.

Comment Manipulation

If comment prefix and suffix strings have been set up through the user profile, they will be used when re−formatting comments. Several checks are made on the supplied strings for use with different formatting jobs:

- ◊ When entering a solid line of asterisks (default key Alt−8), the prefix string is added to the beginning of the line and the suffix string to the end of the line. If either the prefix or suffix string contains a blank, it is removed before being added to the output text.

◊ When block formatting a paragraph of text, any existing comments are first removed from the text, even if they don't include leading or trailing spaces. The output comment strings will be padded with spaces as appropriate if they don't already include spaces.
◊ When formatting inline comments, comments are recognised whether they include spaces or not. The output comment uses the supplied prefix and suffix strings with no modification.

Highlighting

When writing code it is often useful if parts of the code can be displayed in a different colour from the normal text colour. X2 provides the ability to display comments, special keywords, and quoted strings in a different colour. Highlighting is provided on a text line in a hierarchy of precedence, which is:

1. Marked text is always displayed in the *mark* colour, even if the text contains comments or keywords
2. Comments are displayed with the *comment* colour, even if they contain keywords
3. Quoted strings are displayed using the *quotes* colour, even if they contain keywords
4. Keywords are displayed with the *keywords* or *alt_keywords* colour
5. All other text is displayed using the *data* colour.

Comment Highlighting

Before any line is displayed on the screen, it is scanned for comments. If a comment is detected, it will be displayed in a different colour, which can be modified in the user profile. If the comment colour is chosen to be the same as the normal text colour, this feature will be effectively disabled.

X2 does not attempt to detect comments which may span more than one line, nor code which has been commented out with such tricks as the C language *#if 0* preprocessor command. To add this feature would require a language−sensitive file parser, and would slow down processing speed.

Keyword Highlighting

Two sets of keywords may be defined in the user profile, which will be displayed using the *keywords* or *alt_keywords* colour if detected in an uncommented section of a line. Keyword detection is limited to the following rules:

◊ The keyword search is case−insensitive, unless the *keyword_case* setting is *exact*
◊ The character before the keyword and the character after the keyword must not be alphabetic (in the range A−Z)
◊ Keywords are not highlighted if they fall within a commented, quoted, or marked area.

The keyword search logic necessarily involves a certain overhead in the display of each line. The performance penalty will increase as the number of keywords increases. If you find the performance to be unacceptable, this feature may be disabled by removing all occurrences of

the *highlight_keyword* and *alt_highlight_kw* strings from your profile with the *_RESET* keyword.

Language context keywords are sometimes desired to have special capitalisation, either for cosmetic reasons or because the language is case sensitive. The *keyword_trans* profile setting can be used to automatically translate keywords on changed lines into all upper case, all lower case, or into mixed case.

Multiple *highlight_keyword* and *alt_highlight_kw* profile lines may be specified for a given extension. Note that keywords must be separated in the profile by a comma but not a space, as spaces may be considered part of the keyword text.

Cursor Line Highlighting

The default editor highlights the current or cursor line the same as other file lines, but this may be changed in the user profile with the *csr_line* colour setting. Setting it to an explicit colour will only change the portion of the current line that would normally be displayed with the data colour, but using the special keyword *reverse* will cause the entire current line contents to be displayed using the reverse of the normal colours. For example, if a character would normally be displayed with black on white, it would be displayed for the cursor line as white on black.

Splitting Text

Normally when you split a line the line is terminated at the cursor position, and everything to the right of the cursor is inserted as a new line in the file. If the current line contains an unmatched left parenthesis "(" before the cursor position, the new line will be lined up beneath the first non−blank character following the parenthesis. If the current line begins with a bulleted list, i.e. it begins with "− ", the new line will be aligned beneath the beginning of the list text. If the current line contains no left parenthesis and does not indicate a list, the new line will be lined up with the first non−blank character on the old line.

If the current line is marked with a line mark, and the following line is unmarked, the mark will be extended to include both lines after the split.

Auto−Flow

The X2 Editor defines file margins which will cause text to be automatically split when a line gets too long. For example, if the file margins are defined as 1 70 1 77, any character entry that causes the current line to extend beyond position 70 will cause the current line to be split at the first blank before column 70. If the cursor is at the end of the current line, it will be moved to the end of the newly inserted line.

Compiler Errors

When compiling source code, most programmers will see warnings and error messages from the compiler. Instead of writing down the source file, line of the error, and the error message, it is convenient to let the computer do some of the work. The author makes extensive use of

Make files to compile source code, and uses the following inference rule line to speed up detection and fixing of errors:

```
$(CC) $(CCOPTS) $*.c >$*.ERR 2>&1 || x $*.c −ERR
```

This rather cryptic looking line does many things for the user. First, it uses the compiler defined by $(CC) to compile the source code, represented by $*.c, with the options defined by $(CCOPTS). It re−directs the output from the compiler to a file with the same filename as the source code, and an extension of ERR. The directive "2>&1" makes sure that all error messages that would normally be written to *stderr* are also written to the .ERR file. If the return code from the compiler is non−zero, the editor will be invoked on the source file with the −ERR option. −ERR from the operating system command line will automatically load the .ERR file and insert a specially−marked comment at the appropriate line in the source file for each error it finds. Ctrl−N can be used to locate the next error comment.

Compiler error output differs with different compilers and source languages. The compiler error parsing rules are controlled through the *openfile_id* parameter in the user profile, which is defined for all files with extension *.ERR*. Note that the error parsing scans file lines for valid filename characters; all characters that are recognised by the operating system are considered valid **except** the left and right parentheses, which are commonly used in compiler output to delimit such things as error line numbers. The equals sign (=) is also not accepted as it has a special meaning to the editor.

If a file is loaded with the −ERR option, you will hear a short beep from the speaker. This is to alert you that you have an error in your code. Usually when running a long MAKE program under OS/2, you will want to take advantage of multitasking and do something else while waiting. If your MAKE is running in a fullscreen session which is not visible to you, an audible notification of an abnormal MAKE termination can be quite useful.

When all errors have been corrected, there is no need to remove the comment lines which were inserted by −ERR. Simply saving the file will cause the editor to scan for these lines and remove them before saving the file to disk. If they are in the way, they may be removed with Ctrl−O.

When the editor is ended, it will terminate with a special return code of −1. This indicates to the MAKE program that one of its commands failed, so it will not try to continue with the make.

Compiler error parsing is known to work with the following compiler output:

- ◊ IBM C/Set++
- ◊ IBM toolkit IPFC
- ◊ IBM C/2 and early Microsoft C compilers
- ◊ JAVAC Java compiler
- ◊ Microsoft MASM
- ◊ AIX xlc

Hidden Lines

The X2 Editor has the ability to selectively exclude and include lines from the display. The

lines are still part of the file and will be saved to disk with the rest of the file, but are not viewed. In addition, any mark operation (copy, move, delete, shift left or right) that spans one or more excluded lines will only affect the visible lines in the mark. Optionally, a shadow line can be displayed to represent the hidden lines. An ALL command is also provided which works much like XEDIT's ALL command. See ALL for a detailed description of the ALL command.

Saved File Information

When a file is saved with the X2 Editor, additional information is written to the file's extended attributes (EAs). Specific editor settings are saved, so that when the file is next edited, the edit environment can be restored as much as possible. The settings that are saved and restored are:

- ◊ The cursor row.
- ◊ The cursor column.
- ◊ The margin settings, if changed from the defaults.
- ◊ The tab settings, if changed from the defaults.
- ◊ The bookmark setting(s).
- ◊ The tab compression setting, if set with the /NOTABS or /TABS options on the *SAVE* or *FILE* commands.
- ◊ The comment formatting style.

If no saved information is found for a file, or the /TOP option is used, the cursor position will default to the first row of the file. The default file margins, tab settings, and comment formatting style are obtained from the user profile. Extended attribute information is saved in all versions except the DOS version. Extended attributes may be turned off with the *EA* command (see EA). Note that the Windows, AIX, and Linux versions save extended attribute information into a file called **XEAINFO.DTA** in the directory specified by the XPATH environment variable.

Marking Text

The X2 Editor supports three types of marks:

1. Line marks
2. Block marks
3. Word marks

Line marks include the entire contents of the line, while block marks only include a sub−section of a line. Block marks may be defined by specifying two corners of the block with Alt−B, or by moving the cursor with the cursor movement keys while simultaneously depressing the Shift key. Word marks are special cases of block marks. A word mark has the special property that when copied or moved, if the target area is preceded with a space, an extra space will be inserted with the marked word. When deleting a word mark, if the word is preceded and succeeded with a space, an extra space will be removed.

The default X2 Editor follows the E Editor standards for marking text, with one important exception. If a mark already exists, it is removed in response to a request for a new mark, instead of being extended as in the E Editor. For example, the first invocation of Alt−L will define a single line mark. If Alt−L is pressed again with the cursor on another line, the line mark is extended. Another press of Alt−L will cause the previous line mark to be removed, and a single line mark to be started at the current cursor location. Options are available to allow the mark keys to always extend a current mark. For example, to force the Alt−L key to extend a line mark, define it to "MARK LINE EXTEND".

If a block or word mark exists on a single line and text is inserted or deleted before the mark, the mark will be adjusted to stay on the same text.

Recorded Key Sequences

The X2 Editor provides a facility to record keystrokes for later playback. Every key pressed will be remembered between two presses of the record key (Ctrl−R is the default key). The key sequence can then be replayed with another key, which is Ctrl−T in the default configuration. This is a very handy way to write little "on−the−fly" macros to do some kind of repetitive task. Note that the keystroke buffer can hold a maximum of 255 keystrokes. If this limit is hit before the recording is stopped, it will automatically be stopped and a message will inform the user.

When playing back a key sequence, the entire sequence may not be completed. If the sequence contains a Find command which is unsuccessful, it halts and displays a message. This allows fast execution of a sequence by holding down the Ctrl−T key while still providing some file protection by checking for deviations from expected response. The recording can be re−activated by switching to another file, or by defining a new Find string and executing a successful find.

The initial state of the command line and insert mode are remembered when starting a keystroke recording. These values are restored to their original states as part of the playback initialisation. A message "Remembering keys" is displayed in place of the PF line while keystroke recording is active.

Automatic Binary File Detection

When X2 loads a file, it scans the first 80 bytes of the file for the null (hex 00) character. If it finds a null character in the file, the file will initially be displayed in hex mode. You can toggle the display back to text mode with the Alt−H key. See Hexadecimal Mode Screen Layout for details about editor behaviour in hex mode.

The scan for binary files will be skipped if the */BIN* command line option is used to specify binary editing. Invoking The Editor describes the */BIN* option.

Editor Settings

The following settings are provided. They can be tailored through user commands, and their

initial values may be set through the user profile. In each case, issuing the command with no parameters will cause its setting to be toggled, or the keywords **ON** and **OFF** will explicitly set the command.

◊ BROWSE
◊ EA
◊ HEX
◊ INSMODE
◊ LINEND
◊ MSGMODE
◊ SHADOW
◊ SPAN
◊ STATUS
◊ SYNTAX
◊ WRAP

BROWSE

When viewing files that have the Read Only attribute set, the editor will automatically disable any changes to the file. This *Browse Mode* may be turned off, i.e. normal editing mode is turned on, with the **BROWSE OFF** command.

EA

The X2 Editor will normally save file information in a file's Extended Attributes, so that the edit view will be restored when next viewing the file. If this is not desired, setting **EA OFF** will turn off this feature for the current file. The EA feature may be turned off for all files on a disk with the user profile.

HEX

This command provides the default Alt−H key behaviour − it toggles the file view between normal text view and binary or hex view.

INSMODE

InsMode is used to toggle insert/replace editing mode.

LINEND

This command toggles the linend setting. The linend character is used to separate multiple commands; when LINEND is set to OFF, the linend character becomes part of the command.

MSGMODE

This command is mostly used in macros − turning **MSGMODE OFF** suppresses all messages from the display, until MSGMODE is turned back ON.

SHADOW

The X2 Editor provides the ability to selectively exclude lines from the display, just like the XEDIT editor does on VM. When lines are excluded, a "shadow" line can be displayed in their place, to let you know how many lines have been excluded. The shadow line can be suppressed by turning the SHADOW setting OFF. The default value for new files is ON. The SHADOW setting is unique for each file in the ring.

SPAN

When searching for text, sometimes text will be split across multiple lines of the file. Setting SPAN to ON will cause searches to span as many lines as necessary to find text. SPAN is always ON when viewing a file in hex mode, and may be turned on with the SPAN command for other files. It is OFF by default for text files since it causes search performance degradation.

When a file is in text view and SPAN is in effect, lines will be treated as if they are separated by a single space. In hex view, lines are concatenated directly together with no intervening blank.

STATUS

Normally the status line is updated whenever the cursor is moved within a file. This slows down the editor, particularly when using a slow PC. In these cases, performance gains can be made by turning off the status line. Entering **STATUS OFF** will turn the status line off, and **STATUS ON** can be used to restore it. The STATUS setting is global for every file in the ring.

SYNTAX

If any *expand_keyword* lines are found in the profile, they will be converted to *expand_replace* lines in response to the space character. This syntax assistance feature is not always desired, so it can be turned on and off with the *SYNTAX* command. Entering **SYNTAX OFF** will turn syntax assistance off, and **SYNTAX ON** will turn it back on. The SYNTAX setting is unique for each file in the ring.

WRAP

For locating text, it is often useful to locate the text even if it is located above the current position. If a search reaches the end of the file without finding the text, it will continue from the top of file to the current position if the WRAP setting is ON (the default). Setting WRAP to OFF will terminate an unsuccessful search at the bottom of the file. The WRAP setting is unique for each file in the ring.

If WRAP is set ON and a search wraps around the top or bottom of the file, a message is displayed informing the user. WRAP will also cause a backwards (towards top of file) search to wrap around to the bottom of the file.

Tutorial

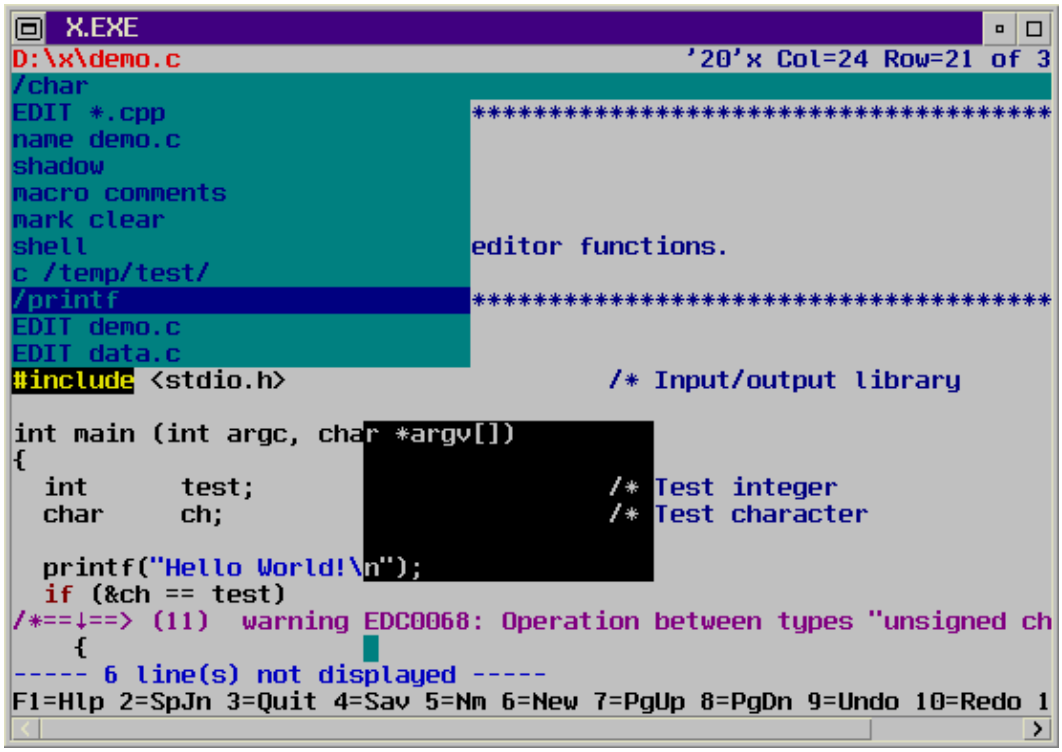
- Screen Areas
- Sample Edit Session
  - Basic Navigation
  - Marking
  - The Command Line
  - Hidden Lines

This section is designed to show some of the capabilities and functionality of the X2 Editor. If you are already familiar with the editor you may want to skip this section.

Screen Areas

The following figure shows a sample editor screen, when editing a small C source file.

Sample Editor Screen



This sample illustrates the main areas of the screen:

Filename

The filename in the sample is in the top left hand corner of the screen, and is *d:\x\demo.c*. The filename is displayed in light red, which in the default configuration is the *mod_filename* colour and means the file has been altered since it was last saved.

Status Area

This area is located on the top line of the screen, on the right hand side. It begins with the hexadecimal representation of the current character (a blank in the sample), and is followed by the file column and row number, the total number of lines in the file, the alteration count, and the insert/replace indicator.

Command Line

This is the line used to execute commands against the editor. It is only displayed when active; when the cursor is in the data the command line is hidden by data. In the sample screen the command line is active; it is the line immediately below the status line which covers the entire screen width, and contains the command */char*.

Command Stack

This is a window of previously executed commands which is displayed immediately beneath the command line. Like the command line, it is only displayed when necessary. A current line is displayed on the command stack window, which can be moved up and down with the cursor keys. Selecting a command stack line will copy the stack contents to the command line.

Data Area

The data area is the area of the screen used to display the file contents. All lines of the data area may be modified by overtyping, as long as the file is not loaded in *Browse* mode. The data area can contain lines with varying emphasis − this example shows *comments*, *quoted strings*, *keywords*, *alternate keywords*, and *error lines*.

Comments

In the sample file, these are delimited by the normal C language strings of /* and */, and are displayed in dark blue. There are two kinds of comments displayed: *Block comments*, which span the entire line, and *Inline comments* which are on the same line as some code.

Marked area

The marked text in the example is displayed with a light grey colour on a black background. There are two main types of marks in X2: line marks and block marks. Block marks are always rectangular and have borders at both lines and columns. Line marks mark the entire line, so do not have column borders. The mark in the example is a block mark.

Quoted strings

The string in quotes, "Hello World!" is highlighted in light blue colour. Note that the mark overlaps part of the string, but does not affect the colour of the part of the string that is not marked.

Keywords

X2 provides the ability to define a list of words that are to be highlighted in a different colour. In the example, the keyword *if* has been highlighted in red.

Alt_keywords

There are two colours that may be used to highlight sets of keywords. The default alternate keyword highlighting is yellow on a black background. The keyword *#include* has been highlighted with *alt_keyword* emphasis.

Error line

The /ERR command line option instructs the editor to look for a .ERR file and insert any compiler errors directly into the source file. These lines are read only, and highlighted in magenta in the sample. The warning message displayed in the example is error EDC0068 from the *VisualAge C++* compiler.

Shadow cursor

When the command line is active, the current location in the data is shown with a *shadow cursor*. The shadow cursor in the example is located on the line immediately

below the E in EDC0068.

Shadow line

When lines have been excluded from the display, their place is marked by a *shadow* line which indicates how many lines have been hidden. Six lines have been hidden in the example. The shadow line itself may be hidden with the *SHADOW* command.

Help line

The last line on the screen is reserved for the help line. This line is overlaid by warning or error messages when necessary, but normally it shows the settings for the main function keys. If you press and hold any of the Ctrl, Alt, or Shift keys the help text will change to show the function keys that are in effect with the various keys. If CapsLock is active the help text will change to all upper case.

Sample Edit Session

This section takes you through a sample edit session. It shows you how you can use some of the default keys to speed up editing tasks, and how to manipulate data with the default keys. Text that you should type is highlighted in *italics*, while keys you should press are highlighted in **bold** text.

Basic Navigation

If you have not already created a profile, do so now:

1. Open an OS/2 fullscreen or windowed prompt
2. Change to the directory containing X.EXE, XPROFILE.EXE, and XPROFILE.DEF
3. *xprofile xprofile.def*
4. *set xpath*=curdir, where curdir is the current directory

Load a new C source file into the editor: *x newfile.c*. The message "New file" should be displayed on the help line.

Create a main function. Type *main* and press the space bar. A skeleton main function should be entered for you.

Define some variables: *int sq, sum=0;*

Duplicate the current line: **Ctrl−K**

Overtype the variable names with another variable called *ctr*

Insert a new line: **Ctrl−Enter**

Insert the following text:

```
for (ctr = 0; ctr < 10; ++ctr)
  {
    sq = ctr * ctr;
    printf("The square of %i is %i\n", ctr, sq);
    sum += sq;
```

```
    }
  printf("The total of the squares is %i\n", sum);
```

While adding the above text, you may need to move around the screen. The following keys may be useful:

Enter
: Move the cursor to the first non−blank of the next line

Left, right, up, down
: Move the cursor one character in the specified direction

Ctrl−Left, Ctrl−Right
: Move the cursor one word in the specified direction

PgUp, PgDn
: Move the current line to the top or bottom of the screen, respectively

End
: Move the cursor to the end of the current line

Ins
: Toggle insert mode

Del
: Delete the character at the cursor

Comment your code. Move to the first brace after the *if* statement, and position the cursor at the end of the line. Add the following text:

```
  // Loop through the integers
```

When you move the cursor down past this line, you should see that the comment has been converted from a quick comment to a regular C language comment, and has been aligned to the right comment margin.

Save your file: **F4**. You should see the filename change from red to magenta, to indicate that it has been saved successfully.

Marking

Now we want to move the calculation of the squares to a separate function. We'll do that by adding a prototype before the definition of the main function. Move the cursor just above the definition of the *main* function, and insert the following line:

```
  int CalcSquare (int num);
```

Mark the above line by pressing **Alt−L** with the cursor still on the line. You should see just one line change to marked emphasis.

Move to the bottom of the file: **Ctrl−End**

Insert a few blank lines after the main function, and copy the marked line: **Alt−C**

Remove the trailing semi−colon and insert beginning and ending braces to define the function. Move back to the calculation of the square from the previous exercise: sq = ctr *

ctr;. Duplicate this line with **Ctrl−K**, then overwrite the top copy to call the new CalcSquare function: *sq = CalcSquare(ctr);*

Position the cursor over the next line and start another line mark with **Alt−L**. You will notice that this extends the previous mark to the current location. You may remove the mark with **Alt−U**, but an easier way is just to press **Alt−L** once again.

Extend the line mark to include the next line (printf) by moving the cursor down and pressing **Alt−L** again.

Move back down to the bottom of the file and move the marked text to its new location. Position the cursor over the line containing the open brace for the function definition, and press **Alt−M**.

If you want to align the text in the new function, press **Alt−<** or **Ctrl−F7** to move the text left, as many times as necessary.

Insert a line at the beginning of the function to define a variable: *int sq;*

Insert a line at the end of the function to return the result: *return sq;*

Press **F4** to save the file again

The Command Line

You may have noticed that the variable name *ctr* in the CalcSquare function does not match the supplied parameter name *num*. This may be fixed with the Change command. Move the cursor to the beginning of the CalcSquare function and press **Esc** to display the command line. Enter: *c /ctr/num/* and press the **Enter** key. Respond to the prompts by pressing *Y* until you reach the end of file. If you are sure you want to change all occurrences of a given string to the end of file, you can supply the * option to the change command.

While still on the command line, type *save* and press Enter to save the file to disk. The default F4 key is set to call the Save command when pressed. Knowing the equivalent command for a key is useful when writing macros or configuring the editor.

The command stack is displayed whenever the command line is active. This is a list of the twenty most recently issued commands, sorted so the most recent is displayed first. Although the command stack contains up to twenty items, you will only see a maximum of ten lines when using the default configuration. You may scroll through the list with the cursor up and cursor down keys. When you do so, you will notice that the current stack line is copied to the command line for modification and/or execution.

Hidden Lines

While editing a large file, it is sometimes useful to see only a few lines and hide the rest. X2 provides this capability through keys and commands. While editing NEWFILE.C, press Ctrl−X several times. You should see the current line replaced by a line that says "1 line(s) not displayed". The number of lines displayed will increase every time you press Ctrl−X. The "line(s) not displayed" line is called a Shadow line, and may be turned on and off with the

Shadow command or by pressing **Ctrl−S**. Press **Ctrl−S** twice to see the shadow line disappear and reappear.

Press **Ctrl−U** to make sure all lines are displayed. Now move to the open brace in the for statement in your main function. Press **Ctrl−A** (exclude area) to hide all the lines with the same or greater indentation. Now move the cursor to the shadow line and press **Ctrl−X**. This will show the five lines that were previously hidden.

There are times when you want to see where you have used a certain variable name, or perhaps you want to see every location where you have called a given function. You can do this with X2's All command. Go to the command line and enter *all /printf/*. You should see just two remaining text lines and three shadow lines. Move to the first printf line and press **Alt−L** to mark the line. Move to the second printf line and press **Alt−L** again to mark all the lines between the two printf lines. Now edit a new file: Press *F6* to display the command line with the command Edit already displayed. Now enter a new filename: *printfs.c*. This will create a second file in the edit ring. You can move back and forth between the two files with the F11 and F12 keys. Press **F12** to move back to newfile.c. Notice that the file is as you left it, i.e. the hidden lines and mark are intact. Move back to printfs.c by pressing **F12** again. Press **Alt−C** to copy the mark. You will see that only the two lines were copied, even though you marked hidden lines between the two visible lines. This allows you to easily copy just a few lines from a large file. Most other commands only work on visible text, so if a line is hidden it can safely be ignored when doing file operations that may destroy data.

Press **F3** to quit printfs.c, and reply **Y** to the prompt confirming that you don't want to save your changes. Press **F4** to save newfile.c and you will notice that all your hidden lines are now visible again.

Default Key Assignments

- Unshifted Keys
  - Alphanumeric Keys
  - Function Keys
  - Special Character Keys
  - Special Keys
- Shifted Keys
  - Alphanumeric Keys
  - Function Keys
  - Special Character Keys
- Control Keys
  - Alphanumeric Keys
  - Function Keys
  - Special Character Keys
- Alternate Keys
  - Alphanumeric Keys
  - Function Keys
  - Special Character Keys

This chapter describes the default settings for each key on the keyboard. Some of the defaults can be changed through profile customisation (see Key Remapping). If no profile is used, or the keys are not changed in the profile, these are the functions which apply to each key.

- Unshifted Keys
- Shifted Keys
- Control Keys
- Alternate Keys

Unshifted Keys

This section describes the functionality behind each key on the keyboard when no modifier keys are depressed; i.e. none of the Shift, Ctrl, and Alt keys are active.

Alphanumeric Keys

The alphanumeric keys perform their normal function. Each key is used to enter the lower case letter inscribed on the key cap.

Function Keys

F1

Browse help file. The default help file is X.HLP, but it can be customised in the user profile. If the cursor is not on the command line, the help file will be loaded at the Top Of File line. If the cursor is on the command line, the first word on the command line will be used as a search parameter to scan the file. If the command line is empty, the special keyword CMD will be used. No changes can be made to the help file when it is loaded via this function.

F2

Split/Join line. If the cursor is positioned after the end of the line, join the next line to the current line at the cursor position. If the cursor is positioned over the body of the line, split the line at the cursor position.

F3

Quit the current file. If changes have not been saved, a confirmation message will be displayed − see QUIT Command for details.

F4

Save the current file to disk.

F5

Change the file name. A command will be displayed on the command line with the current filename already filled in. Simply alter the name and press Enter to rename the file.

F6

Edit a new file. Enter the new filename and press Enter.

F7

Page up, from the current cursor position. If the cursor is on the last screen row the file is scrolled a full page. If the cursor is somewhere other than the last row, the cursor row is moved to be the last row of the screen. Exactly the same as the Page Up key definition.

F8

Page down, from the current cursor position. If the cursor is on the first screen row the file is scrolled a full page. If the cursor is somewhere other than the first row, the cursor row is moved to be the top row of the screen. Exactly the same as the Page Down key definition.

F9

Undo changes to the file. If the current line has been changed, it will be restored to its original state. Otherwise, the last modified line will be restored. Repeated executions of this key will restore line changes until the file reaches its last saved state.

F10

Redo changes to the file. This key will reverse a previous undo action, but only if the undo key was the previous key pressed.

F11

Make the previous file in the ring the current file.

F12

Make the next file in the ring the current file.

Special Character Keys

Backspace

Destructive backspace. Deletes the previous character on the line, and moves the cursor to the left one character. If the cursor is in column one of the line, this key will join the current line to the end of the previous line.

Cursor Down

Moves the cursor down one row. The screen will scroll if necessary. If the cursor is on the command line, retrieves the next oldest item from the command stack.

Cursor Left

Moves the cursor left one column. The screen will scroll if necessary. If in hex display, the cursor will stay in the hexadecimal or text portion of the screen.

Cursor Right

Moves the cursor right one column. The screen will scroll if necessary. If in hex display, the cursor will stay in the hexadecimal or text portion of the screen.

Cursor Up

Moves the cursor up one row. The screen will scroll if necessary. If the cursor is on the command line, retrieves the next newest item from the command stack.

Delete

Delete the character at the current position.

End

Move the cursor to the end of the current line. If the cursor is already at the end of the line, it will be moved to the end of the next line in the file.

Enter

Move the cursor to the beginning of the next file line. If the next line is blank, the cursor is positioned so it will be directly beneath the beginning of the previous non−blank line. If the cursor is on the command line, executes the command specified on the command line. If the command line is empty, moves the cursor to the topmost visible file line on the screen.

Esc

Toggle the command line. If the cursor is in the file, the command line will be displayed and the cursor positioned on the command line. If the command line is active, it will be hidden and the cursor restored to its previous file position.

Home

Move the cursor to column 1 of the current line. If the cursor is already in column 1, move to column 1 of the previous line.

Insert

Toggle insert mode. Characters typed when insert mode is on will be inserted into the file. When insert mode is off, they will replace the previous line characters. Insert mode is indicated with "Ins" on the status line, and by a block cursor. Replace mode is indicated with "Rep" on the status line and by a thin underline cursor. If the current file is being browsed, replace mode is indicated by the text "Brw" on the status line.

Page Down

Page down, from the current cursor position. If the cursor is on the first screen row the file is scrolled a full page. If the cursor is somewhere other than the first row, the cursor row is moved to be the top row of the screen. Exactly the same as the F8 key definition.

Page Up

Page up, from the current cursor position. If the cursor is on the last screen row the file is scrolled a full page. If the cursor is somewhere other than the last row, the cursor row is moved to be the last row of the screen. Exactly the same as the F7 key definition.

Tab

Move the cursor to the next tab position on the line. If insert mode is on, the text will also be moved to the next tab position. This provides a fast way to move text to the right.

If the cursor is on the command line, the tab key provides a filename completion function. Typing the first few characters of a filename and pressing Tab causes the remainder of the filename to be filled in and the cursor positioned to the end of the command line. If there are multiple matches to the supplied string, a popup window will show the matching files and directories. The number of matches that will trigger this window is configurable through the user profile; see Filename Completion

Threshold. If filename completion causes a window to be displayed, the supplied text is drawn with normal emphasis, while the possible completions have bold emphasis. Typing the first letter of any completion will filter the display to just the possible completions that match that character.

Special Keys

Caps Lock
: This key toggles upper case mode on the keyboard as normal. The function key display line at the bottom of the screen will reflect this key. When Caps Lock is on, all text at the bottom of the screen will be displayed in upper case. When Caps Lock is off, the function key text reverts to its normal, mixed case format.

Num Lock
: This key provides its normal function; i.e. the keys on the numeric keypad will enter their numeric values.

Print Screen
: This key provides its normal function; i.e. the current screen contents will be printed to the local printer.

Scroll Lock
: When Scroll Lock is on, the cursor movement keys work differently than normal. Instead of moving the cursor on the fixed screen, the screen scrolls and the cursor stays in the same position on the screen. Scroll Lock only works when the cursor is in the file. If the cursor is on the command line, the Scroll Lock key has no effect on cursor movement.

Shifted Keys

This section defines all keys which are active when the Shift key is also depressed. To activate these functions you must press the Shift key and the target key at the same time. If the Shift key is depressed and held down for a couple of seconds (the exact time varies according to the speed of the local PC), the function key display line at the bottom of the screen will change to show the settings for the function keys when combined with the shift key. Releasing the shift key will immediately change the function display line to its normal text.

Alphanumeric Keys

The alphanumeric keys perform their normal function. Each key is used to enter the upper case letter inscribed on the key cap.

Function Keys

Shift−F1
: Scroll the file left one column. The cursor does not move on the screen, but does move to the next file column.

Shift−F2
: Scroll the file right one column. The cursor does not move on the screen, but does move to the previous file column.

Shift−F3

Scroll the file up one row. The cursor does not move on the screen, but does move to the next file row.

Shift−F4
Scroll the file down one row. The cursor does not move on the screen, but does move to the previous file row.

Shift−F5
Centre the current row on the screen.

Shift−F6
Undefined

Shift−F7
Undefined

Shift−F8
Undefined

Shift−F9
Undefined

Shift−F10
Undefined

Shift−F11
Undefined

Shift−F12
Undefined

Special Character Keys

Shift−Backspace
Destructive backspace. Deletes the previous character on the line, and moves the cursor to the left one character. If the cursor is in column one of the line, this key will join the current line to the end of the previous line.

Shift−Cursor Down
CUA Marking. Extends the current mark (if any) down one row.

Shift−Cursor Left
CUA Marking. Extends the current mark (if any) to the left one column.

Shift−Cursor Right
CUA Marking. Extends the current mark (if any) to the right one column.

Shift−Cursor Up
CUA Marking. Extends the current mark (if any) up one row.

Shift−Delete
Undefined

Shift−End
Mark the current line from the cursor position to the end of the line. The cursor is moved to the end of the line. Nothing is changed if the cursor is beyond the end of the line.

Shift−Enter
Move the cursor to the beginning of the next file line. If the next line is blank, the cursor is positioned so it will be directly beneath the beginning of the previous non−blank line.

Shift−Esc
Toggle the command line. If the cursor is in the file, the command line will be displayed and the cursor positioned on the command line. If the command line is active, it will be hidden and the cursor restored to its previous file position.

Shift−Home

Mark the current line from the cursor position to column 1. The cursor is moved to column 1 of the line. Nothing is changed if the cursor is already in column 1.

Shift−Insert

Undefined

Shift−Page Down

Undefined

Shift−Page Up

Undefined

Shift−Tab

Move backwards to the previous tab position.

Control Keys

This section defines all keys which are active when the Control key is also depressed. To activate these functions you must press the Ctrl key and the target key at the same time. If the Control key is depressed and held down for a couple of seconds (the exact time varies according to the speed of the local PC), the function key display line at the bottom of the screen will change to show the settings for the function keys when combined with the Control key. Releasing the Control key will immediately change the function display line to its normal text.

Alphanumeric Keys

Ctrl−A

Hide area based on indentation. Starting with the current line, all succeeding lines which are indented the same or more will be excluded from the display. Blank lines are included in the area to be excluded.

Ctrl−B

Set bookmark position. If multiple bookmarks are available for this file, a window showing all the bookmarks is displayed. Selecting one and pressing enter will overwrite it with the current cursor position. The first bookmark position is initialised to the starting cursor position when the file is loaded.

Ctrl−C

Toggle comment formatting for the current file. This indicates how in−line comments will be aligned for files which have active comment indicators. The order rotates through:

1. Right alignment
2. Left alignment
3. No comment formatting

Ctrl−D

Delete the current word from the cursor position to the beginning of the next word.

Ctrl−E

Erase the current line from the current cursor position to the end of the line.

Ctrl−F

Repeat previous find command

Ctrl−G

Go to saved bookmark position. If multiple bookmarks are available for this file, a window showing all the bookmarks is displayed. Selecting one and pressing enter will move the cursor to the bookmark line and column. The bookmark remains active, but the previous cursor position is remembered in an *alternate bookmark*. If

the cursor is already on the target bookmark, it is moved to the alternate bookmark instead. The alternate bookmark is also set when the cursor is moved by the Ctrl−F11 function (see Code Functions List).

Ctrl−H

Undefined

Ctrl−I

Undefined

Ctrl−J

Insert a new blank line, and move the cursor to the new line.

Ctrl−K

Duplicate the current line. A copy of the current line is inserted below the current line.

Ctrl−L

Copy the contents of the current file line to the command line. If a block mark exists on the current line, just that portion of the line will be copied. If the command line is not currently active, it will be activated.

Ctrl−M

Undefined

Ctrl−N

Find next compiler error comment. The cursor will be positioned on the next error line in the file. If a column is available in the error line, the cursor will be positioned to that column in the file. Compiler Errors contains more detail about compiler error handling.

Ctrl−O

Obliterate compiler errors. This key will remove all compiler error comments from the current file. They will automatically be removed if the file is saved. Compiler Errors contains more detail about compiler error handling.

Ctrl−P

Open (edit) the file named on the current line. See Customising the OpenFile Function for information on setting up the profile to correctly open files.

Ctrl−Q

Undefined

Ctrl−R

Start/end recording keystrokes for later playback. See Recorded Key Sequences for details.

Ctrl−S

Toggle the SHADOW setting for the current file. If SHADOW is ON, it will be turned OFF, and vice versa.

Ctrl−T

Re−play recorded keystroke sequence. If recording is still active, the current recording is completed before it is re−played. See Recorded Key Sequences for details.

Ctrl−U

Unexclude all. Shows every line in the file.

Ctrl−V

Repeat the last find command, but reverse the direction. This will not affect the find command for subsequent use with Ctrl−F.

Ctrl−W

Find the word at the cursor position. For details on the Find Word algorithm, see FIND_WORD.

Ctrl−X

Exclude/include file lines. If the current file line is not a shadow line, it will be excluded from the display. If the current file line is a shadow line, all the lines represented by that shadow line will be restored to the display.

Ctrl−Y

Bracket/GML tag matching. Match Command contains information about this function.

Ctrl−Z

Undefined

Function Keys

Ctrl−F1

Undefined

Ctrl−F2

Undefined

Ctrl−F3

Convert all text in the marked area to upper case.

Ctrl−F4

Convert all text in the marked area to lower case.

Ctrl−F5

Convert all text in the marked area to mixed case. All text will be in lower case, except the first character of each word, which is upper case.

Ctrl−F6

Undefined

Ctrl−F7

Shift marked text left one column. Text at the beginning of the mark will be lost. Text to the right of a block mark is also shifted to the left.

Ctrl−F8

Shift marked text right one column. Blanks are inserted at the leftmost edge of the mark. Text to the right of a block mark is also shifted to the right.

Ctrl−F9

Enter the current date into the file, in the format YY/MM/DD.

Ctrl−F10

Rotate the case of the current word, through UPPER, Mixed, and lower. A word is considered to be made up of any of the characters a−z, 0−9, and the underscore (_). Any other characters delimit the beginning and end of the word. If the cursor is on a non−alphanumeric character, the case of the previous word will be changed.

Ctrl−F11

Popup Code Functions Window

Ctrl−F12

Popup Ring Window

Special Character Keys

Ctrl−Backspace

Delete the current line from the file.

Ctrl−Cursor Down

Convert the character at the cursor position to lower case, and move the cursor one character to the right.

Ctrl−Cursor Left

Move the cursor to the beginning of the previous blank−delimited word. If the cursor is at the beginning of the line, the cursor will be moved to the last word on the previous line.

Ctrl−Cursor Right

Move the cursor to the beginning of the next blank−delimited word. If the cursor is past the end of the line, it will be positioned beneath the next word on the previous line. If the previous line is also blank beyond the cursor position, the cursor will be moved to the beginning of the next line.

Ctrl−Cursor Up

Convert the character at the cursor position to upper case, and move the cursor one character to the right.

Ctrl−Delete

Delete the current word from the cursor position to the beginning of the next word.

Ctrl−End

Move to the bottom line of the file

Ctrl−Enter

Insert a new blank line, and move the cursor to the new line.

Ctrl−Esc

Under OS/2, this key is used to pop up the desktop window list.

Ctrl−Home

Move to the top line of the file

Ctrl−Insert

Undefined

Ctrl−Page Down

Move to the bottom line visible on the screen

Ctrl−Page Up

Move to the top line visible on the screen

Ctrl−Tab

Undefined

Ctrl−[

Undefined

Ctrl−]

Undefined

Alternate Keys

This section defines all keys which are active when the Alternate key is also depressed. To activate these functions you must press the Alt key and the target key at the same time. If the Alt key is depressed and held down for a couple of seconds (the exact time varies according to the speed of the local PC), the function key display line at the bottom of the screen will change to show the settings for the function keys when combined with the Alt key. Releasing the Alt key will immediately change the function display line to its normal text.

Alphanumeric Keys

Alt−A

Mark area based on indentation. Starting with the current line, all succeeding lines which are indented the same or more will be marked. The mark will be a line mark and will remove any previous mark. Blank lines are included in the new mark. If the

current line begins in column one, the current function will be marked instead of the rest of the file.

Alt−B

Mark a block of text. If there is a single character already marked, it will be extended to the current cursor position. Otherwise, a block mark will be started at the current position.

Alt−C

Copy marked text to the current location. If lines are marked, they will be inserted below the current line. If a block of text is marked, it will be inserted at the current position. The mark will move with the text, so the newly inserted text will be marked.

Alt−D

Delete marked text from the file.

Alt−E

Move the cursor to the end of the marked area. For line marks, this key will move the cursor to the last marked line, and the column will remain constant. For block marks, the cursor will move to the last marked line and the last marked column.

Alt−F

Fill marked area. The marked area will be filled with the character entered in response to the prompt. If the marked area is a line mark, each line will be filled with characters between the block formatting margins.

Alt−G

Undefined

Alt−H

Toggle between hexadecimal mode and text view.

Alt−I

Input an ascending series of integers in the currently marked column. The sequence starts at the top marked line, in the current column. If there is already an integer there, it will be used as the starting value. Otherwise, the sequence starts with the number 1. The output numbers will be padded with blanks so they are right aligned.

Alt−J

Join the current line with the following line. If the cursor is beyond the end of the line, the join occurs at the cursor position. Otherwise, the two lines are joined with a single space between them.

Alt−K

Undefined

Alt−L

Line mark. If a single line is already marked in the current file, the mark is extended to include the current line. Otherwise, any previous mark is cleared and the current line is marked.

Alt−M

Move the marked area to the current position. If lines are marked, they will be inserted below the current line. If a block of text is marked, it will be inserted at the current position. The mark will move with the text, so the newly inserted text will be marked. The previously marked text is deleted from the file.

Alt−N

Enter the current filename at the cursor position.

Alt−O

Overlay mark. If lines are marked, they are copied to the current position in place of any existing lines. If a block is marked, it is copied over top of the current text. The mark is moved to the new position.

Alt−P

Paragraph re−format. For details, see REFORMAT.

Alt−Q

Edit compiler error file. The results from a compile must be piped to a file named fn.ERR. When editing *fn.ext*, pressing this key will cause the error file to be loaded and comments inserted at each line which caused an error. The error comment lines will automatically be removed if the file is later saved. See Compiler Errors for further details regarding compiler error handling in the X2 Editor.

Alt−R

Remove leading blanks from marked area. All text will be aligned to the left edge of the mark. No characters will be removed from any line except blanks. The text to the left of the mark will remain unchanged.

Alt−S

Split the line at the cursor position. The text after the current column will be inserted as the next line, so it lines up with the beginning of the current line. If the current line contains a left parenthesis "(" character, the new text will line up beneath the parenthesis.

Alt−T

Centre text between two comment markers. If no comment markers are found on the current line, any text on the line is centred between the left and right comment formatting margins. If a block mark exists on the current line, the text within the mark will be centred upon the mark.

Alt−U

Unmark, or cancel any current marked area.

Alt−V

Mark a vertical column of data. Starting from the current position, this key will cause a block mark to extend upwards and downwards for as long as a non−blank character is found.

Alt−W

Mark the current word. A block mark is started at the beginning of the word pointed at by the cursor, which extends to the end of the word plus one space. If the cursor is beyond the end of the line, the last word on the line will be marked. If the cursor is pointing at a space between words, the following word will be marked. When looking for the beginning of a word, the editor scans backwards from the cursor position to find the first character which is alphanumeric or an underscore. Any previous mark is removed. See Marking Text for additional information on word marks.

Alt−X

Escape to enter characters in ASCII mode.

Alt−Y

Move the cursor to the beginning of the marked area. For line marks, this key will move the cursor to the first marked line, and the column will remain constant. For block marks, the cursor will move to the first marked line and the first marked column.

Alt−Z

Mark the comment on the current line, or if there is no comment, mark from the current cursor position to the end of the line. If the cursor is already beyond the end of line and no comment is found on the line, this key does not alter the current mark.

Function Keys

Alt−F1
: Undefined

Alt−F2
: Flicker compare. With two files in the ring, this key will position the cursor at the next lines which are different. If the current lines are already different, this key acts as a toggle between the two files.

Alt−F3
: Re−synchronise flicker compare. Starting with two different lines in two files, this key will position the cursor at the next lines which are the same between the two files. If the current lines are already the same, this key acts as a toggle between the two files.

Alt−F4
: Merge changes. When positioned at two different lines in two files, this key will take all the changes from the current file and apply them to the other file in the ring, up until two lines which are the same. Useful after using Alt−F2 to verify that there are changes between two files.

Alt−F5
: Undefined

Alt−F6
: Undefined

Alt−F7
: Undefined

Alt−F8
: Undefined

Alt−F9
: Enter the current date into the file, in the format month dd, yyyy.

Alt−F10
: Undefined

Alt−F11
: Undefined

Alt−F12
: Undefined

Special Character Keys

Alt−Backspace
: Undefined

Alt−Cursor Down
: Move the cursor to the last line showing on the screen.

Alt−Cursor Left
: Move the cursor to the beginning of the previous symbol, where a symbol consists only of alphanumeric characters plus the underscore (_) character. If the cursor is at the beginning of the line, the cursor will be moved to the last symbol on the previous line.

Alt−Cursor Right
: Move the cursor to the beginning of the next symbol, where a symbol consists only of alphanumeric characters plus the underscore (_) character. If the cursor is past the end of the line, it will be positioned beneath the next symbol on the previous line. If

the previous line is also blank beyond the cursor position, the cursor will be moved to the beginning of the next line.

Alt−Cursor Up

Move the cursor to the top line showing on the screen.

Alt−Delete

Undefined

Alt−End

Move the cursor to the next blank−delimited paragraph in the file.

Alt−Enter

Toggle the command line. If the cursor is in the file, the command line will be displayed and the cursor positioned on the command line. If the command line is active, it will be hidden and the cursor restored to its previous file position.

Alt−Esc

Under OS/2, this key is used to cycle through windows on the desktop.

Alt−Home

Move the cursor to the previous blank−delimited paragraph in the file.

Alt−Insert

Undefined

Alt−Page Down

Move the cursor to the next function in the file. See Cursor Positioning for details.

Alt−Page Up

Move the cursor to the previous function in the file. See Cursor Positioning for details.

Alt−Tab

Undefined

Alt−1

Key in a lower left box character (À)

Alt−2

Key in a bottom tee box character (Á)

Alt−3

Key in a lower right box character (Ù)

Alt−4

Key in a left tee box character (Ã)

Alt−5

Key in an intersection box character (Å)

Alt−6

Key in a right tee box character (´)

Alt−7

Insert a comment line containing just spaces. If the file is new and empty, the first line will be replaced by the comment line. Otherwise, the line is added just below the cursor line.

Alt−8

Insert a comment containing a solid line of asterisks. If the file is new and empty, the first line will be replaced by the comment line. Otherwise, the line is added just below the cursor line.

Alt−9

Key in an upper right box character (¿)

Alt−0

Insert a line containing the text "#if 0"

Alt−Pad+

Add a marked column of numbers. The numbers can contain leading and trailing spaces, decimal points, and plus and minus signs. Any other characters will cause the item to be ignored.

Alt−,

Shift marked text left one column. Text at the beginning of the mark will be lost. Text to the right of a block mark is also shifted to the left.

Alt−−

Key in a horizontal box character (Ä)

Alt−.

Shift marked text right one column. Blanks are inserted at the leftmost edge of the mark. Text to the right of a block mark is also shifted to the right.

Alt−=

Expand the current word from previously entered text

Alt−[

Undefined

Alt−

Key in a vertical box character (³)

Alt−]

Undefined

User Profile

- Creating The User Profile
- Comments
- Key Remapping
  - User Profile Key Remap
- Colour Remapping
  - Colour Remapping
  - X−Windows Colour Remapping
- Strings
- Synonyms
- Bracket Matching Characters
- Initial Editor Settings
  - Newline Character
  - Cursor Size
  - Saving Editor Information
  - Enter Key Behaviour
  - Insert Mode
  - Linend Setting
  - Popup Window Scrolling
  - Quick Bookmark Setting
  - Status Line
  - Automatic Bookmarks
  - Multiple Bookmarks
  - Command Line Location
  - Command Stack Window Size
  - Default Extension
  - Default List
  - Escape Character
  - Filename Completion Threshold
  - Linend Character
  - Null Character
  - OpenFile Paths
  - Quit Response When File Modified
  - Right Alt (AltGr) Key
  - Shell Prompt String
  - Beep Behaviour
  - X−Windows Font
- Disk Specific Customisation
  - User Profile Disk Customisation
- File Extension Specific Customisation
  - Default Extension
  - Inline Comment Formatting Control
  - Code Functions List
  - Syntax Expansion
  - Conditional Strings
  - Customising the OpenFile Function
  - Style Formatting
  - User Profile Extension Customisation

X2 allows limited tailoring through the use of a user profile. The user profile is used to customise the screen colours and some keys. It is also used to set file extension−specific parameters such as comment formatting and syntax expansion keywords.

Creating The User Profile

It is **strongly** recommended that you do not modify the default profile, *xprofile.def*. The default profile is frequently updated, and shipped with each new release of the editor. If you make changes, you'll have to re−do all your changes with each new release. A much better way is to create a new profile, say *xprofile.add*, that contains **only** overrides and new definitions. There is no need to copy definitions from *xprofile.def* unless you want to change them, and then you only need to copy the pieces you want to change.

A user profile can have any name. Only those lines beginning with a recognised keyword will be processed. The profile is processed with the XPROFILE command, with the following syntax:

```
XPROFILE profname <prof2 <prof3...>>
```

where *profname* is the name of the profile file, and *prof2* and *prof3* are optional additional profiles. *Profname* will default to *XPROFILE.DEF* in the current directory if it is not specified. Each profile input file will override the settings from the previous profile.

If successful, a file called X.PRO will be created in the current directory. Copy this file to your XPATH and it will be read whenever you start the editor. Note that if the editor cannot find X.PRO when it starts, it will look for X.PRO in the same directory from which X.EXE was started. This allows you to place a default profile with X.EXE on a common disk in a group environment, and still allow people to override the default profile by placing one in their XPATH directory.

The above instructions apply to the OS/2 version of the editor. Under other operating systems the command syntax is the same, but the profile generation program name differs, as does the output profile name. The following table details these differences.

Operating System	Profiler	Profile Name
OS/2	XPROFILE.EXE	X.PRO
Windows NT/95	XPROFWNT.EXE	XW32.PRO
DOS	XPROFDOS.EXE	XDOS.PRO
AIX	xprofile	XUNIX.PRO
Linux	xprofile	XUNIX.PRO
Solaris	xprofile	XUNIX.PRO

The profile name may be changed in two ways. An environment variable called *XPRO* may be set to the desired profile name, or the /P command line option may be used to temporarily override the profile. Under some Unix systems the environment variable XPATH is used by the X−Windows system itself; in such cases the name X2PATH should be used in place of

XPATH.

Comments

Comments may be included in the user profile input file. Comments either start with an asterisk (*) character in column one, or use the sequence /* and */ to delimit the beginning and end of a comment. The latter sequence may be used to denote inline comments in any profile definition line.

Key Remapping

Keys can be remapped with the following sequence:

```
KEY keyname = function
```

where:

keyname
: is the name of the key to be remapped

function
: is the name of the function to be assigned to that key

keyname is converted to upper case before being scanned, therefore it can be entered in mixed case as convenient. Note that keys may be defined globally, or for a specific file pattern. If a file extension has been specified, the key will be defined only for that file type; otherwise, it will be global to all files. User Profile Key Remap describes the keys which can be remapped. Any command may be assigned directly to a key.

User Profile Key Remap

Key	Profile Name	Default Function
F1	F1	Help
F2	F2	SplitJoin
F3	F3	Quit
F4	F4	Save
F5	F5	Rename
F6	F6	"CmdText EDIT "
F7	F7	PageUp
F8	F8	PageDown
F9	F9	Undo
F10	F10	Redo

F11	F11	Previous_File
F12	F12	Next_File
Backspc	Backspace	Backspace
Delete	Delete	DelChar
End	End	Cursor EOL
Enter	Enter	Cursor NextLine
Home	Home	Cursor Col1
Pad +	PadPlus	Keyin +
Page Down	PgDn	PageDown
Page Up	PgUp	PageUp
Tab	Tab	Tab
Shift−F1	s−F1	Scroll Left
Shift−F2	s−F2	Scroll Right
Shift−F3	s−F3	Scroll Up
Shift−F4	s−F4	Scroll Down
Shift−F5	s−F5	CentreLine
Shift−F6	s−F6	Nop
Shift−F7	s−F7	Nop
Shift−F8	s−F8	Nop
Shift−F9	s−F9	Nop
Shift−F10	s−F10	Nop
Shift−F11	s−F11	Nop
Shift−F12	s−F12	Nop
Shift−Del	s−Del	Nop
Shift−Down	s−Down	Mark Extend Down
Shift−End	s−End	Mark Eol
Shift−Home	s−Home	Mark Col1
Shift−Ins	s−Ins	Nop
Shift−Left	s−Left	Mark Extend Left
Shift−Page Down	s−PgDn	Nop
Shift−Page Up	s−PgUp	Nop
Shift−Right	s−Right	Mark Extend Right
Shift−Tab	BackTab	BackTab
Shift−Up	s−Up	Mark Extend Up
Ctrl−A	c−A	Exclude Area

Ctrl−B	c−B	Bookmark Set
Ctrl−C	c−C	Comment_style
Ctrl−D	c−D	DelWord
Ctrl−E	c−E	EraseEOL
Ctrl−F	c−F	Repeat_find
Ctrl−G	c−G	Bookmark Go
Ctrl−H	c−H	Nop
Ctrl−I	c−I	Nop
Ctrl−K	c−K	CopyLine
Ctrl−L	c−L	CopyToCmd
Ctrl−M	c−M	Nop
Ctrl−N	c−N	Errors Next
Ctrl−O	c−O	Errors Remove
Ctrl−P	c−P	Openfile
Ctrl−Q	c−Q	Nop
Ctrl−R	c−R	Keys_record
Ctrl−S	c−S	Shadow
Ctrl−T	c−T	Keys_play
Ctrl−U	c−U	All
Ctrl−V	c−V	Reverse_find
Ctrl−W	c−W	Find_word
Ctrl−X	c−X	Exclude Toggle
Ctrl−Y	c−Y	Match
Ctrl−Z	c−Z	Nop
Ctrl−F1	c−F1	Nop
Ctrl−F2	c−F2	Nop
Ctrl−F3	c−F3	Mark Upper
Ctrl−F4	c−F4	Mark Lower
Ctrl−F5	c−F5	Mark Mixed
Ctrl−F6	c−F6	Nop
Ctrl−F7	c−F7	Mark Shift Left
Ctrl−F8	c−F8	Mark Shift Right
Ctrl−F9	c−F9	Date Long
Ctrl−F10	c−F10	CaseWord
Ctrl−F11	c−F11	FuncWin

Ctrl−F12	c−F12	RingWin
Ctrl−Del	c−Del	DelWord
Ctrl−Down	c−Down	CaseChar Lower
Ctrl−End	c−End	Bottom
Ctrl−Enter	c−Enter	Input
Ctrl−Home	c−Home	Top
Ctrl−Ins	c−Ins	Nop
Ctrl−Left	c−Left	Previous_word
Ctrl−Page Down	c−PgDn	BottomScreen
Ctrl−Page Up	c−PgUp	TopScreen
Ctrl−Right	c−Right	Next_word
Ctrl−Tab	c−Tab	Nop
Ctrl−Up	c−Up	CaseChar Upper
Ctrl−[	c−[	Nop
Ctrl−]	c−]	Nop
Alt−A	a−A	Mark Area
Alt−B	a−B	Mark Block
Alt−C	a−C	Mark Copy
Alt−D	a−D	Mark Delete
Alt−E	a−E	Cursor EndMark
Alt−F	a−F	Mark Fill
Alt−G	a−G	Nop
Alt−H	a−H	Hex
Alt−I	a−I	Mark Integers
Alt−J	a−J	Join
Alt−K	a−K	Nop
Alt−L	a−L	Mark Line
Alt−M	a−M	Mark Move
Alt−N	a−N	Keyin_name
Alt−O	a−O	Mark Overlay
Alt−P	a−P	Reformat
Alt−Q	a−Q	Errors Show
Alt−R	a−R	Mark Align
Alt−S	a−S	Split
Alt−T	a−T	CentreText

Alt−U	a−U	Mark Clear
Alt−V	a−V	Mark Vertical
Alt−W	a−W	Mark Word
Alt−X	a−X	ASCII
Alt−Y	a−Y	Cursor BegMark
Alt−Z	a−Z	Mark EOL
Alt−1	a−1	Keyin À
Alt−2	a−2	Keyin Á
Alt−3	a−3	Keyin Ù
Alt−4	a−4	Keyin Ã
Alt−5	a−5	Keyin Å
Alt−6	a−6	Keyin ´
Alt−7	a−7	Commentline Empty
Alt−8	a−8	Commentline Full
Alt−9	a−9	Keyin ¿
Alt−0	a−0	Input #if 0
Alt−F1	a−F1	Nop
Alt−F2	a−F2	Compare
Alt−F3	a−F3	Compare Sync
Alt−F4	a−F4	Compare Merge
Alt−F5	a−F5	Nop
Alt−F6	a−F6	Nop
Alt−F7	a−F7	Nop
Alt−F8	a−F8	Nop
Alt−F9	a−F9	Date Ordered
Alt−F10	a−F10	Nop
Alt−F11	a−F11	Nop
Alt−F12	a−F12	Nop
Alt−−	a−−	Keyin Ä
Alt−=	a−=	Expand
Alt−Down	a−Down	BottomScreen
Alt−End	a−End	Next_para
Alt−Home	a−Home	Previous_para
Alt−Left	a−Left	Previous_sym
Alt−Right	a−Right	Next_sym

Alt−Up	a−Up	TopScreen
Alt−[	a−[	Nop
Alt−\	a−\	Keyin [3]
Alt−]	a−]	Nop

Colour Remapping

The profile may be used to change the colours used to display practically every area of the X2 Editor screen. Under Unix systems the base X−Windows colours themselves may be remapped to different values.

Colour Remapping

Screen colour areas can be remapped with the following sequence:

```
COLOUR areaname = fg ON bg
```

where:

areaname
: is the name of the screen area to be remapped

fg
: is the foreground colour to be used for this area

bg
: is the background colour to be used for this area

The available colours for use in either the foreground or the background are *Black*, *Blue*, *Brown*, *Cyan*, *Dark Grey**, *Green*, *Light Blue**, *Light Cyan**, *Light Green**, *Light Grey*, *Light Magenta**, *Light Red**, *Magenta*, *Red*, *White**, and *Yellow**. The special foreground colour *Reverse* is only applicable when *areaname* is equal to *popup_cursor* or *csr_line*, and indicates that the popup window cursor should be displayed using the reverse of the normal colour attributes for the window. The table below defines the various screen areas and their default colours.

* The colours marked with an asterisk may cause a blinking display when switching to OS/2's window list and back, if they are assigned as a foreground colour. If this occurs, the blinking may be removed by issuing the *REFRESH* command.

Area Name	Meaning	Default Foreground	Default Background
alt_keywords	The colour used to highlight alternate user keywords	Yellow	Black
browse_data	File text for a read only file	Green	Light Grey

browse_ind	Ins/Brw indicator text for a read only file	Blue	Light Grey
command	Popup command line	Blue	Cyan
command_stack	Popup command stack	Blue	Cyan
comment	The colour used to distinguish comments from normal text or code	Blue	Light Grey
csr_line	The colour used to display regular data on the current cursor line. If set to *Reverse*, the normal colours of the cursor line will be reversed. If explicitly set to a colour, only the data area of the current line will be changed.	Black	Light Grey
data	Normal text data	Black	Light Grey
error_line	Compiler error information lines	Magenta	Light Grey
filename	The colour used to display the filename on the left of the top screen row, when the file has not been changed.	Magenta	Light Grey
function_name	The colour used to display the name of a function that has been recognised through the function_id profile setting.	Magenta	Light Grey
highlight	The colour used to highlight the results of a Find command	Yellow	Magenta
keywords	The colour used to highlight user keywords	Red	Light Grey
mark	Marked area	Light Grey	Black
message	The colour used for messages which will be written over the help line from time to time	White	Light Grey
mod_filename	The colour used to display the filename on	Light Red	Light Grey

	the left of the top screen row, when the file has been altered.		
pfline	The PF display line on the bottom row of the screen	Black	Light Grey
popup_cursor	The colour used to highlight the cursor line when a popup window is active. The special value *Reverse* indicates that the normal uncursored colour scheme will be reversed to highlight the cursor.	Reverse	
prompt	The colour used to display prompt windows.	Yellow	Black
prompt_input	The colour used to display the input text area of prompt windows.	Black	Yellow
protected	The colour used to display protected fields (see LINEFIELDS for details on setting protected fields).	Green	Light Grey
quotes	The colour used to highlight strings delimited with double quotation marks ("). Only the first quoted string on a line will be highlighted.	Brown	Light Grey
shadow	The shadow line used to indicate one or more hidden file lines	Light Blue	Light Grey
shadow_cursor	Cursor position in data area when cursor is on the command line	Black	Cyan
status	Status area on top row of the screen. Contains indicators for the cursor row and column number, file size, insert/replace status,	Blue	Light Grey

	and hexadecimal value of the current character.		
tofeof	The colour used to draw the Top of File and End of File lines	Black	Light Grey
window_bold	Bold popup window lines. Used for lines highlighted with *bold* emphasis (see WINLINE).	Black	Cyan
window_data	Normal popup window lines. Used for lines highlighted with *text* emphasis (see WINLINE).	Blue	Cyan
window_ emphasis	Emphasised popup window lines. Used for showing modified files in the Ring popup window, and for *emphasised* lines (see WINLINE).	Light Red	Cyan
window_title	Popup window title line. Used for displaying the popup window title line. (see WINDOW).	Yellow	Magenta
xwindows_cursor	The colour used to draw the cursor under the X−Windows versions. This colour is only applicable to the X−Windows versions on Linux and AIX. Only the foreground colour is used, even though both foreground and background colours must be specified.	Red	Light Grey

X−Windows Colour Remapping

On Unix X−Windows systems, the mappings of the base screen colours may be changed from their defaults.

```
X-COLOUR colour = xcolour
```

where:

colour
: is the name of the colour to be remapped (see below)

xcolour
: is the X−Windows colour to be used for this colour. On most systems, the X−Windows colours available may be found in /usr/lib/X11/rgb.txt.

The available values for *colour* are the same as the foreground and background colours for the Colour Remapping profile setting. The table below defines the default X−Windows colours for each colour.

Colour	Default X−Windows Colour
Black	black
Blue	DarkSlateBlue
Brown	brown
Cyan	cyan
Dark Grey	DarkSlateGrey
Green	green
Light Blue	Blue
Light Cyan	LightCyan4
Light Green	LightSeaGreen
Light Grey	LightGrey
Light Magenta	magenta
Light Red	red
Magenta	magenta4
Red	red4
White	white
Yellow	yellow

Strings

There are places in the X2 Editor where the default strings may be overridden by user−defined strings. This may be required so the string accurately describes the configuration, or the user may simply prefer some other text. String definition lines in the profile are formatted as follows:

```
STRING str = text
```

where *text* is the text to be displayed. Quotation marks are optional but recommended. Either single quotes (') or double quotes (") may be used. The identifier *str* can be any one of the

following:

helpfile
: Specifies the help file location, where *X.HLP* is the default. The help file is the file that is edited in response to the F1 (Help) key. If not specified, the file will be loaded from the directory specified in the *XPATH* if available, or from the current directory if not.

msg_error_parse
: The message that is displayed when the error parsing function is invoked.

NLS_Keyboard
: The keyboard to use under Windows 32 systems. Possible values are *Dutch Belgian*, *French Belgian*, *Danish*, *Dutch*, *French*, *German*, *Italian*, *Japanese*, *Spanish*, *Swedish*, *Swiss*, or *UK*. If any other value is used, the default US keyboard will be used. This string has no effect with any version except the Windows version.

Synonyms

The X2 Editor provides the ability to define user exits for internal commands. Synonym lines have the following syntax:

```
SYNONYM syn = MACRO macroname <parms>
```

where:

syn
: is the name of the internal editor command which is to be overridden. The supported synonym names are *BEEP*, *CHANGE*, *EDIT*, *ERROR_PARSE*, *EXIT*, *LOAD*, *LOCATE*, *NEWFILE*, *QUIT*, *SAVE*, and *STARTUP*:

  BEEP
  : Executed when the editor issues an alarm, i.e. when a Locate command wraps around the file, or when error parsing is being performed through the /Q edit switch. Either "WRAP" or "ERROR" will be appended to the parameter list.

  CHANGE
  : Executed whenever a change command is issued from the command line, i.e. not from a macro.

  EDIT
  : Executed when a file is being added to the editor. The target filename will be appended to the parameter list. Note that the parameter list contains the command line as typed, including possible wildcards and multiple files.

  ERROR_PARSE
  : Executed when error information is to be added to a file. The current filename will be appended to the parameter list.

  EXIT
  : Executed when the editor is terminating, i.e. after the last file has been quit. Nothing is added to the parameter list. Since there are no files in the editor, most commands (including *EXTRACT*) will fail with a return code of 15 (no documents loaded).

  LOAD

Executed when a file has just been added to the editor. The filename will be appended to the parameter list. Unlike the *EDIT* synonym, *LOAD* is passed the fully resolved filename for the file just loaded.

LOCATE

Executed whenever a locate/find command is issued from the command line, i.e. not from a macro.

NEWFILE

Executed only when a new file has just been added to the editor. The fully qualified filename will be appended to the parameter list. If both *LOAD* and *NEWFILE* synonyms are defined, the *NEWFILE* synonym is executed **first**.

QUIT

Executed when a file is being removed from the editor. The current filename will be appended to the parameter list.

SAVE

Executed when a file is to be saved to disk. The target filename will be appended to the parameter list.

STARTUP

Executed when the editor is starting, but *after* all command line files have been added to the editor. Nothing is added to the parameter list.

macroname

is the name of the macro to be invoked instead of the internal editor command. It is likely that the macro will want to invoke the corresponding internal command as part of its processing. This can be done with the COMMAND keyword.

parms

is a list of optional parameters to be passed to the macro. The parameter list may be appended with certain useful information, as described above.

The STARTUP and EXIT synonyms will not be executed if you invoke the editor with the /Q (quiet) switch. If you write a macro that is intended to be used as a synonym, it is recommended that you prefix all internal commands with the keyword "COMMAND", all macros with the keyword "MACRO", and all external commands with the keyword "SHELL". This restriction is unnecessary when writing macros that aren't intended to be used as synonyms.

Some example uses of synonyms:

```
synonym edit = macro hostedit edit
synonym save = macro hostedit save
```

Used to download a file from the host before loading it into the editor, if the filespec matches a predetermined format, and to upload the file again when it is saved to disk.

```
synonym startup = shell mode co80,32
synonym exit    = shell mode co80,25
```

Used to change the screen size to contain 32 rows while in the editor, and to restore it to 25 rows when the edit session is ended.

Bracket Matching Characters

The Ctrl−Y key is set to match pairs of brackets. Brackets may be defined with the *brackets* keyword, as follows:

```
BRACKETS = "o1,c1;o2,c2; ..."
```

where *o1* and *c1* define the opening and closing string for the first bracket pair, and *o2* and *c2* define the second bracket pair. As many brackets as desired may be entered, but they must come in pairs. Bracket strings may be as long as desired. For example, "#if,#endif;" would be a valid pair of "brackets". Only one *brackets* line is allowed in the user profile. To use either a comma or semi−colon as part of the brackets string, insert a backslash immediately before the character.

In addition to user defined matching characters, X2 does special matching for highlighted tags and conditional strings. See MATCH for further information.

Initial Editor Settings

This section describes settings which are global to every file in the ring. These settings may be changed once the editor is active, but they define the initial state for each setting.

◊ Newline Character
◊ Saving Editor Information
◊ Enter Key Behaviour
◊ Insert Mode
◊ Popup Window Scrolling
◊ Quick Bookmark Setting
◊ Status Line
◊ Automatic Bookmarks
◊ Multiple Bookmarks
◊ Command Line Location
◊ Command Stack Window Size
◊ Default Extension
◊ Default List
◊ ESCAPE Character
◊ Filename Completion Threshold
◊ LINEND Character
◊ NULL Character
◊ OpenFile Paths
◊ Shell Prompt String
◊ Beep Behaviour
◊ X−Windows Font

Newline Character

The X2 Editor recognises any combination of carriage return (CR) and line feed (LF) characters to denote the end of a line. When it reads a file from disk, it remembers which combination of newline characters were used to delimit the line. For newly inserted lines, it will use the combination it finds in the first line of the file. For newly created files, it will use both CR and LF as the newline marker. This may be changed to just LF with the following

profile line:

```
DEFAULT_NEWLINE    = LF
```

Cursor Size

The normal cursor size is an underline bar to represent Replace mode, and a medium rectangle to represent Insert mode. If you find the cursor difficult to see with these defaults, you may want to adjust the sizes to suit your taste. The Insert mode and Replace mode cursor sizes may be changed in the profile:

```
CURSOR_INSERT  = size1
CURSOR_REPLACE = size2
```

size1 and *size2* may be either of *SMALL*, *MEDIUM*, and *LARGE*. The previous profile options to set the cursor size were called *big_cursor* and *small_cursor*. They are still supported, but the preferred method is to specify the cursor sizes separately.

Saving Editor Information

X2 provides the capability to save editor information with a file when it is saved to disk. It does this by writing the cursor position and other data to an extended attribute (EA) for the file. By default this capability is turned on, but you may turn it off in the profile with the following line:

```
EA = OFF
```

Enter Key Behaviour

By default, if the cursor is in the file the Enter key will always move the cursor to the first non−blank character on the next line. If you want Enter to move to the new line when Insert mode is off, but insert a new line when Insert mode is on, add the following line to your user profile:

```
ENTER_INSERT = ON
```

When the cursor is at the last line of a file, and ENTER_INSERT is OFF, or Insert mode is off, pressing the Enter key will not move the cursor. You can make the Enter key insert a new line in this circumstance with the following profile line:

```
INSERT_EOF = ON
```

Insert Mode

The default setting for insert mode is OFF. To start the editor with insert ON, insert the following line in the user profile:

```
INSERT = ON
```

Normally, when the editor is in replace mode, the cursor is displayed as a thin line on the bottom of the line. When insert mode is ON, the cursor takes up half the distance from the bottom of the line to the bottom of the previous line (the *cell height*). If you find it hard to see these cursors, you may enlarge them with the following statement in the profile:

```
BIGCURSOR = ON
```

This will cause the replace cursor to take up half the cell height, and the insert cursor will take up the full cell height.

The Tab key normally moves the cursor to the next tab position, but when Insert mode is on, it will insert spaces to the next tab position. To cause it to always move the cursor without inserting spaces, add the following line to your profile:

```
TAB_INSERT = OFF
```

Linend Setting

By default the LINEND setting is ON, but it can be turned off through the profile:

```
LINEND = OFF
```

Popup Window Scrolling

When displaying a popup window or the popup command stack, the up and down cursor keys are used to scroll through the list. The default editor will stop scrolling if you press the down key while at the bottom of the list, or the up key when at the top of the list. If you want the window to wrap around from the top to the bottom and vice versa, add the following line to your user profile:

```
POPUP_WRAP = ON
```

Quick Bookmark Setting

The bookmark feature may be set to utilise multiple bookmarks for each file. If this has been done, a popup window will be displayed when a bookmark is to be set. A quicker way is to

have the bookmarks pushed down like a stack, and have the new bookmark added to the topmost position. This is accomplished by setting the *quickmarks* profile option:

```
QUICKMARKS = ON
```

Status Line

The default setting for the status line is ON. To start the editor with STATUS OFF, the *status* command must be placed in the user profile:

```
STATUS = OFF
```

Automatic Bookmarks

The X2 Editor provides the capability to automatically generate a bookmark whenever the cursor jumps a long way. A profile setting is provided to allow you to specify how many of these special bookmarks are to be saved, and the minimum number of rows that the cursor has to move before the bookmark is triggered. Automatic bookmarks are displayed for selection with the AUTOBOOKMARK command, which is not defined to a default key. The following command is used to set up automatic bookmarks:

```
AUTO_BOOKMARKS = n,m
```

Where *n* specifies the maximum number of auto bookmarks that are to be saved, and *m* is the minimum number of lines to trigger the bookmark.

As an example, auto_bookmarks could be set to 15,50 which means you have up to 15 special bookmarks, which are set whenever you issue a command that causes your cursor to jump more than 50 lines. If you pressed ctrl−home your last cursor position would be automatically saved, and issuing the autobookmark command would allow you to recover your cursor position from before pressing ctrl−home.

By default this option is set to 0,0 which disables it.

Multiple Bookmarks

By default, the X2 Editor will provide a single bookmark setting. This allows you to quickly set a bookmark with Ctrl−B and go to that saved position with Ctrl−G. If you want more than one bookmark, you can increase the number of bookmarks with the following command:

```
BOOKMARKS = n
```

If you select more than one bookmark, you will be prompted for the bookmark whenever you press Ctrl−B or Ctrl−G.

Command Line Location

When the Escape key is pressed a command line is displayed at the top of the screen, and the command stack is displayed immediately below this. To change the location of the command line to the bottom of the screen, with the command stack immediately above it, use this profile command:

```
CMDLINE = BOTTOM
```

Command Stack Window Size

When you display the command line with the escape key, a window is displayed which shows previously entered commands. By default, this window is ten rows deep and 30 columns wide; however, it may be set to any number of rows between zero and twenty and any number of columns with the following profile commands:

```
CMD_WINDOWSIZE  = n
CMD_WINDOWWIDTH = m
```

Default Extension

If a file is specified with no extension, and it is not found on disk, a search may be made for one or more *default* extensions. If a file with the default extension is found, it will be loaded instead of the file without the extension. By default no default extension processing will be performed, but this capability may be turned on by defining a *DEFAULT_EXT* line in the profile:

```
DEFAULT_EXT = ext1,ext2,ext3,...
```

If a file is specified without an extension, and it doesn't already exist on disk, the supplied extensions will be tried, one at a time. The first one that results in a matching file will be used as the file to be edited. If all default extensions are tried without a successful match, the originally specified file specification will be used.

For example, let's say that the *default_ext* string is set to **C,H,CMD**, and that the filename **TEST** is specified on the command line. X2 will load the first file found from the following list:

1. TEST
2. TEST.C
3. TEST.H
4. TEST.CMD

If none of the above files exist on disk, a new file called **TEST** will be loaded into the ring.

Default List

If the editor is started without a file specified on the command line, it will load an empty file with no name. You can customise it so it will load files from a list, by setting the *DEFAULT_LIST* in your profile:

```
DEFAULT_LIST = fn
```

The format of *fn* is one file per line − the text on each line will be treated as a file to be loaded. Lines that begin with an asterisk (*) will be treated as comments and ignored. If the default list is specified but not found, it will be edited instead of an unnamed file. This allows you to easily create the default list file, which is probably unique for each directory on your system.

Escape Character

The escape character is used when displaying popup windows to control the emphasis of the text in a WINLINE. It is followed by various characters to create text with **Bold**, *emphasised*, or normal text. WINLINE contains more information on how to use the escape character when building popup window lines.

The default escape character is a backwards quote (`) symbol. To change it to another character, insert the following line in the user profile:

```
ESCAPE_CHAR = c
```

Filename Completion Threshold

When the tab key is pressed from the command line, the last word on the command line is expanded to find all matching filenames. By default, if more than one match is found a popup window will show all the matches for user selection. This behaviour may be tailored by setting a threshold in the profile as follows:

```
fn_completion = n
```

This will cause the matching filenames to be displayed on the command line in response to successive presses of the tab key, if the number of matches found is less than or equal to *n*. If more than *n* matches are found the popup will still be used.

Linend Character

The default linend character is the caret (^) symbol. To change it to another character, insert the following line in the user profile:

```
LINEND_CHAR = c
```

Null Character

The null character is used in place of x'00' characters when displaying a file in text mode. The default null character is the question mark (?) symbol, but it can be changed to another character with a line like the following in the user profile:

```
NULL_CHAR = c
```

OpenFile Paths

The default *OpenFile* function will look for a file in the current directory. If you would prefer it to scan directories that have been defined by environment variables, you may set the various paths with the PATHS statement:

```
PATHS = paths
```

The *paths* defines a set of environment variables, where each variable may be separated by a space or semi−colon. For example, *paths* may be set to *INCLUDE;PATH*. In this case, all the paths defined by INCLUDE will be searched for the file, followed by all the paths specified by *PATH*.

This statement has no effect in the DOS version.

Quit Response When File Modified

Controls the available responses when the QUIT command is issued for a modified file. By default, the characters Y, N, and W and any keys that have been set to the FILE, SAVE, or QUIT commands are acceptable responses. To bypass the function key check so that only the characters Y, N, and W will be accepted, set this option to "charonly":

```
quit_modresponse = charonly
```

Right Alt (AltGr) Key

Some NLS Windows keyboards don't automatically distinguish the right alt key as an AltGr key. If you are using an NLS keyboard and the right alt key is not giving different results from the left, you probably need to tell X2 to treat right alt as AltGr:

```
right_alt_is_altgr = on
```

Note that this setting will have no effect if you are using any version except the Windows version.

Shell Prompt String

When the user shells to a command prompt with the SHELL command, it can sometimes be easy to forget that the editor is still active. An indicator is added to the default prompt string to act as a reminder. By default this string is "(x)", but it can be changed with:

```
SHELL_PROMPT = "str"
```

where *str* will be prefixed to the default prompt string for all SHELL commands.

Beep Behaviour

The only time the default editor beeps is when the **/ERR** option is supplied on the command line. If you like to hear a beep when the editor wraps around a file during text searches, add the following to your profile:

```
WRAP_BEEP = ON
```

X−Windows Font

The X−Windows (AIX) version of the editor uses graphics routines to draw its window; therefore, the font it uses may be customised. The default font is "fixed", but it may be changed with the XWINDOWS_FONT profile line:

```
XWINDOWS_FONT = "newfont"
```

To find which fonts are installed on your system, go to directory /usr/lpp/X11.fnt and look at the files with extension .inventory.

Disk Specific Customisation

It is sometimes handy to customise the editor settings by disk. For example, when an OS/2 mounted disk is really resident on an AIX system, it would be nice to automatically save files with blanks compressed into tab characters.

Keywords are available to alter the editor settings based on the disk letter. The settings available may be found in User Profile Disk Customisation.

User Profile Disk Customisation

Keyword	Text	Default	Meaning
disk	disk1,disk2,...	N/A	Defines a list of disks for which the following

			definition(s) apply. There is a maximum of 27 disk letters that may be defined. The special disk * may be used to set disk settings for all disks.
disk_ea	ON | OFF	ON	Controls saving of file information with a file as extended attributes (EAs). *ON* saves the information, *OFF* discards it. Overridden by the *EA* command.
disk_linend	CR | LF | CRLF | ASIS	ASIS	Defines the desired linend character(s) to be used when saving a file. Each line may be terminated with a Carriage Return, a Line Feed, or the two together. The default value of *ASIS* means that lines should be saved with the same termination character as they had when they were read.
disk_tabs	ON | OFF | ASIS	ASIS	Allows file lines to be saved with multiple occurrences of blanks compressed into tab characters. The default value saves lines with no tab compression, which improves performance but increases the amount of disk space required to save the file.

File Extension Specific Customisation

Customisations are often required for specific file types. File types are recognised in one of three ways:

1. By the file extension
2. By the file pattern
3. By the first file line

The file *extension* is defined to be everything after the last period (.) in a file name. If it matches a supplied extension, profile customisation for that extension will be used.

A file *pattern* may be defined in the profile for cases where a file has no extension, or where the portion before the period is more suitable for categorising files. Wildcards may be used in file patterns. An example of this method of typing a file comes from *xprofile.def*:

```
file_pattern = xprofile.*
```

If the above two tests have failed to provide a match, the first line of the file is examined for a shell extension. If the first two characters of the line consist of "#!", the text after the last file separator (slash on Unix systems, backslash otherwise), is used to search for profile customisation. For example, if the first line of a file reads *#!/usr/bin/perl*, then the file "extension" is assumed to be *perl*. This feature is most useful on Unix systems, but is available in all versions except for the DOS version.

A series of keywords is available to set the file extension, and to specify comment formatting and syntax expansion parameters for that file type. Each has the syntax "keyword = text", where *keyword* and *text* are defined in User Profile Extension Customisation.

Default Extension

If desired, you can override the system defaults for various extension−specific settings. This is done by specifying a default filetype of *****. This will be used if no other matching filetype is found for a file. For other extensions which are defined after the default extension in the profile, the values defined for the default extension will be the initial values for the definition. Note that changes to the default file extension of * will only affect file extensions that are undefined or defined later in the sequence.

Inline Comment Formatting Control

Comment formatting is an important part of the X2 Editor. The formatting of inline comments can be controlled through the *comment_formatting* keyword in the user profile. This line has the following format:

```
comment_formatting = flag1 | flag2 | flag3... | flagN;
```

The following flags are available:

ALIGN_QUICK
: Align (without converting) quick_comments to the current margins according to the left/right settings. Incompatible with the *CONVERT_QUICK* and *CONVERT_ALLQUICK* flags.

COLSTART
: Only align comments starting in *comment_column*

CONVERT_ALLQUICK
: Controls whether inline comments indicated by the *quick_comment* string should be converted to use the defined *comment_prefix* and *comment_suffix* strings. If set, any occurrence of *quick_comment* within a line will be converted to the *comment_prefix* string, and the *comment_suffix* string will be added to the end of the line. If *quick_comment* is null, this option has no effect.

CONVERT_QUICK
: Controls whether inline comments indicated by the *quick_comment* string should be converted into regular comment strings. This option differs from the above in that only inline quick comments that have non−blank code before the comment will be converted. If *quick_comment* is null, this option has no effect.

C_CONDITIONAL
: Use C formatting rules to determine left/right alignment. See Inline Comments for details.

CPP_CONDITIONAL
: Use C++ formatting rules to determine left/right alignment. See Inline Comments for details.

IF_MODIFIED
: Only align comments on lines that have been altered. This setting will ensure that comments are only re−formatted if the line has been modified. Simply scrolling through the file with the down cursor key will not alter the file. Handy if you're viewing someone else's code and you don't want to change all their comments.

KEEP_BLANKS
: Keep leading and trailing blanks in right aligned comments. Normally the editor will

strip out any blanks before and after the comment text before pushing the comment to the right comment formatting margin. This option forces it to maintain any blanks that may have been inserted in the comment.

KEEP_TRAILSPACE

If *comment_prefix* contains one or more trailing spaces, then those spaces must be found in the input text for it to be recognised as a comment. By default, any trailing spaces are removed from *comment_prefix* before input text is checked for comments.

LEFT

Format all comments to the left comment margin

NO_BLOCKS

By default, blocks of text that are formatted with the Alt−P key are formatted so that the defined *comment_prefix* and *comment_suffix* strings are inserted at the beginning and end of each output line. To have blocks formatted as just text, include this option for the desired file extension.

NONE

Make no changes to comments. Useful to turn off previously set commenting flags.

RIGHT

Format all comments to the right comment margin

UPPERCASE

Defines whether the code portion of a comment should be upper cased as part of formatting. The rules for upper casing a line are:

1. If the *UPPERCASE* flag is FALSE, the line is not altered
2. Nothing is touched after the first occurrence of the comment_prefix string (if found)
3. Anything within either single (') or double (") quotes is untouched.
4. If a *conditional_prefix* is defined for the filetype, any line beginning with the *conditional_prefix* string is not changed.

For most programming languages, the default of FALSE will be desired. Assembly language is the most likely candidate for a setting of TRUE.

Note that all the formatting flags must be written on a single line, and that all the flags are re−set to their default values whenever a new comment_formatting profile line is encountered. Only one of the LEFT, RIGHT, C_CONDITIONAL, and CPP_CONDITIONAL flags may be specified for any file extension.

Code Functions List

When viewing a file containing source code, the Ctrl−F11 key may be used to display a window containing a list of each function defined in the current file. The list will be sorted alphabetically by function name, and the current function (if available) is selected. Function recognition is set up through the function_id keyword in the user profile. This keyword has the following format:

```
function_id = "keyword", flags, <WORDNUM=w,> <STARTLINE=s,> <MAXLINES=m,>
              <COLUMN=c< &|− c2,>> <NAME_OFFSET=n>;
```

where:

keyword

is a text string that must be found on a line to trigger the following parameters. If *keyword* equals the special string "_RESET", all previous *function_id* settings will be cancelled.

flags

is one or more of the following options, which may OR'd together with a vertical bar (|) character:

ANY

A placeholder option, meaning no restrictions on the placement of the keyword information.

CANCEL

If found, cancels a search for a required keyword.

END_OF_WORD

The keyword must be located at the end of a word, i.e. there is non−blank text before the keyword, and the keyword is either at the end of the line or it is followed by a blank character.

IGNORE

If found on a line, it is definitely **NOT** a function, no matter what other keywords are found.

MATCH

Once the given text is found, use bracket matching rules to find the corresponding character before identifying a function.

NO_DUPES

Indicates that the keyword string must NOT be found before the next *REQUIRED* keyword.

OPTIONAL

If found under the specified conditions, implies that the previously found string is definitely to be treated as a function. This is designed to work with C++ constructors to stop initialisers from inhibiting the constructor line recognition.

PREVIOUS

Starting from the current position, scan backwards for the supplied text.

REQUIRED

Together with the *MAXLINES* parameter, indicates that the keyword must be found within the next *m* lines, or the current line is not a function.

REQUIRED_SET

The keyword is part of a set of words that may indicate a function. One or more of the required_set keywords must be found on the current line.

START_OF_WORD

The keyword string must be found at the beginning of a word, i.e. it must be preceded with a blank character.

w

is the position that *keyword* must be found in the line to be valid. If this parameter is omitted, *keyword* may be anywhere on the line.

s

The starting line number (from the current line) to scan forwards for a *REQUIRED*, *CANCEL*, or *OPTIONAL* keyword. This number defaults to 1, i.e. start looking at the next line; however, it may be set to 0, if a second scan is required of the current line.

m

The number of lines to scan forwards for a *REQUIRED*, *CANCEL*, or *OPTIONAL* keyword.

c

is the file column that *keyword* must start in to be recognised. If this parameter is omitted, *keyword* may be located anywhere on the line.

c2

if present, indicates that *keyword* is restricted to starting between columns *c* and *c2* in the file. If specified with an ampersand (&), it is further restricted to starting in column *c* **or** *c2*. The dash (−) character is used to signify the range of starting columns.

n

if present, indicates the offset in blank−delimited words from the indicator text *keyword*, to the actual function name. This information is used to display the function name in a different colour, and for sorting the function list popup window.

Function recognition can be set up for any language; however, it is mainly intended for use with the C, C++, Rexx, NetRexx, SCRIPT, and ASM languages.

Function Identification Examples

The rules for identifying functions in various programming languages differ widely, making the required profile code sometimes quite complicated. A few examples will hopefully illustrate the purpose behind some of the above flags and parameters.

The first example is commonly used when parsing Rexx source code. It says that if the first word on the line ends with a colon character, it is a function definition line. The name offset of −1 tells the editor to treat the first word **before** the colon as the function name.

```
function_id = ":", END_OF_WORD, WORDNUM=1, NAME_OFFSET=−1;
```

In the next example, we're using the function recognition a bit differently; we again want to highlight lines in which the first word ends in a colon, but this time we want to exclude those lines in which the first word also contains a right parenthesis. The start line offset of 0 means to start the *Cancel* scan at the current line, i.e. the line that has tentatively been flagged as a function. Adding maxlines of 0 means we only want to scan the current line before we stop searching.

```
function_id = ":", END_OF_WORD, WORDNUM=1, NAME_OFFSET=−1;
function_id = ")", CANCEL, STARTLINE=0, MAXLINES=0, WORDNUM=1;
```

The following example is used to scan X2 Editor profile definition files. There are two possible triggers for a "function" − the first word on the line may be either "extension" or "file_pattern". If found, the actual function name is considered to be two words after this text, and could contain the special characters '.', '*', and ',' in addition to the normal alphanumeric characters.

```
funcname_chars = ".*,";
function_id    = "extension",    REQUIRED_SET, WORDNUM=1, NAME_OFFSET=2;
function_id    = "file_pattern", REQUIRED_SET, WORDNUM=1, NAME_OFFSET=2;
```

Our next example is starting to get more complicated, and is used for parsing Java source files. As part of the profile processing we have chosen to extend the base "C" filetype, but we want to handle functions separately. So we start the function id processing with the *_RESET* keyword, which clears all previous definitions. We then specify that any line that contains a semi−colon in **any** position, is to be ignored. If we didn't do this, then any function

call would be flagged as a function *definition*, which is obviously not what we want. We then specify that any line containing either the word "class" or a left parenthesis is **maybe** a function − we have also required that it be followed by a left brace in either column 1 or column 3, up to 15 lines from the line in which the special text was found. We've also told the editor that if we find a right brace in column 3, *before we find the left brace*, then we don't want a function. The *NO_DUPES* flag on the left parenthesis specification, means that we must find a left brace before we find another left parenthesis; otherwise, no function.

```
base_extension = C
function_id    = _RESET;
function_id    = ";",      IGNORE;
function_id    = "CLASS", REQUIRED_SET, NAME_OFFSET=1;
function_id    = "(",      REQUIRED_SET | NO_DUPES, NAME_OFFSET=-1;
function_id    = "{",      REQUIRED, MAXLINES=15, COLUMN=1&3;
function_id    = "}",      CANCEL, MAXLINES=15, COLUMN=3;
```

The standard processing for C source files is very complicated, which reflects the free coding format that is allowed by the C language. We start by specifying that any line that begins with a hash symbol or contains a semi−colon is to be ignored. This covers pre−processor statements (#include) and function calls. We now have three "trigger" strings to flag a function line, and two strings which may cancel the function identification. These are similar in form and function as for the Java processing described above.

```
function_id = "#",         START_OF_WORD | IGNORE, WORDNUM=1;
function_id = ";",         IGNORE;
function_id = " STRUCT ", REQUIRED_SET, NAME_OFFSET=1;
function_id = "CLASS",     REQUIRED_SET, NAME_OFFSET=1, WORDNUM=1;
function_id = "(",         REQUIRED_SET | NO_DUPES, NAME_OFFSET=-1;
function_id = "{",         REQUIRED, MAXLINES=15, COLUMN=1;
function_id = "}",         CANCEL, MAXLINES=15, COLUMN=1;
function_id = ");",        CANCEL, MAXLINES=15, COLUMN=1;
```

The above sample is good for identifying functions in C source code, where the opening and closing braces are always in column one of the file. Some people prefer to write C code with the opening brace following the function definition, e.g.

```
int main (int argc, char *argv[]) {
  ...
}
```

This code is more difficult to parse, but the following profile definitions may help:

```
function_id = _RESET
function_id = "#",         START_OF_WORD | IGNORE, WORDNUM=1;
function_id = ";",         IGNORE;
function_id = " STRUCT ", REQUIRED_SET, NAME_OFFSET=1;
function_id = "CLASS",     REQUIRED_SET, NAME_OFFSET=1, WORDNUM=1;
function_id = "}",         MATCH, COLUMN=1;
function_id = ")",         PREVIOUS | MATCH, NAME_OFFSET=-1;
```

Our last example is the definition for C++ files. As C++ is simply an extension of the C language, we'll want to include all the C function recognition statements, with a few extensions of our own. In order to handle constructors we need to check for a colon in word number one of the following 15 lines. If we didn't include this check, constructors that called other initialisers would fail our checking, as a left parenthesis would be found before the left

brace. Note that we wish to include the colon and tilde characters in our list of function name characters, to handle member functions and destructor functions respectively.

```
base_extension = C
funcname_chars = ":~";
function_id    = ":", OPTIONAL, WORDNUM=1, MAXLINES=15;
```

Syntax Expansion

X2 provides the ability to automatically expand keywords into additional text. This provides a handy way to reduce typing and syntax errors. Every time the space bar is pressed, a check is made on the line text against user defined syntax triggers. If a trigger is found, the specified replacement text is entered, and the cursor re−positioned. Insert mode is always set to allow continued typing. Syntax expansion is defined in the user profile, with the following keywords:

expand_keyword = "text"

Defines a syntax expansion keyword. If *text* is found as the only text on a line when the space bar is pressed, the line will be replaced with *expand_replace* line(s) or an *expand_macro* macro will be executed. If *text* is equal to the special value *_RESET*, any previous expansion keyword settings will be removed from the profile. If *text* contains the text "\1" the text must only match up to this variable. The following word will be substituted for any occurrences of "\1" in the *expand_replace* lines. Note that only one substitution variable may be specified.

If desired, multiple keywords may be specified in *text* by separating them with commas. For example, the line "expand_keyword = kw,keyword" will expand with *expand_replace* or *expand_macro* if **either** *kw* **or** *keyword* is found on the line. To include a comma in the keyword text, prefix it with a backslash.

expand_macro = "text"

Defines the name of a macro which will receive control when a syntax expansion keyword is recognised. There may only be one *expand_macro* line for a given *expand_keyword*, and this keyword is mutually exclusive with *expand_replace*. The macro is responsible for expanding the keyword. The current line is deleted before the expansion macro is executed, but its indentation and contents are supplied to the macro as parameters.

expand_replace = "text"

Defines a syntax expansion replacement line. All replacement lines up until the next *expand_keyword* definition are used to replace the active keyword. If the *expand_replace* text contains the string "\c" the cursor will be placed at that location, **unless** the string is escaped by preceding it with another backslash, i.e. "\\c" will be treated as just the text "\c". Insert mode is always set when a keyword is expanded.

Besides "\1" and "\c", special substitution variables may be included in *expand_replace* strings, as defined in the following table. Note that each variable identification character is **case sensitive**.

Variable	Meaning
\\	Escape, replaced with \
\1	Substitution variable
\B	Full written month, e.g. February
\c	Cursor placement
\CP	The currently defined comment prefix string
\CS	The currently defined comment suffix string
\d	2 digit day number, e.g. 06
\D	Day number, e.g. 6
\f	Unqualified filename, e.g. cdPlayer.ini
\F	Fully qualified filename, e.g. C:\WINDOWS\cdPlayer.ini
\H	Hour of the day, 24 hour clock
\m	2 digit month number, e.g. 02
\M	Minute of the hour, e.g. 09
\S	Second of the minute, e.g. 03
\y	2 digit year number, e.g. 00
\Y	4 digit year number, e.g. 2000

Conditional Strings

Many programming languages include some sort of preprocessor statement to conditionally include blocks of code. For example, the C language compilers provide the *#if*, *#elif*, *#else*, and *#endif* directives. The X2 Editor supports these constructs through the *conditionals* keyword, which has the format str1, str2, str3, ..., strM, strN. These strings are used with the *CONDITIONAL* and *MATCH* commands.

The *CONDITIONAL* command can take one of three parameters: *IF*, *ELSE*, or *END*. With the *IF* parameter the first conditional string, i.e. *str1*, will be input into the current file. With the *END* parameter, the last string (*strN*) will be input. If the *ELSE* parameter is specified, the **second last** (*strM*) parameter is input into the file.

The *MATCH* command takes no parameters. If the cursor is on *str1* the cursor will be moved to the **next** occurrence of **any** of *str2* through *strN*. If the cursor is on *strN* it will be moved to the **previous** occurrence of *str1*. If it is located on any of *str2* through *strM*, it will be moved to the **next** occurrence of **any** of *str2* through *strN*.

The *MATCH* command will recognise strings other than conditionals. If brackets have been defined and the cursor is positioned on one of these strings, the cursor will be moved to the equivalent bracket. Repeated invocations will toggle the cursor between the open and close bracket string. If the cursor is positioned on a colon (:) character, it is assumed to represent the start of a GML tag. If the next character is the letter E, a backwards scan will be made for the tag. "Tags" that may be found inside comments or quoted strings are ignored for the purpose of the match command.

Customising the OpenFile Function

The X2 Editor provides a key (Ctrl−P in the default configuration) that is used to open a file based on information on the current line. By default, it looks for the first text on the line that contains valid filename characters and uses that; however, the default behaviour may be extended by filetype through the user profile. For any extension one or more *openfile_id* entries may be defined, with the following format:

```
openfile_id = "keyword", WORDNUM=w, <PATH=p,> <NO_BLANKS,>
<PATHSEARCH=ps,> <DEFAULT_EXT=xxx>;
```

where:

keyword
: defines a pattern that is used to locate the filename on the line. It contains the variable \f to indicate the beginning of the actual filename; if any text precedes \f it must be found on the line for this entry to be applied. In addition to \f, optional parameters \r and \c may be specified to define a starting row and column number, respectively.

w
: is the starting position that *keyword* must be found in the line to be valid. Note that the *WORDNUM* parameter is **required**.

p
: is an option path string that will be prefixed to the target filename, if found. Normal filename expansion characters such as "=" can be specified; see File Specification.

NO_BLANKS
: an option which specifies that the filename is not expected to contain blanks. If not specified, the default is to take all characters after \f as part of the filename.

ps
: is a string that is found in the source file on another line; if found in the file, the immediately following text is taken to be the path information. Note that the *PATH* and *PATHSEARCH* options are mutually exclusive.

xxx
: the extension to add on to the filename when opening the file. If the keyword defines the \f filename to be afn, then the file to be opened will be called afn.xxx.

Style Formatting

The editor has the capability of re−formatting source code files according to pre−specified rules. This is useful when viewing someone else's source code − it is much easier to concentrate on the content of the code when the coding style is familiar and not distracting. Style formatting is not intended to be perfect; rules may be specified which formats most code so it is close to the desired output, but it is almost impossible to get all code to format perfectly. Still, it is a useful function that can assist in the review of source code files.

Style formatting is set up for a file extension in the user profile. The format of style formatting lines is:

```
  styleword = keyword, flag1 | flag2 | ... flagN;
```

where *keyword* is a keyword that, if found at the beginning of a file line, will cause succeeding line(s) to be formatted according to the specified flags. Specify a *keyword* of *_RESET* to clear previously defined style keywords.

Flags can be:

INDENTTHIS
: Indent code starting with the line beginning with the *keyword*

INDENTNEXT
: Indent the code following a line beginning with the *keyword*

INDENTNEXT1
: Indent just one line following a line beginning with the *keyword*

INDENTNEXT2
: Indent the next two lines following a line beginning with the *keyword*

UNDENTTHIS
: Undent lines beginning with the line that begins with the *keyword*

UNDENTNEXT
: Undent lines beginning with the line after the line that begins with the *keyword*

UNDENTDBL
: Double the normal undent, i.e. undent twice the *code_indent* value

PUSH
: Save the current indentation on a LIFO stack

POP
: Restore the indentation to the previously PUSHed value

Note the style formatting uses the utilities DLL and is unsupported in the DOS version.

User Profile Extension Customisation

Keyword	Text	Default	Meaning
extension	ext1,ext2,...	N/A	Defines a list of extensions for which the following definition(s) apply. Up to 50 file extensions may be defined in the entire profile. If *ext1* is ***** the following information is taken to be the default for any file extension. The supplied list may represent either operating system file extensions or VM host filetypes.
file_pattern	ext1,ext2,...	N/A	A synonym for the *extension* keyword, which is more meaningful when one or more of the "extensions" represent a full filename. If a filename is required in the list, it must be followed by a

			dot (.), even if it has no extension.
a−pfline	"text"	" 2=Comp 3=Sync 4=Merge 9=MDY "	The text to display on the bottom line of the screen when either Alt key has been pressed and held down.
alt_highlight _kw	word1,word2,...	_RESET	Defines an alternate set of keywords which should be highlighted on the screen with the *ALT_KEYWORDS* colour. Refer to Keyword Highlighting for details.
APIFile	<d:\path\> fn.ext	""	Defines the file which contains a list of API expansion text. Required to turn on the API Expansion feature. If no path information is specified, the file is assumed to be in the directory specified in the *XPATH* if available, or in the current directory if not.
autosave	n	0	Defines the number of alterations that can be made to the file before it is automatically saved to disk.
base_extension	extname	"*"	Identifies the previously defined extension data that should be used as the initial data for the current extension. This statement should be located immediately after the *extension* keyword.
c−pfline	"text"	" 3=UC/m 4=LC/m 5=MC/m 7=ShLf 8=ShRt 9=YMD 10=C/Wd 11=Fnc 12=Rng"	The text to display on the bottom line of the screen when either Ctrl key has been pressed and held down.
code_indent	n	2	Defines the number of spaces to indent code on a new line. This value will only be used if the cursor is being moved to a blank line and the first or last character on the previous line is a left brace ({) character.
code_type	text	NONE	Identifies the type of source code contained in the file. Any text is valid.

comment_ anycolumn	true/false	true	Controls whether a comment may be located in any column in the file, or must be located in the *comment_column* column.
comment_column	nn<,t>	40,8	Controls the formatting of inline comments. By default, inline comments that are to be left−aligned will be formatted so that the beginning of the comment is located in column 40 of the file. This option allows you to change that column to *nn*. If the code portion of a line extends past column *nn*, the comment will be aligned on the next available multiple of *t* bytes.
comment_escape	c	""	If the *comment_prefix* string is prefixed by *c*, the string is NOT treated as a comment but as part of the regular text.
comment_ formatting	Flag bits	none	Defines the initial formatting of comments. See Inline Comment Formatting Control for definitions of the various options.
comment_prefix	string	""	Defines the text which identifies the beginning of a comment. This text is used when formatting comments. It also identifies a comment for highlighting purposes. Note that some of the *comment_formatting* flags control exactly how the *comment_prefix* is interpreted.
comment_suffix	string	""	Defines the text which identifies the end of a comment. Note that quotes are only necessary if the text contains either leading or trailing blanks.
conditional_prefix	string	""	Defines a string which identifies the beginning of a conditional string. This string is used by the UPPERCASE option of the comment formatting logic to bypass automatic code upper casing.
conditionals	str1,str2,...strN	"#if 0, #else, #endif"	Defines strings which are to be used with the

			CONDITIONAL command to input conditional compilation *IF*, *ELSE*, and *END* commands. These strings are also used for "bracket" matching; for this function more than one *str2* string may be specified; see Conditional Strings.
end_blanks	on/off	off	When ON, allows trailing blanks to be added at the end of any line. The blanks will be saved on disk as part of the line.
eof_text	text	"==== End Of File ===="	The text to be displayed to mark the end of the file data
expand_keyword	text	_RESET	Defines a syntax expansion keyword. See Syntax Expansion for details.
expand_macro	text	N/A	Defines the name of a macro which will receive control when a syntax expansion keyword is recognised. See Syntax Expansion for details.
expand_replace	text	N/A	Defines a syntax expansion replacement line. See Syntax Expansion for details.
expand_tabs	on/off	on	Controls expansion of tab (hex 09) characters to blanks. When on, blanks are inserted so that the column is moved to the next position that is an even multiple of eight.
fieldtemplate	<c1 [U P]> <c2 [U P]> ...	""	Specifies a default field template which will be applicable to all lines in the file. See FIELDTEMPLATE for more information on fieldtemplate settings.
findword_anycase	on/off	off	Allow finding the word under the cursor in mixed case with the FIND_WORD function. The default value of OFF forces an exact case search.
flowtonew	on/off	off	If ON, flowed text is inserted as a new line; if OFF, flowed text is inserted at the beginning of the next line.

funcname_chars	"charlist"	""	Defines characters which are used in addition to the default set (alphanumerics plus the underscore character) to recognise the beginning and end of a function name in function recognition and highlighting.
function_header	true/false	false	If *true*, scrolling to the next or previous function will move to possible header text above the actual header definition line. If set to the default of *false*, function scrolling will move directly to the function definition line.
function_id	"keyword", flags, <WORDNUM=w,> <STARTLINE=s,> <MAXLINES=m,> <COLUMN=c <&c2>,> <NAME_OFFSET =n,> <NAME_LENGTH =l>;	_RESET	Defines keywords which are used by the editor to identify functions for a given filetype. See Code Functions List for more information about the parameters.
helpfile	fn	""	Defines *fn* as an override to the default, globally defined help file, for just the current filetype.
highlight _keyword	word1,word2,...	_RESET	Defines keywords which should be highlighted on the screen with the *KEYWORDS* colour. Refer to Keyword Highlighting for details.
highlight _quotes	true/false	true	Specifies whether strings which are delimited by either single or double quotation marks should be highlighted with the *quotes* colour.
highlight_tags	"prefix1, suffix1; prefix2, suffix2; ...prefixn, suffixn;"	""	Defines the beginning and ending text that identifies tags which are to be highlighted. Any text on a line between *prefixn* and *suffixn* will be highlighted with *keywords* emphasis. Tag strings must come in pairs, and are separated by commas and semi−colons. To use either a comma or semi−colon as part of a tag string, insert a backslash immediately before the character. To really enter a backslash, escape

			it by entering two backslashes in a row. To allow any alphabetic character to be contained in a prefix string, use the special character sequence "a−z".
indent_keywords	"kw1,kw2,...kwn"	""	Defines a set of comma−delimited words which cause the cursor to be indented by the *code_indent* number of spaces, if it is moved to a blank line and the previous line begins with one of the specified keywords.
input_keywords	"kw1,kw2,...kwn"	""	Defines a set of keywords, delimited by commas, which trigger the *input_macro* if found at the beginning of a line when a new line is input. This check is skipped if the *INPUT* command is issued from within a macro.
input_macro	"macroname <parms>"	""	The name of the Rexx macro to be executed if a new line is added and the current line begins with one of the *input_keywords*. The indentation and the contents of the current line are supplied to the macro as parameters.
key XX	command	varies	Defines the XX key to the specified *command*, for the current file extension only. Usually used as an override to a default key.
keyword_case	exact/any	any	If *exact*, highlight keywords must be found exactly as typed to be highlighted; if *any*, a case−insensitive search will be used to find highlight keywords on a line.
keyword_trans	upper/lower/ mixed/AsIs	AsIs	Controls keyword translation. Possible values are *UPPER*, *lower*, *Mixed*, and *AsIs*, where *AsIs* is the default. Any changed lines containing words from either the *highlight_keyword* or *alt_highlight_kw* lists will have the keywords translated into the specified case.
margins	n1 n2 n3 n4	1 100 1 77	Defines formatting margins. *n1* and *n2* define the left and right

			text margins for auto−flow. *n3* and *n4* define the desired block comment margins.
openfile_id	"keyword", WORDNUM=w, <PATH=p,> <NO_BLANKS,> <PATHSEARCH =ps,> <DEFAULT_EXT =xxx>;	_RESET	Defines keywords which are used to provide a filename for the OPENFILE function. See Customising the OpenFile Function for more information about the parameters.
pfline	"text"	"F1=Hlp 2=SpJn 3=Quit 4=Sav 5=Nm 6=New 7=PgUp 8=PgDn 9=Undo 10=Redo 11=Prv 12=Nxt"	The text to display on the bottom line of the screen when no shift keys are active.
quick_comment	string	""	A string which will be converted to the defined *comment_prefix* and *comment_suffix* strings. A commonly used quick_comment string is //.
save_options	string	""	Provides default options to be used with the SAVE and FILE commands. Any combination of valid save options may be specified, but command line options will override these defaults.
s−pfline	"text"	"F1=ScrL 2=ScrR 3=ScrU 4=ScrD 5=CtrLnVrt"	The text to display on the bottom line of the screen when either Shift key has been pressed and held down.
shadow	on/off	on	Sets the initial value of the SHADOW setting for this file type.
sort_funclist	on/off	on	Controls whether the function list popup window will be sorted.
split_align_paren	on/off	off	If set to ON, split new text will be aligned beneath an open

			parenthesis from the previous line, if found.
styleword	keyword, flags	undefined	Sets a keyword and the formatting style for style formatting. See Style Formatting for details.
syntax	on/off	on	Sets the initial value of syntax assistance for this file type.
tabs	n1 <n2 n3 n4...> <,m>	8	Defines the initial tab stops. If *n1* is the only number specified, tabs will be set every *n1* spaces, beginning in column 1. If the *,m* parameter is specified, soft tab settings will continue from the last hard tab stop specified, every *m* spaces. Note that tab stops must be numeric and they must be specified in ascending sequence. There is a limit of 31 hard tab settings for each file type.
tag_end	text	""	The string that indicates the end of a highlight_tag. For example, for HTML markup, a beginning tag is indicated with <tag>, and the corresponding ending tag is </tag>. For this filetype, the tag_end string will be "/".
tof_text	text	"==== Top Of File ===="	The text to be displayed to mark the beginning of the file data
undent_keywords	"kw1,kw2,...kwn"	""	Defines a set of comma−delimited words which cause the cursor to be undented by the *code_indent* number of spaces, if it is moved to a blank line and the previous line begins with one of the specified keywords.
undo_limit	N	−1	Maximum number of undo lines to save. −1 means save all changes until the file is saved to disk. 0 means save no undo lines. N means save the previous N changed lines, although this option is not yet implemented.
wrap	on/off	on	Sets the initial value of the WRAP setting for this file type.

Commands and Macro Support

- Macro Debugging
- EXTRACT Command
- Locate Text
- Change Text
- Popup Windows
  - List Box
  - Message Box
  - Prompt
  - Password Prompt
- Editor Commands
  - ACCENT
  - ADD
  - ALL
  - ALT
  - APPEND
  - ASCII
  - AUTOBOOKMARK
  - AUTOSAVE
  - BACKSPACE
  - BACKTAB
  - BACKWARD
  - BOOKMARK
  - BOTTOM
  - BOTTOMSCREEN
  - BROWSE
  - C, CHANGE
  - CASECHAR
  - CASEWORD
  - CD
  - CENTRELINE
  - CENTRETEXT
  - CHANGES
  - CLIP
  - CMDLINE
  - CMDTEXT
  - COMMAND
  - COMMENTLINE
  - COMMENT_STYLE
  - COMPARE
  - CONDITIONAL
  - COPYLINE
  - COPYTOCMD
  - COUNT
  - CURR_ALT_PFLINE
  - CURR_CTRL_PFLINE
  - CURR_PFLINE
  - CURR_SHIFT_PFLINE
  - CURSOR

◊ DATE
◊ DELCHAR
◊ DELDUPES
◊ DELETE
◊ DELSYM
◊ DELWORD
◊ DIAG
◊ DOWN
◊ DUPLICATES
◊ E, EDIT, X
◊ EA
◊ EOF_TEXT
◊ ERASEEOL
◊ ERRORS
◊ EXCLUDE
◊ EXITRC
◊ EXPAND
◊ EXT
◊ EXTRACT
◊ FFILE
◊ FIELDTEMPLATE
◊ FILE
◊ FIND_WORD
◊ FORWARD
◊ FT
◊ FUNCWIN
◊ GET
◊ HELP
◊ HEX
◊ HIDEFILE
◊ INPUT
◊ INPUT_ERRORLINE
◊ INSMODE
◊ JOIN
◊ KEY
◊ KEYIN
◊ KEYIN_NAME
◊ KEYS_PLAY, PLAYBACK
◊ KEYS_RECORD
◊ KEYS_WRITE
◊ L, LOCATE
◊ LINECOLOUR
◊ LINEFIELDS
◊ LINEMACRO
◊ LINEND
◊ MA, MARGINS
◊ MACRO
◊ MARK
◊ MATCH
◊ MESSAGEBOX
◊ MSG

◊ MSGMODE
◊ NAME
◊ NEXT, NEXT_FILE
◊ NEXT_ERROR
◊ NEXT_FUNC
◊ NEXT_PARA
◊ NEXT_SENTENCE
◊ NEXT_SYM
◊ NEXT_WORD
◊ NOP
◊ NUMFILES
◊ OPENFILE
◊ PAGEDOWN
◊ PAGEUP
◊ PASSWORD
◊ PFLINE
◊ PRESSKEY
◊ PREVIOUS_FILE
◊ PREVIOUS_FUNC
◊ PREVIOUS_PARA
◊ PREVIOUS_SYM
◊ PREVIOUS_WORD
◊ PROMPT
◊ PUT
◊ QQ, QQUIT
◊ QUIT
◊ REDO
◊ REFORMAT
◊ REFRESH
◊ RENAME
◊ REPEAT_FIND, REPFIND
◊ REPLACE
◊ RESOLVE_FN
◊ RESTORE_FIND
◊ REVERSE_FIND
◊ RINGWIN
◊ SAVE
◊ SCROLL
◊ SETRESULT
◊ SHADOW
◊ SHADOWTEXT
◊ SHELL
◊ SHOW
◊ SHOWLINE
◊ SORT
◊ SPAN
◊ SPLIT
◊ SPLITJOIN
◊ STATUS
◊ STATUSTEXT
◊ STYLE

- ◊ SYNTAX
- ◊ TAB
- ◊ TABLINE
- ◊ TABS
- ◊ TIMER
- ◊ TITLE
- ◊ TOFEOF
- ◊ TOF_TEXT
- ◊ TOP
- ◊ TOPLINE
- ◊ TOPSCREEN
- ◊ UNDO
- ◊ UNDO_BLOCK
- ◊ UNDO_LIMIT
- ◊ UP
- ◊ WINDOW
- ◊ WINLINE
- ◊ WINSELECT
- ◊ WINSORT
- ◊ WINWAIT
- ◊ WRAP
- ◊ nnn
- ◊ /text</< |/& /text2/>>

♦ Command Summary
- ◊ Command Summary (A−H)
- ◊ Command Summary (I−P)
- ◊ Command Summary (Q−Z)

X2 provides the capability to define your own commands through a macro facility. Macros must be written in Rexx, and have a file extension of .X or .CMD. If your macros are installed somewhere on your PATH, they can be invoked directly by simply typing the macro name. If not on your path, the path to the macro must also be specified.

X2 supports Classic and Object Rexx on OS/2, and Object Rexx under Windows 95, Windows NT, Linux, and AIX. The Regina Rexx interpreter is supported as an alternate for Windows 95 and Windows NT, and the only interpreter under Sun Solaris. Rexx is unsupported in the DOS version.

When invoking a macro, type either its name or "MACRO name" on the command line. Note that you should **not** include the file extension unless it is a .CMD file − the X2 Editor automatically suffixes the macro name with .X before searching for it on disk. On the other hand, macros with an extension of .CMD **must** include the file extension on the command line if they are to use the X addressing environment to communicate with the editor.

When a macro is started, the default Address for command resolution is X. Commands that you issue to the operating system environment through the editor will work, but it is recommended that you use *Address CMD* when you wish to execute a shell function. You may use the *Address()* function in a Rexx program to determine the current address environment. Note that if you are using Regina Rexx, *Address CMD* will not work, and you should use *Address SYSTEM* instead.

When you issue a command on the command line, it is first checked against the builtin editor commands. If not found, it is assumed to be a macro name, and the command is issued as a macro. If the macro was not found, it is issued to the operating system as a shell command.

You can specify parameters to a macro on the command line. Facilities exist to communicate with the editor by issuing any editor command from the macro.

If a macro supplies a numeric return code on the Exit statement, this return code will be passed to the calling program, which may be another macro.

Note that macro names can only contain alphanumeric characters, i.e. any of the letters a−z and numbers 0−9. This is because X2 terminates a command at the first non−alphanumeric character. The first non−alphanumeric character becomes the start of the command arguments, if any.

Macro Debugging

If desired, macros can be debugged by inserting Trace statements in the code and watching the results on the screen. If there are many statements they may scroll off the screen and be difficult to read. Any trace statements may be re−directed to a file with the greater−than (>) sign, as follows:

```
macroname parms >macro.out
```

In the above example, *macroname* would be executed normally, with the specified parms. If *macroname* contains any Rexx Trace statements, the results of these statements will be written to *macro.out* instead of to the screen.

IMPORTANT NOTE: If debugging and using interactive tracing, i.e. the Trace ? format, it is possible to cause the session to hang if an invalid Rexx statement is issued. This is a known problem; unfortunately there is no known solution at present.

EXTRACT Command

The EXTRACT command is particularly useful for gaining access to editor variables from macros. If an Extract command is successful, it will set the results in Rexx stem variables. In all cases, stem.0 contains the number of variables set. Use the following syntax:

```
EXTRACT /opt/
```

The table below contains a list of all the available Extract options and the variables they will set. The possible return codes are:

0
: Successful completion

−1
: Insufficient memory

−6
: Invalid parameter

15
No file left in edit ring

Extract Options

This table shows all the available Extract options, and the Rexx variables which will be set in response to the Extract command.

Option	Variable	Contents
ALT	ALT.0 ALT.1 ALT.2	The number of variables returned (2) The number of changes made since the last autosave The number of changes made since the last save
APIFILE	APIFILE.0 APIFILE.1	The number of variables returned (1) The APIFile name for the current file
AUTOSAVE	AUTOSAVE.0 AUTOSAVE.1	The number of variables returned (1) The autosave value
BOOKMARK	BOOKMARK.0 BOOKMARK.n	The number of variables returned The nth bookmark in line,column format, or 0,1 if undefined
CD	CD.0 CD.1	The number of variables returned (1) The current directory
CMDLINE	CMDLINE.0 CMDLINE.1	The number of variables returned (1) 1 if the command line is active, or 0 if the cursor is in the file.
CODE_TYPE	CODE_TYPE.0 CODE_TYPE.1	The number of variables returned (1) The code type, as inferred from the filetype. Possible values are ASM, BASIC, C, HELP, NETREXX, REXX, SCRIPT, and NONE.
COLOUR areaname	COLOUR.0 COLOUR.1 COLOUR.2 COLOUR.3	The number of variables returned (3) The colour area name The foreground colour The background colour
COLOURS	COLOURS.0 COLOURS.n	The number of variables returned The nth colour definition, in the format areaname foreground ON background
COMMENTS	COMMENTS.0 COMMENTS.1 COMMENTS.2 COMMENTS.3	The number of variables returned (3) The comment prefix string defined in the user profile. The comment suffix string. The quick comment string.
CURLINE	CURLINE.0 CURLINE.1 CURLINE.2 CURLINE.3 CURLINE.4 CURLINE.5	The number of variables returned (7) The contents of the current file line, with nulls converted into the null_char 1 if the line is visible, or 0 if it is hidden. The number of hidden lines The shadow text for the current line. 1 if the line is an error line, 0 if

		not.
	CURLINE.6	Up to 40 characters of the line in hexadecimal format
	CURLINE.7	1 if a popup window is active, or 0 otherwise
CURSOR	CURSOR.0	The number of variables returned (4 or 5)
	CURSOR.1	The cursor row in the file.
	CURSOR.2	The cursor column in the file.
	CURSOR.3	1 if the line is visible, or 0 if it is hidden.
	CURSOR.4	1 if the column is visible, or 0 if it is hidden.
	CURSOR.5	The numeric offset from the beginning of the file if hex display is on.
ESCAPE	ESCAPE.0	The number of variables returned (1)
	ESCAPE.1	The escape character.
EXT	EXT.0	The number of variables returned (1)
	EXT.1	The current file extension
FIELDTEMPLATE	FIELDTEMPLATE.0	The number of variables returned (1)
	FIELDTEMPLATE.1	The field template for the current file
FILEINFO	FILEINFO.0	The number of variables returned (15)
	FILEINFO.1	The current file name
	FILEINFO.2	The number of lines in the file
	FILEINFO.3	The current file alteration count
	FILEINFO.4	The current file autosave count
	FILEINFO.5	The current edit mode, "BROWSE" or "EDIT"
	FILEINFO.6	The current syntax filetype
	FILEINFO.7	The current file WRAP setting
	FILEINFO.8	The current file SHADOW setting
	FILEINFO.9	The current file bookmark numbers. If there are too many bookmarks to fit into the available space (80 bytes), then the list will be truncated.
	FILEINFO.10	The current file comment markers
	FILEINFO.11	The current file cursor position
	FILEINFO.12	The current file margins
	FILEINFO.13	The current file tab settings
	FILEINFO.14	The current file code type
	FILEINFO.15	"NEW" if the file is new, "OLD" otherwise
FILENAME	FILENAME.0	The number of variables returned (1)
	FILENAME.1	The name of the currently edited file.
FIND	FIND.0	The number of variables returned (5)
	FIND.1	The text of the find string(s)
	FIND.2	The find arguments
	FIND.3	The name of the file containing the last match. Set to "" at editor initialisation time, and if the file is quit.
	FIND.4	The row number of the last successful match
	FIND.5	The column number of the last successful match
	FIND.6	The string that was found

FLSCREEN	FLSCREEN.0 FLSCREEN.1 FLSCREEN.2	The number of variables returned (2) The line number of the first line displayed on the screen. The line number of the last line displayed on the screen.
FT	FT.0 FT.1	The number of variables returned (1) The current syntax filetype. This is either the file extension or the result of the FT command. It will always be upper case in order to allow easier comparison operations in macros.
FUNCTION	FUNCTION.0 FUNCTION.1 FUNCTION.2	The number of variables returned (2) The text of the current function definition line. The line number of the current function.
HELPFILE	HELPFILE.0 HELPFILE.1	The number of variables returned (1) The help file name
INSERT	INSERT.0 INSERT.1	The number of variables returned (1) 1 if the cursor is in INSERT mode, or 0 if it is in overstrike (REPLACE) mode.
KEY keyname	KEY.0 KEY.1	The number of variables returned (1) The current setting for keyname, where keyname is any key that may be modified through the user profile.
KEYPRESS	KEYPRESS.0 KEYPRESS.1	The number of variables returned (1) The name of the next depressed key. Will wait until the user presses a key before returning.
LASTMSG	LASTMSG.0 LASTMSG.1	The number of variables returned (1) The last displayed message
LINE <n>	LINE.0 LINE.1 LINE.2 LINE.3 LINE.4 LINE.5 LINE.6 LINE.7	The number of variables returned (7) The contents of the file line n, where n defaults to the current line 1 if the line is visible, or 0 if it is hidden. The number of hidden lines The shadow text for the line. 1 if the line is an error line, 0 if not. Up to 40 characters of the line in hexadecimal format 1 if a popup window is active, or 0 otherwise
LINECOLOUR	LINECOLOUR.0 LINECOLOUR.x	The number of variables returned, which varies according to the number of line segments in the line The highlighting attributes for a line segment, in the format start_col end_col index, where start_col is the starting column number, end_col is the ending column number, and index is an index to the extract /colours/ array. If the linecolour command has been used on the current line, index is replaced by

		the fg on bg format.
LINEFIELDS	LINEFIELDS.0 LINEFIELDS.1	The number of variables returned (1) The line fields for the current line
LINEND	LINEND.0 LINEND.1	The number of variables returned (1) The current LINEND setting, and the LINEND character
MARGINS	MARGINS.0 MARGINS.1 MARGINS.2 MARGINS.3 MARGINS.4	The number of variables returned (4) The left file margin The right file margin The left paragraph formatting margin The right paragraph formatting margin
MARK	MARK.0 MARK.1 MARK.2 MARK.3 MARK.4 MARK.5 MARK.6 MARK.7	The number of variables returned. 0 if there is no mark, or 7 if a mark exists. The marked file name. The first marked line number. The last marked line number. The first marked column number. This will be 0 if the mark is a line mark. The last marked column number. This will be 0 if the mark is a line mark. 1 if the marked file is the current file, or 0 otherwise. 1 if the mark is a word mark, or 0 otherwise.
MARKTEXT	MARKTEXT.0 MARKTEXT.x	The number of variables returned, which equals the number of visible marked lines The contents of the Nth marked line. If the mark is a block mark, only the marked portion of the line is returned.
MSGMODE	MSGMODE.0 MSGMODE.1	The number of variables returned (1) The current MSGMODE setting.
NAME	NAME.0 NAME.1	The number of variables returned (1) The name of the currently edited file.
OS	OS.0 OS.1 OS.2 OS.3 OS.4	The number of variables returned (4) The operating system name − "OS/2", "Windows NT", "Windows 95", "AIX", "Linux", "SunOS", or "HP−UX" The operating system version The directory path separator character The separator character between multiple paths
PFLINE	PFLINE.0 PFLINE.1	The number of variables returned (1) The current PF display text when no shift keys are active.
RING	RING.0 RING.1 RING.n	The number of variables returned, which is also the number of files in the edit ring. The first file in the ring, which is also the current file. The nth file in the ring
SCREEN	SCREEN.0 SCREEN.1 SCREEN.2	The number of variables returned (2) The number of rows in the screen The number of columns in the screen
SHADOW	SHADOW.0	The number of variables returned (1)

	SHADOW.1	ON if the SHADOW setting is ON, or OFF if it is OFF.
SHADOWTEXT	SHADOWTEXT.0 SHADOWTEXT.1	The number of variables returned (0 or 1) The shadow text for the current line, if set.
SIZE	SIZE.0 SIZE.1	The number of variables returned (1) The number of lines in the current file
STATUSTEXT	STATUSTEXT.0 STATUSTEXT.1	The number of variables returned (1) The statustext template for the current file
TABS	TABS.0 TABS.1	The number of variables returned (1) The tab settings for the current file
VERSION	VERSION.0 VERSION.1	The number of variables returned (1) The editor version number
WRAP	WRAP.0 WRAP.1	The number of variables returned (1) ON if the WRAP setting is ON, or OFF if it is OFF.
X2PATH	X2PATH.0 X2PATH.1	The number of variables returned (1) The XPATH/X2PATH setting

Locate Text

To search for text in a file, use the **L** command. This command has the following syntax:

```
<L <~>/text</< |/& <~>/text2/> -mecflqs>
```

where:

L

Indicates the locate command. May be omitted if the search delimiter character is a slash (/).

~

Indicates negation. If specified, lines that **do not** contain *text* will be found.

/

The search delimiter. This character marks the beginning of the search string. If it occurs twice in the search string, the second occurrence marks the end of the search string.

text

The text to search for.

text2

An optional second string to search for. If the two strings are separated by an OR character (|), the next occurrence of **either** *text* or *text2* will be found in the file. If the two strings are separated by an AND character (&), the next line containing **both** *text* and *text2* will be found.

-mecflqs

Search options

-

Backwards search

m
: Only scan marked area for *text* or *text2*

e
: Case must match exactly

c
: Case−insensitive search (the default)

f
: *text* must be located at the beginning of a line

l
: *text* must be located at the end of a line

q
: Don't highlight search result

s
: Symbol search. The *target* cannot be part of another word

The *l* option is useful when used with the *m* option for locating blank lines. If a block mark is set in column one of the file, the following command will locate the next blank line in the file:

```
L // lm
```

When specifying search text, either normal ASCII characters may be used, or a hexadecimal representation may be used. When specifying search strings in hex, use the following syntax:

```
'hhhh'x
```

The following rules apply to hex strings:

◊ They must be delimited with single quote (') marks
◊ The second quote must be immediately followed by the character 'x' or 'X'
◊ There must be an even number of digits
◊ Each digit must be in the range 0−9 or A−F

The following return codes may be received from the Locate command:

0
: Successful completion

−1
: Insufficient memory

−6
: Invalid parameter

7
: Target not found

15
: No file left in edit ring

Change Text

To change the text in a file, use the **C** command. This command has the following syntax:

```
C /from/to</ mecflps*N>
```

where:

/
: The search delimiter. This character marks the beginning and end of the search and replacement strings.

from
: The text to search for. The text may be specified in hex notation, see Locate Text for details.

to
: The replacement text. The text may be specified in hex notation, see Locate Text for details.

*mecflps*N*
: Options

  m
  : Only scan marked area for *from*

  e
  : Case must match exactly

  c
  : Case−insensitive search (the default)

  f
  : *from* must be located at the beginning of a line

  l
  : *from* must be located at the end of a line

  p
  : Preserve case. If *from* is all upper case, the replacement string will be converted to upper case. If *from* is all lower case, the replacement string will be converted to lower case. If *from* contains both upper and lower case letters, it will be replaced by *to* directly.

  s
  : Symbol search. The *target* cannot be part of another word

  *
  : Change all occurrences from the current position to the bottom of the current file

  N
  : Change the next N number of occurrences from the current position to the bottom of the current file, without prompting for change confirmation

If the * (change all) option is not used, the first occurrence of *from* will be found and a message asking Yes/No/Last/Go/Quit/Abort will be displayed. After the first change an additional option of Undo will be present. Press the first letter of the option you wish, where the options are as follows:

Yes
: Make the change and proceed to the next occurrence of *from*

No
: Skip this item and proceed to the next occurrence of *from*

Last
: Make the change and terminate the change command

Go
: Change the remaining occurrences of *from* into *to* without further prompting

Quit

Terminate the change command without changing the current occurrence of *from*. The file position will be restored to its original position. The same result will occur if you press the Escape key in response to the prompt.

Abort

Terminate the change command without changing the current occurrence of *from*. The cursor remains at its current (new) position.

Undo

Undo the previous change and position the cursor at the previous occurrence of *from*.

The following return codes may be received from the Change command:

0

Successful completion

−1

Insufficient memory

−6

Invalid parameter

7

No changes made

12

File is read only

15

No file left in edit ring

Popup Windows

The X2 Editor provides several facilities for the macro writer to communicate with the user; one of the most useful ways is through popup windows. Popup windows are defined as windows which appear in the middle of the editor screen to display information and receive a response. The X2 Editor provides the following types of popup windows:

- List Box
- Message Box
- Prompt
- Password Prompt

List Box

You can display a list of items for user selection or reference. This type of popup is the most complicated, and typically uses several commands to set up and display the window. These are:

- WINDOW
  Used to initialise the window's size, title, and the expected number of entries in the list
- SETRESULT
  Used to set the Rexx variable *result*; see below for an example of how to use this command with popup windows
- WINLINE
  Adds a line of text to a window
- WINSORT

Sorts the window lines

◊ WINWAIT

Displays the window and waits for user response

The following example displays a popup window and waits for user response. If a line is selected, the macro is re−invoked with a parameter which determines subsequent action. This old method of using popup windows is a bit cumbersome and difficult to follow, as it requires the macro to be invoked twice and parameters must be passed between the two invocations.

```
/* */
  Parse Source os invoke sourcefn .
  Parse Var sourcefn sourcefn '.' .
  Parse Arg opt .
  Select
    When opt = 'One'
      Then 'MESSAGEBOX You selected line 1'
    When opt = 'Two'
      Then 'MESSAGEBOX You selected line 2'
    Otherwise Do
      'WINDOW 2 40 2 Test Window'
      'WINLINE Line 1\nMACRO' sourcefn 'One'
      'WINLINE Line 2\nMACRO' sourcefn 'Two'
    End
  End
Exit
```

The next example uses SETRESULT and WINWAIT to improve the logic flow of the macro. WINWAIT causes the window to be displayed while the calling macro is still active; when a user selects an entry the variable *result* will be set to a parameter which can be used further down in the program.

```
/* */
  'WINDOW 2 40 2 Test Window'
  'WINLINE Line 1\nSETRESULT One'
  'WINLINE Line 2\nSETRESULT Two'
  'WINWAIT'

  Select
    When result = 'One'
      Then 'MESSAGEBOX You selected line 1'
    When result = 'Two'
      Then 'MESSAGEBOX You selected line 2'
    Otherwise Nop    /* User probably escaped*/
  End
Exit
```

Message Box

A message box is used simply to provide information to a user, and optionally to retrieve a single key in response. The example below shows a message box which displays the text "Hello" centred on line one, and "World" left−aligned on the second line. The return code will be non−zero if the user presses escape, and the key they pressed is returned in the special Rexx variable *result*.

```
/* */
  'EXTRACT /ESCAPE/'
  'MESSAGEBOX Hello' || escape.1 || 'NWorld'
  'MSG Messagebox return code was' rc 'and the result was' result
```

Prompt

A prompt is a special kind of message box, where space is provided for the user to enter text as a response. The response is returned in the Rexx variable *result*.

```
/* */
  'PROMPT Please enter your name'
  If rc = 0
    Then 'MSG Hello' result
```

In a prompt window, the Tab key acts as it does on the command line: the previous word on the line is expanded to a filename from disk.

Password Prompt

Use the password command to retrieve information from the user, but not display it on the screen. Instead of echoing the user input inside the message box, asterisks are used as placeholders. Just like the Prompt command, the response is returned in the Rexx variable *result*.

```
/* */
  'PASSWORD Please enter your password'
  If rc = 0
    Then 'MSG You can trust me with' result
```

Editor Commands

This section contains a description of all commands available from the command line. Some of these commands are only really useful from macros, although all can be issued from the editor command line. Many are not available in the DOS version to save on space. These are detailed in Differences in the DOS Version.

ACCENT

Syntax: ACCENT [ACUTE CEDILLA CIRCONFLEX GRAVE TILDE TREMA UMLAUT]

Description: Input an accented character. This command prompts the user for a key (one of aeiouAEIOUcCnN and space, depending on the option), and enters the appropriate accented letter. This command is only supported under Windows, and is intended to be assigned to a key, usually an AltGr key on an NLS−enabled keyboard.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

ADD

Syntax: ADD

Description: Add a column of marked numbers

Default Key: a−padplus

Return Codes:

0
: Successful completion

15
: No file left in edit ring

ALL

Syntax: ALL </text</ +−mecfls~>

Description: Display only those lines which contain *text*.

Options:

+
: Add lines to current display

−
: Subtract lines from current display

m
: Only scan marked area for *text*

e
: Case must match exactly

c
: Case−insensitive search

f
: *text* must be located at the beginning of a line

l
: *text* must be located at the end of a line

s
: Symbol search. The target *text* cannot be part of another word

~
: Reverse the current include/exclude settings. If specified it must be the first and only parameter, i.e. *text* must be omitted.

Default Key: c−u

Return Codes:

0
: Successful completion

−1
: Insufficient memory

−6
: Invalid parameter

7
: No matches found

15
: No file left in edit ring

ALT

Syntax: ALT altcount <chgcount>

Description: Set the alteration count(s) for the file. *altcount* specifies the number of changes since the last autosave. *chgcount*, if specified, is the number of changes since the last save. If *chgcount* is 0, QUIT will exit the file without confirming potential data loss. Otherwise, the file is assumed to have been modified and the QUIT function will ask for confirmation.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

APPEND

Syntax: APPEND fn

Description: Appends all visible lines in the current file to the supplied *fn*. Any lines which have been excluded from the display are not copied to the new file. If a mark exists in the current file, only visible lines in the marked area are copied.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

ASCII

Syntax: ASCII

Description: Escape to enter characters in ASCII mode.

Default Key: a−x

Return Codes:

0
: Successful completion

15
: No file left in edit ring

AUTOBOOKMARK

Syntax: AUTOBOOKMARK

Description: Move to an automatic bookmark position. Displays a window showing all the automatic bookmarks for selection − selecting an item will move the cursor to that position in the current file. Note that automatic bookmarks must be turned on in the profile, see Automatic Bookmarks.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

AUTOSAVE

Syntax: AUTOSAVE <n>

Description: Set the autosave value for the file. Every time the alteration count reaches this value, the file will be automatically saved to a temporary file on disk. The temporary filename is the same as the current filename, except that the extension is replaced with the

first value from (000, 001, 002, ..., 009) that will create a unique filename. If *n* is 0, the file will never be auto−saved. If *n* is omitted entirely, the file will be autosaved immediately.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

BACKSPACE

Syntax: BACKSPACE

Description: Move the cursor one position to the left, and delete the character in the new position. If the cursor is positioned on the first file column, the current line will be joined with the previous line and the cursor moved to the intersection point.

Default Key: backspace

Return Codes:

0
: Successful completion

15
: No file left in edit ring

BACKTAB

Syntax: BACKTAB

Description: Move the cursor to the previous tab position

Default Key: backtab

Return Codes:

0
: Successful completion

15
: No file left in edit ring

BACKWARD

Syntax: BACKWARD

Description: Scroll the screen a full page towards the top of file. The cursor position on the screen will be unchanged, unless the first page is reached.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

BOOKMARK

Syntax: BOOKMARK <n1 <n2<,n3>> SET GO PUSH>

Description: Set or move to a bookmark position. *n1* specifies the bookmark number. If *n2* is specified, it indicates the new line setting for the bookmark; otherwise, the cursor will be moved to the bookmark number *n1*. If *n3* is also specified, it indicates the new column setting for the bookmark. If *n3* is not specified, the column defaults to column 1.

If no parameters are supplied, or the *SET* parameter is used, a bookmark will be set at the current cursor position. Note that if quickmarks are on the first bookmark is set; if multiple bookmarks are allowed, a window is displayed to prompt for the bookmark number.

The *GO* parameter moves the cursor to a previously saved bookmark. If more than one bookmark is allowed, the user is prompted to select the correct bookmark number from a list of set bookmarks.

The *PUSH* option forces quickmarks behaviour; the current cursor position is added to bookmark number 1, and the remaining bookmarks are pushed down the stack.

Default Keys: c−b, c−g

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

22
: No bookmarks defined

BOTTOM

Syntax: BOTTOM

Description: Move to the bottom line of the file

Default Key: c−end

Return Codes:

0
: Successful completion

10
: The cursor was not moved

15
: No file left in edit ring

BOTTOMSCREEN

Syntax: BOTTOMSCREEN

Description: Move to the bottom line of the screen

Default Keys: c−pgdn, a−down

Return Codes:

0
: Successful completion

10
: The cursor was not moved

15
: No file left in edit ring

BROWSE

Syntax: BROWSE <ON OFF>

Description: Turn the browse setting on or off. If no parameters are specified, the browse setting is toggled.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

C, CHANGE

Syntax: C /from/to</ mecflps*N>

Description: Change occurrences of *from* into *to*. See Change Text for details.

Default Key: none

Return Codes: See Change Text

CASECHAR

Syntax: CASECHAR <**UPPER** LOWER>

Description: Change the case of the current character to UPPER or lower, depending on the parameter, and move the cursor one position to the right. The default parameter is UPPER.

Default Key: c−up, c−down

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

CASEWORD

Syntax: CASEWORD

Description: Rotate the case of the current word, through UPPER, Mixed, and lower. If the cursor is on a non−alphanumeric character, the case of the previous word will be changed, unless the cursor is at or near the beginning of the line, in which case the **next** word will be changed.

Default Key: c−f10

Return Codes:

0
: Successful completion

15

No file left in edit ring

CD

Syntax: CD <d:\path>

Description: Change the current drive and directory for subsequent commands. If *d:* is omitted, the drive remains the same. If *path* is omitted, the current directory for the new drive will be used. If both are omitted, the current directory information will be displayed. The original directory information will be restored upon editor termination.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

CENTRELINE

Syntax: CENTRELINE

Description: Centre the current row on the screen.

Default Key: s−f5

Return Codes:

0
: Successful completion

15
: No file left in edit ring

CENTRETEXT

Syntax: CENTRETEXT

Description: Centre text on the current line between the comment margins, or between the mark if the line is marked with a block mark.

Default Key: a−t

Return Codes:

0
: Successful completion

15
: No file left in edit ring

CHANGES

Syntax: CHANGES

Description: Display only those lines which have been changed in this editing session

Default Key: none

Return Codes:

0
: Successful completion

7
: No changed lines found

15
: No file left in edit ring

CLIP

Syntax: CLIP <COPY CUT PASTE <BLOCK>>

Description: Manipulate the clipboard on 32 bit Windows or Unix systems. Options:

COPY
: Copy the marked area to the clipboard. The file is not changed.

CUT
: The same as COPY, but delete the marked area after successful copy to the clipboard.

PASTE <BLOCK>
: Copy any lines that may be in the clipboard into the current file. Lines are inserted after the current cursor position by default, but will be inserted into existing lines with the BLOCK option.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

6
: No mark defined

15
: No file left in edit ring

18
: Utilities DLL not loaded

19
: No data in clipboard

CMDLINE

Syntax: CMDLINE <**TOP** BOTTOM>

Description: Set the command line position to either the top or the bottom of the screen, where TOP is the system default. If no parameter is provided, the command line setting alternates between TOP and BOTTOM. To change the default through the user profile, see Command Line Location.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

CMDTEXT

Syntax: CMDTEXT text

Description: Display *text* on the command line and position the cursor at the end of *text*.

Default Key: f6

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

COMMAND

Syntax: COMMAND internal command

Description: Execute an internal command, but bypass any possible synonym or macro resolution.

Default Key: none

Return Codes: From internal command

COMMENTLINE

Syntax: COMMENTLINE <**FULL** <c> EMPTY> <INDENT>

Description: Input a block comment line. The default *FULL* option fills the line with asterisks, while the *EMPTY* option contains blanks. The optional parameter *c* will fill the line with the character *c*. The optional parameter *INDENT* will cause the comment line to be inserted with the same indentation as the current line.

Default Keys: a−7, a−8

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

COMMENT_STYLE

Syntax: COMMENT_STYLE <ALIGN_QUICK COLSTART CONVERT_ALLQUICK CONVERT_QUICK CPP_CONDITIONAL C_CONDITIONAL IF_MODIFIED KEEP_BLANKS LEFT NO_BLOCKS RIGHT UPPERCASE>

Description: Modify the automatic comment formatting style. For a complete description of the various options, refer to Inline Comment Formatting Control.

Default Key: c−c

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

COMPARE

Syntax: COMPARE <**DIFF** SYNC MERGE>

Description: Compare two files. The DIFF option stops the cursor at the first non−matching lines, and is the default. The SYNC option will try to re−synchronise the files to common lines. The MERGE parameter copies differing lines from the current file to the next file in the ring, until a set of matching lines is found.

Default Keys: a−f2, a−f3, a−f4

Return Codes:

0
: Successful completion

−6
: Invalid parameter

7
: No matching lines found (SYNC option)

11
: No differing lines found

15
: No file left in edit ring

CONDITIONAL

Syntax: CONDITIONAL <IF ELSE END>

Description: Input conditional compilation *IF*, *ELSE*, or *END* command. The default values are *#if 0*, *#else*, and *#endif* respectively. Conditional strings may be set in the user profile with the *CONDITIONALS* keyword.

Default Key: a−0

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

COPYLINE

Syntax: COPYLINE

Description: Copy the current line. The new line is inserted immediately following the current line.

Default Key: c−k

Return Codes:

0
: Successful completion

15
: No file left in edit ring

COPYTOCMD

Syntax: COPYTOCMD

Description: Copy the current line's text to the command line. If a block mark exists on the current line, just that portion of the line will be copied.

Default Key: c−l

Return Codes:

0
: Successful completion

15
: No file left in edit ring

COUNT

Syntax: COUNT /text</< |/& /text2/> −mecfls>

Description: Count the number of occurrences of the target string(s). See Locate Text for details on specifying options.

Default Key: none

Return Codes:

0 or greater
: Successful completion. The value returned is the number of occurrences found.

−6
: Invalid parameter

15
: No file left in edit ring

CURR_ALT_PFLINE

Syntax: CURR_ALT_PFLINE text

Description: Use the new *text* for the PF display line when the Alt key is active, but only for the current file.

Default Key: none

Return Codes:

0
: Successful completion

−1
: Insufficient memory

−6
: Invalid parameter

15
: No file left in edit ring

CURR_CTRL_PFLINE

Syntax: CURR_CTRL_PFLINE text

Description: Use the new *text* for the PF display line when the Ctrl key is active, but only for the current file.

Default Key: none

Return Codes:

0
: Successful completion

−1
: Insufficient memory

−6
: Invalid parameter

15
: No file left in edit ring

CURR_PFLINE

Syntax: CURR_PFLINE text

Description: Use the new *text* for the PF display line when no shift keys are active, but only for the current file.

Default Key: none

Return Codes:

0
: Successful completion

−1
: Insufficient memory

−6
: Invalid parameter

15
: No file left in edit ring

CURR_SHIFT_PFLINE

Syntax: CURR_SHIFT_PFLINE text

Description: Use the new *text* for the PF display line when the Shift key is active, but only for the current file.

Default Key: none

Return Codes:

0
: Successful completion

−1
: Insufficient memory

−6
: Invalid parameter

15
: No file left in edit ring

CURSOR

Syntax: CURSOR <+|−>row <<+|−>col> BEGMARK CMDLINE COL1 <STAY> DATA ENDMARK EOL <STAY> NEXTLINE TOGGLE

Description: Move the cursor to the *row* and *column* specified. If a plus or minus sign is specified, the offset will be relative from the current cursor position. Instead of numeric parameters, one of the following words may be specified:

BEGMARK
: Move the cursor to the beginning of the marked area

CMDLINE
: Move the cursor to the command line

COL1 <STAY>
: Move the cursor to the first column of the command line. If it is already in column 1,

move it to column 1 of the previous line, unless *STAY* is specified.

DATA

Move the cursor to the data area

ENDMARK

Move the cursor to the end of the marked area

EOL <STAY>

Move the cursor to the end of the current line. If it is already at the end of the line, move it to the end of the next line, unless *STAY* is specified.

NEXTLINE

Move the cursor to the beginning of the next file line. If *enter_insert* is set to ON in the user profile, a new line will be inserted if insert mode is also ON.

TOGGLE

Toggle the cursor between the command line and the data area

Default Keys: end, enter, home, a−e, a−y

Return Codes:

0

Successful completion

−6

Invalid parameter

9

Cursor at TOF/EOF line

10

The cursor was not moved

15

No file left in edit ring

16

Target line is hidden

DATE

Syntax: DATE EUROPEAN LONG ORDERED SORTED USA

Description: Input the current date into the file at the current cursor location. The option controls the date format:

European

Input the date in dd/mm/yy format, e.g. 31/07/96

Long

Input the date in long format, e.g. July 31, 1996

Ordered

Input the date in yy/mm/dd format, e.g. 96/07/31

Sorted

Input the date in yyyymmdd format, e.g. 19960731

Usa

Input the date in mm/dd/yy format, e.g. 07/31/96

The *Sorted* and *Long* options return a four digit year. Under Windows and Unix systems, the century is calculated from a two digit year: if the system year is less than 50 the 21st century is assumed; otherwise the 20th century is returned. This code is only available in versions 1.98 Beta D and above.

Default Keys: a−f9, c−f9

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

DELCHAR

Syntax: DELCHAR

Description: Delete the current character at the cursor position.

Default Key: Del

Return Codes:

0
: Successful completion

15
: No file left in edit ring

DELDUPES

Syntax: DELDUPES <C **E**>

Description: Remove all duplicate lines from a file. If there is a marked area in the current file, the search is restricted to the marked lines. If the mark is a block mark, the search is further restricted to the marked columns within the block. If the *C* parameter is used, case is ignored when comparing lines. The *E* (exact case match) parameter is the default.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

DELETE

Syntax: DELETE <N *>

Description: Delete the current line from the file. If *N* is supplied, the given number of lines will be deleted. If *N* is an asterisk (*), all lines from the current line to the end of file will be deleted.

Default Key: c−backspace

Return Codes:

0
: Successful completion

9
: Cursor at TOF/EOF line

15
: No file left in edit ring

DELSYM

Syntax: DELSYM

Description: Delete the current symbol from the cursor position to the beginning of the next punctuation. If the cursor is on a punctuation mark, this function will delete all punctuation up to the next symbol.

Default Keys: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

DELWORD

Syntax: DELWORD

Description: Delete the current word from the cursor position to the beginning of the next word

Default Keys: c−d, c−del

Return Codes:

0
: Successful completion

15
: No file left in edit ring

DIAG

Syntax: DIAG

Description: Output diagnostics into a file called .XDIAG. The output file consists of the following main sections:

Edit Environment
: General information about the editor's configuration, global settings, and environment

Key Definitions
: All major keys and their current definitions, including both default and modified keys

Files in Ring
: An entry for each file in the ring, including file specific settings and file memory use

Marked Text
: Information about the marked area, if any

Allocated Memory
: A summary of the total memory use by the editor

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

18
: Utilities DLL not loaded

DOWN

Syntax: DOWN <N *>

Description: Move the cursor down one row. If *N* is supplied, the cursor will be moved the given number of lines. If *N* is an asterisk (*), the cursor will be moved to the last line of the file.

Default Key: down

Return Codes:

0
: Successful completion

−6
: Invalid parameter

9
: Cursor at TOF/EOF line

10
: The cursor was not moved

15
: No file left in edit ring

16
: Target line is hidden

DUPLICATES

Syntax: DUPLICATES <**DELETE** ALL EXCLUDE> <C **E**>

Description: Work with duplicate lines in the current file:

DELETE
: Delete all duplicate lines. This option, the default, is identical to the *DELDUPES* command.

ALL
: Show only duplicate lines. Any lines having no duplicate will be excluded from the display.

EXCLUDE
: Hide all duplicate lines. Any lines having at least one duplicate will be excluded from the display.

If there is a marked area in the current file, the search is restricted to the marked lines. If the mark is a block mark, the search is further restricted to the marked columns within the block. If the *C* parameter is used, case is ignored when comparing lines. The *E* (exact case match) parameter is the default.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

E, EDIT, X

Syntax: EDIT <fn1 fn2...> <options>

Description: Add the specified file(s) to the edit ring. See Invoking The Editor for details on options and file name syntax.

Default Key: none

Return Codes:

0
: Successful completion

−1
: Insufficient memory

−2
: Unable to open the file

−3
: Unable to read the file

1
: New file

2
: Line(s) split

13
: Invalid path

EA

Syntax: EA <ON OFF>

Description: Turn the saving of editor information with extended attributes on or off. If no parameters are specified, the EA setting is toggled.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

EOF_TEXT

Syntax: EOF_TEXT text

Description: Change the text used to mark the end of the current file

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

ERASEEOL

Syntax: ERASEEOL

Description: Erase the current line from the cursor position.

Default Key: c−e

Return Codes:

0
: Successful completion

15
: No file left in edit ring

ERRORS

Syntax: ERRORS <**SHOW** NEXT REMOVE>

Description: Handle compiler errors, where *SHOW* (the default) copies error lines from *fn.err* into the current file, *NEXT* moves the cursor to the next error line, and *REMOVE* removes all error lines from the current file. See Compiler Errors for more information on creating and using compiler error files.

Default Keys: a−q, c−n, c−o

Return Codes:

0
: Successful completion

15
: No file left in edit ring

EXCLUDE

Syntax: EXCLUDE <N * TOGGLE AREA>

Description: Exclude the line(s) at the cursor position, with the following options:

N
: Exclude the next N lines from the display, where *N* defaults to 1.

*
: All lines from the current line to the End Of File will be excluded.

TOGGLE
: The current line will be excluded if it is visible, or the lines represented by a shadow line will be shown if the cursor is on a shadow line. Note that if *SHADOW* is *ON*, the shadow line will count as one line excluded, no matter how many lines it represents.

AREA
: Exclude from the current line, all lines with indentation greater than or equal to the current line's indentation.

Default Keys: c−a, c−x

Return Codes:

0
: Successful completion

−6
: Invalid parameter

10
: The cursor was not moved

15
: No file left in edit ring

EXITRC

Syntax: EXITRC nnn

Description: Set the editor exit return code to *nnn*. This option is useful when calling the editor from other environments, where a return code would be a handy way to communicate some information from the editor back to the calling environment. The normal editor return code is 0.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

EXPAND

Syntax: EXPAND

Description: This command is used to expand a few letters of a word into a longer word. If invoked from the data area, it examines previously entered text to find a word that begins with the text that is found at the cursor position. If found, the word at the cursor position is replaced with the full word. Repeated presses of this key will provide alternative expansions for the current word. Insert mode is always turned on for this operation, so no existing data will be overwritten.

If this command is invoked from the command line, it will take the characters on the command line and use them to fill the command line with the first matching command.

Default Key: a−=

Return Codes:

0
: Successful completion

15
: No file left in edit ring

EXT

Syntax: EXT <newext>

Description: If used with no parameter, the current file extension is displayed. If *newext* is supplied, the file extension will be changed to *newext*.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

EXTRACT

Syntax: EXTRACT /opt/

Description: Gain access to editor information from a Rexx program. See EXTRACT Command for details regarding the various options.

Default Key: none

Return Codes: See EXTRACT Command

FFILE

Syntax: FFILE <newname /CR /CRLF /CRCRLF /LF /NOEA /NOTABS /T /U>

Description: Save the currently edited file to disk. Exactly like the FILE command, only it skips the check for a changed filename before saving. All options are described under the SAVE command.

Default Key: none

Return Codes:

0
: Successful completion

−1
: Insufficient memory

−2
: Unable to open the file

−4
: Unable to write the file

−10
: Path not found

−11
: Access denied (file attributes?)

12
: File is read only

15
: No file left in edit ring

FIELDTEMPLATE

Syntax: FIELDTEMPLATE <c1 [U P]> <c2 [U P]> ...

Description: Control default editing of all file lines, through definition of protected and unprotected fields. See the LineFields command for syntax details.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

FILE

Syntax: FILE <newname /CR /CRLF /CRCRLF /LF /NOEA /NOTABS /T /U>

Description: Save the currently edited file to disk. If *newname* is supplied, the file will be saved under that name. Otherwise, the current name will be used. If the save was successful, quit the file. All options are described under the SAVE command.

Default Key: none

Return Codes:

0
: Successful completion

−1
: Insufficient memory

−2
: Unable to open the file

−4
: Unable to write the file

−10
: Path not found

−11
: Access denied (file attributes?)

1
: File already exists

12
: File is read only

15
: No file left in edit ring

FIND_WORD

Syntax: FIND_WORD

Description: Search forwards in the current file for the word located at the cursor position. A word is defined as containing only alphanumeric characters plus the underscore (_) character. Any character which is not in the range A−Z and is non−numeric will be taken as the end of the word for the search. The exception to this rule is the case of dates and times. Any string of the format dd/dd/dd, where dd is a numeric digit and / is the system date separator, will be taken as a "word". Similarly, any string in the format hh:mm:ss or hh:mm:ss.ttt will be taken as a "word".

If the cursor is on the command line, the command line is replaced with a locate command showing the current search string. This is useful if you want to make a small change to the string without having to re−type most of it. A string delimiter is chosen from the set "/&$:\" that is not part of the search string.

Default Key: c−w

Return Codes:

0
: Successful completion

15
: No file left in edit ring

FORWARD

Syntax: FORWARD

Description: Scroll the screen a full page towards the end of file. The cursor position on the screen will be unchanged, unless the last page is reached.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

FT

Syntax: FT ext

Description: Treat the file for formatting purposes as if it had the supplied extension. All syntax expansion and formatting keywords will be used for the pseudo extension.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

FUNCWIN

Syntax: FUNCWIN

Description: Display a popup window containing all functions in the current file.

Default Key: c−f11

Return Codes:

0
: Successful completion

15
: No file left in edit ring

GET

Syntax: GET filename

Description: Copy the supplied *filename* into the current file, starting at the current cursor position. The file, which cannot contain wildcards, will not be added to the edit ring.

Default Key: none

Return Codes:

0
: Successful completion

−2
: File not found

−6
: Invalid parameter

15
: No file left in edit ring

HELP

Syntax: HELP

Description: Browse the help file

Default Key: f1

Return Codes:

0
: Successful completion

15
: No file left in edit ring

HEX

Syntax: HEX <ON OFF>

Description: Turn Hex Display mode on or off. If no parameters are specified, the current hex display setting is toggled.

Default Key: a−h

Return Codes:

0
: Successful completion

15
: No file left in edit ring

HIDEFILE

Syntax: HIDEFILE <ON OFF>

Description: Toggle the Hide setting for the current file, or with no parameters, the current Hide setting is toggled. If Hidefile is ON, the file will become invisible in the ring, unless it is explicitly called by name. Note that the file is automatically discarded when no visible files are left in the ring, **even if it has been modified**.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

INPUT

Syntax: INPUT <text>

Description: Input a new line into the file, after the current cursor position. If *text* is supplied, use that text for the new line.

Default Key: c−enter

Return Codes:

0
: Successful completion

12
: File is read only

15
: No file left in edit ring

INPUT_ERRORLINE

Syntax: INPUT_ERRORLINE text

Description: Input a new line containing *text* into the file, after the current cursor position. The new line will have error highlighting and will be read−only.

Default Key: none

Return Codes:

0
: Successful completion

12
: File is read only

15
: No file left in edit ring

INSMODE

Syntax: INSMODE <ON OFF>

Description: Turn Insert mode on or off. If no parameters are specified, the current insert mode setting is toggled. Insert mode is indicated with either Ins or Rep in the far right of the status line.

Default Key: ins

Return Codes:

0
: Successful completion

15
: No file left in edit ring

JOIN

Syntax: JOIN

Description: Join the current line with the following line. If the cursor is after the end of the current line the join occurs at the cursor position. Otherwise, the lines are joined with a single space between them.

Default Key: a−j

Return Codes:

0
: Successful completion

15
: No file left in edit ring

KEY

Syntax: KEY key = func

Description: Set the specified *key* to the given *func*, where *func* can be any command that may be issued from the command line. The new key is in effect only for the current file; to globally set a key use the user profile.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

KEYIN

Syntax: KEYIN text

Description: Enter the supplied *text* at the current cursor position. This function will not work correctly if entered on the command line; it must be used from a key or a macro only. Note that this command accepts and keeps trailing spaces.

Default Keys: padplus, a−1, a−2, a−3, a−4, a−5, a−6, a−9, a−−, a−\

Return Codes:

0
: Successful completion

−6
: Invalid parameter (no text supplied)

12
: File or current line is read only

15
: No file left in edit ring

KEYIN_NAME

Syntax: KEYIN_NAME

Description: Enter the current filename at the current cursor position.

Default Key: a−n

Return Codes:

0
: Successful completion

−6
: Invalid parameter (no text supplied)

12
: File or current line is read only

15
: No file left in edit ring

KEYS_PLAY, PLAYBACK

Syntax: KEYS_PLAY <N *>

Description: Play back a recorded key sequence. This allows a macro to execute the function normally assigned to the Ctrl−T key. If *N* is supplied, the key sequence will be executed the given number of times, unless it is interrupted by an abnormal key response. If *N* is an asterisk (*), the key sequence will be executed until the cursor reaches either the Top Of File line or the End Of File line, or until an abnormal key response. Recorded Key Sequences contains more information about recording and playing back key sequences.

Default Key: c−t

Return Codes:

0
: Successful completion

−6
: Invalid parameter

9
: Cursor at TOF/EOF line

15
: No file left in edit ring

KEYS_RECORD

Syntax: KEYS_RECORD

Description: Initiate a sequence of recorded keys, which can be played back later with the Keys_play command. Recorded Key Sequences contains more information about recording and playing back key sequences.

Default Key: c−r

Return Codes:

0
: Successful completion

15
: No file left in edit ring

KEYS_WRITE

Syntax: KEYS_WRITE fn.x

Description: Translate the current recorded key sequence into the equivalent Rexx macro. File *fn.x* will be added to the edit ring, and will contain skeleton Rexx code. This command is useful for saving recordings for later playback, or for quickly generating the basis for a more complicated macro.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

18
: Utilities DLL not loaded

L, LOCATE

Syntax: L /text</< |/& /text2/> −mecflqs>

Description: Locate the supplied *text*. See Locate Text for details.

Default Key: none

Return Codes: See Locate Text

LINECOLOUR

Syntax: LINECOLOUR <c1 c2 [fg1 ON bg1 /name] c3 c4 fg2 ON bg2 ...>

Description: Control the colouring of the current line. Columns *c1* through *c2* are coloured with foreground colour *fg1* on a background of *bg1*, where foreground and background colours are described in Colour Remapping. Alternatively, / syntax may be used to define line colours. If a slash is followed by the name of an X2 colour area, e.g. *DATA*, then the currently defined settings for that area will be used to display the given columns.

Multiple sets of columns and colours may be specified in a single command. No parameters removes any previously specified line colouring for the current line. Note that explicitly setting a line's colour overrides any syntax specific colouring that may be applied to a line.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

LINEFIELDS

Syntax: LINEFIELDS <c1 [U P]> <c2 [U P]> ...

Description: Control editing of the current line, through definition of protected and unprotected fields. Everything from column c1 until the end of line or the next linefield will be either **U**nprotected or **P**rotected, depending on the parameter. The *LINEFIELDS* command with no parameters will clear the setting for the current line.

A quick way to set default linefields for every file line is provided with the FieldTemplate command. Note that the LINEFIELDS and FIELDTEMPLATE commands are currently experimental, and may change or be removed in a future release.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

LINEMACRO

Syntax: LINEMACRO <MACRO whatever>

Description: Set a macro which will be executed whenever the user tries to enter a character on the current line. When a key is pressed, macro *whatever* will be called with the following parameters:

```
CHAR c rr nn mode b
```

where

c
: is the new character

rr
: is the current row number

nn
: is the current cursor column

mode
: is either "Ins" or "Rep"

b
: is 0 if editing is allowed, or 1 either if the line is read only or if the entire file is in browse mode

By default, if a linemacro is called for a line change, the line change will be suppressed. To let the change proceed, use the EXITRC command to set the return code to a non−zero value.

To clear a linemacro from the current line, issue the LINEMACRO command with no parameters.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

LINEND

Syntax: LINEND <**ON** OFF <c>>

Description: Turn the linend setting on or off. If no parameters are specified, the linend setting is toggled. The default setting is ON. If *c* is specified, it becomes the new linend character.

Default Key: none

Return Codes:

0
: Successful completion

15

No file left in edit ring

MA, MARGINS

Syntax: Margins <n1 <n2 <n3 <n4>>>>

Description: Set/query file margins. If no parameters are supplied, the current margins are displayed. *n1* through *n4* represent the left and right file margins, and the left and right comment alignment margins, respectively. *n3* and *n4* are also used as the margins for the reformat command. The minimum allowable value for *n4* is 20.

Default Key: none

Return Codes:

0
: Successful completion

9
: Cursor at TOF/EOF line

10
: The cursor was not moved

−6
: Invalid parameter

15
: No file left in edit ring

MACRO

Syntax: MACRO macroname <parms>

Description: Execute the specified *macroname*. The macro must have a file extension of X, although the extension is not specified on invocation. Any parameters which the macro may expect are passed directly to the macro.

Default Key: none

Return Codes:

0
: Successful completion

−3
: Macro not found

Other
: Numeric parameter passed to macro Exit statement

15
: No file left in edit ring

MARK

Syntax: MARK <option>

Description: Create or manipulate a text mark on the text at the current position. The following options are available:

ADD
: Add the column of numbers contained by the marked area, and insert the result at the end of the marked block.

*ALIGN <**LEFT** RIGHT>*
: Align text to the left or right edge of the marked area by removing all intervening blanks. The default option is *LEFT*.

AREA
: Mark the area starting at the current line and ending at the last line with the same or greater indentation.

BLOCK <EXTEND>
: Create a block mark. If the *EXTEND* parameter is omitted, any existing mark will be cleared, unless it is a block mark spanning one character only. With the *EXTEND* parameter, an existing block mark will be extended to the current position. Existing line marks or marks in other files will be cleared regardless.

CLEAR
: Remove any existing mark.

COL1
: Mark from the cursor position to the beginning of the current line. The cursor will be moved to the beginning of the line.

COPY
: Copy the marked text to the current position.

DELETE
: Delete the marked text from the file.

EOL
: Mark from the cursor position to the end of the current line. If there is a comment on the line, mark only the comment.

EXTEND [LEFT RIGHT UP DOWN]
: Extend the current mark one character to the left, right, upwards, or downwards, and move the cursor in the indicated direction.

FILL </c/>
: Fill the marked area with the supplied character. If only one character is supplied, it will be used to fill the mark, or if two or more characters are used the second character will be used. If no characters are supplied, the user will be prompted for a fill character.

INTEGERS
: Input an ascending series of integers, one on each marked line. The starting integer is taken from the first marked line; if it is missing or invalid, it will default to 1. The output numbers will be padded with blanks so they are right aligned.

***LINE** <EXTEND>*
: Create a line mark. If the *EXTEND* parameter is omitted, any existing mark will be cleared, unless it is a line mark spanning one line only. With the *EXTEND* parameter, an existing line mark will be extended to the current position. Existing block marks or marks in other files will be cleared regardless. *MARK LINE* is the default option.

LOWER
: Convert the marked area to lower case text.

MERGE
: Merge the marked area into the current position. Only target characters that are blank will be replaced from the mark.

MIXED
: Convert the marked area to mixed case text.

MOVE
: Move the marked text to the current position. It will be deleted from its previous position.

OVERLAY
: Overlay the marked text onto the current position.

*SHIFT <LEFT **RIGHT**>*
: Shift the marked text one column to the left or to the right, where *RIGHT* is the default.

SYM
: Mark the nearest symbol to the cursor, where a symbol consists only of the alphanumeric characters, plus the underscore character.

UPPER
: Convert the marked area to upper case text.

VERTICAL
: Mark the vertical column of text that contains the cursor position.

WORD
: Mark the nearest word to the cursor.

Default Keys: a−a, a−b, a−c, a−d, a−f, a−i, a−l, a−m, a−o, a−r, a−u, a−v, a−w, a−z, c−F3, c−F4, c−F5, c−F7, c−F8

Return Codes:

0
: Successful completion

−6
: Invalid parameter

6
: No mark defined

12
: File is read only

14
: Current line is blank

15
: No file left in edit ring

MATCH

Syntax: MATCH

Description: Move the cursor to the equivalent string from the set of conditional strings, brackets, and GML tags. See Conditional Strings.

Default Key: c−y

Return Codes:

0
: Successful completion

10
: The cursor was not moved

15
: No file left in edit ring

MESSAGEBOX

Syntax: MESSAGEBOX text

Description: Display a window containing *text*. The window is dismissed after **any** keypress, and the key pressed is set in the Rexx variable *result*.

The supplied *text* may include multiple output lines, by separating each line by a sequence consisting of the *escape* character immediately followed by the letter N. The **first** line is centred on the window; all subsequent lines are left justified.

Default Key: none

Return Codes:

0
: Successful completion

−1
: Insufficient memory

−6
: Invalid parameter

15
: No file left in edit ring

MSG

Syntax: MSG <text>

Description: Display *text* as an editor message. If *text* is omitted, any current message text will be cleared.

Default Key: none

Return Codes:

0
: Successful completion

15
No file left in edit ring

MSGMODE

Syntax: MSGMODE <**ON** OFF>

Description: Turn message display on or off. If no parameters are specified, the msgmode setting is toggled. The default setting is ON. No information message is displayed when this command is executed. If message mode is OFF, then all messages are suppressed, except for messages that require a user response.

Default Key: none

Return Codes:

0
Successful completion
15
No file left in edit ring

NAME

Syntax: NAME <newname>

Description: Change the name of the current file. If *newname* is identified as a host file name, the only modification to the supplied text is to convert it to upper case. If *newname* is assumed to represent a PC file name, it will be formatted to a standard d:\path\fn.ext format. If *newname* is not specified, the current file's name will be added to the command line.

The following criteria must all be met to identify a file as a host filename:

1. The first two characters must be alphabetic
2. The third character must be a colon
3. No backslash must be found in the name

Possible host filenames are *ha:PROFILE EXEC A* and *hb:myfiles.script(member)*.

Default Key: f5

Return Codes:

0
Successful completion
15
No file left in edit ring

NEXT, NEXT_FILE

Syntax: NEXT, NEXT_FILE

Description: Make the next file in the edit ring the current file.

Default Key: f12

Return Codes:

0
: Successful completion

15
: No file left in edit ring

NEXT_ERROR

Syntax: NEXT_ERROR

Description: Move the cursor to the next compiler error line in the file.

Default Key: c−n

Return Codes:

0
: Successful completion

7
: No more error lines found

15
: No file left in edit ring

NEXT_FUNC

Syntax: NEXT_FUNC

Description: Moves the cursor to the next function in the file.

Default Key: a−pgdn

Return Codes:

0
: Successful completion

10
: The cursor was not moved

15

No file left in edit ring

NEXT_PARA

Syntax: NEXT_PARA

Description: Moves the cursor to the next paragraph in the file.

Default Key: a−end

Return Codes:

0
: Successful completion

10
: The cursor was not moved

15
: No file left in edit ring

NEXT_SENTENCE

Syntax: NEXT_SENTENCE

Description: Moves the cursor to the beginning of the next sentence in the file.

Default Key: none

Return Codes:

0
: Successful completion

10
: The cursor was not moved

15
: No file left in edit ring

NEXT_SYM

Syntax: NEXT_SYM

Description: Moves the cursor to the next symbol in the file, where a symbol consists only of the alphanumeric characters, plus the underscore character. If the cursor is past the end of the current line, the cursor will be positioned under symbols in the preceeding line until the end of that line is reached, at which point it will move to the first symbol in the next line.

Default Key: a−right

Return Codes:

0
: Successful completion

10
: The cursor was not moved

15
: No file left in edit ring

NEXT_WORD

Syntax: NEXT_WORD

Description: Moves the cursor to the next blank−delimited word in the file. If the cursor is past the end of the current line, the cursor will be positioned under words in the preceeding line until the end of that line is reached, at which point it will move to the first word in the next line.

Default Key: c−right

Return Codes:

0
: Successful completion

10
: The cursor was not moved

15
: No file left in edit ring

NOP

Syntax: NOP

Description: Null operation. Handy for disabling a key.

Default Keys: s−F6, s−F7, s−F8, s−F9, s−F10, s−F11, s−F12, s−del, s−ins, c−h, c−i, c−m, c−q, c−z, c−[, c−], c−F1, c−ins, c−tab, a−g, a−k, a−[, a−], a−F1, a−F5, a−F6, a−F7, a−F8, a−F10, a−F11, a−F12

Return Codes:

0
: Successful completion

15
: No file left in edit ring

NUMFILES

Syntax: NUMFILES

Description: Displays the number of files in the edit ring on the message line.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

OPENFILE

Syntax: OPENFILE

Description: Open (edit) the file named on the current line.

Default Key: c−p

Return Codes:

0
: Successful completion

15
: No file left in edit ring

PAGEDOWN

Syntax: PAGEDOWN

Description: Scroll the screen towards the end of file. The line containing the cursor will move to the top of the screen, unless the cursor is already at the top of the screen, in which case a full page is scrolled.

Default Key: pgdn

Return Codes:

0
: Successful completion

15
: No file left in edit ring

PAGEUP

Syntax: PAGEUP

Description: Scroll the screen towards the top of file. The line containing the cursor will move to the bottom of the screen, unless the cursor is already at the bottom of the screen, in which case a full page is scrolled.

Default Key: pgup

Return Codes:

0
: Successful completion

15
: No file left in edit ring

PASSWORD

Syntax: PASSWORD text

Description: Display a window with *text* as the title, and collect user input, but display asterisks (*) in place of each input character. When the user presses the **Enter** key, the response text will be copied into the Rexx variable *result*. If the user presses the **Escape** key, *result* will not be set and the return code will be 17. The window will automatically resize itself if the input spans more than one window line.

Default Key: none

Return Codes:

0
: Successful completion

−1
: Insufficient memory

−6
: Invalid parameter

15
: No file left in edit ring

17
: User quit the function

PFLINE

Syntax: PFLINE text

Description: Use the new *text* for the PF display line when no shift keys are active.

Default Key: none

Return Codes:

0
: Successful completion

−1
: Insufficient memory

−6
: Invalid parameter

15
: No file left in edit ring

PRESSKEY

Syntax: PRESSKEY keyname

Description: Press the key named by *keyname*. Designed to be used in conjunction with the EXTRACT /KEYPRESS/ command.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

PREVIOUS_FILE

Syntax: PREVIOUS_FILE

Description: Makes the previous file in the ring the current file.

Default Key: f11

Return Codes:

0
: Successful completion

15
: No file left in edit ring

PREVIOUS_FUNC

Syntax: PREVIOUS_FUNC

Description: Moves the cursor to the previous function in the file.

Default Key: a−pgup

Return Codes:

0
: Successful completion

10
: The cursor was not moved

15
: No file left in edit ring

PREVIOUS_PARA

Syntax: PREVIOUS_PARA

Description: Moves the cursor to the previous paragraph in the file.

Default Key: a−home

Return Codes:

0
: Successful completion

10
: The cursor was not moved

15
: No file left in edit ring

PREVIOUS_SYM

Syntax: PREVIOUS_SYM

Description: Moves the cursor to the previous symbol in the file, where a symbol consists only of the alphanumeric characters, plus the underscore character.

Default Key: a−left

Return Codes:

0
: Successful completion

10
: The cursor was not moved

15
: No file left in edit ring

PREVIOUS_WORD

Syntax: PREVIOUS_WORD

Description: Moves the cursor to the previous blank−delimited word in the file.

Default Key: c−left

Return Codes:

0
: Successful completion

10
: The cursor was not moved

15
: No file left in edit ring

PROMPT

Syntax: PROMPT text

Description: Display a window with *text* as the title, and collect user input. When the user presses the **Enter** key, the response text will be copied into the Rexx variable *result*. If the user presses the **Escape** key, *result* will not be set and the return code will be 17. The window will automatically resize itself if the input spans more than one input line.

Default Key: none

Return Codes:

0
: Successful completion

−1
: Insufficient memory

−6
: Invalid parameter

15
: No file left in edit ring

17
: User quit the function

PUT

Syntax: PUT fn

Description: Puts all visible lines in the current file to the supplied *fn*. Any lines which have been excluded from the display are not copied to the new file. If a mark exists in the current file, only visible lines in the marked area are copied.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

QQ, QQUIT

Syntax: QQUIT

Description: Quit the current file without saving any changes. If the file contains changes, you will **not** be warned that the changes will be lost.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

QUIT

Syntax: QUIT

Description: Quit the current file. If changes have not been saved, the following confirmation message will be displayed:

```
Throw away changes?

    Yes No Write
```

By default, the following responses are possible:

Y
: Throw away all changes to the file and proceed with the Quit operation

N/Escape
: Keep all changes and abort the Quit operation

W
: Write the file to disk before quitting. If the write is unsuccessful, the Quit is aborted.

FILE key
: Pressing any key assigned to the FILE function will cause the file to be written to disk before quitting.

SAVE key
: Pressing any key assigned to the SAVE function will cause the file to be written to disk before quitting.

QUIT key
: Pressing any key assigned to the QUIT function will cause the file to be removed from the edit ring without any changes being saved.

Note that use of the FILE, SAVE, and QUIT keys in response to the confirmation message is only supported if Rexx and the utilities DLL are available. These keys may be turned off in the user profile; see Quit Response When File Modified.

Default Key: f3

Return Codes:

0
: Successful completion

15
: No file left in edit ring

REDO

Syntax: REDO

Description: Redo a line change that was previously undone with the Undo command.

Default Key: f10

Return Codes:

0
: Successful completion

15
: No file left in edit ring

20
: End of undo stack

REFORMAT

Syntax: REFORMAT

Description: Reformat the paragraph beginning at the cursor position. If comment markers are active for this file, the paragraph will be formatted with leading and trailing comment markers. If the first line of text begins with a dash (−), the following lines will be indented to align after the dash, and formatting will end at the next line which begins with a dash. Otherwise, formatting will end at:

1. A blank line
2. A read−only line
3. A line containing no blank characters
4. A line beginning with one of the *highlight_tags* strings
5. Any line beginning with a colon (:) or period (.)
6. End of File

The beginning line is examined for leading indentation − all following text will be indented the same as the current line, unless it begins with a dash.

Default Key: a−p

Return Codes:

0
: Successful completion

9
: Cursor at last file line

15
: No file left in edit ring

REFRESH

Syntax: REFRESH

Description: Refreshes the screen. Necessary in some macros which don't happen to call commands which will refresh the screen. This command also resets the video mode to avoid screen blinking.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

RENAME

Syntax: RENAME

Description: Display text on the command line to rename the current file.

Default Key: f5

Return Codes:

0
: Successful completion

15
: No file left in edit ring

REPEAT_FIND, REPFIND

Syntax: REPEAT_FIND

Description: Repeat the last locate (L) command.

Default Key: c−f

Return Codes:

0
: Successful completion

7
: Target not found

15
: No file left in edit ring

REPLACE

Syntax: REPLACE text

Description: Replace the contents of the current line with the supplied *text*.

Default Key: none

Return Codes:

0
: Successful completion

12
: Line is read only

15
: No file left in edit ring

RESOLVE_FN

Syntax: RESOLVE_FN fn

Description: Resolve a supplied filespec into a format recognisable by the operating system, as explained in File Specification. The resulting filename is displayed on the command line as a message, which may be accessed by a macro through the *EXTRACT /LASTMSG/* command.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

RESTORE_FIND

Syntax: RESTORE_FIND

Description: Restore the previous find text and options. The old parameters are automatically saved whenever a find command is issued.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

REVERSE_FIND

Syntax: REVERSE_FIND

Description: Repeat the previous find (Locate) command, but search in the opposite direction. If the previous search was forwards through the file, this command causes a backwards scan for the same text.

Default Key: c−v

Return Codes:

0

Successful completion

7

Target not found

15

No file left in edit ring

RINGWIN

Syntax: RINGWIN

Description: Display a popup window containing all the files in the ring. Modified files will be displayed with the *window_emphasis* colour.

Default Key: c−f12

Return Codes:

0

Successful completion

15

No file left in edit ring

SAVE

Syntax: SAVE <newname /CR /CRLF /CRCRLF /LF /NOEA /NOTABS /T /U>

Description: Save the currently edited file to disk. If *newname* is supplied, the file will be saved under that name. Otherwise, the current name will be used. The following flags are available:

/CR

All lines are saved with a single Carriage Return (CR) terminating character.

/CRLF

All lines are saved with both a Carriage Return (CR) and a Line Feed (LF) terminating character.

/CRCRLF

All lines are saved with two Carriage Returns (CR) and a Line Feed (LF) terminating character.

/LF

All lines are saved with a single Line Feed (LF) terminating character.

/NOEA

Save the current file without saving extended attribute information.

/NOTABS

All lines will be scanned for tab characters, and any tabs found in the line will be converted into blanks.

/T

Any line which contains 8 or more spaces will be saved with a tab character to take the place of some or all of the blanks. The exception is blanks contained within

quoted strings, which are not converted to tabs.

/U

All lines are saved with a single Line Feed (LF) terminating character, i.e. in Unix format.

Default Key: f4

Return Codes:

0
Successful completion

−1
Insufficient memory

−2
Unable to open the file

−4
Unable to write the file

−10
Path not found

−11
Access denied (file attributes?)

1
File already exists

12
File is read only

15
No file left in edit ring

SCROLL

Syntax: SCROLL <**UP** DOWN LEFT RIGHT>

Description: Scroll the screen one unit in the specified direction. The default option is *UP*.

Default Key: none

Return Codes:

0
Successful completion

−1
Insufficient memory

−6
Invalid parameter

15
No file left in edit ring

SETRESULT

Syntax: SETRESULT text

Description: Set the value of the Rexx variable RESULT to the supplied text.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

SHADOW

Syntax: SHADOW <ON OFF>

Description: Turn the shadow line on or off. If no parameters are specified, the shadow setting is toggled.

Default Key: c−s

Return Codes:

0
: Successful completion

15
: No file left in edit ring

SHADOWTEXT

Syntax: SHADOWTEXT <text>

Description: Change the default text that is displayed when the current line is hidden. If SHADOW is ON and the current line is the **first** of a group of excluded lines, the supplied text will be displayed instead of the normal "N line(s) not displayed" text. To remove the shadow text from a line, simply omit any parameters.

Default Key: none

Return Codes:

0

Successful completion

−1

Insufficient memory

−6

Invalid parameter

15

No file left in edit ring

SHELL

Syntax: SHELL <command>

Description: Shell to the operating system. If a command is supplied, it will be executed and the editor automatically resumed. If the Shell command is issued directly from the editor command line, a prompt message "Press any key to return to X2" will be displayed; or if the command is issued from a macro no such message will be written. If no command is supplied, an operating system prompt will be displayed. Typing *exit* will return to the editor session.

On Windows or DOS systems, the shell command will try to execute the program named by the environment variable *COMSPEC*, or simply *COMMAND.EXE* if *COMSPEC* isn't set. Under OS/2, the shell command will try to execute the program named by the environment variable *OS2_SHELL*, or *CMD.EXE* if *OS2_SHELL* isn't set.

Default Key: none

Return Codes:

0

Successful completion

SHOW

Syntax: SHOW <N *>

Description: Display the line at the cursor position. If *N* is supplied, then the next N lines will be unexcluded. If *N* is an asterisk (*), all lines from the current line to the End Of File will be displayed. If *N* is negative, lines will be excluded from the end of the excluded block. Note that if *SHADOW* is *OFF*, this command will have no effect.

Default Key: none

Return Codes:

0

Successful completion

−6

Invalid parameter

10
: The cursor was not moved

15
: No file left in edit ring

SHOWLINE

Syntax: SHOWLINE <N>

Description: Display the specified line. If *N* is supplied, then line number N will be unexcluded, i.e. made visible. If *N* is absent, the current line will be unexcluded.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

SORT

Syntax: SORT <**A** D> <**E** C> <N>

Description: Sort the lines in the file. If a mark is present, only the marked lines are sorted. If it is a block mark, the sort columns are bounded by the mark. The optional flags are:

A
: Ascending sort (default)

D
: Descending sort

E
: Exact case sort (default)

C
: Mixed case sort

N
: Numeric sort. The data will be treated as signed numbers, so plus and minus signs may be used.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

SPAN

Syntax: SPAN <ON OFF>

Description: Turn the span setting on or off. If no parameters are specified, the span setting is toggled.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

SPLIT

Syntax: SPLIT

Description: Split the current line at the cursor position.

Default Key: a−s

Return Codes:

0
: Successful completion

15
: No file left in edit ring

SPLITJOIN

Syntax: SPLITJOIN

Description: Split the line at the cursor position, or join with the next line if positioned after the end of the line

Default Key: f2

Return Codes:

0
: Successful completion

15
: No file left in edit ring

STATUS

Syntax: STATUS <ON OFF>

Description: Turn the status line on or off. If no parameters are specified, the status setting is toggled.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

STATUSTEXT

Syntax: STATUSTEXT template

Description: Use the supplied *template* to draw the normal (full length, text mode) status line. The template may include any of the following special identifiers:

a
: Substitute the current number of alterations since the last save

c
: Substitute the current file column number

r
: Substitute the current file row number

t
: Substitute the current file total number of rows

x
: Substitute the hexadecimal representation of the current file character

The default *template* is (note the trailing blank):

```
"'\x'x Col=\c Row=\r of \t Alt=\a "
```

Default Key: none

Return Codes:

0

Successful completion

−6

Invalid parameter

15

No file left in edit ring

STYLE

Syntax: STYLE

Description: Format the coding style for the current function. If text is marked, the style formatting is restricted to the marked text. Requires *styleword* settings from the profile.

Default Key: none

Return Codes:

0

Successful completion

15

No file left in edit ring

18

Utilities DLL not loaded

SYNTAX

Syntax: SYNTAX <ON OFF>

Description: Turn syntax assistance on or off. If no parameters are specified, the setting is toggled.

Default Key: none

Return Codes:

0

Successful completion

15

No file left in edit ring

TAB

Syntax: TAB <MATCH>

Description: Moves the cursor to the next tab position. If *Insert* mode and *tab_insert* are both ON then spaces will be inserted up to the next tab position. If the *MATCH* parameter is supplied, the cursor will be moved to align with the next blank−delimited word on the

previous line. The same rules for inserting spaces apply with the *MATCH* parameter.

Default Key: tab

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

TABLINE

Syntax: TABLINE

Description: Inputs a special line containing a scale of the file columns, plus a letter "T" for every hard tab stop and a "t" for each soft tab stop. You may overwrite this line with the space, T, and t characters. If you do, when you move the cursor from the tab line the new tab settings will be read from the line and it will be re−written. The tab increment value is taken to be the first small "t" found after the last hard tab stop.

Default Key: none

Return Codes:

0
: Successful completion

12
: File is read only

15
: No file left in edit ring

TABS

Syntax: TABS <n1 n2 n3...> <,m>

Description: Set/query tab settings, where n1, n2, n3 are hard tab settings and m is an increment to generate soft tab settings. If no parameters are supplied, the current tab columns are displayed. If *n1* is the only number specified, tabs will be set every *n1* spaces, beginning in column 1. If the *,m* parameter is specified, the tab settings will continue from the last stop specified, every *m* spaces. Note that tab stops must be numeric and they must be specified in ascending sequence. There is a limit of 31 tab settings for each file.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

TIMER

Syntax: TIMER <<<hh:>mm:>ss cmd>

Description: Set a command to be executed every *N* seconds, where *N* is calculated from the hours, minutes, and seconds specified as the first parameter. The values for *hh*, *mm*, and *ss* must be numeric, but are not limited to 60, so for example, 100 may be specified to execute a command every 100 seconds.

If no options are specified, any previous timer setting is cleared.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

TITLE

Syntax: TITLE text

Description: Set the window title for an X−Windows editor window to *text*. This command is only available in the X−Windows Unix versions of the editor.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

TOFEOF

Syntax: TOFEOF <ON OFF>

Description: Turn display of Top Of File and End Of File lines on or off. If no parameters are specified, the setting is toggled.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

TOF_TEXT

Syntax: TOF_TEXT text

Description: Change the text used to mark the beginning of the current file

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

TOP

Syntax: TOP

Description: Move to the top line of the file

Default Key: c−home

Return Codes:

0
: Successful completion

10
: The cursor was not moved

15
: No file left in edit ring

TOPLINE

Syntax: TOPLINE N

Description: Position line *N* at the top of the screen

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

10
: The cursor was not moved

15
: No file left in edit ring

TOPSCREEN

Syntax: TOPSCREEN

Description: Move to the top line of the screen

Default Keys: c−pgup, a−up

Return Codes:

0
: Successful completion

10
: The cursor was not moved

15
: No file left in edit ring

UNDO

Syntax: UNDO <N *>

Description: Undo the previously changed line(s) in the current file. If *N* is supplied, it specifies the number of lines or blocks that will be restored. If *N* is an asterisk (*), all changes back to the last save will be removed.

Default Key: f9

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

20
: End of undo stack

UNDO_BLOCK

Syntax: UNDO_BLOCK

Description: Treat all line changes up until the next *undo_block* as a single change for undo purposes. This command is only valid when issued from a macro.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

UNDO_LIMIT

Syntax: UNDO_LIMIT N

Description: Set the undo limit for the current file to the value of *N*, where N may be any positive or negative whole number.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

UP

Syntax: UP <N *>

Description: Move the cursor up one row. If *N* is supplied, the cursor will be moved the given number of lines. If *N* is an asterisk (*), the cursor will be moved to the Top Of File line.

Default Key: up

Return Codes:

0
: Successful completion

−6
: Invalid parameter

9
: Cursor at TOF/EOF line

15
: No file left in edit ring

WINDOW

Syntax: WINDOW [_RESET rows cols maxlines title]

Description: Create a popup window with *rows* number of rows, *cols* number of columns, and set aside space for *maxlines* total lines. The window title will be *title*, which is limited in length to the width of the window minus the space required to show the line number. Note that the window will be empty until lines are added to it with the *WINLINE* command.

If the *_RESET* option is used, a previous window definition will be cleared. User Defined Popup Window contains more information about popup windows.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

23
: Popup window already defined

WINLINE

Syntax: WINLINE linetext\nselcmd

Description: Add a line to a user defined popup window. *linetext* defines the text to appear on the window, and *selcmd* is the command that will be executed if the line is selected. During processing, all occurrences of double backslashes ("\\") are converted to single backslashes and the following character is ignored. This allows input of text containing "\n" in either *linetext* or *selcmd*.

linetext may contain control sequences to modify the colour used in the line. Each sequence must begin with the *escape* character, and is followed by one of:

B (Bold)
: Display the following text with the *window_bold* colour

E (Emphasis)
: Display the following text with the *window_emphasis* colour

S (Selection)
: Use the following character as the key for list selection from the keyboard

T (Text)
: Display the following text with the *window_data* colour

Escape char
: Display two escape characters as a single escape character

This command can return −6 (invalid parameter) under any of the following conditions:

- ◊ If no text was supplied
- ◊ If the supplied text doesn't contain the separator text *"\n"*
- ◊ If a user popup window hasn't been defined
- ◊ If the maximum number of lines has been exceeded

User Defined Popup Window contains more information about popup windows. ESCAPE Character discusses setting the escape character in the user profile. See Appendix B. Sample Macro to Create a Popup Window for an example macro that uses WINLINE commands to create a popup window.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

WINSELECT

Syntax: WINSELECT <N>

Description: Select line *N* in a user defined popup window, where *N* must be numeric. If *N* is greater than the total number of rows in the window, the last window line is selected. If *N* is omitted, the last entered WINLINE will be selected.

Default Key: none

Return Codes:

0
: Successful completion

−6
: Invalid parameter

15
: No file left in edit ring

21
: No popup window defined

WINSORT

Syntax: WINSORT <**A** D>

Description: Sort the lines in a user defined popup window. The optional parameters are used to define an Ascending or Descending sort, where the default is Ascending. Lines will be sorted by Selection escape sequence if defined, or by the first character of the line if not. If a previous WINSELECT command has been issued, the selected line will move with the sort to maintain the same line contents.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

21
: No popup window defined

WINWAIT

Syntax: WINWAIT

Description: Wait for an existing popup window to be dismissed. Useful to allow a macro to display a window and wait for a response without losing control. Best used in conjunction with the SETRESULT command to get a response.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

17
: User quit the function

21
: No popup window defined

WRAP

Syntax: WRAP <ON OFF>

Description: Turn the wrap setting on or off. If no parameters are specified, the wrap setting is toggled.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

nnn

Syntax: nnn

Description: Move the cursor to line *nnn* of the file. If *nnn* is larger than the file size, the cursor is moved to the last line of the file.

Default Key: none

Return Codes:

0
: Successful completion

15
: No file left in edit ring

16
Target line is hidden

/text</< |/& /text2/>>

Syntax: /text</< |/& /text2/>> −mecflqs

Description: A short form for the L command to locate *text*. See Locate Text for details.

Default Key: none

Return Codes: See Locate Text

Command Summary

The following tables contain an entry for each command that is recognised by the X2 Editor command line. Commands that are supported by the OS/2 version are marked with a check mark (´) in the first column, commands that are supported under Windows NT/95 are marked in the second column, and supported DOS commands are marked in the third column. Supported Unix commands are marked in the last column, where Unix commands apply to the Linux, AIX 4.1, AIX 4.2, SunOS, and HP−UX operating systems if Rexx is installed.

◊ Command Summary (A−H)
◊ Command Summary (I−P)
◊ Command Summary (Q−Z)

Command Summary (A−H)

Command	OS/2	Windows NT/95	DOS	Unix
ACCENT		´		
ADD	´	´	´	´
ALL	´	´	´	´
ALT	´	´	´	´
APPEND	´	´	´	´
ASCII	´	´	´	´
AUTOBOOKMARK	´	´	´	´
AUTOSAVE	´	´	´	´
BACKSPACE	´	´	´	´
BACKTAB	´	´	´	´
BACKWARD	´	´	´	´

BOOKMARK	´	´		´
BOTTOM	´	´	´	´
BOTTOMSCREEN	´	´	´	´
BROWSE	´	´	´	´
C	´	´	´	´
CASECHAR	´	´		´
CASEWORD	´	´	´	´
CD	´	´		´
CENTRELINE	´	´	´	´
CENTRETEXT	´	´	´	´
CHANGE	´	´	´	´
CHANGES	´	´	´	´
CLIP		´		´
CMDLINE	´	´	´	´
CMDTEXT	´	´	´	´
COMMAND	´	´		´
COMMENTLINE	´	´		´
COMMENT_STYLE	´	´	´	´
COMPARE	´	´		´
CONDITIONAL	´	´	´	´
COPYLINE	´	´	´	´
COPYTOCMD	´	´	´	´
COUNT	´	´		´
CURR_ALT_PFLINE	´	´		´
CURR_CTRL_PFLINE	´	´		´
CURR_PFLINE	´	´		´
CURR_SHIFT_PFLINE	´	´		´
CURSOR	´	´	´	´
DATE	´	´	´	´
DELCHAR	´	´	´	´
DELDUPES	´	´	´	´
DELETE	´	´		´
DELSYM	´	´	´	´
DELWORD	´	´	´	´
DIAG	´	´		´

DOWN	´	´		´
DUPLICATES	´	´	´	´
E	´	´	´	´
EA	´	´		´
EDIT	´	´	´	´
EOF_TEXT	´	´	´	´
ERASEEOL	´	´	´	´
ERRORS	´	´	´	´
EXCLUDE	´	´		´
EXITRC	´	´		´
EXPAND	´	´	´	´
EXT	´	´	´	´
EXTRACT	´	´		´
FFILE	´	´		´
FIELDTEMPLATE	´	´		´
FILE	´	´	´	´
FIND_WORD	´	´	´	´
FORWARD	´	´	´	´
FT	´	´	´	´
FUNCWIN	´	´	´	´
GET	´	´	´	´
HELP	´	´	´	´
HEX	´	´		´
HIDEFILE	´	´		´

Command Summary (I−P)

Command	OS/2	Windows NT/95	DOS	Unix
INPUT	´	´	´	´
INPUT_ERRORLINE	´	´		´
INSMODE	´	´		´
JOIN	´	´	´	´
KEY	´	´		´

KEYIN	´	´	´	´
KEYIN_NAME	´	´	´	´
KEYS_PLAY	´	´		´
KEYS_RECORD	´	´	´	´
KEYS_WRITE	´	´		´
L	´	´	´	´
LINECOLOUR	´	´		´
LINEFIELDS	´	´		´
LINEMACRO	´	´		´
LINEND	´	´	´	´
LOCATE	´	´	´	´
MA	´	´	´	´
MACRO	´	´	´	´
MARGINS	´	´	´	´
MARK	´	´	´	´
MATCH	´	´	´	´
MESSAGEBOX	´	´		´
MSG	´	´		´
MSGMODE	´	´	´	´
NAME	´	´	´	´
NEXT	´	´	´	´
NEXT_ERROR	´	´	´	´
NEXT_FILE	´	´	´	´
NEXT_FUNC	´	´	´	´
NEXT_PARA	´	´	´	´
NEXT_SENTENCE	´	´	´	´
NEXT_SYM	´	´	´	´
NEXT_WORD	´	´	´	´
NOP	´	´	´	´
NUMFILES	´	´		´
OPENFILE	´	´	´	´
PAGEDOWN	´	´	´	´
PAGEUP	´	´	´	´
PASSWORD	´	´		´
PFLINE	´	´		´

PLAYBACK	´	´		´
PRESSKEY	´	´		´
PREVIOUS_FILE	´	´	´	´
PREVIOUS_FUNC	´	´	´	´
PREVIOUS_PARA	´	´	´	´
PREVIOUS_SYM	´	´	´	´
PREVIOUS_WORD	´	´	´	´
PROMPT	´	´		´
PUT	´	´	´	´

Command Summary (Q−Z)

Command	OS/2	Windows NT/95	DOS	Unix
QQ	´	´	´	´
QQUIT	´	´	´	´
QUIT	´	´	´	´
REDO	´	´	´	´
REFORMAT	´	´	´	´
REFRESH	´	´	´	´
RENAME	´	´	´	´
REPEAT_FIND	´	´	´	´
REPFIND	´	´	´	´
REPLACE	´	´		´
RESOLVE_FN	´	´	´	´
RESTORE_FIND	´	´	´	´
REVERSE_FIND	´	´	´	´
RINGWIN	´	´	´	´
SAVE	´	´	´	´
SCROLL	´	´	´	´
SETRESULT	´	´		´
SHADOW	´	´	´	´
SHADOWTEXT	´	´		´
SHELL	´	´	´	´

SHOW	´	´		´
SHOWLINE	´	´	´	´
SORT	´	´	´	´
SPAN	´	´	´	´
SPLIT	´	´	´	´
SPLITJOIN	´	´	´	´
STATUS	´	´		´
STATUSTEXT	´	´		´
STYLE	´	´		´
SYNTAX	´	´		´
TAB	´	´		´
TABLINE	´	´	´	´
TABS	´	´		´
TIMER	´	´		´
TITLE				´
TOFEOF	´	´		´
TOF_TEXT	´	´	´	´
TOP	´	´	´	´
TOPLINE	´	´	´	´
TOPSCREEN	´	´	´	´
UNDO	´	´	´	´
UNDO_BLOCK	´	´		´
UNDO_LIMIT	´	´	´	´
UP	´	´		´
WINDOW	´	´	´	´
WINLINE	´	´	´	´
WINSELECT	´	´		´
WINSORT	´	´	´	´
WINWAIT	´	´	´	´
WRAP	´	´	´	´
X	´	´	´	´
nnn	´	´	´	´

Hexadecimal Mode Considerations

If an input file contains the null character or Hexadecimal mode has been initiated with the Alt−H key, the display changes to show the input file as a sequence of 16 byte "lines". Two views of each line are shown: a set of four groups of eight hexadecimal integers each, and the character representation of the line. An example of hexadecimal mode representation of a file is shown in the following figure.

```
54686520 71756963 6B206272 6F776E20    *    The quick brown
666F7820 6A756D70 73206F76 65722074    *    fox jumps over t
6865206C 617A7920 646F672E 0D0A0D0A    *    he lazy dog.CRLF
```

Hexadecimal Mode Screen Layout

In the above example, CRLF represents two carriage return/line feed pairs. Changes can be made to either representation of the line. When moving the cursor over the file, a shadow cursor is displayed which follows the real cursor on the opposite representation of the line. When the cursor is on the hex representation, the shadow cursor will show the equivalent position on the character portion. When the cursor is on the character representation of the line, the shadow cursor will show the position in the hex portion.

When replacing characters in Hexadecimal mode, the cursor position is important. When over one of the hex sections of the line, only the characters 0−9 and A−F are valid. Lower case letters are converted to upper case. The replacement causes the hexadecimal value of the character at that position to be changed. This change will be reflected in the text section of the line. If the cursor is positioned over the text portion of the line, no conversion is done and all characters are valid. The hexadecimal representation of the character will be changed in the hex section.

In text mode the status line shows the row and column of the cursor in the file, where column 1 starts at the left edge of the screen. In hex mode the offset in bytes from the beginning of the file is shown instead. The offset will increment by one for every **two** hex characters on the line, until the cursor reaches the text section. Then it will increase normally until it reaches the end of the text section.

Some default keys will work differently in Hexadecimal mode; most notably, you cannot insert or delete characters in hex mode. The full list of changed keys is outlined below.

Cursor Left
: Moves the cursor left one character, unless the cursor would move to a non−modifiable section of the line. If so, it moves to the end of the previous modifiable section. The cursor will remain in either the hexadecimal or text portion of the screen; when the left edge of either section is reached the cursor is moved to the right edge of the same section, on the previous line.

Cursor Right
: Moves the cursor right one character, unless the cursor would move to a non−modifiable section of the line. If so, it moves to the left edge of the next modifiable section. The cursor will remain in either the hexadecimal or text portion of the screen; when the right edge of either section is reached the cursor is moved to the left edge of the same section, on the next line.

Delete
: Does nothing.

End
: Moves the cursor to the right edge of the text view area.

Home
: Moves the cursor to the left edge of the first hexadecimal input area.

Insert
: Does nothing.

Tab
: Moves the cursor to the beginning of the next input section on a line.

Shift−tab
: Moves the cursor to the beginning of the previous input section on a line.

Ctrl−Backspace
: Does nothing.

Ctrl−Cursor Left
: Moves the cursor to the beginning of the previous input section on a line.

Ctrl−Cursor Right
: Moves the cursor to the beginning of the next input section on a line.

Ctrl−Enter
: Does nothing.

Alt−7
: Does nothing.

Alt−8
: Does nothing.

Editor Differences Between Operating Systems

- Differences in the Windows NT/95 Version
- Differences in the DOS Version
- Differences in the Unix X−Windows Versions
- Differences in the Linux Curses Version

Most of the above commentary describes the behaviour of the OS/2 version of the X2 Editor. In most cases, it applies equally well to all versions; however, there are differences between the versions which are driven by operating system differences or environmental limitations. This section outlines the main differences, using the OS/2 version as the reference point. Note that some commands are not supported in all versions; a summary of each command and its supported platforms can be found in Command Summary.

- Differences in the Windows NT/95 Version
- Differences in the DOS Version
- Differences in the Unix X−Windows Versions
- Differences in the Linux Curses Version

Differences in the Windows NT/95 Version

The Windows NT and Windows 95 operating systems do not support file extended attributes, so the editor must use a different technique to remember file settings between edit sessions. The EA information is saved in a file called XEAINFO.DTA in the directory specified by the XPATH. This technique has the disadvantage that EA data is not transferred with a file when it is copied or moved to a different directory, or if it is renamed. Also, the XEAINFO.DTA file will continually grow larger as files are edited and then deleted from the system.

The Alt−Enter key is usurped by the operating system to toggle the DOS window between a Windowed and FullScreen session, so it is not available to the editor.

Differences in the DOS Version

The main differences between the DOS version and the OS/2 version are the lack of macro support and the memory management routines. The DOS version does not support Rexx macros; therefore many of the commands that are only useful from macros are undefined. These are outlined in Command Summary.

The DOS file system does not support long filenames, nor extended attributes. Therefore the editor's ability to remember file settings such as cursor position does not apply to this version.

The DOS version is restricted to a 640K memory address space, which makes it difficult to edit large files.

The *PATHS* profile option is not supported under DOS.

Differences in the Unix X−Windows Versions

The Unix versions are the only versions that run in a true windowed session. As such, the screen size may be changed simply by re−sizing the window with the mouse. Care must be taken with the Alt−F4 key, and also with the system menu on the X−Windows screen. Either one can cause your editing session to be terminated, **without** any warning about loss of data.

The Unix versions are the only versions to support the *xwindows_font* profile variable to change the screen font, and the *x−colour* profile setting to change the X−Windows colour mappings.

The X−Windows versions are the only versions in which the editor draws the cursor instead of using the system cursor. It is drawn with a line to represent the Replace mode cursor, and as a box around the current character when in Insert mode. These versions use the xwindows_cursor colour variable to draw the cursor.

Differences in the Linux Curses Version

The Linux version comes in two flavours: a curses based fullscreen version that is primarily intended for emergency or quick use; and an X−Windows version that provides a **much** nicer editing environment. It is recommended that you use the X−Windows version whenever possible; the above text describing differences in the Unix X−Windows versions applies equally well to the X−Windows version on Linux.

The Linux fullscreen version uses curses for screen and keyboard support. The curses support is not ideal; many keystrokes are not supported, and only eight colours are available. The curses version colours are mapped according to the following table.

Profile Colour	Mapped Colour
Black	Black
Blue	Blue
Brown	Yellow
Cyan	Cyan
Dark Grey	Black
Green	Green
Light Blue	Blue
Light Cyan	Cyan
Light Green	Green
Light Grey	White
Light Magenta	Magenta
Light Red	Red
Magenta	Magenta

Red	Red
White	White
Yellow	Yellow

Many keystrokes are unrecognised or changed under the Linux curses version. These include:

- The ctrl−arrow, ctrl−enter, and ctrl−keypad keys are dead
- F11 is interpreted as shift−F1, and F12 as shift−F2
- Ctrl−F keys are interpreted as the base F keys
- The left Alt key works as expected, except the Alt F keys are intercepted by the Operating System
- The right Alt key has no effect

Other changes in the Linux version include:

- The changing PF display that occurs when one of the shift keys is pressed and held down does not work. The only PF line that is displayed is the unshifted one which is defined in the user profile.
- There seems to be no support for a big cursor, so the cursor is the same whether in Insert or Replace mode. You need to check the Ins/Rep indicator in the top right corner.
- Displaying characters with a value of 0x80 or higher causes a flashing string of characters to be shown

Appendix A. Rexx Program to Measure Editor Load Times

```
/*********************************************************************/
/*                                                                   */
/* Test the execution time of three editors.  We need the following files  */
/* in the current directory:                                         */
/*                                                                   */
/*   T1, T2, T3 − Sample data files                                  */
/*   E.EXE      − EOS2 editor                                        */
/*   E.EX       − EOS2 default profile                               */
/*   T2.EXE     − T editor, OS/2 version                             */
/*   X.EXE      − X2 Editor                                          */
/*                                                                   */
/* Written by B. Thompson, December 31, 1993                         */
/*                                                                   */
/*********************************************************************/

  Say 'You will have to press F3 27 times quickly for this test...'
  Parse Pull .
  Call check_editor 'T1'                                    /* Small file*/
  Call check_editor 'T2'                                   /* Medium file*/
  Call check_editor 'T3'                                    /* Large file*/
  Call lineout 'TESTTIME.OUT'                          /* Close output file*/
  Say 'Results are in TESTTIME.OUT'
Exit


/*********************************************************************/
/*                                                                   */
/* Check each editor against a supplied filename.  We do three iterations  */
/* for each editor, and average the results.                         */
/*                                                                   */
/*********************************************************************/

CHECK_EDITOR:  Procedure
  Parse Arg fn .
  Call lineout 'TESTTIME.OUT', 'Testing against' fn
  total. = 0                                          /* Initialise totals*/
  Do iteration = 1 To 3
    total._e = total._e + timeit('E' fn)
    total._t = total._t + timeit('T2' fn)
    total._x = total._x + timeit('X' fn)
  End                                                          /* End do*/
  Call lineout 'TESTTIME.OUT', ' Average time for E was' ,
    Format(total._e / 3,2,2) 'seconds'
  Call lineout 'TESTTIME.OUT', ' Average time for T was' ,
    Format(total._t / 3,2,2) 'seconds'
  Call lineout 'TESTTIME.OUT', ' Average time for X2 was' ,
    Format(total._x / 3,2,2) 'seconds'
Return


/*********************************************************************/
/*                                                                   */
/* Invoke the editor and time the execution.  The user will have to queue  */
/* some F3 keystrokes to make sure we quit as soon as the file is loaded   */
/* in each editor.                                                   */
/*                                                                   */
```

```
/*****************************************************************************/

TIMEIT:  Procedure
  Parse Arg editor fn .
  Call Time('R')                                        /* Start a timer*/
  editor fn
Return Time('E')
```

Appendix B. Sample Macro to Create a Popup Window

```
/*********************************************************************/
/*                                                                   */
/* Create and manage a popup window to enter bookmaster tags into a file.  */
/* The WINLINE command requires that the window text be separated from the */
/* resulting command by the text "\n".  This macro is a sample to show how */
/* to use the editor's popup window interface.                       */
/*                                                                   */
/* Requires version 1.61 or later of the editor.                     */
/*                                                                   */
/* Written by B. Thompson, June 7, 1995                              */
/* Fix problem inserting bookie tag at beginning of line, April 29, 1996   */
/* Insert tag around block mark, April 29, 1996                      */
/* Updated to illustrate bold and emphasised attributes, August 30, 1996   */
/*                                                                   */
/*********************************************************************/

  Parse source . . macroname '.' .                          /* Who are we?*/
  Parse Arg parm .

  Select
    When parm = 'HP1' | parm = 'HP2'
      Then Do                                                  /* Italics*/
        gml_tag = 'hp' || substr(parm,3) || '.'
        'EXTRACT /CURLINE/'
        'EXTRACT /CURSOR/'
        'EXTRACT /MARK/'
        wordlen = 0
        If (mark.0 > 0) & (mark.2 = mark.3) & (mark.4 > 0)
          Then Do                                  /* Tag the marked block*/
            cursor.2 = mark.4 - 1
            wordlen = mark.5 - mark.4 + 1
          End
          Else Do While Substr(curline.1,cursor.2,1) <> ' '
            cursor.2 = cursor.2 - 1
            If cursor.2 = 0
              Then Leave               /* We hit the beginning of the line*/
          End                                                 /* End do*/
        If cursor.2 > 0
          Then Do
            beginning = Substr(curline.1,1,cursor.2)
            ending    = Substr(curline.1,cursor.2+1)
          End
          Else Do
            beginning = ''
            ending    = curline.1
          End
        If wordlen > 0
          Then Do                                   /* Parse over the mark*/
            word = Substr(ending,1,wordlen)
            ending = Substr(ending,wordlen+1)
          End
          Else Do
            Parse Var ending word ending
            ending = ' 'ending                     /* Put the blank back in*/
          End
        'REPLACE' beginning':'gml_tag || word || ':e'gml_tag || ending
```

```
        If wordlen > 0
          Then Do                                          /* Move the mark*/
            'CURSOR' mark.2 mark.4 + length(gml_tag) + 1
            'MARK BLOCK'
            'CURSOR +0 +'wordlen − 1
            'MARK BLOCK'
          End
      End
    Otherwise Do                                        /* Create the window*/
      'EXTRACT /ESCAPE/'
      'WINDOW 6 50 6 Bookmaster Tags'
      'WINLINE Head level 0 (h0)\nKEYIN :h0.'
      'WINLINE Head level 1 (h1)\nKEYIN :h1.'
      'WINLINE Head level 2 (h2)\nKEYIN :h2.'
      'WINLINE Head level 3 (h3)\nKEYIN :h3.'
      'WINLINE' escape.1'EItalics' escape.1'T(hp1)\nMACRO' macroname 'HP1'
      'WINLINE' escape.1'BBold' escape.1'T(hp2)   \nMACRO' macroname 'HP2'
    End
  End                                                      /* End select*/
Exit
```

Appendix C. Sample Profile for EOS2 Users

```
/*****************************************************************************/
/*                                                                           */
/* XPROFILE.E − Sample profile for configuring the X2 Editor so it looks     */
/* and behaves as closely as possible to the default EOS2 configuration.     */
/* This is not a perfect mapping since not all EOS2 functions are            */
/* reproduced in X2.                                                         */
/*                                                                           */
/* If you have Rexx available, you may wish to set key f6 to "Macro boxes"   */
/* to get drawing capabilities.  Boxes.x is available through xmacros.zip.   */
/*                                                                           */
/* This profile may be run standalone or as overrides to the default         */
/* profile, depending on whether syntax assistance is required:              */
/*                                                                           */
/*   xprofile xprofile.e               − No syntax assistance              */
/*   xprofile xprofile.def xprofile.e  − Syntax assistance                 */
/*                                                                           */
/*****************************************************************************/

key a−b     = Mark Block Extend
key a−l     = Mark Line Extend
key a−1     = Openfile
key a−f10   = Previous_File
key c−del   = EraseEOL
key c−enter = Cursor +1 0
key enter   = Input
key f2      = Save
key f4      = File
key f5      = Nop
key f6      = Nop
key f7      = Rename
key f8      = "CmdText EDIT "
key f10     = Next_File
key pgdn    = Forward
key pgup    = Backward

PFLine   = "F1=Help  2=Save  3=Quit  4=File                  7=Name  8=Edit  9=Undo  10=Next"
s−PFLine = "F1= Scrl  2=Scrl   3=Scrl   4=Scrl   5=CenterLine                          "
c−PFLine = "                           3=UpperMark  4=LowerMark                       "
a−PFLine = "                                                                      10=Prev"

colour alt_keywords    = Light Grey on Blue
colour browse_data     = Light Grey on Blue
colour command         = Light Grey on Brown
colour command_stack   = Light Grey on Brown
colour comment         = Light Grey on Blue
colour data            = Light Grey on Blue
colour filename        = Light Grey on Black
colour pfline          = Cyan on Black
colour highlight       = Light Grey on Blue
colour keywords        = Light Grey on Blue
colour mark            = Blue on Light Grey
colour message         = Light Red on Black
colour mod_filename    = Red on Black
colour quotes          = Light Grey on Blue
colour shadow_cursor   = Black on Brown
colour status          = Light Grey on Black
colour tofeof          = Light Grey on Blue
```

```
extension          = *
comment_formatting = NONE
alt_highlight_kw   = _RESET
highlight_keyword  = _RESET
```