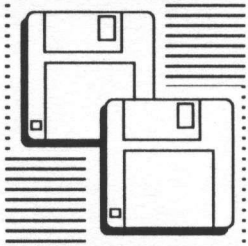
Ed Hak



Douglas
Communications

ATARI ST

EdHak

Contents

Contents, cont'd

Contents, cont'd.

INTRODUCTION

EdHak is a very versatile editor that runs as either a Desk Accessory (DA) or as a stand-alone PRG. Just change the extender from .ACC to .PRG and it runs as a normal program. It runs in ST low, medium, and high resolution, as well as TT low, medium, and high resolution. With it you can edit ANYthing, including text or binary files of ANY size, disk sectors, and RAM.

Starting with version 2.1 EdHak is copyrighted, commercially distributed software, NOT shareware. If you wish to let friends try out EdHak, just give them the demo or an earlier shareware version of it. You are also encouraged to upload the demo version with its documentation to bulletin boards for others to try out. The demo version has various features disabled, such as saving files or saving configuration changes to disk. The ED_HAK.NIC file on your disk includes a Douglas Communications icon for those of you who use an alternative desktop that can use NeoDesk icons.

Please see the README.TXT file on your disk for any changes since this documentation was printed.

If you have less available RAM than needed to run EdHak as an accessory, but still want some way to be able to do simple text editing like writing letters and addressing envelopes from an accessory, try the stripped down freeware version of EdHak called DIARY20S.ACC which is also on your disk. It only takes up 38K of ram (compared to about 77K for the full version of EdHak). This was accomplished by decreasing the buffer size to 4K and removing a number of features, most notably the word wrap, macros, launching other programs, and all hack, block, search, and configuration functions.

An even smaller and more limited accessory on your disk is EdWin, which can only hold one screen of text, but it can be resized and moved anywhere on your screen.

INSTALLATION

EdHak can run as either a desk accessory (ACC) or as a stand-alone program (PRG). The only difference is how YOU name the file. Although there may only be an EDHAKxxx.ACC

file on your disk, you can copy that file to another disk and name it EDHAKxxx.PRG. The x's in these names represent the version number, but it is not a necessary part of the name. Once you have installed EdHak as ACC and/or PRG, you will probably want to configure it to your personal taste, such as setting the buffer size, whether to use word-wrap, whether to automatically save BAKup files, etc.

If you are familiar with the 'fastload' bit available with TOS 1.4 and later, you should not set the fastload bit for EdHak. However, the use of CodeHead's shareware program PINHEAD to speed up loading of all programs, especially during bootup, is recommended.

Installation As ACCessory

Simply copy the EDHAKxxx.ACC file from your disk to the root directory of your boot floppy disk or to the root directory of your boot partition if you have a hard disk. This is the same place you will find your AUTO folder and any other desk accessories that you usually have available on your desktop. When you reboot your computer, EdHak should show up in your list of Desk Accessories. If not, make sure that you are not trying to load more than six ACC's, since that is the limit of the Atari Desktop.

EdHak functions fine within the CodeHead utility, MultiDesk, so if you want to be able to access EdHak as an accessory without having it always taking up a lot of your RAM or you want access to more than six ACC's, you may want to use MultiDesk or any comparable desk accessory manager. Please note that if you use MultiDesk Deluxe and have EdHak loaded as a non-resident accessory, the contents of your EdHak edit buffer will be lost when you close EdHak. So in that case you may want to get into the habit of using Alt-Q or menu:Quit to exit rather than Alt-X so you get prompted to save any changes to your file.

Installation As PRG

If there is not an EDHAKxxx.PRG file on your disk due to lack of space, just copy the EDHAKxxx.ACC file to another disk and rename it to EDHAKxxx.PRG (or whatever name you wish that ends with .PRG). Then you can run it like any other PRG by double-clicking on its icon or name depending on how you have your files displayed.

Once you have a PRG version of EdHak, you can use the desktop Install Application option to set up your DESKTOP.INF file to automatically call up EdHak anytime you double-click on any desired type of file. For instance, if you wanted to have EdHak automatically load in a .TXT file whenever you double-clicked on a .TXT file, you would (1) highlight the EDHAKxxx.PRG icon or filename in your desktop directory window by single clicking on it, then (2) click on the Install Application... menu item under the desktop Options menu, and (3) type TXT for the document type and choose normal boot status and GEM as the application type, then (4) click on Install, and finally (5) click on Save Desktop under the desktop Options menu. Then whenever you double-click on a filename or icon with a .TXT extender, EdHak will run and load that file for you to view or edit it. This can be done for any type of file, not just TXT files.

BASIC OPERATION OF EDHAK

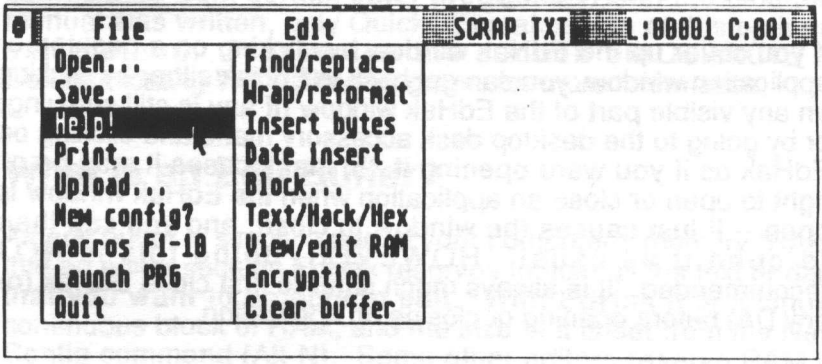
This program is basically a text editor desk accessory (often referred to as a 'DA' or 'ACC') or stand-alone program ('PRG') that opens up a half or full screen window. When half screen size, you can move it up or down on your screen so as to be able to see anything else on the screen. The size is toggled like any GEM window using the 'full' button at the top right corner.

When running as a desk accessory, it is available from within any other GEM program that has a menu bar, so you can get to it whenever you want to jot something down or view or make changes in a file without going to get your favorite word processor or disk sector editor. Also, as an ACC any text in the buffer stays there until you Clear it or reboot, regardless of whether you have the EdHak window open. (Note that this is not true if you have EdHak loaded into MultiDesk Deluxe as a non-resident accessory. In that case you will lose the buffer contents when you exit EdHak, so be sure to save your work before exiting.)

The amount of text/data you can fit into the edit buffer ranges from 4K to as much RAM as you have depending on how you have EdHak configured. The default is 10K (10240 characters).

Menu vs. Keyboard

The 'Menu' button near the left end of the window title bar gives access to the file, editing, and configuration commands. The drop down menu goes away just by moving the mouse outside the menu area, with no clicking needed.



The Double-Wide Drop-Down Menu

The drop-down Menu built into the title bar of the EdHak window gives you access to just about all the possible commands. However, once you get used to what the commands do, you may want to just use the equivalent keyboard commands, which are generally combinations of the <Alternate> key with some other key. The first upper case letter of each Menu command is the key you use along with <Alt> to perform the same command from the keyboard. For instance, to save a file, you would type Alt-S, holding the <Alt> key down while hitting the <S> key. Another example is Alt-T for toggling between Text and Hack display modes.

Another reason you may want to learn keyboard commands is that some are not available from the menu at all or they require fewer steps if done from the keyboard. For example, Alt-M is the command to Merge a file into the current buffer. To perform this from the menu you would click on Open... and then you would be given the option of opening a new file or merging into the current file. Another example is Shift-Alt-C, which lets you set or change the clipboard path. This command is not available at all from the menu.

On-Line Help

While editing, you can always hit the Help key or Alt-H or use the drop-down menu if you don't know what you're doing.

Window Handling

If you cover up the EdHak window by clicking on a desktop or application window, you can get back to EdHak either by clicking on any visible part of the EdHak window (if any is still showing) or by going to the desktop desk accessory menu and clicking on EdHak as if you were opening it. In many cases it may be all right to open or close an application when the EdHak window is open -- it just causes the window to close, and you can then re-open it as usual. HOWEVER, doing this is NOT recommended. It is always much safer to first close EdHak (or any DA) before opening or closing an application.

Clear Buffer (Alt-C, menu: Clear buffer)

This command wipes out everything currently in the edit buffer, but it will always ask for confirmation before doing it as well as asking if you want to save it if it has changed since it was last saved. This allows you to create a new file, but you will need to remember to give it a new name, if desired, when saving it to disk.

Quit -- There Must Be 50 Ways to Leave Your EdHak

(Alt-Q, Alt-X, Shift-Alt-X, Esc, menu: Quit, mouse: CloseButton)

You wouldn't think Quitting would require any explanation, would you? But there are a few things to know. If you use Alt-Q, Esc or menu:Quit, then you will always be asked if you want to save your work if any changes have been made since the last time it was saved. If no changes have been made, it will just ask if you Really want to quit.

If you use either Alt-X or clicking on the window's close button (upper left corner), then EdHak will just plain quit with no questions regardless of whether you have saved your changes or not. If you are running EdHak as a desk accessory, failing to

save changes is not too terrible, since you can just re-open EdHak and everything will still be there (until you reboot). But beware that if you are running it as a PRG, then you better be sure anything you care about has been saved to disk, because everything in the buffer is lost as soon as you quit.

The optional Shift-Alt-X command is for use when EdHak has been called from certain other programs. As of the time this manual was written, only QuickCIS makes use of this optional exit method. Please see the section on QuickCIS in Communicating With Other Programs for more information.

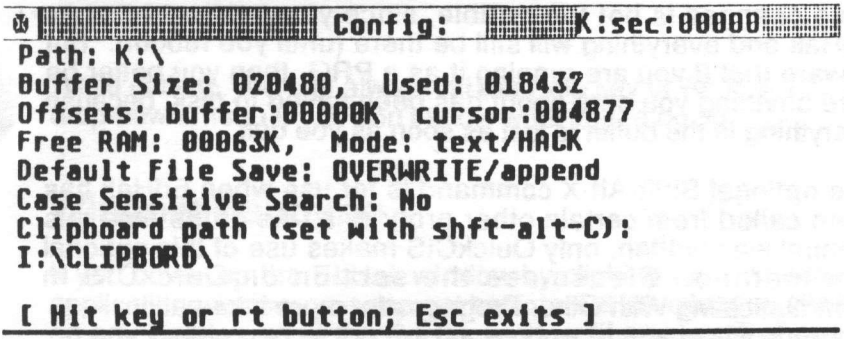
What is an Edit Buffer?

An edit buffer is simply a part of your computer's memory (RAM) that an editor such as EdHak reserves to hold all the text or data that you want to be able to edit. With EdHak it is a single continuous block of RAM, and the size of it is set from the New Config command (Alt-N). Some other editors reserve RAM in smaller blocks and just grab more blocks as needed to hold bigger files. With most of these editors, if a file is bigger than all the RAM in your machine, you are out of luck. However, with EdHak you can edit ANY size file as long as you have enough space on your disk (hopefully a hard drive if you deal in big files) to hold both the original file and a BAKup copy of it. This is because EdHak is smart enough to keep track of what part of the file it has in its buffer and use the disk to hold the rest of the file. For more details on this please refer to the Extended Files section in the manual.

CONFIGURATION

(Alt-N, menu: New config?)

This provides user-alterable buffer size (4K minimum, maximum limited only by your RAM) and many other options. This command first shows your current status and settings: current path (filename is always displayed in the window title bar), buffer size, buffer filled, buffer offset (for a file larger than the buffer how far from the start of the file is the start of the buffer), cursor offset (how many bytes from the start of the buffer is the cursor), free ram (largest available block), text/hack mode, default file save method (Overwrite or Append), default case sensitivity for the Find command, and Clipboard path.



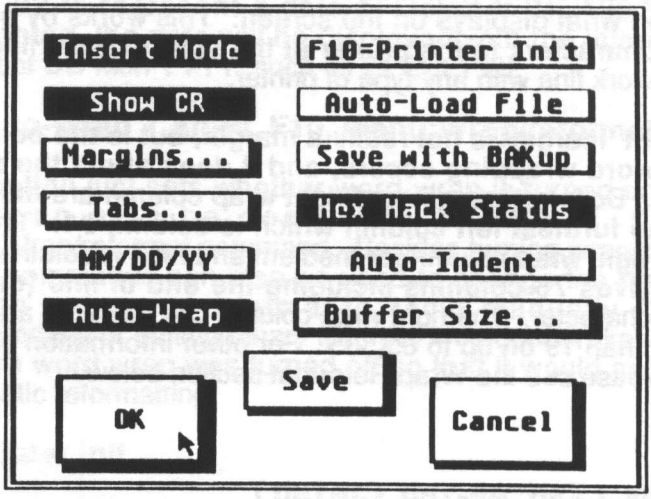
1st Config Screen
Title indicates now editing disk sectors on drive K:

The text/hack display mode is set while editing with either the Alt-T command or the menu:Text/Hack/Hex command. The default file save method is set to whatever your last response was when saving to an already existing file. That is affected by any of the file or block saving commands such as Alt-S, menu:Save or menu:Block. Similarly, the default case sensitivity for the Find command is set to whatever your last response was when performing a search. Therefore, it can be changed using the Alt-F or menu:Find/replace commands. If the clipboard path has not yet been set, the line following the "Clipboard path (set with shft-alt-C):" will be blank. The Shift-Alt-C command to change the clipboard path functions when you are editing, not while you are looking at this status screen. The settings of all these items can be saved to the config file as future defaults if the Save button is used in the second Config screen.

When done viewing the first config status screen, you can either exit back to your editing by hitting the Escape key or continue on to the next Config screen by hitting any other key or the right mouse button.

In the second Config screen (a dialog box with many buttons) you get to check and change many more EdHak settings and save them to the EDHAK.CFG configuration file as future defaults if desired, along with the settings shown in the first Config screen. The settings in the second Config screen include insert/overwrite text entry mode; whether the 'cr' end-of-line symbol is displayed; left print margin and right wrap column; number of spaces between tabs; date format; word-wrap on/off;

whether macro 10 is sent as a printer initialization string; whether the current file will automatically load at startup; whether a BAK file is always kept when a file is Saved; hex versus decimal display of the Hack mode status line; auto-indent on/off; and last but not least, buffer size. These will now be described in more detail.



2nd Config Screen, with some butttons turned on

Insert Mode (<Insert> key)

If the "Insert Mode" button is highlighted it means typing will insert characters at the cursor. If not highlighted you are in overwrite mode and your typing will overwrite any text at the cursor. While typing you can tell which mode you are in from the shape of the cursor. If it is a block shape covering the whole character you are in insert mode, but if it is just an underline shape you are in overwrite mode. This can also be toggled with the Insert key.

Show CR (Control-Return)

This sets whether or not the Carriage Return symbol, the little 'cr' end-of-line character is displayed. This is also toggled by hitting Control-Return anytime you are in edit mode (not while looking at this configuration display).

Margins (Alt- <, Alt- >)

No, you don't have to hold down the shift key for the Alt- < and Alt- > commands to work -- this could have said Alt-comma and Alt-period, but it seemed like the < and > symbols made more sense for left and right margins. Please note that the left 'margin' refers only to what gets sent to the printer and has no effect on what displays on the screen. This works by sending space characters to the printer at the start of each line, so it should work fine with any type of printer.

The right 'margin' is not really a margin, but is the column at which word wrapping occurs, and it does affect the screen display. Both left margin and right wrap column are measured from the furthest left column which is column #1. Thus the default right wrap column for medium and high resolution is 75, which gives 76 columns including the end of line (carriage return) character. The right wrap column can be set to any value greater than 19 on up to 65,535. For other information on word wrap, please see the Wrap/Reformat section below.

Tabs (Tab, Alt-Tab, Ctl-Tab)

In insert mode, the Tab key inserts whatever number of spaces you have it set for (default is 5). In overwrite mode, it skips over that number of spaces rather than erasing anything. Thus overwrite mode can be useful for editing existing tables.

To change the spacing between tab stops, use either the Alt-Tab command or the menu:New config:Tabs command. All tabs have the same spacing between them. If you wish to convert spaces to real tab characters or vice versa, it is best to do it with the Control-Tab command described in the section on Conversions. Alternatively you could use the Find/replace command, replacing the same number of spaces with the tab character or vice versa, but this will be much less likely to leave your columns aligned properly, since it will always use the same number of spaces rather than attempting to line up everything per the tab settings. If you do wish to enter the tab character [CHR(9)] as either the search or replace string, you can do it by hitting Ctl-I as text, 9 as decimal or 09 as hex.

Date format (Alt-D, menu:Date insert)

When the Alt-D or menu:Date insert command is given, the current system date is inserted at the cursor in one of two possible formats. If the MM/DD/YY config button is highlighted, it will use the US dating shorthand on month/day/year, such as 10/21/92 to represent a date of October 21, 1992. If not highlighted, the date will be displayed with the international format of DD Mon YYYY, such as 21 Oct 1992.

Auto-Wrap (Alt-W, F10, menu:Wrap/reformat)

This button just sets whether word-wrap is turned on or not. This can also be done with the Alt-W, F10, or menu:Wrap/reformat command. Besides turning wrap on or off, those commands can also be used to change the right wrap column or force a reformat to occur manually. If doing programming or editing binary files you would most likely want to be sure word wrap was turned off so that it would not do any automatic reformatting.

Printer Init

This sets whether a printer initialization string is sent before every print operation. This can be used to automatically set your print quality, line spacing, or other printer options at the start of any print operation. Macro F10 is where the printer initialization string is stored. Please see the section on Macros for more details on saving a macro by marking a block and saving it with Ctl-F10, and see the section on Printing for details on controlling your printer.

AutoLoad

This option allows you to specify a file for EdHak to load into its buffer automatically upon bootup (or when the PRG is run), so when you open EdHak the file is already there without you having to go find and open it. This is especially useful if you are always editing the same file such as a diary. You set this up by: (a) load the file of interest into the buffer, then (b) go to New Config (Alt-N), and verify that the correct filename and path are shown, (c) select Auto-Load File in the second Config screen, and finally (d) Save the configuration. If EdHak can't find the specified file upon bootup, it just skips over that routine and no harm is done.

Save with BAKup

This sets whether a backup (.BAK) file is kept whenever saving with Overwrite. When saving with BAKup, EdHak first deletes the previous .BAK file (if there is one), then renames the current disk file with a .BAK extender, and finally saves the new version of the file to disk. If any error occurs in saving the new version, the prior version will be renamed back to the original filename and there will be no .BAK file left. If you like to maintain some level of risk in life, or if you are just short on disk space, then don't bother with this.

Hex Hack Status

When in Hack mode the status display on the right side of the window title bar shows the offset from the start of the buffer and the value of the character at the cursor. If the Hex Hack Status button is highlighted, these two numbers will be shown in hexadecimal. If this button is not highlighted, they will be shown in decimal.

Auto-Indent (Alt-Return)

If this button is highlighted hitting the return key while typing will place the cursor at whatever column that text begins on the prior line, rather than placing it at the far left in column #1. This can be useful for programming or for indenting whole paragraphs. The Alt-Return command toggles back and forth from auto-indent to non-indented returns while performing a Return.

Buffer Size

Please note that changing the buffer size does not take effect until you re-run the PRG or reboot the ACC (or reload the ACC into MultiDesk). Also note that if you start dealing with a buffer contents larger than about 20K, you will start to notice a slowdown when inserting or deleting near the beginning of the buffer. This happens despite the fact that a very fast machine code routine is used for moving everything around in memory. For more info, read the tech notes below. If you deal with large files often, you may want to stick to a buffer size of 20K and just let EdHak scroll back and forth so it only has to deal with 20K at a time.

When saving a new buffer size, the EdHak ACC or PRG file can now be named anything. If it doesn't find the expected name (EDHAKxxx.PRG or EDHAKxxx.ACC where xxx is the current version number) it puts up a file selector. Note that it will only look in the path that the program ran from, regardless of what you try to do via the file selector.

OK, Save, Cancel (from Configuration Dialog)

OK keeps the changes you made (other than buffer size) in effect until you reboot or rerun the PRG. To make the changes permanent you must pick Save. Note that everything except the buffer size gets saved to an EDHAK.CFG file, while the buffer size gets written into the main EDHAKxxx.ACC or PRG file. The Save function here also saves any text macros you have created as described in the Macros section below. For the saved configuration to be read upon bootup by EdHak, the EDHAK.CFG file must be in the same path (folder or root directory) as the main EdHak file itself.

CURSOR CONTROL

Page up/down	Shift-arrow
Top/Bottom of screen	ClrHome
Top/Bottom of buffer	Shift-ClrHome
Start/End of line	Shift-Arrow
Start/End line in window	Shft-Ctl-Arrow
Goto start/end of block	Alt-1 / Alt-2
Goto Line # / Offset #	Alt-G
Goto Next/Prev word	Ctl-Arrow

Excerpt from First Help Screen

The arrow keys move the cursor around the screen one character at a time left and right, and one line at a time up and down. When you get to the top or bottom of the window, hitting the arrow again will force the window contents to scroll one line

at a time, to move the desired part of the file into the window. If you have a line that goes beyond the right edge of the window and go to arrow into that "extended" part of the line, the window will shift over to show the part of the lines beyond the first window. Lines may be up to 65,535 characters long, but chances are you will never have any need to handle a line that long!

To move the cursor left or right by one full word at a time, hold down the Control key while hitting the left or right arrow.

To shift the screen contents up or down by one "page", hold a Shift key down while hitting the up or down arrow. When the word "page" or "paging" is used in this context of scrolling the window contents, it refers to whatever number of lines are contained in the current window size minus one. So if the EdHak window is in its half-screen size, which can hold 10 lines of text, paging will move the contents up or down by 9 lines, thus leaving one line of overlap to make it a little easier to see the continuity from one windowfull of text to the next. In its full-screen size with 22 lines of text, paging will scroll up or down by 21 lines. Paging can also be done by clicking the mouse in the gray area of the vertical slider bar.

The Shift-Right Arrow and Shift-Left Arrow combinations place the cursor at the end or the start of the current line, respectively. If the Control key is held down along with a Shift key, the cursor won't go any further than the right or left edge of the window, which can be useful if editing a line that is longer than the width of the window.

To quickly move the cursor to the start of the buffer, hit Shift-ClrHome. To move to the end of the buffer hit Shift-ClrHome again. To just move to the start of the top line of the current window, hit ClrHome without a shift key. Once the cursor is there, hitting ClrHome again puts the cursor at the end of the window.

If you know a line number that you want to go to and you are in text mode rather than hack mode, the Alt-G command lets you pick the desired line. Note that lines are counted from the beginning of the buffer, so if you are editing a file larger than the buffer with the beginning of the file outside the buffer, the line number shown in the window title bar and used by the Goto Line command will not be the true line number of the file.

If you are in Hack mode and want to go to a certain offset from the start of the file, the Alt-G command lets you do it. If you are editing a file larger than the buffer and the beginning of the file is outside the buffer, you can figure out what offset to use in the Goto command by checking the config status screen (Alt-N) for the Buffer Offset. Subtract the buffer offset from your desired offset to calculate the offset within the buffer, which is what the Goto command needs.

Bookmarks can be useful things. Currently EdHak just has two "bookmarks." These are the beginning and end of a block, even if it is not highlighted. The commands Alt-1 and Alt-2 take you to the start or end of the last block that was marked. If you want to be able to quickly go back to some spot in the buffer, you can hit F1 to mark that spot and later hit Alt-1 to go right back to that spot. If you want to mark a second spot, hit F2. Since this will highlight the whole area between the first bookmark and the second, hit F3 to unhighlight the block so you don't accidentally delete it with the Backspace or Delete key. Then Alt-2 will take you to the second bookmark when desired. Note that these bookmarks only function properly within the current buffer, so if you shift to a different part of a file larger than the buffer, Alt-1 and Alt-2 will no longer take you back to the original spots in the file.

ENTERING TEXT & DATA From The KEYBOARD

All plain text (ASCII) characters can be entered with a single keystroke or with a Shift-keystroke. This includes the full lower and upper case alphabet, numbers 0 to 9, and all the punctuation and other symbols shown on the keys of your keyboard. However, the ASCII character set only covers some of the 256 possible characters, so EdHak provides other ways to enter the rest of the characters. The commands to do this are described here, and a more complete explanation of this subject can be found in the section "ASCII and the Character Set" later in this manual.

Text Entry (A-Z, 0-9, etc.)

As you type single keystrokes on your keyboard each character gets entered into the edit buffer and is displayed on the screen in the EdHak window. If you are in Insert mode (cursor is block shaped), entering each character pushes over any text after it to make room for the new character. In Overwrite mode (cursor is underline shaped) each character you type replaces the character that used to be there.

If word-wrap ("Auto-Wrap" button in configuration dialog) is turned on, you can just keep typing and EdHak will automatically move to the next line when needed without you needing to hit the Return key.

Simple editing is done with the Backspace key or the Delete key. Backspace erases the character before the cursor and moves all subsequent text back to fill the space. Delete is different only in that it erases the character the cursor is on.

If you hit Backspace from the beginning of a line or Delete at the end of a line, you remove the carriage return/linefeed separating the lines and they are joined into one line. If you have word wrap turned on, that line and the rest of the paragraph are then automatically reformatted to so that the lines fit within whatever line length (right wrap column) you have set in your configuration. The same thing happens if you are Overwrite mode and type in any character at the end of a line, since that character will replace the carriage return (and invisible linefeed) that used to be there.

Date Insertion (Alt-D, menu: Date Insert)

This command inserts the current date at the cursor using one of two formats. This assumes you have your ST's date set properly. The two possible date formats are (1) MM/DD/YY, as in 05/03/93 for May 3, 1993, and (2) DD Mon YYYY as in 03 May 1993. The date format is chosen in the second Config screen. If the MM/DD/YY button is highlighted, you get the date in that format, and if it is not highlighted, you get the International format, DD Mon YYYY.

Control Chars & Other Non-Text Characters
(Ctl-A -> Ctl-Z, Alt-Esc, Alt-I, menu: Insert bytes)
(& Alt-numeric keypad combinations with TOS 2.06, or higher)

The Atari keyboard and operating system and those of many other computers are designed to create the character values #1 through #26 with the Control key combinations Ctl-A through Ctl-Z. Many programs use these Control key combinations for commands, but in EdHak they are left alone since they provide a very easy way to enter some very useful printer control codes into your text. For instance, Ctl-M produces the Carriage Return character (#13), Ctl-J is the Linefeed character (#10), Ctl-L is the Formfeed character (#12), and Ctl-I is the Tab character (#9). Since the keyboard has more keys than just A-Z, some other weird charcters can be produced with other Control key combinations, but you probably won't find much need for them -- it is mentioned just as a warning in case you wonder what is going on when you try something like Ctl-period or Ctl-backslash. For more complete information on this subject, please see the section on "ASCII and the Character Set" later in this manual.

Escape Character (Alt-Esc)

One special character has its own keyboard command. The Escape character (#27) can be output with Alt-Esc as well as the normal Control-[(which puts out character #27, just like Control-A puts out character #1 and Control-Z puts out character #26). This can be useful for putting in printer control codes which often require the Escape character at the start of any printer command. See the section on Printing for more information.

Insert Bytes -- Hex, Decimal, or Text String
(Alt-I, menu: Insert bytes)

This is for direct entry of all 256 characters (#0 - #255). To enter ones above #127 you need to enter their hex or decimal values (unless you know some key combination that I don't that will do it!). For instance, to get the trademark symbol (little 'TM') on the screen, you need character #191 (decimal), which is also known as CHR(191), which is 'BF' in hex. Don't expect your printer to output a 'TM' though. On my printer, character #191 is an italics question mark.

Note that here and anywhere a string is to be entered, such as for search/replace, entering hex or decimal byte values gives access to all 0-255 possible characters. Hex digits must be entered in pairs or the string will be interpreted simply as a string of characters rather than hex values. Decimal byte values must be separated by some non-digit character, such as a comma.

If you are fortunate enough to have an STe or TT with version 2.06, 3.06 or later of TOS, there is another easy way to enter any of the 256 single characters. You just hold down the <Alternate> key, type the decimal number of the character you wish to enter using the numeric keypad, and then release the Alt key.

Text Macros

(F9, Cntrl-FuncKey, Alt-FuncKey, menu: macros F1-10)

This refers only to text & data macros -- there are no command macros. In other words a macro is simply a string of characters that you can insert whenever you want without having to type them all in. This might be useful for entering your name and address at the top of letters or keeping various printer set-up strings handy for different types of printing needs. There are 10 macros available (one for each function key), and each can hold up to 80 characters.

Hitting F9 or using the menu command shows 3 alert boxes which provide reminders on how to create and use macros, and then the ten macros are shown, one per line (display is truncated to what will fit on one line).

To store a string of text as a macro, first select a block as described below under Select Block, then copy the block to a macro with Cntl-FuncKey. For instance, to set up the macro for F1, select a block and then hit Control-F1. Then when you want to insert that macro, hit Alt-F1. Note that if you want the macros to still be there after rebooting or after quitting the PRG, you need to save your configuration, which is done from the second Config screen.

The only one of the 256 characters that can not be part of a macro is the null character (#0), since null is interpreted by EdHak as the marker for the end of the macro.

Macro #10 (Alt-F10) is special in that it can be used as a default printer initialization string which will automatically be sent to the printer before every print command, IF you have selected that option in the Config dialog. This can be useful if you always wish to tell your printer to do something like perforation skips or print in NLQ mode or any combination of printer commands up to 80 characters long. See the section on Printing for more info. If you have more than one printer setup string that you'd like always available, you can save the others as macros too (by typing each into your edit buffer, marking it as a block, and hitting Ctl-FuncKey). Then using Alt-FuncKey you can insert any one you want to use at the beginning of the file or block before printing it.

Word Wrap & Reformatting
(Alt-W, F10, menu:Wrap/reformat)

This command first gives you an alert box with the choices of Wrap!..., Cancel, or Turn Off. With Wrap turned On (which is the initial default), EdHak will word wrap to a default line length of 75 characters, plus the CR/LF for a total of 76 (or 64 if in 'Hack' mode, or appropriately shorter lengths in low resolution). This length can be changed with the Alt- > command or via the New config menu item.

Word wrap occurs automatically when adding to the end of your text and you hit the word wrap column, or when inserting text in a line and the end of the line reaches the wrap column. With Wrap turned Off, you can create/edit lines of any length up to 65,535 columns. Turning Wrap off also can be useful if you get annoyed by the constant redrawing of the the rest of the screen while inserting text. In this case, all the text you insert will be on one line until you force it to wrap using the Wrap! command. Regardless of whether Wrap is turned on or off, you can always force a wrap (reformat a paragraph, block, or whole buffer) with the Wrap! button of the alert box. You can get very quick at hitting F10, Return, Return to reformat the current paragraph. EdHak assumes paragraphs are separated either by a blank line or a line indented by at least one space or tab character. EdHak automatically maintains two spaces after periods when wrapping.

To change the column at which word wrap occurs, please see the Config: Margins command above (Alt- >). Word wrap can also be turned on/off using the Config:Auto-Wrap button in the Config dialog.

EDITING

Block Cut / Paste and all Delete functions

This covers all block functions, which includes deletion of whole or partial lines.

Select Block (F1 & F2 or mouse)

A block can be selected either by using the mouse or using F1 and F2. To use the mouse, hold down the left button while dragging the mouse either up or down to the end of the desired block. If you drag to the top or bottom of the window, the text will scroll while continuing to select the block, so it is easy to select blocks larger than the size of the window. If you use the function keys, F1 sets the start of a new block (thus cancelling any prior block selection). F2 sets the end of a block, which can be either before or after the start. Note that if you hit F2 again (or shift-left button) after moving the cursor outside the existing block, it will extend the block to that new point, but if the cursor is within an existing block, F2 will shrink the block. Also note that the last character of a block is always the character preceding the cursor -- this makes it easy to select an entire line without including the first character of the next line.

The keyboard commands Alt-1 and Alt-2 take you to the start or end of a block. Since these work even when a block is not highlighted, they can be used as general bookmarks.

Deselect Block (F3 or mouse)

To deselect a block either hit F3 or use the mouse to set the end of a block at its starting point.

Cut Block
(F4, Backspace, Delete, Alt-B, menu: Block...)

This command deletes a block that you have already selected and places it into the paste buffer. Each time you use this command or perform a Reformat, or if EdHak does an automatic word wrap, you lose anything that was already in the paste buffer. If you just want to make a copy of a block rather than deleting it, first cut it, then hit UNDO or F5 (which pastes it), and finally put the cursor where you want the copy to go and paste it again. The paste buffer will remain intact until you do any further

deletion of any type or until you cause a word wrap to occur. If you try to cut a block larger than the paste buffer can hold, you will be asked if you wish to Continue without any Undo or Paste capability (thus losing the block forever) or Cancel.

If you wish to save a block to its own disk file, select a block and then use one of the Save File commands described above.

Paste Block (F5, UNDO, Alt-B, menu: Block...)

This inserts the contents of the paste buffer at the cursor. You may then need to reformat using the word wrap command (F10, Alt-W or menu: Wrap/reformat)

Delete Whole Line or End of Line (Shift-Delete, Alt-Delete)

Deleting a whole line with Shift-Delete (or multiple lines in sequence) or deleting to the end of a line with Alt-Delete has the same effect as doing a block cut of the line or lines. So, to delete or move a few entire lines, it may be easier to just hit Shift-Delete a few times rather than marking them as a block and cutting them. As long as you don't move the cursor between deletions (and don't force a word wrap), each new line will just be appended to the paste buffer without losing its previous contents.

Find / Replace (Alt-F, menu: Find / replace, F6)

When you choose this command, you are first asked if you want to just Find something or perform a search & replace operation. This defaults to Find. Next you enter the string to search for. Normally, this would be entered as a string of characters, including any control characters such as Cntrl-M/Cntrl-J for carriage return/linefeed or Cntrl-I for the tab character, etc. Alternatively, you can enter any of the full 0-255 character set by using their two-digit hexadecimal values or 1-3 digit decimal values.

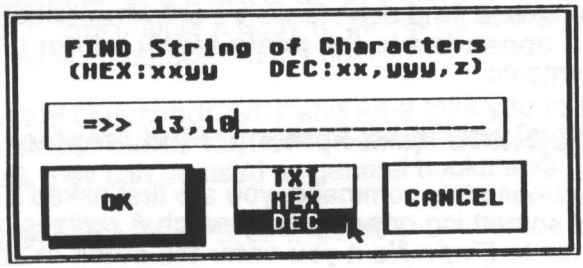
For instance, to enter CHR(155) as a hex value, you would type '9B' without the quotes and click on the Hex button. For decimal, you would type in '155' and click on the Dec button. The null character would be '00' in hex or '0' in decimal. If you are a masochist, you could also use hex values to enter normal text -- the hex string for 'XYZ' would be '585960'. The letters A-F in hex strings can be entered in either upper or lower case.

For a listing of all 255 characters and how to enter them see the section titled, "ASCII and the Character Set".

After entering the search string, you are asked whether to make the search case sensitive or not. This initially defaults to No, which means that it will match either upper or lower case versions of the search string. Anytime you change the case sensitivity of the search, that becomes the new default until you reboot the ACC or rerun the PRG. To keep the current default in future sessions you would need to save your configuration with the New Config command.

If you have chosen Replace, then you get to enter the replacement string. Then you pick whether to Query (i.e., ask you what to do when it finds each match) or just go ahead and replace every match it finds without asking. The default is always Query. If you pick Query, then each time it finds a match you get to tell it whether to (a) skip to the Next match without replacing this one, (b) Replace this match, or (c) Quit searching.

Once a search string has been entered, you can always just hit F6 to find the next match.



Searching for CR/LF in Decimal

As a final note, if you need fancier search/replace capabilities, you might try another program on your disk called SRCHREP.PRG, which allows various types of wildcards in search strings (alpha, decimal, hex wildcards) and replace strings (leave unchanged, make lower case, and make upper case wildcards), as well as replacing only every Nth match when doing a global replace. Using this, you could do things like take a file that is all upper or lower case and capitalize the first word of each sentence. The SRCHREP program is freeware, but it is limited to a 64K file size.

INPUT & OUTPUT

(Disk Files, Disk Sectors, Printer, Modem, MIDI)

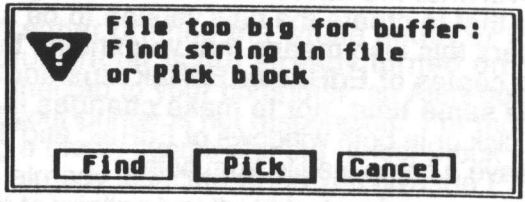
This section covers reading and writing to the disk, and sending output to the printer (parallel) port, modem (serial) port, and music (MIDI) port. A more detailed discussion of dealing with disk sectors and binary files is contained in the Hacking section below.

Open File / Disk

(Alt-O, Alt-R, menu: Open...)

This lets you read a file of ANY type (text, doc, prg, tos, rsc, etc.) or any range of disk sectors into the edit buffer. If the file is anything other than ASCII text, you will see funny characters on the screen for each non-ASCII byte.

If the file is larger than the buffer, you will be asked what portion of the file to load. You can even search for something and the part of the file containing it will be loaded. You can use the keyboard or mouse to set which part of the file to load, or where to start searching -- zero loads the first part of the file, '1' loads the file starting at a point 1K (1024 bytes) into it, '2' starts 2K into it, etc. The amount of the file that gets loaded is 2K less than the buffer size. So, with the default 10K buffer, it will load in 8K. More information on large files is given in the following Extended Files section.



This lets you choose where to start editing a large file

If you already have stuff in the edit buffer that hasn't been saved, you will be asked if you want to save it first. After taking care of that, you will be asked if you want to open a New file or Merge a file into the current one at the cursor. Opening a new file will destroy everything currently in the buffer.

If you choose to merge a file into the current one and there is not enough room in the buffer, the merge will not be done at all -- there is no provision to merge just part of a file. You can always get around that by first saving the current file, loading the one you wanted to merge, saving a block from that one, loading the original file, and then merging the block that you had saved.

If you leave the filename portion of the file selector blank, then the disk drive you have chosen is opened as if it were a huge file, and you get to choose which block of sectors to load into the edit buffer. Search/loading also works here. For information on opening and editing files other than ASCII text files (including disk sectors), please see the Hacking section below.

Extended Files (ANY file larger than buffer)

Even with EdHak's small (10K) default buffer size, it will handle ANY size file by loading in only a portion of the file that you want. It is "limited" only in that you can't choose a specific portion beyond the first 99 megabytes, just because the biggest number the dialog box lets you enter is 99999. If anyone finds this to be a limitation, let me know!

Block zero is the first 8K block of the file, and the dialog box will allow you to go up to a block starting at about 1K from the end of the file, which would give you over 8K of free space to add text to the end of the file. When you go to save the file, EdHak will take care of stringing together the text from the original file with the block of it that you edited to create a new file. This means that it won't overwrite the latest version of the original on disk, since it needs that to create the new version to be saved. For you power users this also means that you should be careful if you have two copies of EdHak or EdHak plus another editor running at the same time, not to make changes in the same document on disk or in both windows of EdHak, and then expect to be able to save a complete file from both.

Since you can only load an 8K block of a file that is bigger than 10K, it will require more work to do some things, like doing a cut/paste from the end of a big file to its beginning. To do such a thing you would need to first save the file after the Cut, then move to the other portion of the file (saving the changed portion), and finally do the Paste. Or to handle a block larger than the paste buffer size, you can first save the block to disk and then use the Merge feature to paste it at the desired location.

Save File / Disk
(Alt-S, Shift-Alt-S, Alt-A, Shift-Alt-A, menu: Save...)

Without the shift key, this gives you a file selector to choose the filename you wish to save the file as. It will default to the filename you last opened or saved if there was one. If you try to save to a filename that already exists on the disk, it will warn you of this and ask if you want to Overwrite the old file, or Append to the end of that file. The Append option is handy if you are using it as an actual diary, or if you want to add text to the end of a large file without having to read that file into the buffer first. Note that this initially defaults to Overwrite (without a backup), but then defaults to whichever you last used. Also, using the configuration options described above, you can force it to initially default to Append and/or always save with a BAKup copy if you wish. No matter how a file or block is saved or appended, it will update the time/date stamp on the disk file. With the shift key combinations, the file is saved with NO questions or warnings.

If you have a block selected, you will be asked if you want to save just that block or the entire buffer. It defaults to saving only the block.

If you already know you want to append to an existing file, you can take a shortcut and just hit Alt-A, which will force the current file (or a selected block) to be appended to whichever file you choose from the file selector.

Print (Alt-P, menu: Print...)

This allows either printing the whole buffer or just a marked block. If your printer is not already turned on, or if it is not properly connected to your computer, an Alert box will instantly tell you that your printer is not responding. For more information on selecting a block, please see the Select Block section below. If a block is already selected when you give the Print command, it will default to printing just that block unless you click on the 'All' button in the alert box.

Note that you can send any printer control codes you want to set left margin, bold, near letter quality, or any other things your printer can do. All you have to do is look at your printer manual to find out what codes do what. For example, on my printer (which is pretty much Epson compatible) the code to set the perforation skip is Esc-'N'-'x', where x is the number of lines to

skip. To get the escape character, type Control-[or Alt-Esc, then type upper case N. Then type Control-I to set a perforation skip of 9, since 'I' is the ninth letter of the alphabet (don't worry about the funny character that appears - that's just what the ST uses for displaying this control code which is normally non-printable).

If you have one or more printer control strings that you want to be able to send to the printer without typing them in each time, this can be done using macros. If you have the Printer Init selected in the config options, macro F10 will automatically be sent at the start of any print operation. Please see the sections on Macros and Configuration below for more details on this.

EdHak has no built-in support for the Atari laser printer. So if you use a printer such as this with no resident font, you will need to use some printer emulation software such as Laser Brain (available from many bulletin boards).

If you need to print out of the serial (modem) port, please see the Upload section below.

Upload to Modem or MIDI Port
(Alt-U, menu: Upload)

This command prints the buffer or just a marked block directly out the serial RS232 port (modem port) or out the MIDI music port, just as if you had Printed it to the printer. It uses no transfer protocol, not even the ASCII protocol. Note that it treats the output as if it were going to a printer, so it first sends the printer setup string (macro #10) if you have the configuration set up to do that. Then it sends spaces at the start of each line if you have a left print margin set to anything higher than zero, unless you are sending to the MIDI port. So you may need to change your configuration temporarily before using this command -- but you don't need to reboot to make the new configuration active. As with printing, a carriage return is sent at the end of the transmission to make sure the receiving device gets the last line.

You could use the upload to modem command to compose and send messages from within a GEM terminal program or to send modem control commands to your modem without running a

terminal program at all. With the upload to MIDI command you could send a segment of a MIDI file from your buffer to your midi device to see what sounds are created. Note that MIDI files are not text, so you should toggle into Hack mode to be able to see everything in the file.

COMMUNICATING WITH OTHER PROGRAMS

(or multiple copies of EdHak)

Using Disk Files to Import / Export

The most straightforward and compatible method to transfer text or other data among the greatest number of current programs is to use intermediate disk files. This means one program saves a block of text or data to a disk file (exports it), and another program reads in that file (imports it). In general it makes no difference what filename you use for this, as long as you are able to find the file when running the program that is going to import it. Besides allowing text transfer between EdHak and other applications, this can be used within EdHak itself or between two copies of EdHak both running on your machine at the same time.

One example of where you might use this method is in preparing text to enter into a desktop publishing program like Pagestream or Calamus. You can create all the text for a document within EdHak, then save the file to disk giving it any name you like. Then open Pagestream (or just close the EdHak window if you already have Pagestream open underneath it), make sure it is in text entry mode, and choose Import Text with carriage return at end of each line.

Clipboard

Beginning with version 2.30 EdHak supports the GEM clipboard when doing a block save or merge. It also supports the existing Stalker/Steno clipboard which until versions 3.01 & 2.01 was slightly different than the Atari GEM standard clipboard, because no standard had been set by Atari until 1991. Note that EdHak does not use the clipboard for its own internal cut/paste commands because it does them in RAM which is much faster if

you don't have a ramdisk or harddrive to use for your clipboard. So, if you wish to export or import text or data using the clipboard you need to use the block save or file merge commands in EdHak.

The Atari clipboard is just a path (often to a folder called 'CLIPBORD', but not necessarily) that is stored in your computer's memory that any program can find out about. If a program supports the clipboard, before reading or writing to the clipboard it will ask the computer's operating system where the clipboard is (i.e., what disk drive it's on and what folder it is in). If no path has been set yet, then the program will automatically decide where to put the clipboard or as in EdHak the user will get to decide where to have the clipboard. For the best speed it should be located on a Ramdisk or a hard drive.

Using the clipboard is basically the same as using an intermediate disk file as described above except that with the clipboard the program is able to locate the file rather than making you find it, because the file will always be in one place and it will have a known name.

Copying TO the Clipboard (Alt-S, menu:Save or menu:Block)

First you need to highlight a block in EdHak by using F1 and F2 or click-dragging the mouse. After highlighting the desired block you would copy it to the clipboard most quickly by hitting Alt-S (file save), <Return> (yes, just save the block), <Return> again to save it to the default SCRAP.TXT clipboard file, and finally if SCRAP.TXT already exists click on Overwrite to have your block be the only thing in (on?) the clipboard. By choosing Append instead of Overwrite you could combine a number of separate blocks into a single clipboard file. This can all also be done from the menu by clicking on either Save... or Block... instead of the keyboard Alt-S command.

Although this method is not as automatic as in other applications where you would simply use a copy command to copy a block to the clipboard, this EdHak method does give you more complete control over all the clipboard files. There are advantages to both approaches, and it is expected that future versions of EdHak will provide an option to use the clipboard in the more normal way with cut and copy commands that directly use the clipboard.

Reading FROM the Clipboard (Alt-M, menu: Open)

Decide where in your document you want to put the contents of the clipboard, and place the cursor there. Then use the keyboard Alt-M (file merge) command (or menu:Open command and click on Merge) and hit <Return> to select the default SCRAP.TXT clipboard file. If there is room in your buffer to insert the file, it will be copied from the clipboard into the document in your buffer at the cursor location.

Setting or Changing the Clipboard Path (Shift-Alt-C)

The keyboard command Shift-Alt-C can be used anytime to set the Clipboard path or change the existing Clipboard path. Changing the clipboard path will not affect any files that are in the old clipboard, but when you try using the clipboard from any application after that, it will only find the new clipboard until you change the clipboard path back to the old one. This feature can be used to set a temporary path for EdHak to save blocks to or merge files from. For instance, after capturing a bunch of e-mail from an electronic bulletin board into EdHak (either directly from STalker or by loading the captured file from any terminal program into EdHak after the session), you could change the clipboard path to a folder where you keep copies of correspondence. Then highlight each person's e-mail and append it to a file containing all your correspondence with that person using Alt-A or the Save... command from the menu. When done saving all the mail, change the clipboard path back to wherever you normally have it.

Clipboard Cautions

If you use a number of different applications that use the clipboard, you may see a variety of SCRAP files there. Text files will generally use the filenames SCRAP.TXT or SCRAP.ASC depending on whether a carriage return/linefeed is expected at the end of each line. EdHak assumes the SCRAP.TXT name since it assumes a CR/LF at the end of each text line. Since EdHak does not automatically delete other kinds of SCRAP files when it writes SCRAP.TXT, you might need to do that on your own when exporting data to another application if that application first tries reading SCRAP.ASC rather than SCRAP.TXT and a prior SCRAP.ASC file exists in the clipboard.

"Kwiksend" (Alt-K, menu: Quit)

Version 2.13 introduced the the first "Kwiksend" feature (sorry, but 'Q' was already being used for Quit). Kwiksend is a way to send a block of text of ANY size from EdHak running as an ACC to any other open application or accessory window that can accept text input from the keyboard, without having to save it to an intermediate disk file. Since EdHak uses your computer's keyboard buffer to send the text, your other application or accessory receives the text just as if you typed in the characters on your keyboard.

To use Kwiksend, first select a block you wish to send using either F1 and F2 or click-dragging the mouse. Then either hit Alt-K or choose Quit from the menu.

Kwiksend will then give you the option of stripping out the Carriage Return / Linefeed at the end of each line. This can be useful when sending text into something like Pagestream where you may have a different column width than you had in the original text. When using the option to strip CR/LF, it will only strip the first of a series of them, so paragraphs separated by one or more blank lines will remain as separate paragraphs.

The Kwiksend function has a built-in initial delay of about a half second to allow the receiving application to be fully ready to receive text after the EdHak window closes. This even makes Kwiksend able to send text into ST Writer (which can not access desk accessories from its edit window). The only trick is that you must hit the 'E' to go into the ST Writer Edit mode **Immediately** after the last EdHak Kwiksend command, which is choosing whether to strip carriage returns or not.

Once the sending has begun into the receiving application's window, you can abort it instantly at anytime by hitting the Alt key.

Getting Kwiksend to work properly with certain other applications has been somewhat tricky. There are still a few word processors, especially European ones like Tempus or That's Write, with which you might have a problem of lost characters, at least if you try to send more than a couple lines of text. Please see the COMPAT.TXT file on your disk for the latest details on this if interested.

STalker

(terminal program by Eric Rosenquist, distributed by Gribnif)

Beginning with version 2.3 of EdHak, it can function as the capture buffer for STalker if you have EdHak installed as a desk accessory. This works either with the EdHak window open or closed. To make this work you must set the STeno name in the STalker preferences to be the filename of your EdHak accessory MINUS THE .ACC EXTENSION (e.g., use EDHAK230 rather than EDHAK230.ACC). Then save STalker's settings and quit Stalker. The next time you run STalker it will use EdHak as its capture buffer. When the EdHak buffer fills, it will ding (if you have your monitor volume turned up) and give you the options of saving it to disk and/or clearing it to continue capturing.

If the file you are capturing to in EdHak is named "CAPTURE.TXT," then anytime the buffer fills it will automatically append itself to that file and then clear the buffer to continue capturing.

Other functions that STeno may do with STalker, such as serving as a type-ahead buffer, are not yet implemented in EdHak.

QuickCIS

(CompuServe Navigator by Jim Ness)

Beginning with version 2.24 of EdHak and version 1.70 of QuickCIS, you can use EdHak as the message editor for QuickCIS if you have EdHak loaded as a desk accessory. The two things you must do to make this happen are (1) name your copy of EdHak that you want to use for this purpose, EDHAK.ACC, and then reboot your computer (or reload that copy of EdHak into MultiDesk), and (2) configure your QuickCIS to use EdHak as its editor by clicking in the appropriate preference box in QuickCIS.

This interface was worked out with the author of QuickCIS because the internal editor in QuickCIS is very limited compared to the capabilities of EdHak. The demo version of EdHak, available for free on CompuServe, will still function as the QuickCIS editor, so Jim Ness expects future versions of QuickCIS to continue to use EdHak for easy message editing.

When using EdHak from within QuickCIS, there is an optional exit command you can use (Shift-Alt-X), which allows you to easily toggle back and forth between the QuickCIS message screen and EdHak.

Hybrid Arts Music & MIDI Software

Hybrid Arts has been including the stripped down freeware version of EdHak called DIARY20S.ACC in their MidiTrack software package to provide a way for musicians to enter lyrics or other text into music being composed. Currently, this is done by saving the desired text from DIARY to a disk file which can then be imported into MidiTrack. Since DIARY will operate as a desk accessory, this can all be done without leaving MidiTrack. If you would like a more direct (instantaneous) text transfer method in the future, you should ask Hybrid Arts to incorporate support for the direct transfer method that EdHak uses with the CompuServe navigator, "QuickCIS," as described below in "Offer to Other Programmers."

Offer to Other Programmers (Interfacing directly with EdHak)

On your disk is a file called OFFER23.TXT, which describes two methods of passing text or data between your application and EdHak.ACC. This can allow your program to have access to a good editor without you having to write one yourself. Both methods make use of the GEM AES message pipe to pass instructions between the programs. In the first method buffer location and size are passed so that EdHak can instantly copy the appropriate section of memory from your application into its buffer for editing and then back to your application for whatever purpose you want. This is the method used between EdHak and QuickCIS.

The second method involves passing EdHak a filename and path, so data can be transferred automatically between the applications via a disk file.

Also, EdHak.PRG can always be invoked via a GEM pexec call with the desired file to open given as a command line parameter.

Launch a Program (Alt-L, menu: Launch PRG)

This can be quite useful or quite deadly (crash-causing) depending on what you try to do with it. It lets you execute another program (PRG, TOS or TTP) without leaving EdHak, and then automatically return to EdHak as soon as you quit the other program. This works best when running EdHak as a PRG, but many things will also run if running as an ACC. However, if you are running one application, then open the EdHak accessory and try to run yet another program from the EdHak Launch command, you will often find there is not enough RAM left to open it, since the first application might have already taken most of the available RAM. You can check on this by looking at EdHak's first Config screen (Alt-N, menu:New? config) to see how much Free RAM there is before trying to run something else. In this situation it is best to use the Launch (or Execute) command of the first application if it has one (such as in Flash or Tempus).

In general, most programs will run ok from EdHak running as a PRG, fewer will run if your are running as an ACC from the desktop, and far fewer will run if running EdHak as an ACC within another application. When running as an ACC, it seems that problems generally occur if you try to run any application that has a menu bar, but other programs like DC Format, PicSwitch, ArtST, ProCopy, ARC Shell, and STicker all seem to work fine.

Running EdHak from other Programs or from a Command Line

When you have a copy of EdHak that has been renamed with a .PRG extender, you can call it from any other application that allows executing or launching a program from within it. Some examples include the Flash terminal program, the Tempus editor, the ARCShell archiving utility, and of course EdHak itself. Furthermore, some programs such as command line interpreters (CLI's) and programming shells allow passing a filename along with the call to run another program. When a filename is passed to EdHak, it will automatically load that file into its edit buffer when it opens up. This allows EdHak to be installed and used as the default editor in some programming shells. This is essentially what the Atari desktop does after you have used the Install Application option.

CONVERSIONS

This section covers various types of conversions you can do with EdHak, including converting among different types of text files, conerting tabs to spaces and spaces to tabs, encryption and decryption, and conversion to and from hexadecimal.

Text files of Various Types
(Macintosh, Unix, DOS, 8-Bit Atari, etc.)

If you ever (a) use a word processor and try doing something with a saved DOC file in another word processor or editor or (b) trade text documents with friends or (c) download files from bulletin boards, then you have undoubtedly run into strange looking things on your screen, and you have entered... the ASCII Twilight Zone.

As described more completely in the section "ASCII and the Character Set," ASCII provides a standard for text representation; however, it leaves a couple things somewhat up in the air. Most importantly, the character combination to represent the end of a line or paragraph has gone in different directions on different types of computers.

The Atari ST/TT computers have basically followed suit with the DOS (IBM compatible) machines and use a Carriage Return/Linefeed combination (CR/LF which is character #13/#10) for the end-of-line indicator. Macintosh software uses only the Carriage Return (CR), and last we checked Unix software used only the Linefeed (LF). If you had a Classic 8-bit Atari, it used character #155 for the end-of-line, which on the ST shows up as a cent symbol.

Many utilities have been written by many programmers to convert among these text formats. Although EdHak may not be the ultimate in user-friendliness in performing these conversions, it can do them all if you know how to use its Find/replace command.

If you have a text file that does not display properly in EdHak's text mode, then it is probably one of these other file types. For instance, if everything is on one very long line, then either there is something other than CR/LF being used at the end of each line or it is a related type of word processor file that has nothing

at the end of each line but only has something at the end of each paragraph. To find out what is going on and what to do about it, just convert to Hack display mode (Alt-T: Text/Hack). Look at what should be the end of a line and see if there are any funny characters there, such as a little 'cr' (Carriage Return) or a musical note (the ST's Linefeed representation) or a cent symbol (8-bit Carriage Return) or something else.

If you find one (or more) of these, or any other funny character, put the cursor on it and write down the decimal or hex number (hex has a '$' before it) following the little 'c:' located at the right side of the the window title bar. This number is the character value of that (hopefully) end-of-line character. Now move the cursor back to the beginning of the buffer and do a Find/replace operation using Alt-F or menu: Find/replace. For the Find string put in the character or characters you wrote down, remembering to click on DEC or HEX depending on whether it was a decimal or hexadecimal number. For the replace string put '13,10' without the quotes and click on DEC. Then Replace All and switch back to text display mode to see how it looks. If done correctly, it should now be displaying nicely on your screen.

If you did not find any sort of end-of-line characters, you probably just need to do a Wrap/reformat command which will put CR/LF at the appropriate places in your document.

Some other types of files are harder to decipher, but not impossible. Word processor DOC files will typically start off with some non-text header that could contain information on margins, page length, font type and size, etc. If all you want is the text without formatting information, you can just mark that header as a block and delete it. If the document had right-justified text in it, there will probably be some weird looking characters separating words, rather than plain blank spaces. Again, if you just want the text from this document, you can make a note of what those characters are in Hack mode and then replace strings of them with a single space character.

If you have a text file that you want to give to a friend who has different type of computer, you might try helping him/her out by converting the file the other direction into the most approprite format for their machine. For example, to create a Mac text file you would find all the CR/LF's and replace them with CR's.

Tabs <=> Spaces (Ctl-Tab, Alt-Tab)

The key combination Control-Tab gives you the choice of converting from Spaces to Tabs or from Tabs to Spaces or Canceling. Note that EdHak displays tab characters as just what they are, a single little clockface which is character #9. It is up to you to make sure before using this command that the tabstop interval is set the way you want it using the Alt-Tab or menu: New config command. This command does NOT simply exchange a tab character with the number of spaces per tabstop. When converting to tabs it checks each tabstop and if that position is preceded by at least two blank spaces a tab is substituted for as many spaces as there are, up to the spaces-per-tabstop in your current configuration. When converting to spaces, it finds each tab character, changes it to a space and continues inserting spaces up to but not including the next tabstop.

Please forgive the lack of speed on this one -- a speed increase is one of the many things on the list for the future.

Encryption (Alt-E, menu: Encryption)

This command lets you make anything in your buffer confidential by encrypting it using a password of your choice. You MUST remember your password to be able to unencrypt it, which is done using the same command. You can even encrypt it more than once using different passwords, and then unencrypt it using the same passwords in any sequence you like. One major caution on encryption: If your password is more than one character long (which is certainly a good idea), then do NOT use the Append, Merge, Block Save features or load/save a file larger than your buffer because the decryption will not work properly unless you get very lucky (e.g., the file length is an exact multiple of the password length.) If you need to use encryption and the file of interest is larger than your buffer, you can either (a) reconfigure your buffer to be large enough for the whole file, or (b) use some other encryption utility such as one available in various popular compression utilities like ARC, LHARC or ZIP.

Hex <=> Characters
(Alt-T, menu: Text / Hack / Hex)

This command converts your entire edit buffer contents from characters to hexadecimal values or vice versa. You MUST remember to convert back to characters before saving if you want the original file to be in the form it started as. Otherwise, you can always reload the hex file back into EdHak for re-conversion from hex to characters.

Note that conversion TO hex requires the buffer to be twice as large as the original file, since two hex digits are needed to represent each original character. When converting from text to hex, EdHak automatically also changes the display into Hack mode and doubles the cursor offset from the start of the buffer so the cursor is at the hex representation of the same character it had been resting on.

When converting from hex to characters, EdHak automatically halves the cursor offset from the start of the buffer to place the cursor on the character where it started. However, it does not automatically take you out of Hack display mode, since the type of data you would most likely be editing in hex mode would not be plain text. If it is text, you can easily make the final conversion from Hack display to Text display mode with the Text/Hack button of the Alt-T command.

[Area left blank, preceding Hacking section]

HACKING

This section covers the editing of binary (non-text) files, disk sectors, and RAM. Information is given that should allow the novice "hacker" to have some fun and learn a few things about the structure of files, disk directories, and RAM. The term "hacking" is used here with its earlier meaning of exploring computer operation and finding creative solutions, rather than the more recent twisted meaning with destructive connotations.

Toggle Text/Hack/Hex mode
(Alt-T, menu: Text / Hack / Hex)

For most word processing needs you will want to use Text mode, which starts a new display line for every CR/LF (character #13/#10) combination in the file (or any time you hit Return while entering text from the keyboard). Text mode displays every character, 0-255, EXCEPT the CR/LF combination, which it either displays as the little 'cr' symbol or as a blank (null) space at the end of each line depending on your configuration. Also, while in Text mode, the current Line and Column of the cursor are shown in the top window bar, measured from line zero at the top, and column one at the left edge.

Hack mode displays all 0-255 characters including the CR/LF combination as individual characters. In Hack mode, CR and LF are treated just like any other characters, and every line is displayed with a length of 64 (or 32 in low res) unless you change the right wrap column while in Hack mode. This is similar to the displays found in disk sector editors, except that it does not automatically display the whole window in hexadecimal.

The hex or decimal value of the character at the cursor is shown in the top window bar, preceded by a '$' sign if in hex. To the left of this character value is 'n' which is the number of bytes (characters) you are from the top of the buffer -- this is often called the offset. The first character in the buffer therefore has an offset of zero.

In the sample display below, the cursor is positioned in the second line on one of the null (character value = zero) bytes at the end of an alert string in a binary file. You might find character strings like this near the end of executable program

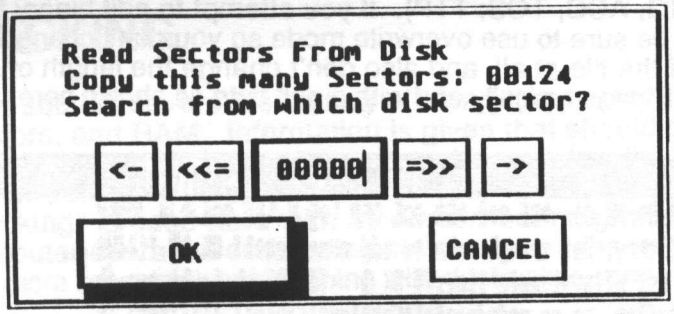
files (PRG, ACC, TOS, TTP). If you attempt to edit binary files like this, be sure to use overwrite mode so you don't change the length of the file at all, and also don't change the length of any strings. Strings typically end with a null byte as shown here.

Sample Hack Mode Display

If you need to look for a string of hex or decimal values, you can let the Find command do it for you rather than to trying to do a visual search of the screen. However, for those who prefer to see everything in hex, you can convert the entire buffer to hex using the Alt-T command. After working with it in hex format, you would want to convert it back from hex to characters using the same command. Please see the section on Conversions for more details on the hex <=> character conversion procedure.

Disk Sector Editing
(Alt-R, Alt-O, menu: Open...)

You can use EdHak as a disk sector editor! To do this, start as if you were going to open a file, but leave the filename part of the item selector empty and just specify which drive you want. Then open it and you will be asked which sector to start reading from. Sector zero is the boot sector, and this is followed by two copies of the FAT (file allocation table), then the root directory, and finally the actual data of all the files on the disk and any subdirectories (lists of files in folders).



Search-Loading Disk Sectors

Unless you are using a strange format for your drive (i.e., something not supported by GEM) or have a very large hard drive partition, files are stored in 1K chunks called 'clusters,' each of which consists of two 512-byte 'sectors'. Some hard drives with large partitions (e.g., over 32 megs) may be set up with larger clusters and sectors such as 2K clusters and 1K sectors. Regardless of the size, as long as it is a format supported by GEM and not something like Magic Sac or Macintosh format, EdHak should deal with it just fine. EdHak will load as many sectors as will fit into the buffer and still leave you a couple K free in case you want to add to it, just as if you had loaded part of a file that was too big for the buffer.

If you don't know anything about disk structure, do NOT try saving this back directly to disk sectors. If you want to save it as a file, then give it a filename when saving and all will be safe. If you DO know what you are doing, then this disk editing feature provides some different options from anything you are likely to find in a typical disk/sector editor, since you can view/edit more than one sector at a time, toggle into text mode, insert, append to files, etc. EdHak can also serve as a substitute for programs that let you save the boot sector, FAT, and root directory to a disk file in case they ever get corrupted and you want to restore them.

Following is some basic information on the structure of the boot sector of a disk and the structure of disk directories which you may find useful or at least fun to investigate. For a more in-depth explanation of these and related subjects please see a more complete reference work such as "Atari ST Disk Drives Inside and Out," by Uwe Braun, Stefan Dittrich, and Axel Schramm, published by Abacus Software.

Note that any 2-byte or 4-byte numbers in the boot sector or directory sectors of an ST disk are stored in "reverse" sequence to mimic the way DOS and the Intel microprocessor chip (IBM compatible) store numbers rather than the way the ST's Motorola 68000 series chips store numbers. This is done to allow the ST to read and write IBM disks. As an example, the hex number $800 (decimal #2048) is normally stored in the ST in 2 consecutive bytes with the high order byte first as 08 00. However, if that same number were used in the boot sector or disk directory, you would find it appearing reversed as 00 08.

Boot Sector Table

Byte # (In hex)	Contents
$00-$01	2-byte branch command to boot program if present
$02-$07	Reserved fill bytes or loader (some viruses use these bytes)
$08-$0A	Disk serial number
$0B-$0C	Number of bytes per sector, usually 512
$0D	Number of sectors per cluster, usually 2.
$0E-$0F	Number of reserved sectors, usually 1.
$10	Number of File Allocation Tables (FAT's), usually 2.
$11-$12	Max number of directory entries in root directory, usually 112.
$13-$14	Total number of sectors on disk, usually 1440 for DS floppy.
$15	Medium description, not used.
$16-$17	Number of sectors per FAT, usually 5 (Apple File Exchange prefers this to be 3)
$18-$19	Number of sectors per track on, usually 9 for floppy.
$1A-$1B	Number of sides on disk, 1 or 2 for floppy.
$1C-$1D	Number of hidden sectors, usually zero.
$1E-$1FD	Other information used for boot disks.
$1FE-$1FF	2-byte checksum of boot sector contents.

Disk Directory Sectors

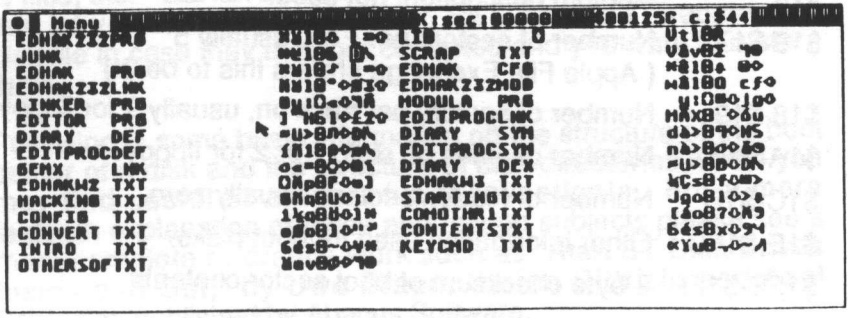
Each directory entry contains 32 bytes. As mentioned above, beware that any 2 or 4-byte numbers are stored "backwards" with the highest order byte last. Following are tables showing the structure of the directory and its contents as well as a sample display of the edit buffer when containing directory sectors.

Disk Directory Structure

Byte # (decimal)	Contents
1-8	8-character filename
9-11	3-character file extension (type)
12	1-byte file attribute bit flags (see table below)
13-22	10 reserved bytes
23-24	2-byte time stamp (see format below)
25-26	2-byte date stamp (see format below)
27-28	2-byte Number of the first cluster that the file uses.
29-32	4-byte File size, in bytes

File Attribute Flags

Bit #	What it means if set = 1
0	Read Only
1	Hidden File
2	System File
3	"File" name is actually the disk volume name.
4	"File" name is actually a folder name.
5	Archive bit, file has changed (since last backup)



Editing Directory Sectors of Disk

Time / Date Stamp Format

The time stamp is stored as follows. The highest order bit comes first.

Bit #	Contents
Time:	
0-4	Seconds divided by 2, 0-29 (5 bits isn't enough to count to 60)
5-10	Minutes, 0-59
11-15	Hour of day, 0-23
Date:	
0-4	Day of Month, 1-31
5-8	Month of Year, 1-12
9-15	Years since 1980, 0-127

View / Edit RAM
(Alt-V, menu: View / edit RAM)

Yes, you can load any part of your ST's memory into the edit buffer, either by searching for a sequence of characters or by choosing which specific block of memory to load. Then you can edit it, save it to a file, or even write it back RAM in its original or some other location. Hack away! but don't complain to me if your machine crashes after you change something in RAM. The RAM you can access includes addresses from zero up to whatever your operating system thinks is the top of physical RAM ("phystop"). This includes all the system variables, screen ram, etc., but not Read-Only Memory (ROM). Note that with many ramdisks phystop is lowered to fit the ramdisk above it, so phystop becomes lower than the actual top of physical ram. To get to that RAM in a ramdisk you can use the disk editing feature described above.

Some interesting and potentially useful addresses in your ST's memory are listed below. For a more complete list you should find one of the TOS reference books available from your Atari dealer or by mail order. A couple examples include:

Compute's Atari ST, Volume 3, "TOS" by Sheldon Leemon, published by Compute! Books.

Atari ST Internals by K. Gerits, L. Englisch, R. Bruckmann, published by Abacus Software.

Some System Variables

Name	Decimal	Hex

phystop 1070-1073 $42E-$431

This contains the address of the physical top of RAM, which is actually the first non-usable byte at the high end of memory. It is often adjusted downward by ramdisk programs to fit the ramdisk above it.

membot 1074-1077 $432-$435

The address stored here is the lowest currently available byte in memory.

memtop 1078-1081 $436-$439

The address stored here is the highest currently available byte in memory.

seekrate 1088-1089 $440-$441

The 2-byte integer stored here contains the default seek rate for any connected floppy disk drives. Only the first (lowest order) two bits of this word are used for this as follows:

Bit 1	Bit 0	Seek Rate
0	0	6 ms
0	1	12 ms
1	0	2 ms
1	1	3 ms (default)

Some System Variables, cont'd

timr_ms 1090-1091 $442-$443

This 2-byte integer is the number of milliseconds between system timer ticks.

fverify 1092-1093 $444-$445

This 2-byte integer is just a flag that determines whether write verify is turned on for floppy disks. If zero, verify is off, otherwise verify is on which is the default.

almode 1096-1097 $448-$449

This 2-byte integer is a flag that determines whether video output is the US standard NTSC 60 Hz (if zero) or European standard PAL 50 Hz (if non-zero).

v_bas_ad 1102-1105 $44E-$451

The address stored here is the starting address of screen RAM, which is a 32K memory area containing the image displayed on the screen.

nflops 1190-1191 $4A6-$4A7

This 2-byte integer is the number of actual floppy drives connected to the computer, which can be zero, 1 or 2. If you plug in a floppy drive but don't reboot, you will need to change this number for the drive to be recognized by the operating system.

drvbits 1218-1221 $4C2-$4C5

This is a 4-byte bit flag in which each bit that is set indicates the logical drives that are connected. If just one floppy drive is connected, both bits zero and 1 are set, since the operating system uses it as logical drives A and B. Depending on version of TOS, this might only allow up to drive P.

end_os 1274-1277 $4FA-$4FD

The address stored here is the first byte past the area of RAM used by TOS. This is the starting address of the transient program area.

ASCII and the Character Set

(See Appendix A for full Table)

What is ASCII? And what is all this hoopla about "all 256 characters"?? ASCII stands for American Standard Code for Information Interchange. Every character you see on a computer screen (not just Atari) is stored in a byte. A byte is a unit of storage on a computer that holds 8 bits of information. A bit is just a tiny recording of either 1 or 0, on or off, yes or no, something or nothing, black or white (good or evil?), depending on how you prefer to look at it. If you are mathematically inclined you can figure out that with 8 things that each have two possible values, there are 256 total possible combinations (or is it permutations?). Therefore, each byte can hold one of 256 possible values numbered from 0 to 255.

The values go from 0 to 255 because programmers assume the bits to stand for either 1 or 0 and they use them to count. If all the bits are 'off' this means zero; if the first bit is 'on' this means a value of one; if the second bit is 'on' it means a value of two; if both the first and second bits are 'on' that equals three; just the third bit 'on' means a value of four. As higher bits get turned on they represent bigger values, each bit being twice as big as the previous bit. When they are all 'on' it adds up to 255.

The software you run on your computer, including the built-in operating system (TOS on the Atari ST/TT) translates this value into the characters used for words (A-Z & a-z), punctuation such as !;':"?-., numbers (0-9), other printable symbols like @#$%*, and other funny symbols that your printer wouldn't know what to do with. ASCII is simply an agreed upon method to translate between the value of a byte and the character it goes with. Appendix A has a table listing all 256 values and the character that goes with each of them. Actually, the ASCII standard only covers the first half of the values (up through #127). Values from #32 up to #126, are the 'printable' characters that a normal word processor is willing to show you. Values below #32 are where the printer control codes are found. For instance, a value of #13 has a function of Carriage Return (CR), which moves the print head back to the start of the line, and the value #10 is Linefeed (LF), which moves down to the next line. Value #12 is Formfeed (FF) which moves the paper to the start of the next page.

The characters associated with these control codes and values from #127 to #255 vary with different types of computer (ST is different from Mac which is different from IBM). Also, for any of you who deal with older 8-bit machines like the Classic Atari (800, XL, XE, etc.) or Commodore, beware that they only use part of the full ASCII standard. If you have to do anything on IBM mainframes you might run into something called EBCDIC, which is a totally different translation method from ASCII, but luckily it is possible to translate between them.

TECHNICAL NOTES

Text Display Speed and Clipping

EdHak v 2.2 now uses the built-in GEM VDI graphics text output routine rather than writing directly to screen ram as it did in versions 1.8 - 2.1. As long as the character output is forced to be at byte boundaries (i.e., some multiple of 8 pixels from the left edge of the screen), this VDI method is plenty fast, and when combined with a software accelerator such as QuickST or TurboST, the speed is about as fast as can be done. (My hat is still off to Tempus, which is still faster, but it doesn't run as an ACC.) By using the VDI text output, compatibility should generally be better. Also screen redraws when moving windows around are more complete, since GEM draws partial characters when necessary (i.e., when the clipping rectangle didn't happen to be on a character boundary), whereas the old EdHak text output routine could only draw complete characters.

Memory Management & Buffer Size

The buffer size must be saved to the main prg or acc file as part of the BSS length in the file header (rather than reading it in from a separate config file) to be able to safely allocate the proper amount of memory. Other desk accessories that use a more typical method of allocating memory (dynamic allocation using a Malloc call) are not handled properly by TOS and will steal gobs of extra ram if you ever try switching between low and medium resolutions, because TOS does not deallocate that ram before allocating it again. STeno, which I also consider to be a fine editor, gets around this by letting the user deallocate the ram when desired.

EdHak handles the text buffer as simply one continuous block of RAM, and so to insert or delete near the beginning of that can require moving around a large chunk of ram. Despite the fast machine code routine that handles this, it obviously isn't fast enough when you get above about 25K. The way other word processors get around this is to dynamically allocate small chunks of ram as needed and link them together with pointers, thus requiring much fewer actual bytes to be moved. If that approach is used in an ACC things can get screwed up in a low<->med resolution switch. Incidentally, the slowdown is exactly the same thing you run into with Flash's buffer, if you ever use that. This slowdown is one main reason why I have begun work from the ground up on an all-assembly language version which will use pointers and blocks of RAM (but will still be crash proof for resolution changes). The beginnings of this are in the program EdWin in the Freeware folder of your disk.

Paste buffer, Line pointers, and Word wrap

The paste buffer, which is used to hold cut blocks and deleted lines and characters, is not very large. It is equal to 1/4 the size of the edit buffer (2560 bytes in the case of the normal 10K buffer). This buffer is also used for word wrap operations, so anything in the paste buffer is lost as soon as any word wrap is done. This buffer is the same size as the block of memory that is used to store pointers to the start of each line. Since each pointer is four bytes long, the default 10K edit buffer can handle up to 2560/4 = 640 lines. If you load a file with LOTS of blank or short lines, you could exceed that, in which case the file will load, but the last line will just be one long line containing all the rest of the file, but the text will not have been corrupted at all. I hope to rewrite all this code for version 3.0 to get rid of the need for the line pointer buffer.

Note on Extended Lines
(longer than one window width)

Be sure to turn off Wrap before trying to edit a line that extends beyond the first window, unless you want it to wrap as soon as you hit a key (or as soon as the line end hits the wrap column if you've set it to wrap at column 90 or something). Simply moving the cursor with the arrow keys beyond the furthest left or right

columns of the window will move you to the next or previous section of the line. Shift-left or right arrow takes you to the start/end of the line, no matter how far over it is. Another command, Shift-Control-arrow, takes you just to the left or right edge of the current window. Clicking with the mouse by the edge of the window will also take you to the next/previous section of the line.

Other Software on Your Disk

A number of demo, freeware, and shareware programs are included on your disk. These are for your own use and to give to friends, upload to BBS's, add to user group disk libraries, etc. If you do distribute any of these, please be sure to include the documentation file that goes with each of them so people know how to use the software.

EDHAKDEM.ACC is a demo version of EdHak which is fully functional as ACC or PRG except that it can't save files to disk and it is limited to a 4K edit buffer size.

EDWIN.ACC is a tiny fully assembly language text editor that can only handle one window of text -- i.e., its buffer is only 2K and it has no scrolling capability. However, it takes up almost no memory (10K) and is fine for writing little notes, printing envelopes, etc. It can also run as a PRG.

DIARY20S.ACC is a stripped down version of an earlier EdHak that takes up much less memory than the full EdHak. It lacks features such as word-wrap, block functions, search/replace, and all hack functions.

SRCHREP is a program to do interesting wildcard search/replace functions, including replacing every Nth occurrence of a match. Its main limitation is that it can't handle files larger than 64K.

SORTER is a program that can sort lines based on any desired column range. It is definitely not the fastest of sorters, but if you need to sort a list and don't have something better then give it a try. It is also limited to 64K.

How to reach Douglas Communications:

Douglas Communications,
PO Box 119,
Stockport,
Cheshire SK2 6HW.

24 Hour Fax/Answerphone Service: 061-456 9587

Limited Warranty & Disclaimer

Douglas Communications warrants to the original purchaser of this software product that the magnetic recording media of the disk will be free of defects in materials and workmanship for ninety days from the purchase date. If the original disk should be discovered to be defective in that time, Douglas Communications agrees to replace it for free. Send the original disk to Douglas Communications along with proof of the date of purchase. This warranty does not apply to disks that have been abused or misused. (Neglect of your disk, however, is OK, as long as it is properly stored.)

This software and documentation have no other warranty expressed or implied. You use it at your own risk. It has been tested with many other software products and various hardware configurations, but since it can not possibly be checked with every hardware/software set up in existence, Douglas Communications can only offer to try to fix any incompatibilities or bugs that you find and report to the author. In no event shall Douglas Communications be liable for more than the purchase price of EdHak.

Thank You - for buying EdHak!

Also, I want to thank Mike Olin of Softhouse Computers in Garden City, Michigan, for many helpful suggestions, creation of the .NIC desktop icon, use of various ST's for testing, and general support throughout months of creation and revision.

All my good friends deserve many thanks for putting up with me always running away from them to try "one" more change!

Thanks also go to the beta testers for keeping things from getting too badly screwed up as new features were added.

I must also thank all the users of various shareware versions of Diary and EdHak who registered and thus made me believe that I had actually created something worthwhile. This even includes those of you who dared to find and report bugs!

Finally, thank you Denise Crawley.

Sincerely, Craig Harvey.

Atari ST / TT EdHak Character Chart

Char	Dec	Hex	Key(s)	Name
	0	00	[Alt-I]	Null
↑	1	01	Ctl-A	
↓	2	02	Ctl-B	
→	3	03	Ctl-C	
←	4	04	Ctl-D	
■	5	05	Ctl-E	
■	6	06	Ctl-F	
■	7	07	Ctl-G	Bell
✓	8	08	Ctl-H	Backspace
○	9	09	Ctl-I	Tab
♀	10	0A	Ctl-J	Line Feed
♪	11	0B	Ctl-K	Vert Tab
F F	12	0C	Ctl-L	Form Feed
C R	13	0D	Ctl-M	Return
♫	14	0E	Ctl-N	
☼	15	0F	Ctl-O	
▶	16	10	Ctl-P	
◀	17	11	Ctl-Q	
↕	18	12	Ctl-R	
‼	19	13	Ctl-S	
¶	20	14	Ctl-T	
§	21	15	Ctl-U	
▬	22	16	Ctl-V	
↨	23	17	Ctl-W	
↑	24	18	Ctl-X	
↓	25	19	Ctl-Y	
→	26	1A	Ctl-Z	
E S	27	1B	Alt-Esc	Escape
∟	28	1C	Ctl-\	
↔	29	1D	Ctl-=	
▲	30	1E	Ctl-6	
▼	31	1F	[Alt-I]	

Char	Dec	Hex	Key(s)
	32	20	Space
!	33	21	!
"	34	22	"
#	35	23	#
$	36	24	$
%	37	25	%
&	38	26	&
'	39	27	'
(	40	28	(
)	41	29	)
*	42	2A	*
+	43	2B	+
,	44	2C	,
-	45	2D	-
.	46	2E	,
/	47	2F	/
0	48	30	0
1	49	31	1
2	50	32	2
3	51	33	3
4	52	34	4
5	53	35	5
6	54	36	6
7	55	37	7
8	56	38	8
9	57	39	9
:	58	3A	:
;	59	3B	;
<	60	3C	<
=	61	3D	=
>	62	3E	>
?	63	3F	?

Char	Dec	Hex	Key(s)
@	64	40	@
A	65	41	A
B	66	42	B
C	67	43	C
D	68	44	D
E	69	45	E
F	70	46	F
G	71	47	G
H	72	48	H
I	73	49	I
J	74	4A	J
K	75	4B	K
L	76	4C	L
M	77	4D	M
N	78	4E	N
O	79	4F	O
P	80	50	P
Q	81	51	Q
R	82	52	R
S	83	53	S
T	84	54	T
U	85	55	U
V	86	56	V
W	87	57	W
X	88	58	X
Y	89	59	Y
Z	90	5A	Z
[	91	5B	[
\	92	5C	\
]	93	5D	]
^	94	5E	^
_	95	5F	_

Char	Dec	Hex	Key(s)
`	96	60	`
a	97	61	a
b	98	62	b
c	99	63	c
d	100	64	d
e	101	65	e
f	102	66	f
g	103	67	g
h	104	68	h
i	105	69	i
j	106	6A	j
k	107	6B	k
l	108	6C	l
m	109	6D	m
n	110	6E	n
o	111	6F	o
p	112	70	p
q	113	71	q
r	114	72	r
s	115	73	s
t	116	74	t
u	117	75	u
v	118	76	v
w	119	77	w
x	120	78	x
y	121	79	y
z	122	7A	z
{	123	7B	{
|	124	7C	|
}	125	7D	}
~	126	7E	~
Δ	127	7F	[Alt-I]

Char	Dec	Hex	Key(s)
Ç	128	80	[Alt-I]
ü	129	81	Insert-
é	130	82	Bytes
â	131	83	for all
ä	132	84	chars.
à	133	85	
å	134	86	
ç	135	87	
ê	136	88	
ë	137	89	
è	138	8A	
ï	139	8B	
î	140	8C	
ì	141	8D	
Ä	142	8E	
Å	143	8F	
É	144	90	
æ	145	91	
Æ	146	92	
ô	147	93	
ö	148	94	
ò	149	95	
û	150	96	
ù	151	97	
ÿ	152	98	
Ö	153	99	
Ü	154	9A	
¢	155	9B	
£	156	9C	
¥	157	9D	
ß	158	9E	
ƒ	159	9F	

Char	Dec	Hex	Key(s)
á	160	A0	[Alt-I]
í	161	A1	Insert-
ó	162	A2	Bytes
ú	163	A3	for all
ñ	164	A4	chars.
Ñ	165	A5	
ª	166	A6	
º	167	A7	
¿	168	A8	
⌐	169	A9	
¬	170	AA	
½	171	AB	
¼	172	AC	
¡	173	AD	
«	174	AE	
»	175	AF	
ã	176	B0	
Õ	177	B1	
Ø	178	B2	
ø	179	B3	
œ	180	B4	
Œ	181	B5	
À	182	B6	
Ã	183	B7	
Õ	184	B8	
¨	185	B9	
´	186	BA	
†	187	BB	
¶	188	BC	
©	189	BD	
®	190	BE	
™	191	BF	

Char	Dec	Hex	Key(s)	Char	Dec	Hex	Key(s)
ij	192	C0	[Alt-I]	α	224	E0	[Alt-I]
IJ	193	C1	Insert-	ß	225	E1	Insert-
א	194	C2	Bytes	Γ	226	E2	Bytes
ב	195	C3	for all	π	227	E3	for all
ג	196	C4	chars.	Σ	228	E4	chars.
ד	197	C5		σ	229	E5	
ה	198	C6		µ	230	E6	
ו	199	C7		τ	231	E7	
ז	200	C8		Φ	232	E8	
ח	201	C9		Θ	233	E9	
ט	202	CA		Ω	234	EA	
י	203	CB		δ	235	EB	
כ	204	CC		∞	236	EC	
ל	205	CD		φ	237	ED	
מ	206	CE		ε	238	EE	
נ	207	CF		∩	239	EF	
ס	208	D0		≡	240	F0	
ע	209	D1		±	241	F1	
פ	210	D2		≥	242	F2	
צ	211	D3		≤	243	F3	
ק	212	D4		⌠	244	F4	
ר	213	D5		⌡	245	F5	
ש	214	D6		÷	246	F6	
ת	215	D7		≈	247	F7	
ן	216	D8		°	248	F8	
ך	217	D9		∙	249	F9	
ם	218	DA		·	250	FA	
ף	219	DB		√	251	FB	
ץ	220	DC		n	252	FC	
§	221	DD		²	253	FD	
^	222	DE		³	254	FE	
∞	223	DF		¯	255	FF	

EdHak
Keyboard Command Summary

Please see the full command descriptions for more information.

Cursor Positioning

Arrow keys	Left, Right, Up, Down
Shift-Up/Down arrow	Page up/down
Shift-Left/Right arrow	Go to start/end of line
Cntrl-Left/Right arrow	Go to previous/next word
Shft-Ctl-Left/Rt arrow	Go to left/right edge of window (useful for long lines)
ClrHome	Toggle top/bottom of screen
Shift-ClrHome	Toggle top/bottom of buffer
Alt-G	Go to line or byte offset
Alt-1 / Alt-2	Go to start/end of block

Editing

Shift-Delete	Delete line
Alt-Delete	Delete to end of line
F1/F2	Set Start/End of Block (or use mouse)
F3	Clear block marks
F4/Backspace/Delete	Cut marked block
F5 or UNDO	Paste block (or Deleted lines)
Alt-B	Block... (cut/paste/save)
Alt-W or F10	Word wrap (turn on/off or reformat Line, Paragraph or All)
Alt-F	Find/Replace (all chars 0-255)
F6	Find next match
Alt-C	Clear Text Buffer
Alt-D	Insert system date at cursor
Alt-I	Insert byte string (all 0-255)
Alt-Esc	Insert Es character (#27, $1B)
Alt-FuncKey	Insert macro (F1-F10)
Alt-V	View/Edit RAM

File Handling & General

Alt-O or Alt-R	Open file... (or disk sectors)
Alt-S	Save file... (or disk sectors)
Shift-Alt-S	Save file to current filename
Alt-A	Append to file...
Shift-Alt-A	Append buffer to current filename
Alt-P	Print
Alt-U	Upload to modem or MIDI port

(Save, Append, Print & Upload allow doing only a marked block)

Alt-H or HELP	Help (summary of all this info)
Alt-L	Launch PRG from within EdHak
Alt-E	Encrypt/Decrypt buffer with password
Alt-Q or Esc	Quit with query (same as menu Quit)
Alt-X	Quit without query (= Close button)
Shift-Alt-X	Alternate exit (for QuickCIStoggle)
Alt-M	Merge file (insert at cursor)
Alt-K	"Kwiksend" block to other window
Shift-Alt-K	Send buffer to EDHAK.ACC, if installed

Configuration

INSERT	Toggle overwrite/insert mode
Alt- <	Set left print margin (inserts spaces)
Alt- >	Set right word wrap column (min 20)
Alt-Tab	Set spaces between all tabs
Cntrl-Tab	Convert Tabs <=> Spaces
Cntrl-Return	Toggle display of 'cr' at end of lines
Alt-T	Toggle text/hack & Convert hex/char
Cntrl-FuncKey	Save marked block to macro F1-10
Alt-N	New config? (Show/Set/Save settings)
Shift-Alt-C	Set/change Clipboard path

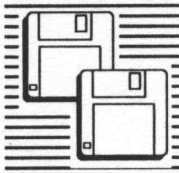
Douglas Communications,
PO Box 119,
Stockport,
Cheshire SK2 6HW.

24 Hour Fax/Answerphone Service: 061-456 9587

Index

STulker 30

Ed Hak



The ST & TT Desk Accessory and Program that is there when you need to Edit Anything!

- **TEXT**
- **DATA**
- **BINARY FILES**
- **DISK SECTORS**
- **RAM**

- ❑ **Colour/Mono, All Resolutions**
- ❑ **File size is not limited by RAM**
- ❑ **Easy control by keyboard or mouse**
- ❑ **Great for many word processing needs**
  (Word Wrap, Block cut/paste, Search/replace, Macros)
- ❑ **Great for use as a Diary**
  (Autoload, Append, Date & Encrypt commands)
- ❑ **Great for use within other applications**
  (Desktop Publishing, MIDI, Comms, etc.)
- ❑ **Great for Hacking**
  (Enter/search/replace all 256 chars in Hex or Decimal)

Douglas Communications
PO Box 119, Stockport, Cheshire SK2 6HW

If this software is defective in any way, please return it to the above address for immediate replacement.

Unauthorised copying, hiring, lending, exchange, public performance and broadcasting of the software is strictly prohibited.

Manufactured in the UK