PAT

Ronald W Anderson

CONTENTS

INTRODUCTION

What's in a name? For those of you who might wonder where PAT came from (the name, that is), after much deliberation and a lot of cute acronyms such as PSEUDO for Programmer's Screen Editor Under Development Occasionally, I kept coming back to two requirements. First, the name had to be short, and secondly I wanted it to reflect the fact that this is an editor for both Programs and Text. PATE for Program And Text Editor came to mind, but I thought that might bring visions of goose livers, so it was shortened to PAT. Much later in hindsight, I might have called it Text And Program Editor or TAPE, but that might have other connotations too!

I have used some number greater than ten editors on a number of different computers, and PIE, written by Tom Crosley in versions for the Apple II, and the 6800/6809 FLEX operating system, had struck me as being better as a general all round tool than any of the others. There are TEXT editors and there are PROGRAM editors and one would think them to be so different and special that no one could write one that would serve both purposes. That is a lot of nonsense. There is no reason that one editor can't serve both purposes. One of my primary design goals has been to keep it simple and avoid frills that only a few users would need now and then. I have concentrated on making the most often used functions as simple to use as possible.

The main goal in the design of PAT was not just to copy an editor that I like very well, but to try to incorporate some improvements as I went along. Let me outline what I believe to be the best features of PAT. Please note that throughout this manual the notation ^ will be used to mean "press and hold the CONTROL key". That is, when you see ^T that means to press and hold the CONTROL or CTRL key and then press T.

1. Clean Screen Layout - Pat puts nothing on the terminal screen but one status line at the very bottom. If the terminal supports reverse video the line may be displayed in that mode. On some terminals, half intensity works better, and you may like it for being less obtrusive. In at least one case, reverse video half intensity was available and seemed the best. The choice is yours by means of the TERMCON file. If no different video attribute is available, the line may be displayed normally. The status line has line and column indicators and mode indicators as well as a display of remaining memory in the edit buffer. This same status line is used whenever an edit command requires user input in the course of editing a file.

2. Minimum Keystrokes - PAT is a SCREEN editor, not a line editor. The main tool of a screen editor is the cursor, and the cursor is used to advantage by all good screen editors to mark start and end of blocks of text that are going to be copied, moved or deleted, and for many other functions. In short, if it can be done with the cursor, you won't be asked to type in information.

3. Speed - PAT will allow you to type successive commands that move the screen around in the text, and it will abort rewriting the screen to do another command. For example, in many screen editors, if you bump the top of the text window with CURSOR UP motions, the screen will move up one line in the text (or the text will move down in the screen). Some editors will rewrite the screen completely three times if you bump the top of the screen three times in a row. PAT will interrupt the screen refresh essentially as soon as it starts in order to handle another command. When the commands stop coming, the screen will be rewritten.

4. Ease of maintenance - This is a feature for the author, but it means that errors or bugs that are found by users can be fixed quickly. PAT in this present version is written in the Whimsical language. It has been "translated" from a PLuS version running under os9 on a 68020 based computer. The translation and testing were done on a 68008 based computer running os9 and SK*DOS. The representation of the procedures in a high level language makes the program listing much more understandable to others, as well as to the author after the passage of some time. As a matter of interest, The code is approximately 2700 lines. The source listing is 50 pages, and the object code is just under 20K.

5. Big Buffer - The main edit buffer size is 100,000 bytes, and I will, on request, supply versions with larger edit buffer. With 100,000 you can edit some 30 pages of single spaced typewritten material within the buffer. The source text file of PAT is about 65K and there are no library files used. Files larger than the buffer can be edited by writing a portion of the buffer to the output file and reading more from the input file until the whole file has been processed.

6. Good File Handling - The input file is never renamed until the output file has been written. At that point the input file is renamed to its original name plus the extension .BAK. If there is already a .BAK file it is deleted. The new file is renamed from .SCR to the original input file name. Power failures never result in confusion since the input file is not renamed at the start of the edit session. In addition to the normal file functions, PAT allows reading a "side file" into the file being edited (at the cursor position of course). It also allows writing a previously marked block of data to a file.

7. Easy Setup - Want to set tab stops every three columns? Just type ESC 3 ^T. Tab positions will be set and displayed on the screen. When you are sure that is what you want, hit a key and you are back to normal editing. If you want to set the tabs to specific and unevenly spaced columns, you type ESC ^T and bump the cursor along the top line with the space bar until you find the column you want to be a tab stop, then type any other key. Continue across the screen until all tabs are set and hit RETURN. The tabs will be

displayed. Hit any key and see the remainder of the tabs (right scroll). Hit a key again and return to editing.

8. Reasonable operating modes - In the normal OVERLAY mode, the terminal screen is analogous to a piece of paper in a typewriter. You don't have to enter CR'S to get down three lines below the current end of the text, just bump the cursor down there and type. If you want to type something in the middle of a blank screen just go ahead. PAT takes care of blank lines by inserting CR, and inserts spaces out to the start of the text.

    a. In INSERT mode characters are inserted at the cursor and the text to the right is pushed along until it hits the line length limit. Both of these modes are operative along with either of the two sub-modes that follow. Insert mode is indicated by a blinking cursor while overlay mode is indicated by a solid cursor when the user's terminal has such features available. INSERT mode is also indicated by the INS status on the status line.

    b. PMODE - (Paragraph MODE) - is used primarily for text. In this mode you never have to think about a CR. Text fills a line and when a word overflows that line it is moved to the start of the next line, even if you are inserting text in the middle of a paragraph, provided you have "split" the line at that point. Also, when in this mode, tabbing ignores the tabs that have been set ONLY WHILE WITHIN TEXT, and tabbing jumps the cursor to the beginning of the next or preceding word. Outside of text, the cursor tabs to the set tab columns. (This mode is close to the one referred to as "Document Mode" in Wordstar terminology).

    c. AMODE - (Auto indent MODE) - In this mode, when the cursor moves to a new line, it aligns itself with the first printable character on the line above it. This mode is extremely useful for programming and typing outlines. There are tab and backtab keys that also aid in this mode. The tabs, when in this mode, adhere to the preset tab columns.

9. Normal Text File - Some editors generate a "peculiar" text file containing "Soft Carriage Returns", "Hard Spaces" etc. Some use no carriage returns within a paragraph so that each paragraph is one very long line. Though such editors generally work fine within their own "system", the files that they produce are hard to LIST to the terminal or edit with other editors. PAT produces "standard" text files. In SK*DOS that means that each line as you see it on the screen while you edit, will be saved with the standard (CR) line terminator. The SK*DOS LIST utility and others assume this to be the form of the input file, and therefore all the SK*DOS utilities work with PAT files with no special handling.

10. Paragraph Formatting - Features are available to format paragraphs to the currently set margins. One of the configuration parameters that you may include in your terminal configuration file is the paragraph indent desired in formatting paragraphs. A paragraph may be formatted with or without justification. If justification is on, spaces will be added to each line to make the right margin even. Regardless of the status of Justify, words will be rearranged within the paragraph to the maximum number that will fit on each line, and the first line will be indented. Generally after formatting a paragraph the cursor will be found on the first character of the next paragraph so the process can be repeated. The format operation is done a paragraph at a time so you can view the text and skip formatting of tables, etc. A new feature of PAT allows you to change the default paragraph indent to any value from zero to the line length. A paragraph that has been formatted with justification may be reformatted without it, and all the added spaces will be removed. Formatting a paragraph will automatically put two spaces after a period that is followed by one or more spaces. Only one of those spaces will be "paddable". That is, the justification routine won't add a space to both of the spaces after a period making the spacing very much wider than the other spaces.

A lately added feature of PAT allows you to move a title to the center of the text width, or to move it to the right or left edge of that area.

11. Long Lines - PAT allows lines as long as 127 characters. The screen will automatically shift to display the right end of lines as you type or otherwise move the cursor past column 80. When the right screen is being displayed, anything you do to move the cursor back to the left end of the line will cause the screen to return to its original position. Note that you can cause the screen to shift to the right simply by moving the cursor past column 80, regardless of line length.

There is one more feature that should be mentioned in the introduction. Did you ever assemble or compile a program and get a message such as ERROR #202 in line #237? Some editors do not associate a number with each line. Though PAT doesn't display numbers, on every line, the line indicator on the status line does display the current cursor line number. Each line is associated with a file line number. In the case of the above error message, you load the file into PAT and type ESC 237^G and before you can blink that line is about 1/3 of the way from the top of the screen and the cursor is at the beginning of it.

It should also be made clear what PAT is NOT. PAT is not a full text formatter. There is no provision for paging or header and footer formatting of pages. It is assumed that the user will embed word processing commands in the text and run the output through one of the several text processors that are available. The formatting that is available, of course is only useful if the printer to be used is of the non-proportional spacing type. For use with a text formatter set up for proportional spacing, use only the Format mode without justify. Commands to be interpreted by the text formatter as start of paragraph may be used in the text. For purposes of formatting a paragraph, the paragraph ends if the first character of the next line is a CR (i.e. a blank line), if it is one of the symbols usually used to indicate a formatter command (comma, period, or colon), or if it is a space (leading space on a new line). This last is included so the user may format paragraphs without blank lines between them, starting a new paragraph with one or more spaces.

GETTING STARTED WITH PAT

OK, so here you sit in front of the terminal with the PAT manual alongside and the disk in your hand. What now? First of all, write-protect the original disk. There are a number of files on the supplied disk. The main one is PAT.COM. The disk also includes the source file PAT.WHM and the manual chapter files. Copy PAT.COM to your working drive or partition and you are all set except for configuring for your terminal.

NOTE: If you have bought the special version of PAT for the PT68K-2 computer with the monochrome or CGA adaptor, IBM clone keyboard and monitor, you don't need to worry about terminal configuration files. That version is pre-configured. You can skip the next few paragraphs and Appendix A. The supplied version of PAT will read a hardware configuration byte in SK*DOS and properly set itself up to run on either the CGA or the Hercules hardware.

The remainder of the files on the disk are for use in configuring PAT for your terminal. Unfortunately there are as many sets of cursor control codes and screen clear codes as there are terminal manufacturers. If your terminal is among those for which there is a configure file, (see Appendix A) your job is a simple one. If for example you have a Televideo 910, 925, or 950, just copy the TERMCON.TVI file into a file called TERMCON.TXT in the root directory of your drive 0 (which should be your system drive or partition and you are done. If your terminal is not listed in Appendix A read the information there "Configuring for a Different Terminal".

When you think you have the proper configuration file, simply type PAT JUNK. PAT will load, and a file named JUNK will be opened in your working directory. Note that you can edit a file with any extension except .SCR that you specify. If you don't specify any, the default is .TXT. You can always specify the logical drive number. Default is the working drive. If the screen is jumbled you have made a mistake in the TERMCON file. Regardless of what you see, you should be able to exit back to SK*DOS by typing ESC Q ^^.

Now with the file open, you will see the status line at the bottom of the screen and the cursor at the top. The status line will indicate MEM NNNNN where NNNNN is a number that indicates how much memory is available. MEM is the amount of memory left outside of the text on the screen. If you are editing a new file, it will be one less than the size of the edit buffer. It will not change until the screen updates as it moves text off of the screen into the edit buffer. Now how about editing a short text?

First turn off PMODE (explained below) by typing ESC^P. Now type a short line of text and a CR. Note that the column indicator increments as you type, and when you enter a CR at the end of the first line, you will see LINE 2 COL 1 displayed. So far so good, but what about moving the cursor around? The cursor motions are controlled by normal "letter keys" used along with the CTRL key. The cursor moves are arranged in a Diamond pattern on the keyboard. To move the cursor up to the start of your first line, just type control E (hold the control key down and type E). Note that you can move the cursor all around the screen without changing the text that is there. If you are a touch typist you will soon get used to holding the control key with your little finger and operating other keys with your remaining fingers without looking at the keyboard.

A few words about PAT's operating modes is in order early, to avoid confusion. PAT has two MAIN modes of operation, OVERLAY and INSERT. These modes are mutually exclusive. That is you can have one or the other at any one time. In OVERLAY mode the text on the screen is "fixed" and you can put the cursor anywhere and overtype (or overlay) what is already there, with new text. In INSERT mode, the text is movable. You place the cursor somewhere and text is inserted at the cursor pushing text ahead of the cursor to the right. These modes are "toggled" by typing ^P.

There are two "sub-modes", both of which may be off, but only one of which may be on at any one time. These sub modes are independent of the status of OVERLAY/INSERT. You can use either of them with OVERLAY or INSERT. These modes are PMODE, (short for Paragraph Mode) in which you simply type text with no regard to its position on the line. When you get to the end of a line (the current line length, which defaults to 65 on startup), the word that is being typed, automatically moves to the start of the next line. This allows you to type text very quickly since you can look at your copy and do not have to refer to the screen to see if a CR is necessary. When this mode is on, PMODE appears on the status line. AMODE (for Auto indent mode) is useful for outlines or programs that have several indent levels. When you type a CR in AMODE, the cursor aligns itself with the first non-space character on the line above. You can use this in conjunction with the TAB and BACKTAB features to great advantage. When AMODE is on, the word AMODE appears on the status line. When both of these modes are off, the status line has no sub-mode indicator showing. When they are both off, the line length has no significance. You may type or move the cursor anywhere on the screen. AMODE is toggled on and off by ESC^A, and PMODE is toggled by ESC^P. Turning PMODE on will automatically cancel AMODE and vice versa.

Now try inserting some text in the middle of your now thoroughly scrambled line. First position the cursor somewhere in the line and then type ^P. Note that the word INS appears on the status line to let you know that you are in INSERT mode. The cursor also blinks or changes shape when you are in insert mode, as an additional reminder (except on a very few terminals that do not have codes to turn the cursor off and on). Provision is made for some terminals that allow you to change the appearance (shape) of the cursor so that you can distinguish modes by the shape if you wish. Now type a few characters and note that they are inserted in your line, and that they push the text to the right ahead of them. If you type too many, the text to the right of the insert will disappear off the end of the screen, and perhaps the screen will shift to the right (text shifts to the left). Eventually you will bump the end of the line and the error message "NOT ENOUGH ROOM" will appear on the status line. If this happens you can split the line (see below). After any error message you must type ESC to resume editing. Also note that being in INSERT mode in no way inhibits your freedom to move the cursor wherever you want.

If you are inserting at the end of the text on a line, in PMODE, a new line will be inserted automatically when you reach the end of the line. The best way to insert text is to split the line at the point where you want to insert, and then rejoin or reformat the paragraph when you are done. While you are in insert mode, type a few backspaces (^H). Note that the character to the left of the cursor disappears and the cursor moves back one space pulling the text at the right along with it. This is the inverse of inserting text. Type ^P to leave the INSERT mode (the ^P toggles INS on and off alternately). Now type a few ^H strokes and note that the cursor moves back and blanks are inserted (the text to the right is not moved to the left to take up the space vacated by the deletion of a character). Of course that is perfectly consistent with being in the overlay mode again.

There is another way to delete characters. To delete the current character (at the cursor) and pull all the characters to the right back, just type ^J. This command works identically in either OVERLAY or INSERT mode.

Refer to figure 1 and try the various cursor motions. The tabs default to every fifth column so a tab right or left will move the cursor 5 positions. Note that cursor left is not the same as backspace. Moving the cursor generally does nothing to the text on the screen.

Now type some text on the screen. When you have at least one screen full of text you can experiment further with the cursor control keys. Note particularly the ^D key functions to move the cursor to top left of the screen if it is not already there. If it is there, it will move to bottom left of screen (last line). This is useful for moving to lines near the top or bottom of the screen.

The cursor express left and right ^] is useful also. If the cursor is anywhere within the text on the line it will go to the end on a ^]. If it is at or past the end, it will go to the start.

So much for cursor motions. Now type ESC ^P and the PMODE flag will appear on the status line. Position the cursor below any text already on the screen and start typing. Note the L=65 on the status line. That means the line length is set to 65 characters. Remember that when a word runs over that column it is automatically moved to the start of the next line. You don't have to remember to hit a CR every time the "bell" rings. (The bell may be toggled on and off by typing ESC ^B). Just type along, and when you get to the last line of the screen, be prepared for a surprise. The text will move up 2/3 of a screen as the last word on that line is moved to the next line. The text above the screen is not lost, but has been put in the larger edit buffer. At this point also note that the MEM indicator shows less than it did initially. You have put some of your text in the edit buffer, off of the screen.

Now, leaving PMODE on, put the cursor on a line of text and use the left and right tab keys ^A and ^G respectively. Note that tabs no longer go to the specified tab columns but to the start of the next or previous word. This feature is called WORD TAB and it is enabled in the PMODE only. Should you tab off the right end of a short line, tabs revert to the tab columns. Now left tabs will observe tab columns until you get back to the end of the text on the line and then revert to word tabbing again to get you to the start of the last word on the line.

In the discussion that follows, it will help to picture a long scroll containing the entire text that you are editing. The CRT screen should be thought of as a "window" that can look at a portion of that text. You can move the cursor up and down within the window, and you can move the window up and down in the text. You will see that the control keys in the TOP row (E,R,T,Y) move either the cursor or the screen UP in the text, while the keys in the bottom row (C,V,B,N) move the cursor or the screen DOWN in the text.

How do you get back to the start of the text you are editing? The screen function keys handle that for you. ^T will always get you to the Top of the file. ^B will find the Bottom. ^R will move the window up by one screenful, and ^V will go down by the same. Note that the cursor will stay in the same relative position on the screen when you go up and down by one screen, a feature without obvious function until you are editing a program and need to go up three screens to find a variable name and then return to the place where you were editing. ^Y and ^N move the screen up and down 1/3 screen respectively. If the line on which the cursor is positioned remains on the screen, the cursor will move with it and remain on the same text line. If this seems strange, it has utility while reading through a text file as for proofreading, marking the current line with the cursor as you go. You can bump the cursor to the bottom of the screen and then move the text and cursor up (screen down) with a couple ^N instructions and continue. Of course ^E will move the screen up when the cursor bumps the top line and ^C will move it down at the bottom. ^U will move the cursor line (and the cursor) to the top of the screen. ^^ (control carat) will move the cursor line and the cursor to the bottom of the screen.

There is one other screen motion that we have not mentioned. Use the command ESC 100^L to set the line length to 100, and turn PMODE on if it is not already on. Now type a long line of text without using a carriage return. Note that when you get to the end of the screen, the text shifts to the left (by 48 columns). Then it continues as you type more characters. When you get past column 100 PMODE does its usual trick and wraps to column zero of the next line. The screen updates and shifts back to the left side of the text. If you don't need lines longer than 78 or 79 characters, you will never see the window shift. If you need longer lines you will be able to have them. The window will shift regardless of PMODE being on. If you move the cursor or type past the right end of the screen, it will shift to the right end of the line. When you type a CR, use ^], or move the cursor to the left beyond column 48, the screen will move again to the left end of the line.

That about covers the basic things about PAT. You now know how to move the cursor around and how to move the screen around. If you want to quit at this point and save your file for further experimental editing, just type ESC ^^ (that's control up-arrow) and the file will be saved as JUNK. Should you ever want to quit editing and throw away the text you have just entered, just type ESC Q ^^ The 'Q' is for Quit, and any letter following an escape as part of the instruction may be either upper or lower case. Note particularly that it is not ESC ^Q, but just plain letter Q or q. Also note that the CTRL key and the up-arrow key are far apart on most terminals making it very difficult to quit editing a file accidentally!

FILES

There are several situations that can occur with regard to the text file that you are editing when using PAT. Let's review them one at a time.

First there is the situation where the file is to be generated for the first time. That is, there is no input file. Suppose the file is to be called "LETTER". Simply type "PAT LETTER". Pat will check to see if the file exits on the disk. If it does not, as is the present case, PAT will open the file "LETTER.TXT" as an output file. When you exit the edit session, the text will be written to the output file.

Now suppose you want to edit "LETTER" again. You simply type "PAT LETTER". Pat will find the existing file and open it for input. The file will be read into the edit buffer completely if it will fit, or partially if it will not. A file named LETTER.SCR (for SCRatch) will be opened. If the input file is too big for the edit buffer, you will be given a warning message. You will edit the file until you want to get the remainder into the edit buffer and then put the cursor below the point where you have edited to your satisfaction and type ESC N ^^. The portion of the file above the cursor line will be written to LETTER.SCR and more of the file (if there is more) will be read from LETTER.TXT. When you finish editing, the contents of the edit buffer will be added to LETTER.SCR. Any remaining portion of the file still not read from LETTER.TXT will be copied to LETTER.SCR, and both files will be closed. Now the input file "LETTER.TXT" will be renamed "LETTER.BAK" and "LETTER.SCR" will be renamed "LETTER.TXT".

Lastly, if you edit a file "LETTER.TXT" and there already exists a "LETTER.BAK" file, when you finish PAT will delete the old backup file, rename the input file to LETTER.BAK, and rename the LETTER.SCR file to the original name "LETTER.TXT".

If you want to edit a file into a second file with a completely different filename, you type "PAT LETTER LETTER2". "LETTER.TXT" will be loaded into the edit buffer and the output will be written to "LETTER2.TXT". The input file "LETTER.TXT" will remain unchanged. PAT checks for the existence of a file with the same name as the specified output file and reports an error if it finds one.

If you choose to abort an edit and use the Q option while an output file is open and not fully written, (i.e. editing a large file) You will end up with a partial file as the output. It is recommended that you complete the edit session by exiting normally if you have started to write an output file.

Should something happen during your session, (loss of power,etc.) so that the edit is forceably aborted, you will find the file named "FILENAME.SCR" on the disk with no information in it. Because no files are renamed until the output file is

successfully written, you will find your files intact as they existed before you started the edit session. You may delete the empty .SCR file but should you forget, PAT will do it automatically for you when you again edit the file.

ERROR HANDLING

If an error occurs while doing a file operation such as reading or writing a side file, loading the input file or writing or renaming files on exiting an edit session, the error message will appear on the status line. In order to clear the message, you must hit the ESCape key. The program will, in general, return to its status before you attempted the file operation. That means that, for example, if you try to read a side file and it does not exist, you will have a chance to name it correctly and read it in. If you attempt to write to a sidefile that already exists, you will be given a second chance.

There is one class of problem that deserves special mention. The output file FILENAME.SCR is named and opened when you start an edit session. This problem mainly concerns users of floppy disk only. If for some reason that file cannot be written, for example for a DISK SPACE FULL or a FILE WRITE ERROR, you may change disks and try again. You may not, however, simply tell the editor to exit and save the file. That is because the file is already open on the disk that went bad. If you have to change disks because of an error, mark the top and bottom of your text file and write the whole file to a file of a different name on the new disk. That will insure that a new file is opened, written and closed on the new disk. Then re-insert your bad disk and try to exit the edit session with an ESC Q^^. That should cause the open output file to be closed. This procedure should result in minimum damage to the disk that had the error. Once you have closed a file that caused a "disk full" error, you may delete it from that disk. If there was a file write error, you can copy as much of the disk as possible to a new disk, and reformat it.

INSERT, DELETE, and ESCAPE FUNCTIONS

Now you are going to edit JUNK again. Type PAT JUNK. Note that the first time JUNK didn't exist so a new file was created. This time, however, PAT will report "LOADING FILE" and then display the top screenful of text in the file.

Position some text on the screen and put the cursor on a line that has text on the adjacent lines (above and below). Now type ^I and notice that a blank line is inserted ABOVE the cursor line. It may be easier to think of the line as being inserted at the cursor, with the line that was previously at the cursor moved down to make room for it. Type ^I three more times in rapid succession and note that there are four blank lines. Now put the cursor on the first (topmost) blank line and type ^K a few times. Notice that lines are deleted or Killed. The line at the cursor is deleted and the lines below are moved up. Delete the remaining inserted blank lines.

Now position the cursor on a line that contains text and delete it via ^K. Without moving the cursor type ^O. Note that the line has been restored. Now move the cursor up or down a few lines and type ^O again. Note that the line has been duplicated on the line above the cursor. ^K not only deletes a line but saves it to a "hold buffer". ^O inserts a line and puts the contents of the hold buffer in it. This combination is of much value while editing. Should you delete one too many lines, you can restore the last line deleted. If you want to move a line (as while debugging a program) just delete it and move the cursor to where you want it and type ^O. If you want to duplicate a line without having to delete it, use ^L rather than ^K, move to where you want the line copied and again use ^O. The ^L copies a line to the hold buffer without deleting it.

Now without moving the cursor, type ESC 4^I. (ESC means the escape key on the terminal). Note that when you operate the ESC key the cursor moves to the status line after the word ARG:. Any command that requires a modifier of some sort, or some information beyond the function code, is started with ESC, which moves the cursor to the status line. Notice that the action resulting from entering the above code was to insert 4 lines. Multi-line insert is handy when working on a program and inserting a number of lines of code. You can open up 20 lines, type in the code and close the extras by deleting. ESC 4 ^K will delete four lines just as ESC 4^I will insert four lines. ESC 3000^K will delete 3000 lines. If the file is shorter than the number of lines specified, all lines to the bottom of the edit buffer will be deleted. (Lines that were not read in from the input file will, of course NOT be deleted).

Just as it is possible to delete whole lines, it is also possible to delete from the cursor to the end of a line. ESC ^E will Erase the characters to the end of the line.

There is also a "word delete function. If you type ^_ (control underline) the word in which the cursor is located, or the word immediately following the cursor (if the cursor is on a space immediately preceding the word) will be deleted and the text closed up.

There are many other ESCape functions. ESC 50^L will change the line length from the default 65 to 50. It will not change the text that is already present, but any new text entered with that line length and in PMODE, will be adjusted to that new length. Any text that is formatted now will be formatted to the new line length also.

ESC 99^G will Goto line 99 of the file. Though not useful for editing straight text, this feature is used a great deal in fixing Assembler code or source code for compilers. Most assemblers and compilers report ERROR XYZ on LINE 1327. If you want to find that line, what better way could there be than ESC 1237^G. Note, though, that if you edit the file, line numbers change. Suppose you have multiple errors in an assembler source file. Editing the first one you find that you have to add two lines of code to fix the rror. Now all lines below this fix will have their line numbers increased by 2. Please note that you can go to a line by number from anywhere in the file. You do not have to be at the top of the file to go to a line.

ESC ^S will split the current line at the cursor. The character that is at the cursor and all to the right will go to the next line. The cursor will be in position to start typing additional text at the split point. This is particularly useful when inserting text in the PMODE since PMODE will then wrap a word to the next line when a line is full, as mentioned previously.

ESC ^J will Join two lines. The cursor must be on the first, and it doesn't matter in which column the cursor is located. Text will be joined to the end of the line from the next line. If the two lines won't fit on one, you will receive an error message which you must acknowledge by hitting ESC in order to proceed. In this case you might want to reformat the paragraph. If the second of the two lines contains leading spaces, they will be delete before joining the lines. This is a special feature of PAT that I have not seen in other editors. Generally when editing a program with indented lines, when you join two lines you have to delete the extra spaces from the beginning of the second line either before or after joining the lines. I find that I always have to delete spaces, so I made PAT do it automatically.

We have mentioned turning on the PMODE with ESC ^P previously. AMODE is turned on (cancelling PMODE) by ESC ^A. AMODE is useful in preparing outlines or programs that are indented in similar form, for example in Pascal or "C". In this mode, a CR will cause the cursor to align itself with the first non-space on the previous line. That is, it will always go to the current indent level. If the previous line is blank, the cursor will align itself with the first non-space on the line above that. When used in conjunction with tabs, this features makes it quite easy to type in program source code or outlines.

TAB set is another ESCAPE function. Type ESC 3^T and the tab header and indicator lines will appear across the screen indicating tabs by vertical bars "|". The argument for this function is the number of columns separating each tab position. Type any key, and you will return to edit mode.

If you want tabs set up in specific columns and not evenly spaced, type ESC ^T and the "tab header line" will appear. The cursor will go to the line below the tab header (which indicates column number for each column). Now type spaces for non-tab columns and anything else for tab columns. When you have created your own tab line hit a CR and the new tabs will be displayed in the original format. If you like them, hit a key again and you will have tabs at those columns. The tab header line can be used for a guide if you have specific columns in mind for tab settings.

Remember that the tab line is longer than the screen, and when you bump the right edge of the screen while setting them, the tab line will scroll horizontally so that the first column displayed will be column 49. When you are in the irregular tab setting mode, you redisplay the entire tab line. That is, the first 80 columns are displayed. Typing a key causes horizontal scrolling so you can see the right end of the line, and one more keystroke returns you to normal edit mode.

Tabs, as mentioned previously will tab to the set columns provided PMODE is not on. PMODE would not be used for programming, outlines or tables in text, where word wrap would be a nuisance. In PMODE, however, the tab keys will cause the cursor to move to the first character of the next or previous word for right (^G) or left (^A) tab respectively, as long as the cursor is within the text. As soon as the cursor leaves the text (past the end of the text in the line), it reverts to tabbing to the preset tab columns. When it is tabbed back (left) into the text, it reverts to word tabbing again.

Perhaps it is time to mention one special control function. Sometimes you want to display as much of a particular paragraph or program procedure as will fit on the screen. Simply put the cursor on the first line to be displayed and type ^U and the cursor line with the cursor will be moved to the top of the screen.

There are several other ESCAPE functions that are important enough to deserve chapters of their own in this manual. The following two chapters cover these functions.

SEARCH and REPLACE

Two vital functions in modern screen editors are the capabilities to search for a text string (a string is a combination of ascii characters which may or may not be a word, such as "this" or "XYLO" or "123"), and to replace that string with another. To experiment with this function, type in some text or load a text file that already exists into PAT and go to the top of the file. Now type "ESC the^Z". PAT will search for the word "the" and the cursor will move to the letter "t" of that word. If the word is present on the screen, the cursor will just move to it. If the word is below the present screen, the screen will move down in the file so the word is on the 8th line of the screen and the cursor will move to it. The word "the" is now in a buffer called the SEARCH buffer and it remains there.

Suppose you want to find the word "program" in a specific place in your text. With the cursor at the top of your file, you type ESC program^Z and PAT finds the first place where "program" appears. You determine that that is not the place you are looking for. Just type ^Z and PAT will find the next occurrence of the word. You can repeat this for as long as you like. Each time you repeat this action, PAT will write the word that is in the search buffer on the status line to refresh your memory. The word will only remain there while PAT is searching, so if the search is short, you will have to look quickly to see the word. The search string can include spaces (even leading or trailing ones) if that will help eliminated unwanted string matches. For example ESCthe ^Z (with a space after the e), will eliminate matches on "there, then, them" etc. It is important not to be too skimpy with the search string in order to avoid undesired matches. The letter t by itself will match every occurrence of the letter t in the text.

Suppose as I wrote this manual I had used the name XYZ for the editor and that I wanted to change it to PAT wherever it appeared in each chapter file. I would load the file and type ESC XYZ^Z. Now the cursor would go to the first occurrence of XYZ. Without moving the cursor I would now type ESC PAT^X. That would result in the word XYZ disappearing and the word PAT appearing in its place. The word PAT has now been put in a REPLACE buffer. Now if I want to look for more XYZs I can continue alternate ^Z and ^X, and if I were not sure I wanted to change EVERY occurrence of XYZ to PAT I would do just that. ^Z goes to the next occurrence and ^X causes it to be changed to whatever is in the REPLACE buffer. (The replace buffer can contain nothing, i.e. ESC^X will make the replace buffer "empty" so that the effect will be to delete the search string.)

Note also that as long as the cursor is on the first character of an occurrence of the search string in the text, ^X will cause the substitution of the contents of the Replace Buffer. It doesn't matter whether the cursor got to the string via the ^Z method or if you simply put the cursor there via cursor motion keys.

Backward Search

Backward search works exactly like forward search except that it searches for the previous occurrence of the search string rather than the next occurrence. Suppose you are typing along in the middle of a program and want to see where you last used the label INPUT1. Just type ESCINPUT1^Q. PAT will search back to the last occurrence of "INPUT1" before the position of the cursor when the search was invoked. Once a search string has been defined either using ESCstring^Q or ESCstring^Z, it may be used over and over again. You may use ^Z and ^Q in any order to search forward and backward for the search string.

It is usual practice in "C" programs to put the main() function at the top of the program and work your way down to lower and lower level functions. That is, all function calls are to functions below the present one. (This is not a RULE in C, but it is generally done that way). If that is the case, you can find a function very quickly by going to the bottom of the file and doing a backward search for the function name. Since the function is generally defined below all calls to it, you will find the function on the first try! The reverse holds for Pascal programs where a procedure must be defined before you call it. For those programs you start at the top and use forward search.

Global Changes

If, in editing a file, I am certain I want to replace ALL XYZs with PATs, I will start at the top of the file, type ESC XYZ^Z then after the cursor is at XYZ, I will type ESC PAT^X. After the first occurrence has changed and I am sure I didn't misspell PAT, I type ^W and ALL the occurrences are changed right down to the bottom of the edit buffer. When no further occurrences are found, the screen updates with the cursor after the last occurrence that was changed.

Note that SEARCH only searches for a string AFTER the present cursor position. (This can be used to great advantage to limit the search range at times). If you want to find ALL occurrences of the string, be sure you start at the very top of the file. Of course, if the input file is very large and won't fit in the edit buffer, a global search will only find the occurrences that are in the edit buffer. To reach the rest of the occurrences you will have to write the first part of the file out to the output file and read the remainder of the input file.

Note that a global change only works in the forward search mode.

Case Insensitive Search

A special feature has been added to PAT to allow you to search for a word with no regard to whether the word is capitalized or not, or for that matter whether it is all in upper case. The key sequence ESC ^D toggles this mode on and off. When in the "ignore case" mode, the indicator "UL" (for Upper/Lower) appears on the status line indicating that the case will be ignored. Global search and replace may be used with this mode on, but results could be confusing. You wouldn't, for example want to replace all occurrences whether capitalized or not, with another word that is not capitalized. PAT will not match the capitalization of the search string when it replaces the word with the replace string.

BLOCK HANDLING

Most editors have a way to do something to several lines of text at once. These rely on some way to "mark" a block of text. PAT works with full lines of text only, but it is easy to operate on too much or too little and fix the result by deleting part of a line.

In order to do something with a block you have to MARK it first. To do that, you put the cursor on the first line of the block to be marked and type ESC T<cr>. All block functions are identified by a mnemonic command letter and a CR. ESC T marks the Top of the block. Now put the cursor on the last line of the block and mark the Bottom by typing ESC B<cr>. You will note that when a block is marked the MEM status disappears and the status T=mm and B=nn appear in its place. Also, when both ends of a block ar marked, the block will usually appear different than the rest of the text. If your terminal supports reverse video or half intensity and you have used one of those attributes for your status line, the block will be displayed with the same attribute as the status line. On your status line, mm is the line number of the top line and nn the line number of the bottom line of the block. Note that it is permissible to mark the bottom line of the block before marking the top line. Convenience will dictate the order of marking. It is not permissible to mark a TOP line below a BOTTOM line, and doing so will cause unpredictable results.

Now that the block is marked, you can do one of four things with it. You can Copy it to another place in the file by putting the cursor at the place where you want the block copied (consistent with other functions, the block will be copied ABOVE the cursor line.) Now just type ESC C<cr>. If you want to MOVE the block (copy it to a new location and erase it from its original location) put the cursor at the point where you want it inserted and type ESC M<cr>. PAT will ask you if you want to delete nn lines. It means, do you want to delete the block in its original place. Answer Y or N (upper or lower case). The third function is Delete Block. If you want to delete a block, the cursor position doesn't matter, just type ESC K<cr>. In this case again, you will be asked if you want to delete nn lines.

The fourth function is sometimes a very useful one. Suppose you would like to write a block of text from a file out into another disk file. Just type ESC >FILENAME (cr). The right arrow points at the filename, indicating that you want to write the block to the file, essentially the same usage as in SK*DOS I/O redirection. The cursor position has no bearing on the operation. The block is Copied to the file. If you want the block deleted you must use the ESC K<cr> sequence after writing it to the file. The file write and Copy functions leave the block marked in case you want to copy it again. Of course Move and Delete destroy the block in its original location, so it does not remain marked.

NOTE: You must not copy or move a block into a place within itself. The destination must be outside of the marked block. PAT at this stage of its development will let you TRY to do this, and the result will be the deletion of some of the lines in your text file.

CAUTION: While it is acceptable to edit the text after marking a block, there are a few restrictions that must be observed to insure that the block remains correctly marked. It is permitted to edit the file, inserting or deleting lines anywhere, but it is not permitted to justify or reformat a paragraph that is only partially within a marked block. The operation will take place, but there is a great possibility that words will be moved into or out of the marked block. A paragraph that is wholly within a block my be reformatted to a different width, possibly changing the number of lines. The Format procedure will automatically adjust the last line of the block to take this into account. Note that when a block is copied to a location above itself, the line numbers are compensated so the original block still remains marked and you can copy it again, move it, or delete it. If you edit above the block the line number shown for the Top and Bottom of the block will be adjusted accordingly so that the block remains correctly marked. If you edit within a block any change in the number of lines in the block will cause the Bottom line number to be adjusted.

To unmark a block you have copied or to correct an error type ESC U<cr>.

One other function has been provided. It is possible to shift a marked block of text left or right by n spaces using the command ESC Sn<CR>. The value of 'n' determines how far the block is shifted. if n has no sign (i.e. is positive) the block is shifted right n spaces (by inserting leading spaces in the lines in the marked block). If n is negative (leading minus sign), the block is moved left by deleting leading spaces. Note that PAT won't left shift a line beyond the leading spaces. No printable characters can be lost. When right shifting, however, PAT doesn't check to see whether you have exceeded the maximum line length, and you must be careful not to do so. This is a useful function when restructuring the source code of a program causes a change in the desired indent level of a block of code.

A late addition to PAT allows you to move a portion of a marked block for such purposes as spreading or squeezing the columns in a table. The ESC Sn command now moves only the portion of the block from the cursor column to the right. Text left of the cursor column is not moved. The block remains marked so you can use this command repeatedly to move columns in a block. As mentioned above, no text can be deleted. Text will only be moved left to the cursor column. You can make use of this feature to align "ragged" text in one column, by moving it sufficient columns to put the text on each line in the cursor column. When you are done aligning columns, use ESC U CR to unmark the block.

It is also possible to read a block of text IN from a file. To do this you simply put the cursor where you want the file inserted (above the cursor again) and type ESC <FILENAME (cr). The "arrow" in this case points FROM the filename indicating that text is to be input FROM the file. FILENAME will be read and its contents inserted at the cursor.

To summarize, all block functions use ESC X CR. where X is the single letter function code:

```
     T mark (T)op of block
     B mark (B)ottom of block
     C (C)opy marked block to above cursor
     K (K)ill marked block
     M (M)ove marked block to above cursor
     U (U)nmark block
     Sn (S)hift block n spaces (-n is to left)
        does not shift text to the left of the cursor column
     >filename - write marked block to file
     <filename - read file to above cursor.
```

BIG FILES

We all have times when we want to keep a lot of text together in one big file. It is possible to edit a file longer than the text buffer with PAT, (though with the large buffer such an occurrence is not very likely). When such a file is loaded into the edit buffer, the warning "NOT COMPLETELY LOADED" will be shown. Hit ESC to acknowledge that you have seen the message and edit normally. PAT will load characters into the edit buffer until it reaches a preset "buffer full" condition leaving room for about 6000 more, but MEM will show more than that because some of the edit buffer is transferred to the screen as you work on it.

As you edit a file, the MEM indicator will show you how much memory is left in the main edit buffer. The text on the screen has to fit this buffer as well when you save the file. When the buffer approaches a full condition, a warning will appear where MEM normally is displayed. When you see this warning you should take action immediately to reduce the amount of text in the buffer. To write part of the file to the output file just position the cursor on the first line that you want to remain in the edit buffer and type ESC N^^. PAT will write the text above the cursor to the file and move the remainder to the top of the edit buffer. Note that file line numbers will be retained. If you write the first 100 lines of the file to the output file, the first line in the edit buffer will be 101.

If there is more input file to be read, the edit buffer will automatically fill to its limit again allowing you to do further editing before saving more of, or the whole file. To end the edit session you can simply type ESC ^^... If more input

file remains unread, PAT will first write the edit buffer to the output file and then copy the remainder of the input file to the output file. It is strongly suggested that you not abort (ESC^Q) an edit session if you have already written part of the output file. Simply exit normally and let the remainder of the file be written. If you must quit an edit with a partial output file, the file will be closed and left named with a .SCR extension. You can then delete the .SCR file after exiting the editor. If you forget, PAT will delete it for you when you again edit this file.

A late addition to PAT has been the ESC W^^ command. This one works just like the ESC N^^ command except that new text is not input from the input file regardless of whether or not more is available. This command might be useful to insert a very large amount of text in the middle of a long file.

FORMATTING PARAGRAPHS

There are two terms used here that will bear some explanation and definition. The term FILL will be used to describe the process of putting as many words on each line of a paragraph as will fit within the currently selected margin. When you type text in the PMODE, fill takes place automatically. If you edit a finished paragraph and insert or remove words, generally the lines will become irregular in length. Formatting the paragraph with JUSTIFY off will restore its original appearance with as many words as will fit on each line.

JUSTIFY means to add spaces between words to make all the lines come out the same length (to the currently set right margin). JUSTIFY is a mode that is turned on and off by the command ESC ^\\. When JUSTIFY mode is on, the indicator JU appears in the status line at the bottom of the screen.

A paragraph is formatted by placing the cursor at the start of the first line and typing ^\\. The paragraph will be formatted and the cursor will "try to find" the start of the next paragraph. If you are within a number of paragraphs of text without tables and intervening text that is not to be formatted, the cursor is usually left precisely at the start of the next paragraph where you can again enter ^\\ to format the next paragraph. If Justify mode is on, the paragraphs will be right justified. The fill and justify routines remove the previous justification spaces before reformatting the paragraph. If this were not so, it wouldn't be practical to edit and modify a paragraph that was already justified, since the routine would continue to add spaces to the already present ones! The format routine removes all extra spaces except at the end of sentences. If a period is followed by a space, the routine adds a second "marked" space that is not "padded" with further spaces in the process of justifying the paragraph. When the paragraph is justified by formatting it with JUSTIFY off, only the paragraph indent and the extra spaces at the end of sentences are left in the text. All other extra spaces are removed. It is good typing practice to put two spaces after sentences. PAT takes care of this automatically for you as well as removing any extra spaces you might have accidentally left in the text. The end of a sentence is defined as a period, exclamation point or question mark. If you want to delete all extra spaces AND remove the paragraph indent there is an Unjustify command ESC ^U. This command also packs as many words as will fit on each line.

If a paragraph is formatted with ESC ^U the paragraph indent is not done. That is, the first line is not indented by the number of spaces set up as PARINDENT. If you use ^\\ the indent is done. If you are formatting a paragraph with ESC ^U you can start formatting on any line, since the first line is not indented. This is a quick way to clean up ragged looking text after modifying a paragraph extensively. The paragraph indent default value is in the TERMCON.TXT file for the terminal and it may be set by changing that value. PAT also allows you to change the value by typing ESC n^R. N may be as small as zero and as large as the line length. Negative values will cause unpredictable results. Note that changing the indent by using ESC nn^R will not change the default indent. Next time you use PAT it will again come up with the default indent from the TERMCON file.

I am open to suggestions as to whether to add any other ways of detecting the end of a paragraph.

You will probably need to experiment with this feature of PAT for some time to become familiar with what you can do with it. For example, you can change the right margin and reformat a paragraph to a narrow format to make room for a diagram in the text. If you only want to narrow part of a paragraph, split it with an inserted blank line, reformat part of it and then delete the blank line. Being able to format without a paragraph indent will help in situations where you want to reformat parts of a paragraph to a different width.

There are three other commands that can be used to format text. Sometimes it is desired to move a title or heading to the center or right of the text. ESC ^H will move the text on the cursor line to the center of the text, i.e. centered between the left edge and the line length displayed on the status line. ESC ^N will move the text so that it ends at the right margin. ESC ^Y will undo either of the above and move the text back flush with the left margin. (Note the three keys are "in a row" with Y the leftmost, H the center, and N the rightmost).

DESCRIPTION OF KEY FUNCTIONS

General Notes

Control keys that change the contents of the screen or window will cause the window to be completely or partially rewritten as necessary to get the correct information in the window. Note that if you repeat a key that changes the information in the window, the screen rewrite aborts instantly. That is, if you have text above the window, and type ^E five times rapidly, the screen will move up 5 lines and then it will be rewritten. This is a very significant feature of PAT. It saves a great deal of time for the user.

Some of the commands use the ESC (escape) key as a prefix. For example, ESC 5^I means that you type the ESC key, the number 5 and then ^I. (This inserts 5 blank lines above the cursor). Generally the ESC key puts the cursor on the status line at the bottom of the screen and you type characters there, terminating the input with a control character or Carriage Return.

CURSOR CONTROL

^E - Cursor Up

This key moves the cursor up one line. It does what it has to do to accomplish this. If the cursor has room on the screen, It simply moves up one line. If it is at the top of the screen, it "pushes" the screen up in the text so that the next line is visible. If the cursor is at the top of the file, it does nothing in response to this command.

^C - Cursor Down

This key moves the cursor down one line. It behaves like the cursor up key with a few exceptions. If the last line of the file is at the bottom of the screen and the cursor is moved down past the end of the screen, the text moves up 15 lines (for a 24 line monitor) placing blank lines below the text. Moving the cursor down below this screen will move the text up and insert more blank lines. It is not possible to move the cursor so far down that the screen goes blank. The last line of the text will remain. If you must skip more than a screenful of blank lines, temporarily put something on one of them, add your text below the break and delete the temporary characters.

^S - Cursor Left

This key moves the cursor one space to the left without changing the text in any way. The cursor can not be moved past the left end of the line with this key.

^F - cursor Right

This key moves the cursor to the right. When the cursor reaches the 80th column, the screen "scrolls" horizontally so that columns 49 to 128 are displayed. The cursor can not be move past column 128.

^D - Cursor Home / Bottom

This key moves the cursor to the upper left corner of the text window if it is not already there. If it is there, it moves to the lower left corner of the window. The text in the window does not move. This key is handy for getting to the top or bottom of the screen in a hurry.

^] - Cursor Begin / End

This key moves the cursor to the end of a line if it is within the text, and to the beginning if it is at or past the end of the text on the line. On some terminals the right bracket (]) is represented as the upper case of the left bracket. In this case, you will probably have to use CTRL SHIFT along with the "bracket key" to get this function to work.

^G - Right Tab

This key moves the cursor to the next tab position right. Tab positions are defined either by the text already on the line or by the tab settings (see ESC n^T below). If the editor is in the PMODE, tabs within the text move the cursor to the first character of the next word to the right. If you tab past the end of the text, tabs revert to the selected tab columns. In AMODE, tabs always move to the marked tab columns. This is really the best of both worlds. If you are in a paragraph, you can tab three words right by typing ^G three times. It becomes almost automatic. Now go to a blank line and tab to the set columns for entering a table. Tab Right will cause the screen to shift at column 80 but you cannot right tab past the end of a line (column 128).

^A - Tab Left

This key tabs left to the first character of the previous word when the cursor is within text and the editor is in PMODE. When outside of the text, the cursor moves to the previous tab set position. See remarks about Tab Right above. Tab Left is very useful in the AMODE for programming or typing an outline. Tab Left will not tab past the beginning of the current line (column 1).

SCREEN CONTROL - DESCRIPTION

^R - Screen Up

This key moves the screen up in the text by one line less than the number of lines that are displayed on the screen. That is, the first line on the screen becomes the last line after ^R. The cursor stays exactly where it was before the ^R. This feature is useful both in program editing and text editing. You can go up two or three screens to refer to some previously entered text (or program) and go back down the same number of screens, and the cursor will be right at the spot where you were editing before you went away from that spot. Obviously you can't screen up beyond the start of the file.

^V - Screen Down

This key moves the window down in the text by one line less than the number displayed in the window. That is, the bottom line before the ^V becomes the top line. The window will not shift down below the point where the last line of the file is the top and only non-blank line on the screen.

^U - Current Line Up

This key moves the present cursor line (with the cursor) to the top of the screen. (actually it moves the window down in the text but psychologically, you are moving the cursor line to the top of the window, so it fits here on the top row that is associated with UP commands. This is useful for fitting as much of a particular paragraph or program procedure on the screen as possible. Put the cursor at the first line and move it to the top of the screen.

^^ - Current Line Down

This key moves the present cursor line (and the cursor) to the bottom of the screen. This is not a very logical choice of key for this function but at the time it was added, this was just about the last available key.

^Y - Screen Up 1/3

This key moves the screen up in the text by 1/3 screen. If the cursor was high on the screen it will remain with the line where it was before the move. Of course, if the cursor line before the move falls off the bottom of the screen, the cursor will stick at the bottom line.

^N - Screen Down 1/3

This Key moves the screen down by 1/3 screen. The cursor sticks with the line if it stays on the screen. This is useful for proofreading the text. Read along and follow your current line with the cursor. When you get to the bottom, do a couple of ^N's and the cursor will be 2/3 of the way up the screen and at the line you have just read. Now continue reading and bumping the cursor.

^T - Top of File

This key moves the top of the file into the window, with the first line of the file at the first line of the window. Repeated use of this key will do nothing further.

^B - To Bottom

This key moves the bottom of the file to the line 1/3 of the way down the screen and puts the cursor on the next line below the present end of the file. Successive ^B will not change the situation.

EDITING FUNCTION KEYS

^H - Backspace

This key has two different actions depending on whether you have the INSert mode on or off. When the Insert mode is off (Overlay Mode) the cursor moves back one space and replaces the character at that position with a space. In the INSert mode, the cursor moves back a space, deletes the character at that space, and brings the rest of the text to the right of the cursor back with it. If this sounds confusing, the Backspace key operation follows what the cursor does when entering text in the respective modes. The action is what you would expect in these modes.

NOTE: The backspace WILL backspace past the left margin and go to the end of the text on the previous line. You will appreciate this when you want to fix the last character you typed before a Carriage Return. You can back up through a line and into the next one above if you need to. Also note that it is not possible to leave a line with several spaces on it or spaces after the last printable character. PAT automatically 'cleans up' a line when the cursor leaves it. If there are spaces followed by a CR, PAT will remove all the spaces and the CR will be in the first position on the line. When the cursor moves TO a blank line, PAT puts spaces out to the cursor so that if you type something there, it will be preceded by enough spaces to put it there when it is printed. It is not possible to leave trailing spaces on a line with PAT.

^J - Delete Character at Cursor

The character at the cursor will be deleted and text to the right of the cursor will be shifted to the left to fill the space where the deleted character was.

^_ - Delete Word at Cursor

The word in which the cursor is currently located will be deleted including any punctuation not separated from it by a space. That is, if the cursor is in a word, the word delimited by spaces at each end will be deleted, and one of the spaces with it. The effect is to remove the word and leave one space between the previous word and the next word. If the cursor is located on the space preceding a word, that word will be deleted. If it is located in an area of multiple spaces, one space will be deleted.

^I - Insert a Line

A blank line will be inserted where the cursor presently is located, and the text on that line will be moved down. The last line on the screen will move off to the bigger edit buffer. It is not lost.

^K - Kill (delete) a Line

The line on which the cursor is located will be deleted. Text below it will move up to fill the space made by deleting the line. The last line of the display will be moved up from the big edit buffer. If you are at the bottom of the file, the last line will be a blank line. Insert and Kill may be repeated rapidly to insert or delete a number of lines.

Note that ^K has another function. When a line is deleted, it is not lost, but goes to a Hold Buffer (only ONE line will fit in the hold buffer and it will always be the last one deleted). This fact may be used to advantage in two situations. First, if you accidentally delete the wrong line, or one too many lines, you can "restore" the line with the command that follows this one below. Secondly, this is a QUICK method to move a line. You delete the line, put the cursor on the line below where you want to insert it and type ^O.

^O - Output Hold Buffer

This command will output the contents of the hold buffer at the cursor line. The line that was there will move down to make room for the inserted line. This is just like the ^I function except that the inserted line contains text. If you have not used ^K to delete anything, you will get a blank line.

^L - Push Line to Hold Buffer

This command lets you copy a line. It puts the line in the Hold Buffer but it doesn't delete it. Now you can move the cursor and use ^O to duplicate the line.

Note that the above are handy to handle a line or two. There are other ways to mark a large block of text, delete it, move it, or copy it. These are discussed later.

ESC ^S - Split a Line at Cursor

Put the cursor on the first character you want to move to the next line and type ESC^S. The character at the cursor and

everything to the right will be put on a new line below the present one, and the text below will be moved down a line. The cursor remains at the Split point so you can continue to type there. This is handy for inserting more text while re-editing already written material.

ESC ^J - Join Lines

Put the cursor anywhere on a line of text. ^J will join the next line to the current one with one space between, IF THERE IS ENOUGH ROOM on the line. If not, you will get an error message. All error messages remain on the status line until you press ESC, after which you can resume editing. If you want to join lines but they won't fit, you can split the second line at a point where what is left will fit. Any extra spaces at the start of the second line will be deleted when the lines are joined.

ESC nn^I - Insert Multiple Lines

Rather than type ^I 12 times, you can insert 12 blank lines by typing ESC (cursor will move to the status line following the prompt ARG:) where you type in 12^I. The number of blank lines that you specify will be inserted.

ESC nn^K - Kill (delete) Multiple Lines

Works like the above command except that it can delete multiple lines. As a precaution, the status line shows the prompt DELETE 12 LINES? (or whatever number you have specified). Answer with a Y, either upper or lower case, and the lines will be deleted. An N or other response will QUIT the function. If you delete more lines than are left in the edit buffer, all of the text below (and including) the cursor line, will be deleted, i.e. to the bottom of the edit buffer. If you are editing a large file, any part of the file not yet read from the input file will of course, not be deleted.

FORMATTING COMMANDS

ESC nn^L - Set Line Length (for PMODE)

Use this to set the line length at which words typed in will wrap to the next line when in PMODE. The line length appears on

the status line at the bottom of the screen, as L=65, the default value. Note that line length has no function other than in PMODE and in formatting paragraphs. If you type ESC ^L and forget the argument, L will be changed to the default value. There was a possibility of hangup previously since in this case L was set to zero. A subsequent attempt to justify a paragraph caused an exit to the operating system or the monitor.

ESC ^P - PMODE On/Off

This command toggles the PMODE on and off. The default mode is PMODE. That is, when you start editing a file with PAT, it comes up in PMODE. When PMODE is on, the word PMODE appears on the status line. When it is off, you must use a CR to get to the next line.

ESC ^A - AMODE On/Off

When this command is used, AMODE is toggled on and off. If PMODE is on, it is turned off by AMODE and vice versa. The

utility of AMODE is that when you type a CR, the cursor moves to the column under the first printable character in the line above. That is, it goes to the same indent level as the previous line. This is useful for programming or outlines.

ESC x^F - Place Bookmark

X is a letter from A to F. Upper or lower case may be used. This command marks a line so that it may be returned to easily. See ESC nn^G below.

ESC nn^G - Goto Line nn

When you use an assembler or a compiler, generally error messages include the file line number on which the error occurs. You can load the program source file and Goto the line in question. If you have marked a place in the file with one of the six bookmarks that are available (see ESC x^F) you can use ESC x^G, where x is a letter A to F, to return to a marked place. A late addition is the ESC ^G command with no argument. It causes you to return to the place of the last edit activity, i.e. the last place you inserted or changed something in the text.

ESC n^T - Set Regular Tabs

This command sets the tabs every N columns (including the first). It causes the screen to be cleared and a "tab header" to be displayed. The tab positions are displayed under this header line (which shows tab column numbers) as vertical bars. If you want irregular tab columns, type ESC^T and the tab header line will be displayed. The cursor will be at the first column of the line below the header, and you can move it along with the space bar, typing any other character where you want a tab set. When you reach column 79, the next character typed will cause the screen to shift to display the right end of the 128 column tab header. Simply continue defining tabs as before. When you are done, type CR and the header line will rewrite back at the beginning of the line, and tabs will be displayed as vertical bars on the next line. Typing any key will shift the screen and show tabs beyond column 79. When you have verified them type any key to return to editing. Note that if you stop short of the end of the tab header in defining tabs (by hitting CR) any previously defined tabs past that point will be retained. You can clear all tabs quickly by using ESC 127^T.

ESC ^C - Toggle Caps Mode

You may not need this command. If you do, you will be very glad for its existence. Some terminals do not have a CAPS LOCK key such that all letters a to z are converted to upper case but the remainder of the keyboard remains the same. Caps Mode does that with software. If your terminal doesn't have a Caps Lock key or it has a (UGH) Shift Lock, you will need this function.

ESC ^B - Toggle Bell

The BELL column is defined as five less than the set line length. PAT initializes with the bell off. You may toggle it on and off with this command.

There is no indication on the status line of the state of this toggle. You will know by the end of every line typed in, whether or not BELL is on.

ESC ^D - Toggle ignore case on Search

Pat normally matches upper / lower case on searching for a string. ESC ^D toggles a flag that tells PAT to ignore case so that a search for "THE" will find "The" and "the" as well. When the "ignore case" flag is on, the indicator "UL" (for Upper and Lower case) appears on the status line. Remember that when UL appears, case is ignored.

ESC n^R - Set paragraph indent for Justify mode.

ESC ^\\ - Toggle Justify Mode
^\\ - Format current paragraph with indent
ESC ^U - Format present line to end of paragraph no indent.

PAT has paragraph formatting capability via the ^\\ command. If Justify mode is on, the letters JU appear on the status line. Put the cursor anywhere on the first line of a paragraph and type ^\\ and the paragraph will be formatted with full justification (that is, the left and right margins will be even. The first line will be indented by the value of the PARINDENT constant which is configurable via the Terminal configure file and settable with the ESC n^R command indicated above. Spaces will be added within each line to make them all the same length. The length of the line will be the current line length setting. The cursor will generally be left at the first line of the next paragraph so that this command may be repeated. There is no GLOBAL format command because the program cannot always tell the start of a paragraph from something else such as a table, a heading, etc. The end of a paragraph is detected by any of the following:

1. A line starting with a comma, period, or colon. These are used by most text formatting software to indicate a formatting command.

2. A blank line (a CR in the first column).

3. A line starting with a space.

The second case should be obvious. If you end a paragraph with a CR, and then skip a line, the program has no trouble detecting the end of the paragraph. The third case is for those who don't want blank lines between paragraphs. When you type the text, simply indent one space at the start of a new paragraph. See the caution below regarding this method of delimiting paragraphs.

If the Justify mode is off when you issue the Format command, the words will be moved around, lines split and rejoined, so that all the lines contain all the words that will fit within the presently set margins. The only difference from the case of Justify mode being on is that no spaces will be inserted between words. Regardless of mode, Format command will put two spaces after any period that is followed by a space (primarily for ends of sentences). This sometimes causes unsightly gaps after abbreviations such as P. O. Box... If you want to fix these, simply delete the extra space and insert it elsewhere in the line to maintain justification.

If a paragraph has been formatted with Justify on, formatting it with Justify off will remove all the extra spaces that were added to make the lines come out even. The paragraph indent will still be added to the line on which the cursor is at the format command. Formatting with ESC ^U will produce identical results except that the indent will not be done on the first line. You can use this command to reformat parts of paragraphs. Some users of PAT, particularly if they have a text formatter program that works with a printer that has proportional letter spacing, will want to have the formatter do the justification. However, the capability to maximize words on each line will be appreciated after re-editing a section of text.

CAUTION: If you do not use blank lines between paragraphs, an attempt to unjustify a paragraph with justify off or with ESC ^U will run all of your text together into one big paragraph. BE CAREFUL.

If you have one of the many printers that provide only constant pitch printing (10 per inch or 12 per inch), you may use PAT to pre-format paragraphs so you can get an idea of how long the text will be. Note, however, that PAT has no provision for page headers, footers, or page numbers. Such functions will have to be handled by a text formatter.

ESC ^H - Center Text

This command will cause the text on the cursor line to be centered between the set margins. The text will be moved to the center and spaces will be inserted between it and the left margin.

ESC ^N - Right Justify Text

The text on the cursor line will be shifted to the right margin and spaces will be added in front of it.

ESC ^Y - Left Justify

This command will move the text on the cursor line to the left margin. It is used primarily to undo the two preceding commands.

SEARCH AND REPLACE

Search and replace functions are easy to use. First you enter the search string. As soon as it is terminated, the search begins, and if the string is found in the text, it will be on the screen with the cursor at the first character. Then the Replace string is entered. When it is terminated, the string will be replaced. If you decide that the "found" string is not the one to be replaced, you can repeat the search and look for the next occurrence of it. There is also a "GLOBAL" replace mode in which all occurrences from the cursor to the end of the file are replaced.

ESC TEXT^Z - Enter Search String and Search Forward

Type ESC and the cursor goes to the status line. Now type the text string you want to find. Spaces may be included in the text string for the search. This is useful when searching for such words as "the". ESCthe ^Z with a space after the word "the " distinguishes the word "the " from the words "them" or "these". The control Z terminates the search string and causes the search to start. The search proceeds forward from the position of the cursor at the start of the search. Search honors lower case and capital letters. That is, a search for "school" will not find "School".

ESC ^D - Toggles the "ignore case" mode

The word "SCHOOL" or "School" will match the word "school". This mode is indicated by the "UL" on the status line (for Upper/Lower).

ESC TEXT^Q - Enter Search String and Search Backward

Type ESC and the cursor goes to the status line. Now type the text string you want to find, terminated by ^Q. The control Q terminates the search string and causes the search to start. The action is identical to that for ESC ^Z described above except that the search is backward in the text starting at the present position of the cursor.

ESC TEXT^X - Enters Replace String

This string is entered in the same way as the search string.

The cursor should be at the found occurrence of the search string since it will be replaced after you enter this string. Note that the replace string is terminated with ^X. The replace string may be a null. That is you may type ESC^X in which case the search string will be replaced with nothing at all, in other words, it will be deleted.

^Z - Repeat the Search Forward

A search string must be defined or you will get an error message Nothing to search for". With this command, you can search through a text or program for all the places where the string occurs and decide at each one what you want to do with it.

Note: searches all proceed from the present cursor position. If you want to look at all occurrences of a particular string, be sure you start at the TOP of the file and search DOWN, or at the bottom of the file and search UP.

^Q - Repeat the Search Backward

This works just like ^Z above, except that the search for the currently defined search string proceeds backward in the text from the current position of the cursor.

^X - Repeat Replace

This command will use the previously defined replace string. The cursor must be at the first character of an occurrence of the search string in the text. It may have arrived there by the completion of a successful search for the string or you may have put it there by moving the cursor. If you type this command and this condition is fulfilled, the current occurrence of the search string will be replaced with the current replace string.

^W - Global Replace

After finding and replacing the first occurrence of a string, and without moving the cursor, type ^W and all occurrences from the cursor position to the bottom of the file will be found and replaced. You can stop the global replace by typing ^C. If you let it go to completion, the screen will update with the last replacement on the screen and marked by the cursor. Note that ^W will do nothing unless it immediately follows a ^X instruction. This is for safety. Otherwise it would be possible to make undesired changes after a ^Z by accidentally typing a ^W. Also note that Global Replace only works in the Search Forward mode.

BLOCK FUNCTIONS

Blocks are important in most editors. You "mark" a block of text, and then you can do one of several things. PAT will let you DELETE a marked block, COPY it to somewhere else in the text, (and leave it where it was), MOVE it to somewhere else (removing it from its original position) WRITE it to a file, or UNMARK it if you change your mind. You can also READ a BLOCK from a file into your text at a selected place. All of the block functions are terminated by a CR. Blocks are defined as one or more whole lines. If you have anything but the dumbest of terminals and you have enabled an altered video attribute for the status line, (reverse video or low intensity, for example), a marked block will appear in hat attribute as soon as you have marked both ends of the block.

ESC T<CR> - Mark Top of block

Place the cursor on the first line that is to be included in the block and type ESC T<CR>. The number of that line will appear on the status line (displacing the MEM NNNNN temporarily) as T=NNN.

ESC B<CR> - Mark Bottom of block

Put the cursor at the last line in the block and type ESC B<CR>. The reminder B=NNN will appear after the T=NNN on the status line. These prompts will remain there as long as the block is marked.

ESC K<CR> - Kill Marked Block

The position of the cursor is unimportant when you issue this command. The marked block is deleted. It disappears from the text that is being edited and the screen is rewritten without it. Of course, since the block disappears, it becomes unmarked and the MEM NNNNN status reappears.

ESC M<CR> - Move Marked Block

Place the cursor at the line where you want the block to be inserted (it will be inserted above the line where the cursor is placed. Again, since the block disappears from its original location, it will be unmarked.

ESC C<CR> - Copy Marked Block

Place the cursor at the line where you want a copy of the marked block. It will be inserted above the cursor line, and also remain in its original location. Since the original block is still there, it will remain marked and you can copy it again any number of times. When you are done copying, just use the ESC U command described below, to unmark the block.

ESC Sn<CR> - Shift Block left or right

This command shifts a marked block 'n' spaces. If n is positive (no sign) the shift is to the right. If n has a minus sign directly preceding it, the shift is left. Left shifts will not cause printable text to be deleted. The left shift for any line is stopped when leading spaces run out. Right shift does not check for the line exceeding the 128 character maximum line length. It is assumed that this feature will be used primarily for adjusting indenting of a program source, where it is handy after restructuring the logic and leaving a number of lines at the wrong indent level.

The shift command only moves text at or to the right of the cursor column. To move all text be sure the cursor is at the beginning of the line. Text cannot be deleted by such a move. Text will only move left as far as the cursor column and only spaces can be deleted.

ESC U<CR> - Unmark Block

This command does just what you would think. If you make a mistake in marking a block or have copied it and want to unmark the original, this is the command to use. When the block is unmarked, the MEM NNNNN display returns to the status line.

ESC >FILENAME<CR> - Write Block to File

The arrow pointing at the filename is the key for this command. It opens a file named FILENAME (default extension .TXT and default drive, working drive). Then it copies the marked block to that file and closes it. Since the block is not deleted from the file being edited, it is left marked. After copying it to a file you can unmark it or delete it.

ESC <FILENAME<CR> - Read Block From File

The arrow pointing FROM the filename indicates that the file is to be read IN to the text being edited, AT the cursor (again above the current cursor line). The same defaults apply as above.

EXIT FUNCTIONS

ESC ^^ - Exit

This is the normal exit function. It does just what is described earlier in this manual under the heading FILES. That is, it first writes the output .SCR file and closes it. If PAT was invoked with only one filename, that is, not a different input and output filename, it then deletes the input file and renames the .SCR file to the name of the input file. If PAT was invoked with different input and output filenames, the output file was opened with the specified file name and no files are deleted or renamed. Note that this command is ESC CTRL up-arrow. The up-arrow on some terminals is typed by using SHIFT 6. If the up-arrow is a "capital 6" on your terminal, you may or may not have to operate the SHIFT key in addition to the CTRL key to get a control up-arrow. Some terminals shift automatically since ^6 is not a valid ASCII code. Others do not.

ESC Q^^ - Quit edit

This mode simply abandons the current edit. It will always leave the input file as it was before you started the edit. If the file is too big to fit in the edit buffer, part of it may have been written to a .SCR file. Using this command in that case will leave a partial output file with the extension .SCR. If you have done a lot of work on the early part of the file and then fouled up a later part, the partial output file may be of some value. Generally it is better if you have already written to an output file, to use the normal exit mode.

BIG FILES

ESC N^^ - New text

This command requires that the cursor be positioned with some thought. It causes all of the file above the cursor to be written to an output file and if the input file is not empty, more of it is read into memory. If the remainder of the file will fit in the edit buffer limit, it is all read in. If not, the New command may be repeated later after writing more text to the output file. This is a way to edit very large files, though in general it is better to split files into "chapters" or "libraries" so that each file will fit the edit buffer. Most text formatters allow you to specify a starting page number, and you can print your files out in sequence, each time specifying the next page number as the first for the next file. This feature is a hold-over from the 6809 version of PAT in which the large text buffer was only 29K. With the capability of 100K or 200K of buffer, this will not be needed very often.

ESC W^^ - Write Old Text

This command is essentially the same as the ESC N^^ immediately above except that it Writes the text above the cursor to the output file but it does NOT read in more text from the input file. It simply moves the remaining text up to the top of the buffer. This command is useful if you want to add a large amount of text (or read in a side file) in the middle of a very large file. If you are careful about what you write out, you can read a side file into an almost empty buffer and thereby insert a very large amount of text in the middle of a large file.

VERSION NUMBER

ESC ^V - Get Version

This command will show the Version (revision number) of your copy of PAT. If you write concerning a bug, please include this version number.

CONCLUSION

Well, that is about all there is to PAT. This manual is brief for the most part because PAT is simple to use. You will see after a short learning period that PAT has a feature called Typeahead. Typeahead means that you can continue to type text in while the screen is updating and PAT will not miss any characters. While it may at first be distracting to continue to type and not see the screen update, you will soon get used to the screen "catching up" with you. If you are a touch typist, you will probably avoid looking at the screen during the update when word wrap is taking place.

If you have any questions please write the author. I will do everything possible to incorporate any suggestions in future versions of this manual, and will consider suggestions for improvements to PAT itself, though part of the usefulness of PAT is derived from its simplicity. On the basis that a new feature usually brings about a long debug session, an idea for an additional feature is going to have to be a real winner for me to decide to add it.

Please look at appendix B. It is a short summary of all the functions of PAT. If you configure PAT for a terminal not included in the configuration files, I would appreciate a copy of your configuration file so I can include your terminal in future releases of this software.

Lastly, I hasten to mention that my simple word processor JUST is an excellent companion to PAT for word processing applications.

APPENDIX A

Configuration for Various Terminals

There are several terminal configuration files on the disk supplied with PAT. These files can be generated using any editor, since they only require printable characters. PAT interprets these configuration files to produce the proper control sequences for a terminal. All of the configuration files are named termcon.xxx where xxx generally denotes the name of the terminal for which the configuration file will work. As an aid to keeping track of which control string does what, comment lines are allowed in the termcon file. Each comment line including blank ones must start with an asterisk (*) in the first column of that line.

OFFSET - Line 1

The OFFSET is the value that the terminal requires to be added to the binary code for the desired cursor position. It is always the same for both Line (Y) and Column (X). Most terminals use decimal 32, the value of the SPACE character, so that the cursor position ASCII codes do not overlap the ASCII control codes $00 through $1F. Some terminals use 0. Use a decimal number for this value. ANSI terminals do not use an offset and you can leave the value zero if your terminal expects the row and column information as ASCII strings as the ANSI terminals do.

CURSOR POSITIONING - Line 2

This is the sequence that places the cursor at a desired row and column on the screen. In this sequence, if your terminal expects a binary value for the row and column, as do nearly all the non-ANSI standard terminals, substitute ^y for the ROW or LINE value and ^x for the COLUMN value. Many terminals use the sequence ESC = ^y ^x. When you use control characters in a sequence, you must represent the control character as an up arrow (^) followed by the character. That is, ^C represents control C. Note that you must use the upper case character if the control code uses a letter. See table 1 for the equivalence of decimal, Hex, and Octal values to control characters. Note that ESCape is hexadecimal 1B or decimal 27 which corresponds to ^[. A printable character is used without any up arrow in front of it. No spaces are to be used to separate the parts of the control string. That is, the string given above should be entered as ^[=^y^x. The ANSI terminals require that the row and column be sent in string form. If this is the case, use ^r to represent the row or line and ^c to represent the column. The ANSI standard cursor positioning string is:

```
^[[^r;^cH
```

That means, in turn, ESC, literal [, row as an ascii string, e.g."15", literal ;, column as an ascii string "76" and lastly literal letter capital H. The quotes are not part of the string. Please be careful defining this sequence for your terminal. The others are all simple if you get through this one.

CLEAR SCREEN - Line 3

This is the code that clears the screen of the terminal. You may use a code that will also HOME the cursor, or a code that will not. PAT assumes that it has to place the cursor before it starts rewriting the screen. The Televideo terminals use ^[* .

Note that the above are all the terminal control codes that are absolutely REQUIRED by PAT. The following ones speed up the screen refresh. There are also some default configuration constants after the terminal control codes. If your terminal does not support a particular feature, or if you don't need a terminal initialization or restoration string, leave that line blank.

ERASE TO END OF SCREEN - Line 4

The code that clears the screen from the current cursor position to the end is required here. The initial release version of PAT does not use this code, but it may be used in future releases.

ERASE TO END OF LINE - Line 5

The code that clears the current line from the cursor position to the end of that line is required here. This code improves the performance of PAT greatly if it is available. Pat substitutes the printing of spaces to the end of the line if this code is not present.

SET ATTRIBUTE - Line 6

This code should "turn on" half intensity or reverse video. The attribute selected here is used on the status line, making it distinctive so it can be seen as apart from the text. Note that the Televideo 925 and 950 (and many others) have a peculiarity. They use a character position for the code to turn reverse video on and off. PAT will now acommodate these terminals. If you select an attribute that does use a character position on the screen, you must set the MODAL flag at the end of the configuration file (see below) to -1. Be careful. The Televideo terminals take the character space to set reverse video, but not to set half intensity. The value of the MODAL flag may have to be changed when you change the selected attribute.

Note also that on some terminals you first select an attribute by means of some other code, and then use a different code to actually SET the attribute. The Select Attribute code should be placed in the TINIT slot below, and the SET code should be put here.

CLEAR ATTRIBUTE - Line 7

See the comments above under Set Attribute.

CURSOR OFF / INSERT MODE SHAPE - Line 8

If your terminal allows turning the cursor off you can have it blink when in insert mode and be solid in overlay mode. If you want to do this, place the code here.

If you are unfortunate enough to have a terminal with a cursor that blinks all the time, or if your terminal doesn't allow turning the cursor off, but has commands that can alter the appearance of the cursor, you might want to choose one cursor shape and put the command to turn on that shape here. I suggest using the "standard" shape for the overlay mode and a different one here for the insert mode. You must also set the switch (last item in the configuration file) properly depending on your choice here.

CURSOR ON / OVERLAY MODE SHAPE - Line 9

If you elected to put the cursor off code on line 8 you will want to use the code that turns the cursor on here. This code may be the same as the cursor turn off code on some terminals. PAT will correctly keep the cursor in the proper state if this is the case. If you supply cursor on and off codes, the cursor will blink in insert mode and be steady in overlay mode.

If you have changed the shape of the cursor on line 8 you will want to put the code here to restore the cursor to its normal shape. PAT will restore the cursor to this shape on EXIT regardless of the status of insert/overlay.

TERMINAL INITIALIZATION CODE - Line 10

If your terminal has a select attribute code (as the ADDS terminal in the example program does) you will put that code here. If you have some other initialization code for your terminal and it fits, you can put that here as well.

RESTORE TERMINAL - Line 11

You may want to cancel the attribute selected, or select another one when you exit PAT. You can put the code for that here.

Default Parameters

The following are not control codes but set-up parameters. Simply type the number that is asked for. That is, if you like a paragraph indent of five spaces just type 5 all by itself on that line.

SCREEN WIDTH - Line 12

This byte is necessary so that you can tell PAT the width of the display screen for your terminal. For most terminals this constant is 80.

PARAGRAPH INDENT - Line 13

You can set the number of spaces to be indented at the start of a paragraph here. You can use 0 but there must be a value here. Values 3 through 5 are normal.

TAB COLUMN SPACING - Line 14

Insert the number of columns you want for the initial tab separation. If you are primarily using PAT for program editing, you might want 2 or 3. If you write a lot of outlines or tables in text editing you might want 5 or 6. Note that you can easily change the tabs while running PAT. This initial setting is for your convenience.

SCREEN HEIGHT - Line 15

The requirement here is the number of lines on your screen that will be space for edit lines. Since the status line takes up just one line, this will be the total number of lines that your terminal can display minus 1. If your terminal is a 24 line terminal, you will enter 23 for the screen height. Don't count the 25th line that some terminals use for a set-up or terminal status line as one of the available lines.

MODAL - Line 16

Check the manual for your terminal. If the video attribute you want to use for the status line uses a character position on the screen to set and clear that attribute, set MODAL to -1. If no character position is used to set the attribute, set MODAL to 0. Be careful. Many terminals (particularly most Televideo models and their emulators) use a character position on the screen to set or clear reverse video, but they do not use a character position to set and clear half intensity. The setting of the MODAL flag must therefore be consistent with the video attribute that you have selected, and changing the video attribute can change the required value for the MODAL flag.

SHAPE FLAG - Line 17

If you chose a blinking cursor for insert mode indication, (lines 8 and 9 above) put a 0 here. If you chose to change the cursor shape to indicate the mode, put -1 here. (That is, literally the characters -1 or 0.

THAT'S ALL

Now save the edited file and copy it to your system disk or partition of the hard disk as TERMCON.TXT. PAT presently looks at drive 0 root directory for this file.

Control Characters

Decimal	Hexadecimal	octal	Control
0	0	0	^@
1	1	1	^A
2	2	2	^B
3	3	3	^C
4	4	4	^D
5	5	5	^E
6	6	6	^F
7	7	7	^G
8	8	10	^H
9	9	11	^I
10	A	12	^J
11	B	13	^K
12	C	14	^L
13	D	15	^M
14	E	16	^N
15	F	17	^O
16	10	20	^P
17	11	21	^Q
18	12	22	^R
19	13	23	^S
20	14	24	^T
21	15	25	^U
22	16	26	^V
23	17	27	^W
24	18	30	^X
25	19	31	^Y
26	1A	32	^Z
27	1B	33	^[
28	1C	34	^\\
29	1D	35	^]
30	1E	36	^^
31	1F	37	^_

Appendix B

PAT Control Functions

Cursor Motions

```
   ^E up one line
   ^C down one line
   ^S left one character
   ^F right one character
   ^D toggles top and bottom of screen
   ^] cursor toggles beginning - end of current line
   ^G tab right - tab to tab col in AMODE, next word in PMODE
   ^A tab left  - Ditto.
```

Other Control Functions

```
   ^H backspace erases previous character. It is possible to
      backspace past the start of a line.
   ^P toggle insert mode.
   ^I insert a blank line above the cursor line.
   ^K delete cursor line.  The line is saved to the Delete
      Buffer.
   ^_ delete word in which cursor is located.
   ^O insert a line (like ^I) and then fills it with the
      contents of the delete buffer.
   ^L save a line to the Delete buffer without deleting it.
   ^J delete current character and move rest of line left.
   ^\\ format a paragraph.  Action is different depending on
      whether Justify mode is on or off. See ESC commands below.
   ^Z search for next occurrence of a string defined using
      ESC (string) ^Z or ESC (string) ^Q.  See below.
   ^Q search for previous occurrence of a string defined using
      ESC (string) ^Z or ESC (string) ^Q.  See below.
   ^X replace present occurrence of a string just found by
      using ^Z or ^Q with the string defined by using
      ESC string^X. See below.
   ^W Global change.  After using ^Z and ^X, if you want all
      occurrences of the string changed, type ^W.
```

Window Motions

```
   ^T top of file
   ^R up one screen
   ^Y up 1/3 screen
   ^E is normally a cursor up function but it will move the
      screen if used on the top line when the top line does not
      already display the first line of the file.
   ^B bottom of file
   ^V down one screen
   ^N down 1/3 screen
   ^U move cursor line (and cursor) to top of screen.
   ^^ move cursor line (and cursor) to bottom of screen.
   ^C normally a cursor down function, but if the cursor is on
      the bottom line, will move the screen down one line.
```

ESCAPE Commands

```
   ESC ^A - toggle (A)MODE
   ESC ^B - toggle BELL on and off
   ESC ^C - toggle (C)aps lock
   ESC ^D - toggle ignore case on search
```

```
ESC ^E - erase from cursor to (E)nd of line.
ESC x^F - Set Bookmark (F)lag (x may be a thru f)
ESC n^G - (G)oto Line n
ESC x^G - (G)oto Bookmark (a thru f)
ESC ^G - Goto point of last edit.
ESC ^H - Center text on the cursor line
ESC n^I - (I)nsert n blank lines at cursor
ESC ^J - (J)oin lines at end of first line with warning if
         text won't fit.
ESC n^K - delete n lines starting at cursor.
ESC n^L - set (L)ine length to n - for automatic word wrap.
ESC N^^ - save file above cursor.  Move (N)ew file in from
          input file.
ESC ^N - Right justify text on the cursor line
ESC ^P - toggle (P)MODE
ESC n^R - Set pa(R)agraph indent value.
ESC ^S - (S)plit line at cursor.
ESC n^T - set (T)abs
ESC ^\\ - toggle JUSTIFY MODE
ESC ^U - Fill a paragraph without the paragraph indent.
ESC ^V - displays the revision number of your copy of PAT on
         the status line.  Hit ESC to continue editing.
ESC ^^ - save file with backup.
ESC Q^^ - for (Q)uit.  closes input file and exits without
          saving files.
ESC W^^ - write file above cursor to output file without
          moving new text in from input file.
ESC ^Y - Left Justify text on the cursor line.
ESC str1 ^Z - Find next occurrence of str1.
ESC str1 ^Q - Find previous occurrence of str1.
ESC str2 ^X replace string found by ESC Z or ESC Q
              command with str2.
ESC T<cr> - mark (T)op line of block for deletion, copy or
            move.
ESC B<CR> - mark (B)ottom line of block.
ESC U<CR> - (U)nmark presently marked (or partially marked)
            block.
ESC K<CR> - (K)ill marked block (and unmark).
ESC C<CR> - (C)opy marked block above present cursor line.
ESC M<CR> - (M)ove marked block above present cursor line.
ESC Sn<CR>- (S)hift marked block n spaces (- for left).
ESC >filename <cr> - write marked block to file.
ESC <filename <cr> - read file into edit buffer at
       cursor.
```

Appendix C
Custom Key Configuration

Version 4.0 of Pat-68K for serial terminals, has the feature of user configurable command keys. Presently, the capability is limited to exchanging, swapping or scrambling the key assignments for simple Control key functions and for ESC control key sequences. You may not assign a key outside of the control code range (0-31 decimal, 0-1F Hexadicimal). The configuration table is included in the latest TERMCON.TXT file, the terminal configuration file. Because of the way that the table is interpreted, the position in the table determines the function. The key named in the line after the comment describing the function will be assigned to that function. The first 32 entry table is for the simple control commands. For example if you swap the ^E and ^C in the table, the Cursor Up and Cursor Down commands will be swapped. Do Not delete any of the entries completely. If you do so, all the commands below that point will be displaced by one, which could be a catastrophe. While experimenting ALWAYS keep a copy of the original and working TERMCON. Rename it to TERMCON.OLD and if you make a fatal error and make Pat inoperative, you can reset the computer and delete your new attempt, then rename TERMCON.OLD to TERMCON.TXT and you will be back where you started.

It is important to understand that if you don't keep a copy of your old and working TERMCON, and the new one doesn't work, you have no way of editing it to fix it, unless you can rename the old TERMCON and revert to a working editor again. Then you can fix the problem in the new file and go through the renaming process to try it again, etc.

I've added this feature with some fear and trembling because I don't want to receive calls from users who have gotten "lost" and can't help themselves. If you must change key assignments, be VERY careful. Try a simple swap first to be sure you understand the process. Again, DON'T try to change the order of the commands, just the control codes associated with them.