ROMPLUS+ IN SLOT 5

Mountain Hardware, Inc.

LEADERSHIP IN COMPUTER PERIPHERALS

KEYBOARD FILTER

SOFTWARE MANUAL

KEYBOARD FILTER
SOFTWARE MANUAL

Manual written by Barney Stone

TABLE OF CONTENTS

INTRODUCTION

The Apple II* computer is certainly one of the finest personal computers available today. However, even the best system can be improved by the addition of features not included in the original design. Part of the true power of the Apple II lies in the easy expandability of the system afforded by the 8 slot expansion bus located to the rear of the computer.

The Mountain Hardware ROMPLUS+ is an example of this kind of expandability, and the Keyboard Filter, by Andy Hertzfeld, will add numerous features to your Apple II when used with the ROMPLUS+. The Keyboard Filter, a 2K ROM chip which is essentially an extension to the Apple's monitor, includes machine language programs which add lower case letters, keyboard macros, improved cursor movement, "StopList" and "Endlist", and other features to the Apple. Each feature is controlled by the user through simple keyboard commands. The actual programs on the chip are "transparent" to the user, who need have no understanding of machine language in order to make full use of the Keyboard Filter's capabilities.

In the pages that follow you will find simple examples which demonstrate the various functions of the Keyboard Filter. Once you have followed the instructions for installation of the Keyboard Filter and ROMPLUS+, it is suggested that you go through the various examples in this manual, experimenting at least briefly with each feature as you read about it. This is the only sure way of learning everything which your Keyboard Filter can do for you and your Apple II.

NOTE: Although the Keyboard Filter will work on a system of almost any memory size, some decisions had to be made that were memory size dependent (such as the default location of the font table). In such cases, we have opted for convenience

*Apple II is a trademark of Apple Computer, Cupertino, CA.

for those with 48K systems. This is in light of the continuing decline in memory prices. It was also assumed that most users will either have or plan to have a disk drive as part of their system. Therefore, some of the features and software discussed in this manual will not work properly if your system does not have 48K RAM and a disk drive. Wherever possible, we have included patches to permit use with different system configurations.

BACKGROUND

The monitor program of the Apple II, which resides in the standard ROM memory chips supplied with the computer, normally controls all of the system's input and output. The advantage of this lies in the ease of use of the computer, especially when compared with the earlier hobbyist computers, many of which required the tedious entry of monitor programs through front panel switches before the computer even "knew" how to "listen" to a keyboard. A major feature of the Apple II's monitor lies in the way in which it handles input. It simply does not want to deal with lower case letters. When it receives the code for a lower case letter, whether from its keyboard or from another input source, it converts that letter to upper case before storing it in the computer's memory. Thus, normal keyboard entry of lower case letters required the design of a software system which would "grab" the keyboard input, interpret it, and if necessary, go around the monitor in order to store the desired codes into memory. Once such a system was designed, it was apparent that it was useful for far more than just the entry of lower case letters, and the ensuing work led to the development of the Keyboard Filter.

The Keyboard Filter has two main operating modes; one which uses the Apple's high resolution graphics display, and the other which uses the normal text display. In the graphics mode, the system will display upper and lower case letters, using either the standard font (located on the Keyboard Filter chip) or one or more user-defined character sets residing in RAM memory. Text and high resolution graphics may be freely inter-mixed anywhere on the Apple's screen. The display will mimic the normal Apple text display in almost every way, including cursor movement, screen copying, text "windows", etc. Scrolling, however, will be slowed somewhat due to the necessity of re-writing the entire

graphics screen each time it is scrolled. Also, the cursor does not blink in this mode.

In the text mode on an Apple, lower case letters will appear as various punctuation marks or numerals, making text with lower case letters unreadable. By using the Keyboard Filter's shift lock, however, all of the other features of the Keyboard Filter (cursor movement, keyboard macros, StopList, etc.) may be used without using the graphics memory area, and with full-speed scrolling. The cursor will blink normally in this mode.

On those Apples which have been modified with a new character generator ROM for upper and lower case display (such as the one sold by APPLECATIONS UNLIMITED), the Keyboard Filter's text mode will provide for keyboard entry and normal display of lower case letters, along with all of the other Keyboard Filter features (except for user defined characters and mixed text and graphics), and full speed scrolling.

As you can see, we have striven to provide the greatest flexibility possible for use of the Keyboard Filter, and most users will find different options to be valuable for different applications. We have also tried wherever possible to make the procedures for choosing and switching Keyboard Filter options as simple as possible.

INSTALLATION

The Keyboard Filter is designed for use on Mountain Hardware's ROMPLUS+. It may be used in any of the ROMPLUS+'s six sockets, however, we recommend that it be installed in socket #1, and in the examples which follow it will be assumed that this recommendation has been followed.

NOTE: Your Keyboard Filter may already be installed on the ROMPLUS+. If so, skip to the next section.

NOTE: ROM chips are sensitive to static electricity. The following procedure should not be attempted in a high static environment!

To install the chip, first remove the ROMPLUS+ from your APPLE. REMEMBER - THE APPLE'S POWER MUST BE TURNED OFF WHENEVER INSTALLING OR REMOVING BOARDS FROM THE APPLE!

Next place the ROMPLUS+ on a dry flat surface. Locate the socket labeled #1. The notched end of the Keyboard Filter chip goes toward the left side of the board. See Figure 1.

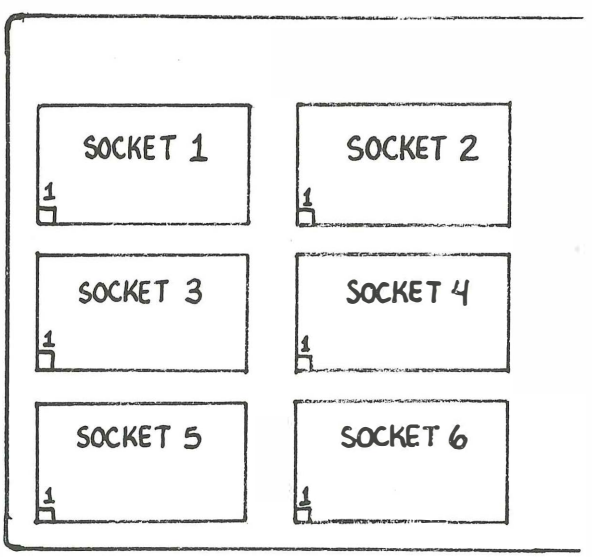


FIGURE 1

Now, slowly and carefully install the chip into the socket, being careful not to bend any of the chip's pins. If a pin should become bent, carefully pry the chip back out of the socket (taking care not bend any more pins). Gently straighten the bent pin (don't break it off!), and try again.

Finally, refer to the ROMPLUS+'s instructions for installation of the board into the computer.

YOUR DEMO DISK

The demonstration disk provided with the Keyboard Filter includes a number of sample and utility programs which you may find useful and informative. Before we begin learning about the Keyboard Filter, you may wish to try some of these demos to get an idea of what the Filter is all about.

The following is a list of the files with a short description of each one. Note that not all of the files listed when you CATALOG your disk are programs. Some are data files, one is an EXEC file, and two are portions of a program which should not be RUN except when they are called for by the main program.

1. KEYBOARD FILTER PROGRAMS. This is a simple program which will turn on the Keyboard Filter and "sign on" when you boot your demo disk.

2. QUICK TOUR is the first program you should run. As with the others on this disk, it will initialize the Filter. It will then take you on a quick tour of the Keyboard Filter and its capabilities. Where appropriate, interactive routines have been included so that you may begin learning how to use the Filter right away. All of the programs on this disk may also be studied as examples of using the Filter.

3. GRAPHICS DEMO is a simple program which illustrates how to mix text and graphics using the Keyboard Filter.

4. MACRO EDITOR is discussed later in this manual. This program is used in defining your own Keyboard Macros, a type of computer shorthand.

5. FONT EDITOR, also discussed below, is used in creating your own character sets.

6. FEDIT & CEDIT are parts of the FONT EDITOR program which are called in with the CHAIN command when they are needed. If one of these programs is run without running Font Editor, it will automatically run Font Editor for you.

7. MANIMATION is an example of using the Keyboard Filter for simple animated graphics. Parts of MANIMATION are also used in the QUICK TOUR program.

8. There are also several ".MTAB"'s and ".FONT"'s on the disk. These are sample macro tables and character fonts for use by the demo programs or in your own programs. ".MTAB"'s are created by the MACRO EDITOR program, and ".FONT"'s by the FONT EDITOR. Information on using these programs is included below.

9. SLOT FINDER is an EXEC file. This file can be EXEC'd into your own programs and used as a subroutine. It will search for a Mountain Hardware ROMPLUS+ card and Keyboard Filter, and if it finds one, tell you which slot it is installed in. Slot Finder may be used with Integer Basic or Applesoft II. See "Using SLOT FINDER" in the Appendix.

INITIALIZATION

NOTE: A special initialization routine must be used if you wish to use the Keyboard Filter in the graphics mode with a RAM version of Applesoft II Basic (i.e., loaded from tape or disk). See "USING THE KEYBOARD FILTER WITH RAM APPLESOFT".

Once the Keyboard Filter and ROMPLUS+ have been installed in your computer, they will have no effect on the normal operation of the system until they are called upon by a simple initialization routine. (In this section we will deal only with the initilization procedures for the Keyboard Filter. For a complete discussion of all of the ROMPLUS+ board's initialization options, refer to the ROMPLUS+ manual.)

The ROMPLUS+ is turned on from Basic with the "PR#(SLOT)" command or "IN#(SLOT)" command, where (SLOT) is the number of the peripheral connector in which the ROMPLUS+ is installed. At this point the initialization firmware on the ROMPLUS+ begins to look for a special character which will tell it what to do. Until it receives that special character (Shift/Control/M or Shift/Control/N), it continues to have no effect on the Apple's normal operation.

The Keyboard Filter itself may now be turned on by holding down the "SHIFT", "CONTROL", and "M" keys simultaneously, then typing the number of the socket in which the Keyboard Filter is installed (usually socket #1), then an A,B,C or D, depending on which initialization option you desire (this will be explained below).

Thus typing: PR#(SLOT)(RETURN)
(Shift/Control/M)1A

will turn on the Keyboard Filter (assuming, as we will, that it is in socket #1), and run initialization procedure "A".

NOTE: A "SYNTAX ERROR" will be displayed following the first "RETURN" after initialization of the Keyboard Filter. This is normal, and should be ignored. If you find this annoying, simply type a (Control/X) or a backspace before hitting RETURN.

NOTE: When the PR#(SLOT) command is used within a program which also makes use of the Apple's Disk Operating System (DOS), the PR#(SLOT) command must be used within a print statement which begins with a (Control/D). This is the standard protocol for use of input or output commands with the DOS. The following example, included within a program, will turn on the Keyboard Filter without interfering with the DOS:

```
10 D$="":REM THERE IS A (CONTROL/D) WITHIN THE QUOTATION
MARKS

20 SLOT=3:REM ASSUMES ROMPLUS+ IS IN SLOT #3. This may
be replaced by GOSUB 11000 to automatically set SLOT if
Slot Finder is part of your program.

30 M$="":REM THERE IS (SHIFT/CONTROL/M) WITHIN THE QUOTES

40 PRINT D$;"PR#";SLOT

45 PRINT:REM NECESSARY BECAUSE THE FIRST CHARACTER
AFTER A PR#(SLOT) OR IN#(SLOT) WILL NOT BE INTERPRETED
AS A COMMAND

50 PRINT M$;"1A"

60 END
```

See "Using SLOT FINDER" in the Appendix.

The Keyboard Filter has four initialization options: A,B,C and D. Options A and B make use of the Apple's high resolution graphics to create a full upper and lower case character set and to facilitate user-defined characters. Option A is used for first time initialization, that is, if the power has been turned off since the last time the Filter was used, etc. Option B is used to re-initialize the Filter after it has been disconnected, such as by a "RESET".

Option C is the first time initialization which does not use the high resolution graphics screen (i.e., it uses the text mode - see above), and Option D is the non-graphics or text mode re-initialization.

A = FIRST TIME INIT WITH GRAPHICS
B = RE-INIT WITH GRAPHICS
C = FIRST TIME INIT - TEXT MODE
D = RE-INIT - TEXT MODE

A first time initialization (Option A or C) sets the following default values for Keyboard Filter functions:

1. Type font in use = 0 (font on Keyboard Filter chip)
2. Address for font #1 = 6000 (Hex)
3. Address for keyboard macro table = 800 (HEX)
4. Color = White
5. Inverse mode is OFF
6. Video display is switched to Page 1

The following default values are set by both first time and re-initializations:

1. Shift Lock ON
2. Raw Mode OFF
3. Overstrike Mode OFF
4. Shift Key Mod OFF

A re-initialization (Option B or D) preserves the previously set values for the above list of parameters.

It is also possible to switch between the graphics and non-graphics modes once the Keyboard Filter has been initialized by typing (Control/Shift/M)1(B or D), with the B or D depending on which option you wish to switch to.

NOTE: Since the Keyboard Filter makes use of the high resolution graphics display, the commands GR, HGR, HGR2, HCOLOR, and TEXT may produce unusual results. In particular, lo-res graphics cannot be used with the Filter in its graphics mode. If you wish to mix text and graphics (high resolution only), it is suggested that you first try the examples discussed in the section headed "Mixing Text and Graphics", then copy the methods used in those examples.

GETTING STARTED

Now it is time to sit down at the computer and begin learning how to use the Keyboard Filter. (If you have not already installed your Keyboard Filter, please refer to the section on "Installation".)

First turn the power on and hit the "RESET" key, then (Control/B) and "RETURN". (If you have the new "Auto Start ROM" monitor chip, or if your system is an Apple II Plus, just insert the Keyboard Filter demo disk and turn on the power.) As you probably know, this will put the system into the Basic operating language (the following examples will work with Integer Basic or Applesoft II).

Now, assuming that your ROMPLUS+ is in slot #3, type:

PR#3 (RETURN)

Next, assuming that your Keyboard Filter is in socket #1, type:

(Control/Shift/M)1A (RETURN)

The Keyboard Filter will now sign on with a non-blinking cursor in the screen's upper left-hand corner. If you enter a (Control/X) at this point, you will avoid getting a SYNTAX ERROR, as mentioned above.

NOTE: The SYNTAX ERROR message is due to the fact that the ROMPLUS+ firmware is passing the (Control/Shift/M) through to Basic. Another way to avoid the error is to type a backspace or two.

Since you have used the "A" initialization option, you are now displaying text on your Apple's high resolution graphics page #1.

NOTE: The non-blinking cursor indicates that you are in the graphics mode. Also, the "color killer" circuit which normally eliminates color from the Apple's text display on all but the oldest Apples, will no longer be functioning, since you are actually in a graphics mode. If you are going to do extensive non-graphics work on a color TV, it is suggested that you simply turn off the color at the set.

Since the Keyboard Filter "comes up" with the shift lock ON, any letters which you now type will be displayed in upper case. To turn off the Shift Lock, enter a (Control/L) (L for Lock). Now you are in the normal upper and lower case mode. For a single capital letter, first hit the ESCAPE key, and then the letter you want capitalized. For a shift lock, enter a (Control/L) (hold down the CONTROL and "L" keys simultaneously). A second (Control/L) will turn off the shift lock. The Filter comes up in the Shift Lock mode because the user will frequently wish to start out with a Basic or DOS command (such as CATALOG), and such commands must be entered in upper case.

NOTE: As mentioned above, your first carriage return after initialization will result in a SYNTAX ERROR unless you first enter a (Control/X).

Now we will take you through a review of the Keyboard Filter features, using each of them in the immediate mode. Later you will see how to use these same features in the deferred mode (i.e., within a program).

The mode of operation which you are now in will simulate the normal Apple text mode in most ways, including text windows, scrolling, etc. Basic programs may be written, loaded or run. However, the system will only accept capital letters as Basic or DOS commands. Use of lower case letters in commands will result in a SYNTAX ERROR. Lower case letters may, however, be used within print statements.

The only deviations from normal operation involve certain display commands, as listed below:

1. The TEXT command will put the computer into the text mode, with the Keyboard Filter still operating. Any lower case letters on the screen when the TEXT command is entered will be displayed as punctuation marks or numerals. Typing POKE -16304,0 will return you to the graphics mode without clearing the screen. (You may need to hit (Control/P) once or twice in order to switch back to your original display page.)

2. CALL -936 and the Applesoft HOME command will home the cursor, but will not clear the display. They will, however, clear the regular test screen, which although it is not being displayed, is being used by the Keyboard Filter. The result of this is that any text on the screen when one of these commands is entered will be erased, rather than be copied, by running the cursor over it. To clear the screen in the immediate mode with the Keyboard Filter, enter a (Control/Z). To clear the screen from within a program, use the command CALL -13376.

3. FLASH & INVERSE will not function properly and may cause the display to become garbled. Use (Control/I) to get inverse video (see below). The flash feature is not available with the Filter in the graphics mode.

UPPER AND LOWER CASE LETTERS

First let's play with the upper/lower case features. If the letters you type are displayed as capitals, then you are in the shift lock mode. This mode may be toggled (turned on and off) with a (Control/L) (L for Lock). If you are not in the shift lock mode (letters you enter are displayed as lower case), you may get a single capital letter by preceding the letter with the ESCAPE (ESC) key. Later on we will describe a simple modification for your Apple which will enable you to use the regular keyboard shift key as on a standard typewriter.

COLORED AND INVERSE TEXT

Next type a (Control/T) (T for Tint) and the number 1. Any characters you now type will be displayed in purple. (Control/T) with a number from 0 to 4 will change the color of any text displayed following the (Control/T), until the next (Control/T) command. The colors displayed (on most TV's) will be:

(Control/T) 0 = White
(Control/T) 1 = Purple
(Control/T) 2 = Green
(Control/T) 3 = Blue
(Control/T) 4 = Orange

Now type a (Control/I) (I for Inverse). This command toggles the Keyboard Filter's inverse mode. Whatever you type or display now will be displayed in reverse video, with the background color depending on the last (Control/T) setting. A new (Control/T) will change the background color.

If you enter an illegal color number, the Filter will beep at you and ignore the (Control/T).

Both (Control/T) and (Control/I) may be used within a program. The procedure for doing this will be explained below.

NOTE: Early Apple II's (generally those with serial numbers below 5000) have only 2 high resolution colors. In those systems, color #1 will be the same as color #3, and color #2 the same as color #4.

NOTE: The less-than-wonderful quality of the colored letters is a direct and unavoidable result of limitations inherent in the Apple II's high resolution graphics. Even so, they are usually quite readable, and will generally look better if double spaced.

CURSOR MOVEMENT

One of the features of the Keyboard Filter is improved cursor movement. The new cursor controls have been designed to simulate those of the new Apple monitor chip, in order to keep unavoidable confusion to a minimum. The new monitor uses the ESCAPE (ESC) key to initiate cursor movement. Since the Keyboard Filter requires the ESC key for a shift key, we have used (Control/E)(E for Escape) to initiate the Keyboard Filter's cursor moves. Once a (Control/E) has been entered (no RETURN is necessary), the following keys will result in the listed cursor moves:

I = UP
M = DOWN
J = LEFT
K = RIGHT

Note that the direction of movement was chosen to match the keys' relative positions on the keyboard:

```
  I
J   K
  M
```

These are the same keys used for these movements by the new Apple monitor.

NOTES:

1. These cursor movements are like the Apple's (Control/A,B,C or D) cursor moves in that they do NOT copy text from the screen.

2. The REPEAT key will work with these commands.

3. While the system is in the cursor movement mode, any combination or number of cursor moves may be made, with or without the use of the REPEAT key.

4. Hitting any key other than I,J,K or M will end the cursor movement mode. The key which ends the cursor movement mode is also entered and displayed normally. (Incidentally, this is NOT the case with the new Apple monitor, which loses the first non-move key entered.) The keys A,B,C or D will produce the Apple II's normal cursor movement, just as if the ESCAPE key and the letter key was pressed. These four keys will also terminate the cursor movement mode.

Try some of these cursor moves now. Practice using the REPEAT key to "fly" the cursor around the screen. Remember:

* (Control/E) begins the cursor movement mode
* I,J,K, and M move the cursor
* The L key will have no effect, i.e., it will neither move the cursor, nor turn off the cursor movement mode
* Any other key ends the cursor movement mode

SCREEN COPYING

One of the Apple II's special features is the ability to copy or read text off of the screen into the computer's memory, simply by moving the cursor over the text you wish to copy, making changes, insertions or deletions as you go. This feature is known as screen copying, and it is particularly handy when writing or editing programs. The improved cursor controls discussed in the previous section will make screen copying much more convenient. In addition, the (Control/W) command may be used to instantly copy from the current cursor position to the end of the current line. This can be particularly helpful when editing or copying long program lines. It has the same effect as holding down the right arrow and repeat keys until the cursor reaches the end of the line.

If you wish to edit a line with strings (expressions within quotation marks) in it, first enter:

POKE 33,33 (RETURN),

Then list the line or lines you want to work on. This command will narrow the width of the scrolling window, making it easier to copy lines without including a lot of extra spaces in your strings. At any time when you wish to copy from the current cursor position to the end of the cursor's current line, just enter a (Control/W). The cursor will immediately move to the end of the line. To continue copying onto the next line, hit the "→" (right arrow) key or the space key once (do NOT hit RETURN!). Now you will be at the beginning of the next line. To copy this entire line, just enter another (Control/W). Do not hit "RETURN" until you are finished entering or copying your entire program line.

When you are finished editing, type: POKE 33,40 (RETURN) to re-set the width of the scrolling window.

If the line or lines you are editing do not include strings, it is not necessary to change the width of the window. In that case, the POKE 33,33 and POKE 33,40 commands are not necessary.

NOTE: Entering these pokes is an ideal application for Keyboard Macros. See below.

PAGE SWITCHING & CLEARING

The Apple II computer supports two separate "pages" of high resolution graphics (see memory map in Applesoft II manual). However, the availability of these graphics screens for use in your Apple is dependent upon your memory size and usage. The following rules will help you to determine which of the graphics pages are available for use in your system.

1. Only graphics page one is available in systems with less than 24K of RAM.

2. When using RAM Applesoft page one is not available.

3. When using the DOS, page two is only available in 48K systems.

NOTE: Graphics page 1 cannot be used with RAM Applesoft; that is, Applesoft and page 1 can be used together only if you have the Applesoft Firmware (ROM) Card. Since the Keyboard Filter normally "comes up" on page 1 (otherwise it would not work with systems smaller than 32K), and since RAM Applesoft uses the same address space as page 1, a special procedure must be used to run the Keyboard Filter with RAM Applesoft. See the section headed "Using the Keyboard Filter with RAM Applesoft".

If you are using your Apple in a configuration which does have both pages of high resolution graphics available, the Keyboard Filter will make it much easier to use this feature. The command (Control/P) (P for Page) will switch from the current page to the alternate page. This feature is particularly handy for keeping notes available for quick reference.

For instance, list a program segment, or if you have a disk drive, catalog your current disk.

Now enter a (Control/P) to switch pages. If it is the first time you have accessed this page it will probably be filled with a pattern of random symbols. Entering a (Control/Z) (Z for Zap) will clear the page. You may now enter, list or run anything you wish (so long as nothing you do switches the display page) without disturbing the information which you left on the other page. To refer back to that information, simply enter another (Control/P). Unless you enter a graphics command, what you do on one screen will have no effect on the other.

NOTE: (Control/Z) should not be used from within a program, since it will leave an extra cursor in the HOME position. Instead, use CALL -13376. BUT : CALL -13376 MUST be the last command on a line, otherwise, it will not work properly.

NOTE: Switching to a page not available in your current system configuration will probably "hang" the system. In that case, reset and start over.

NOTE: The commands (Control/Z) (immediate mode) and CALL -13376 (from within a program) replace the CALL -936 and HOME commands when using the Keyboard Filter. (Control/Z) will clear the current scrolling window and move the cursor to the upper left hand corner of the window.

NOTE: If you are using RAM Applesoft, you must remember not to switch to page 1, since this might "bomb" Applesoft. See section headed "Using the Keyboard Filter with RAM Applesoft."

NOTE: Page switching can confuse screen copying. If you write text on both pages, the Filter will not know which text to copy. The remedy is to clear one of the pages before doing program editing or anything else which may require screen copying.

STOPLIST and ENDLIST

StopList and EndList are two of the most useful features of the Keyboard Filter, particularly when working with Integer Basic.

To see how StopList and EndList work, load a long Applesoft or Integer Basic program into the computer, then list the program normally. While the list is running, enter a (Control/S) (S for StopList) from the keyboard. The listing will stop at the end of the current line. Now hit any other key, and the listing will continue until you hit another (Control/S). This is the StopList feature.

Once you have stopped the listing, if you do not wish to continue listing the program, simply enter a (Control/C) instead of re-starting the listing. The (Control/C) will end the listing and return you to the command mode. This is the EndList feature.

NOTE: The StopList and EndList commands will function at any time when the Apple is writing to the screen, not just during listings. Thus StopList is equivalent to the normal usage of (Control/C) and CONtinue when a Basic program is writing to the screen. It is also particularly useful when cataloging a disk with more than one screen of programs. The StopList and EndList sequence (Control/S, Control/C) will stop the catalog and return you to the command mode.

NOTE: The StopList feature is identical to the similar feature on the new Apple II Plus monitor. EndList is a Keyboard Filter exclusive.

NOTE: The EndList feature assumes that you are using the disk system (DOS). If you are not using DOS, enter the following pokes to make EndList work properly:

```
POKE 976,76
POKE 977,3
POKE 978,224
```

NOTE: Entry of these pokes is an ideal application for Keyboard Macros. See below.

NOTE: StopList and EndList will continue to work when the system is in the RAW mode (see below).

THE "RAW" MODE

As we have seen, the Keyboard Filter makes extensive use of the Apple's control characters for its function switching (see list in Appendix or on Quick Reference Card). As a result, in many applications the Apple or your own programs may require the use of the same control characters which the Keyboard Filter uses. Also, there are times when you will want to imbed Keyboard Filter commands within program lines where they can be interpreted in the deferred mode (when the program is run), rather than having them executed immediately as they are typed in.

In order to accommodate these situations, the Keyboard Filter includes a "raw" mode, in which only three control characters (the Control/R which toggles the Raw Mode, the Control/S for StopList, and Control/C for EndList) are interpreted by the Keyboard Filter. All other control characters are passed through for normal use by the Apple.

The Raw Mode is particularly helpful when writing programs to make use of the various features of the Keyboard Filter. For instance, if you want a program to print a line in colored letters, you may use the following procedure:

1. Type a (Control/R) to enter the Raw Mode.

2. Enter your program's print commands, including the Keyboard Filter commands such as (Control/T) for color switching, (Control/F) for font switching, or (Control/I) for inverse switching. Depending on which font you are using to write your program, you may or may not see the control characters displayed, but as long as they are within quotation marks, they will be included in the program.

3. Be sure to include the commands to re-set the system to normal white characters at the end of your print statement. Otherwise, any further display will be in the last color set by the program.

NOTE: When entering a line in this matter you will NOT see the colored or inverted letter, etc. which you are calling for. This is because the Keyboard Filter is not interpreting them as you type them in with the system in the Raw Mode. However, when you RUN the program with the system NOT in the Raw Mode, the Keyboard Filter will interpret your commands properly. Of course, this will not be the case if you run the program with the Raw Mode turned on.

NOTE: If the system is NOT in the Raw Mode when you enter the program line, you will see the colored, inverted, etc., letters as you enter them, but the control characters will not be stored in the program line, and thus will not switch the system when the program is run.

NOTE: This same procedure will work for putting colored letters, etc., within a program name on a disk catalog or within a Keyboard Macro (see below). As mentioned above, the system will stay in the last called for mode when the catalog has been finished or the Macro printed, and the mode switching commands will only function when the system is NOT in the Raw Mode.

"MTAB" = "MACRO TABLE

KEYBOARD MACROS

Keyboard Macros are one of the Keyboard Filter's most innovative features, providing a capability previously found only in computers far more expensive than the Apple II. Essentially, Keyboard Macros allow you to create your own system of "shorthand" for your computer. A single keyboard macro consists of a string of up to 63 characters. These may include control characters as well as regular alpha-numeric characters. Macros may include single or multi-line Basic or disk commands. Each macro is assigned to a specific symbol on the Apple's keyboard. It can then be automatically typed in its entirety simply by entering a (Control/S) (S for String) and then the proper key for that macro. A total of up to 60 macros are defined using a simple program called Macro Editor, which you will find on the program disk included with the Keyboard Filter. Macro tables may be stored and loaded by name from a disk. The Macro Editor program (see below) will suffix your table name with ".MTAB". You can create multiple ".MTAB's" for different applications although only one may be in use at any one time. Macros can be created to make full use of the Keyboard Filter's colored and inverse letters, lower case letters, and multiple user-defined character sets. (However, you should not attempt to imbed one macro inside another, since this can make the Filter very, very confused.)

NOTE: The (Control/S) command is used by the Keyboard Filter for both StopList and Keyboard Macros, however this does not create a problem. The Keyboard Filter "knows" to interpret the (Control/S) as a StopList command only when the computer is printing something onto the screen, and is ignoring keyboard input.

To see how the Keyboard Macros work, turn on the Keyboard Filter, then load the sample macro table included on the demo disk. To load the table, simply type:

BLOAD SAMPLE.MTAB (RETURN)

NOTE: Remember that all Basic and DOS commands must be entered in UPPER CASE letters! (Otherwise, you will get a SYNTAX ERROR.)

Now you have a number of Keyboard Macros at your disposal. To see a list of the macros available, type the command U from the Macro Editor. This will display the used and unused macro commands.

NOTE: It is suggested that you try these macros with the graphics mode turned on, since some of them use lower case and colored letters. Also, those macros which include DOS (disk) commands will return a SYNTAX ERROR if the DOS is not loaded.

NOTE: To use a Keyboard Macro to enter a line in a Basic program, first enter the line number, then, without hitting return, enter a (Control/S) and the proper key for the macro you wish to use. This will enter the macro and display it as though it had been typed from the keyboard. If the macro does not include a RETURN, you will have to enter one manually in order to enter the finished line into your program.

NOTE: Macro tables will often have to be protected by setting LOMEM. See Appendix.

NOTE: The (Control/S) command should NEVER be used from within a program.

DOES NOT
WORK IF > ⇒ F.P.

DEFINING MACROS

Keyboard Macros are usually defined through the use of the Macro Editor program included with the Keyboard Filter. (To see how macros can be defined without using Macro Editor, see Advanced Programming Techniques in the Appendix.) Incidentally, Macro Editor is a good example of a program which uses features of the Keyboard Filter to create a unique display.

The Macro Editor program should be reasonably self-explanatory. If you wish to create a new macro table (or .MTAB), begin by using the command for clearing the table. If you already have a table which you wish to add to, first use the Clear Table command, then load your table using the Read command. To create new macros, use the Define command. If you wish to use inverse, color or font switching, or to include control characters within a macro, enter a (Control/R) to turn on the Raw Mode (see above). Then enter your macro, complete with control characters and Keyboard Filter commands. If you wish your macro to include a carriage return, enter a slash mark (/) for the return. If you need a slash mark printed in the macro, enter two of them together (//). Only one of them will be printed, and you will not get a RETURN.

NOTE: Be sure to re-set the colors, etc., to normal within the macro - otherwise, whatever is displayed following the macro will remain in the mode set within the macro.

NOTE: Macro Editor interprets upper and lower case characters as the same letter. Therefore you cannot assign one macro to a capital letter and a second to the corresponding lower case letter. However, where two symbols are displayed on the same key (N, P, numerals, etc.), the shifted character may be defined separately from the normal (non-shifted) character.

The Macro Editor program will warn you if you try to define a macro to a key which already has been defined. It will also warn you by ringing the bell if you try to define a macro longer than the maximum of 63 characters. When you have defined all of the macros for your table, enter a RETURN for the next macro, and the program will return you to the command menu. You may now save the macro table to a disk for future reference, or simply exit the program. In either case, the table which you have defined is now loaded and ready for use.

The Macro Editor program also has commands which will list an individual macro or all of the keys to which macros have been assigned, and to display the size of the current macro table.

OVERSTRIKE MODE

If a character is typed or printed on the Apple when the cursor is positioned over an existing character, normally the pre-existing character is erased and replaced by the new character. With the Keyboard Filter's Overstrike Mode, the new character will be superimposed over the pre-existing one. The main applications of this feature will be in printing foreign languages and in underlining. By defining a special character set which includes the various punctuation which you need, you can use the Overstrike Mode to superimpose punctuation marks over other letters. The Overstrike Mode is toggled by (Control/O)(O for Overstrike).

NOTE: If you use the Apple's screen copying with characters which have been printed using the Overstrike Mode, only the last character printed on any one space will be read by the cursor. Therefore, if you anticipate using screen copying, you will probably want to print the punctuation mark first, so that the base letter will be read by the cursor.

NOTE: The Overstrike mode may be used with multiple fonts. See Special Graphics and User Defined Characters, below.

MIXING TEXT AND GRAPHICS

As mentioned above, normal operation of the Keyboard Filter makes use of the Apple's high resolution graphics display. This gives the added benefit of being able to mix text and graphics anywhere on the high resolution screen, instead of being limited to four lines of text at the bottom of the screen. (Obviously there is no way to mix text with low resolution graphics, since there is no way to mix the high and low resolution graphics modes.)

In order to write a program to mix text and graphics, follow the simple procedure listed below:

1. The Keyboard Filter must be initialized when the program is run, so initialize it either manually or within your program.

2. Turn on high resolution graphics as you normally would. For instance, from Applesoft, enter HGR or HGR2. (Don't use HGR if you are using RAM Applesoft!) Poke -16302,0 will switch the system to full screen graphics, so that you will not have four lines of normal (non Keyboard Filter) text at the bottom of the screen.

3. Write your program to draw your graphics as you would normally.

4. Use the TAB, VTAB and HTAB commands to move the cursor to the locations where you wish to print text.

5. Use regular print statements to print your text.

6. Remember - although you are in a graphics mode, you are simulating the text mode. If you print text on the bottom line without ending the print statement with a semi-colon, the entire screen (graphics, text and all!) will scroll up one line.

NOTE: To use lo-resolution graphics from the Filter, use the regular GR command and write your graphics as you would normally. HOWEVER - when you wish to return to displaying text - DO NOT USE THE TEXT COMMAND! Instead, simply POKE -16297,0. Of course, this only applies when using the Filter in its graphics mode. When using the Filter's text mode, the normal TEXT command will work properly, i.e., it will not interfere with the Filter.

For a complete example of using mixed text and graphics, see the "Mixed Text and Graphics" demo on the disk supplied with the Keyboard Filter.

SPECIAL GRAPHICS AND USER DEFINED CHARACTERS

The Keyboard Filter supports the use of multiple user defined character sets. These character sets are useful in many applications, including foreign languages, fancy, sideways or upside down lettering, and for animation. Using the Font Editor program supplied with the Keyboard Filter (see below), users can easily define their own character sets or even make the Apple simulate the character graphics systems used by several other personal computers. There are also several sample fonts on the demo disk supplied with the Keyboard Filter. These fonts show how special characters might be used for upside down or backwards lettering. They also include several special graphics characters, such as playing card and mathematics symbols.

To use one of these fonts, load it into RAM by typing "BLOAD" and the name of the font. For example,

```
BLOAD MIRROR.FONT
```

will load the backwards font as font number one. (Font #0 is the font on the Keyboard Filter chip.) To load a second RAM font, type:

```
BLOAD UPSIDE DOWN.FONT,A$6400
```

This will load the upside down font as font number two. The A$6400 tells the computer to load the font at Hexidecimal address 6400. This is where font #2 will normally be loaded. No address is necessary for font #1, since all of the fonts will default to loading at 6000 (hex), and so forth up to your maximum RAM capacity.

NOTE: For information on changing the default addresses for these fonts, see Appendix. The default address MUST be changed for use in systems with less than 32K of RAM, or 32K systems with the DOS.

To display characters from a RAM font, enter a (Control/F) (F for Font) and the number of the font you wish to use. Anything typed or printed now will be displayed in the new font. To see the special graphics characters included in the font, go into the Raw Mode (Control/R) and try the various control characters from (Control/N) through (Control/Z) except, of course, for Control/R, which will turn the Raw Mode back off. These characters are "real" to the computer, and may be copied by running the cursor over them, just as regular letters and numbers are. Font switching and special graphics characters can both be used in Keyboard Macros using the procedures described under "Raw Mode" and "Keyboard Macros".

NOTE: There was no room for special graphics characters on the Keyboard Filter chip, so the above will not work with font #0.

NOTE: Special graphics characters in the RAM fonts are assigned to ASCII codes from 14 through 31 (decimal). When working in Applesoft, the "CHR$" function may be used to print these characters. For instance, "PRINT CHR$(14)" will print the special graphics character assigned to ASCII code 14 (decimal).

To create animation using the Keyboard Filter, simply create characters which, when combined, will draw your various scenes and figures, then use print statements to do the actual drawing on the screen. Remember when creating a moving scene to erase the previously drawn position of anything moving. Do this by printing blanks at the appropriate positions or by re-drawing your background scene. Also remember that in many cases you will have adequate memory to store more than one alternate character set. The (Control/F) command may then be used to switch between fonts, even in the middle of your animation. Studying the Animated Catalog demo included with the Keyboard Filter will prove helpful in learning these techniques.

USING THE FONT EDITOR PROGRAM

The Font Editor program (which may be found on the demo disk) provides an easy way to define your own character sets. The program is reasonably self explanatory.

Before using the program, you may wish to sketch the characters you wish to enter, remembering that they must be drawn within a 7 (horizontal by 8 (vertical) dot matrix. Graph paper laid out in 7 x 8 blocks will be useful for this task.

There is one problem which you must keep in mind when creating your characters. Due to a peculiarity inherent in the Apple's high resolution graphics system, if you plan to use your font in color, all vertical lines in each character must be on either odd or even vertical lines of the matrix. Otherwise, portions of the character you define may disappear when it is displayed in color. (This is not a problem when the characters are displayed in normal white on black or the inverted black on white.) The Font Editor program will help you to keep track of this by displaying alternate vertical columns in different colors. The program also gives you the opportunity to set whether the character will be displayed beginning on an odd or an even line. The program keeps track of the chosen orientation for each letter by setting or clearing the high order bit (otherwise unused by the Filter) of each character's ASCII CODE. (If you don't know what all of that means, don't worry about it.) More on using this feature in a moment.

Run the Font Editor. This program (which is a good example of a program written for use with the Keyboard Filter) will begin by initializing the Filter in the graphics mode and displaying a menu of command options. Begin with the Read command and load a font, such as MIRROR, UPSIDE DOWN, or COLOR. You may now use the Font command to display the entire font, or the Display command to display a single character. The Display

mode will show the chosen character in all of the colors available.

To edit or change a character, enter the Edit mode.

NOTE: COLOR.FONT is identical to the font included on the Keyboard Filter chip, except that it includes a number of special characters which could not be included on the chip due to space limitations.

NOTE: The Font Editor program actually consists of three programs which are Chained together. The first program includes the Slot Finder routines which initialize the Keyboard Filter. The second (called FEDIT) includes all of the command modes except for the Edit mode, which makes up the third portion of the program (called CEDIT). You should always start by RUNning FONT EDITOR. Remember, YOU MUST LEAVE THE PROGRAM DISK IN THE DRIVE SO THAT THE THREE PORTIONS OF THE PROGRAM CAN CALL EACH OTHER WHEN NECESSARY. Otherwise, the program will crash when it does not find its other parts. If your font is on another disk, put that disk in just for the Read command, then replace the program disk. When you are finished with your font, you may put another disk into the drive and Save the font onto that disk. At all other times you should leave the program disk in the drive.

Now that you are in the Edit mode, follow the directions on your screen and choose a letter to edit (don't worry about messing up the font - we can restore the character to its original shape after we play around with it, and of course we will not affect the font on the disk unless we Save a modified font with the same name). The computer will now switch to low resolution graphics and display an enlarged representation of the character you have chosen in a 7 x 8

grid. A row of letters at the bottom of the screen lists the available commands. The list is shown below, or you can display it on your screen by entering the "?" command.

```
? = PRINT LIST OF COMMANDS
Q = QUIT - RETURN TO MAIN MENU
N = NEW - CHOOSE ANOTHER CHARACTER TO EDIT
S = SHIFT CHARACTER UP, DOWN, LEFT OR RIGHT
R = DELETE A HORIZONTAL ROW (0 = TOP, 7 = BOTTOM)
C = DELETE A VERTICAL COLUMN (0 = LEFT-MOST, 6 = RIGHT)
P = ENTER PLOT MODE
G = ENTER MOVE (GO) MODE
E = ENTER ERASE MODE
H = SET HIGH ORDER BIT FOR COLOR FONTS (SEE ABOVE)
B = BLANK OUT (CLEAR) THIS LETTER
U = UNDO - RESTORE THIS LETTER TO ITS ORIGINAL STATE
```

The Plot mode will move the cursor up, down, left or right, leaving a point defined as part of your character at each position which it enters. The Plot mode uses the same I,J, K and M cursor pad which the cursor movement mode uses. The Erase mode moves the cursor in the same fashion, but erases any previously plotted points which it passes over. Finally, the Go mode allows you to move the cursor anywhere within the grid without affecting any of the points which it passes.

Delete a horizontal row (the R command), will delete the row specified by the number you type. The top row is 0 and the bottom row is 7. When you delete a row, all higher numbered rows move up one row. Likewise, when you delete a column (the C command), you specify a number to indicate which column is deleted. The left-most column is 0 and the right-most column is number 6. When you delete the column, all higher numbered columns move left one column.

The rest of the commands should be self explanatory. Again, remember that you will not change the font on the disk unless you save the font you are editing using the same name that it was saved under previously.

If you Quit the Edit mode and return to the main menu, once again you will find that most of the commands are self explanatory. Perhaps the one exception is the Move command. This command allows you to change the letter to which a character you have defined is associated. For instance, if you Move the letter "T" to the number "8", then every time you typed or displayed an "8" (in this font), the letter "T" (or whatever character had been associated with the letter "T") will be displayed instead of an "8".

USING THE KEYBOARD FILTER WITH RAM APPLESOFT

The Keyboard Filter was designed to "come up" on high resolution Page 1 when it is initialized in the graphics mode. This was necessary so that the system would work in computers with less than 48K of RAM memory. However, the RAM versions (both tape and disk) of Applesoft use the same memory area as Page 1. Therefore, a way must be provided to use the Keyboard Filter so that it will not use Page 1 and interfere with RAM Applesoft.

The following procedure will allow you to use all of the features of the Keyboard Filter in the graphics mode with RAM Applesoft - AS LONG AS YOU DO NOT SWITCH TO PAGE 1! If you do, you will probably have to re-load Applesoft. (Although Applesoft may SEEM to work if you immediately switch back to Page 2, it will probably bomb eventually - in most cases only after you have spent a lot of time typing in a program but have not yet saved your work onto disk or tape!)

NOTE: If you accidentally switch to Page 1, DON'T TYPE ANYTHING! Immediately enter a (Control/P) to switch back to Page 2. With a bit of luck, you will not have any problems.

NOTE: If you are going to use the Keyboard Filter in the text mode (initialization option C), then no special procedures are required. Applesoft may be loaded before or after initialization of the Keyboard Filter.

1. To use RAM Applesoft with the Keyboard Filter's graphics mode, first initialize the Keyboard Filter, as described above, using the "A" initialization option.

2. Use the (Control/P) command to switch the system to Page 2.

3. Now LOAD Applesoft from tape or use the FP command to load from a disk.

4. The screen will be covered with a random display at this point. Type POKE -16304,0 to clear the screen. (Careful! You won't be able to see what you are typing until the screen has been cleared! If you get a bell and the screen does not clear, you have made a typing mistake. Try the POKE -16304,0 again.)

5. Applesoft Basic may now be used with the Keyboard Filter as long as you do not switch to Page 1.

NOTE: If you are using the Keyboard Filter in a system with a disk drive and less than 36K of RAM (for all intents and purposes, if your system does not have 48K of RAM), then you cannot use the Filter's graphics mode with RAM Applesoft. The reason is that RAM Applesoft uses the Page 1 address space and the DOS takes up Page 2.

INPUT & OUTPUT TO PERIPHERALS

Input from and output to peripherals is normally handled by the Apple through use of the PR#(SLOT) and IN#(SLOT) commands. Unfortunately, these commands turn off any previously addressed slots, so that normally only one peripheral board can be activated at any one time. Thus the command PR#(SLOT), when used to turn on a printer, would turn off the Keyboard Filter.

In order to get around this problem, the Keyboard Filter was designed with its own input and output handling capabilities. The Filter manages its own set of input and output switches or "hooks", similar to those used by the DOS.

Once the Keyboard Filter has been initialized, output to a peripheral is switched on by using a (Control/Q)(SLOT). For example, to turn on an Apple communications or parallel interface card in slot #1, enter the command:

(Control/Q)1.

Output to the printer can be turned off with the normal "PR#0" command.

Similarly, input to the Apple with the Keyboard Filter turned on is controlled by a (Control/K)(SLOT) command. To switch input from the keyboard (as it normally is with the Apple) to a device connected to slot #1, enter the command:

(Control/K)1.

Input can then be switched back to the keyboard by using the normal "IN#0" command.

NOTE: Both the (Control/Q)(output) and the (Control/K) (input) commands may be used in either the immediate mode or from within a program. To use them within a program, have the program PRINT "(Control/K)(slot)", and so forth.

NOTE: These routines will NOT work with peripheral cards which use the $C800 ROM address space, such as the Hi-Speed Serial Card, Heuristics Speechlab, or Mountain Hardware Real Time Clock.

Please see the section on conflicts with other $C800 chips, Page 57.

NOTE: If your disk drive is installed in slot #6, a (Control/K) 6 command to the Filter will boot your disk.

USING THE "SLOT FINDER" PROGRAM

Included on the demo disk supplied with the Keyboard Filter is a text file called SLOT FINDER. This file is set up as a subroutine to be appended to your own programs. As part of your program, it will search for a ROMPLUS+ card and Keyboard Filter chip. If it finds them, it will pass the number of the slot in which the ROMPLUS+ board is installed back to your main program as the variable SLOT.

To add SLOT FINDER to your program, first load your program into RAM. Next you must be sure that none of your line numbers will conflict with SLOT FINDER, which uses lines 11,000 through 11,080. Finally, with the demo disk in your disk drive, type:

```
EXEC SLOT FINDER.
```

Your disk drive will come on for a few moments, and depending on whether you are in the MON or NOMON mode of the DOS, you will either see the lines being entered into your program or a series of BASIC prompt signs. When you get your cursor back, list the program. You will find that SLOT FINDER is now a part of the listing. The final step in setting up your program for SLOT FINDER is to add a line near the beginning of your program to GOSUB 11000. You may now save your program with SLOT FINDER for later use.

SLOT FINDER was designed to work with either Integer Basic or Applesoft II. In addition to SLOT, the subroutine will also affect the Integer Basic variables SADDRESS, S1 and S2. In Applesoft, the affected variables will be SL, SA, S1 and S2.

Following is a complete listing of SLOT FINDER. If your program already uses line numbers from 11,000 to 11,080 you may re-number these lines and type them in anywhere in your program (usually at the end of your listing). No changes are

necessary to use the subroutine with different line numbers. However, you will have to change the GOSUB command to reflect the new starting line number of the subroutine.

SLOT FINDER

```
11000 REM FIND OUT THE SLOT OF THE ROM CARD
11010 FOR SLOT = 1 TO 7
11020 SADDRESS = -16384 + 256 * SLOT
11030 S1 = PEEK (SADDRESS + 95)
11040 S2 = PEEK (SADDRESS + 96)
11050 IF S1 = 157 AND S2 = 240 THEN RETURN
11060 NEXT SLOT
11070 PRINT "SORRY...I CAN'T FIND A ROM CARD"
11080 POP:END
```

APPENDIX

APPENDIX I - MEMORY MAP

A Note About Mountain Hardware's ROMPLUS+

In its first incarnation, the Keyboard Filter was simply a 2K ROM chip designed for one of the Apple's two spare ROM sockets. The chief attraction of this approach was its low cost, but it soon became apparent that it had many problems. For one thing, ROM Applesoft uses that address space. Also, empty sockets create vacuums that yearn to be filled. Significant new software systems, Pascal for example, uses the $D000-$DFFF address space (the addresses assigned to the empty ROM sockets). Another consideration is that the Apple DOS's I/O structure makes any program that connects through the KSW and CSW I/O switches want to sit on a peripheral card. For these reasons we decided to design a special peripheral card to hold the Keyboard Filter, and to have the Filter use the $C800-$CFFF address space. While this does add to its expense, it is necessary to keep address space conflicts to a minimum and to ensure that the Filter will work with future systems software (e.g., new versions of the DOS). In addition, Mountain Hardware's ROMPLUS+ can hold up to six 2K ROM chips, and its on-board RAM and TTL inputs make it a very useful device in its own right.

The Keyboard Filter uses no RAM Locations except for the High Resolution screens, optional font and macro tables and the local RAM on ROMPLUS+. A small zero page scratch area is used, but the filter is designed to always save and restore the contents of these locations.

The following is a complete memory map of the Keyboard Filter:

ADDRESS	FUNCTION
$0-$9	Zero page scratch area. These locations are never overwritten - you can treat the system as though they are not used by the Filter.
$800-???	Default address of the Keyboard Macro table. The size of a typical table is about 600 bytes.
$2000-$3FFF	Hi-Res graphics, Page 1
$4000-$5FFF	Hi-Res graphics, Page 2
$6000-???	Default address of the font table. Each font is 1K bytes long. You may have as many as you want (limited only by memory size).
$C800-$CCFF	The code for the Keyboard Filter
$CD00-$CFFE	Font 0, the on-chip font
$CF00-$CF15	Variables used by the filter. Note that they share some of the same address space as Font 0.
$CF16-$CF55	The 64-byte keyboard input buffer

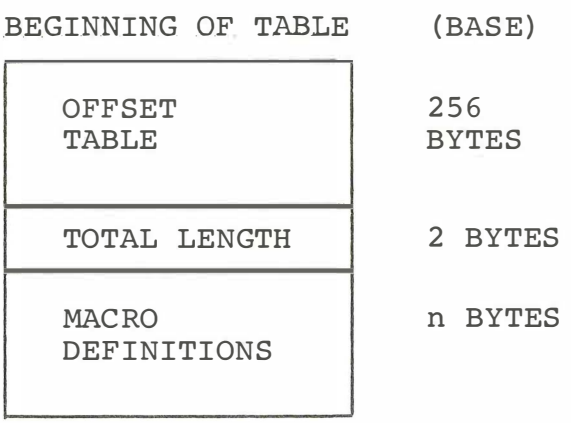
NOTE: Both Macro and Font tables are completely relocatable. They contain no address sensitive information and may be located practically anywhere. Only their default addresses are given here.

APPENDIX II - THE MACRO TABLE FORMAT

The current set of keyboard macros is defined by a relocatable data structure called, not very surprisingly, the keyboard macro table. Usually, you will define and alter macro tables by using a program called the Macro Editor. For more curious users, this section describes the internal format of the macro table.

A macro table contains no address sensitive information; it may be located anywhere in memory. It is pointed to by a 2 byte pointer at $CF0E and $CF0F. This pointer defaults to $800, but may be changed to point wherever you like.

A macro may be associated with any of the 128 ASCII codes. The first 256 bytes of the macro table contain 128 2-byte offsets to the beginning of the actual macros. The offset for the character with ASCII value X is stored at address 2 * X higher than the beginning of the table. All of the offsets are also relative to the beginning of the table.



Following the offset table are 2 bytes indicating the total length of the table. These are used to keep track of the length for editing purposes.

After these length bytes is the beginning of the actual macro data. It is simply a string of characters with their high-order bits clear, except for the last character of a macro, which has its high-order bit set. These macros are stored sequentially. Thus to find the definition for the macro associated with the character with ASCII code X, we first look up the offset at BASE + 2 * X and BASE + 2 * X + 1. If this offset is 0, it means the macro is undefined. Otherwise, we find the actual character data beginning at BASE + OFFSET and ending with the first subsequent character which has its high order bit set.

Finally, it should be mentioned that the maximum length of a single macro is 63 bytes (characters). This limit is caused by the amount of buffer RAM on the board; it is not inherent in the format. When a macro with more than 63 characters is attempted, a bell rings to signal an error and the macro is ignored.

APPENDIX III - FONT FORMAT

The nature of the Apple's high resolution graphics display mode leads to a certain natural font format for hi-res graphics characters. If each character is defined by a 7 x 8 grid or bit matrix, you obtain 24 lines of 40 characters, an exact simulation of the Apple's text mode. Consequently, most hi-res character generators use the same basic font format, and the Keyboard Filter is no exception.

The information supplied in this section is not really necessary for a casual user. The Font Editor included on the demo disk supplied with the Keyboard Filter provides a powerful, convenient way to define fonts without knowing any of the information in this section. However, if you want to write your own font editor or are simply curious, please read on.

Each character is defined by 8 consecutive bytes, one for each horizontal line of the character. The seven low order bits define whether or not each of the seven dots in the line are on or off. The rightmost (low-order) bit maps to the leftmost dot (as usual in Apple hi-res graphics). The high-order bit is used to provide orientation information, as described below.

A font consists of 128 character definitions, one for each legal ASCII code. Thus an entire font occupies 128 * 8 = 1024 bytes. Character definitions are stored sequentially. The data for character X is stored 8 * X bytes past the beginning of the font.

Many fonts can be used at the same time by using the (Control/F) command to switch between them. The first character of the first font is pointed to by address $CF0C. It defaults to

$6000, but can easily be changed (fonts are completely relocatable). Thus if font 1 begins at $6000, font 2 starts at $6400 and font x (x=>1) starts at $6000 + X * $400.

One of the unique features of the Keyboard Filter is its ability to draw reasonable looking colored letters. The high-order bit of each byte in a character definition is used to help achieve this. A character should be defined so that all vertical lines in the character are either even or odd lines, but not a mix of the two. The high bit is used to specify whether the character wants to be even or odd. If the high bit is off, the character is even; if it is on, the character is odd. The Keyboard Filter automatically performs the appropriate shift when in color mode to preserve as much information as possible. The Font Editor program has a feature to help set this bit properly.

APPENDIX IV - ADVANCED PROGRAMMING TECHNIQUES

Changing the Default Parameters

When a first time initialization occurs, the current address of the font table is set to $6000 and the current address of the keyboard macro table is set to $800. These defaults are thought to be natural and convenient, but they will sometimes need to be changed to suit the needs of a particular application. This section describes how this is done.

The current font table address is stored at $CF0C (low byte) and $CF0D (high byte), while the current macro table address is kept a $CF0E (low byte) and $CF0F. We can change these to whatever we desire simply by poking to these addresses.

For example, to change the current address of the font table to $4000, you would enter:

```
POKE -12532,0
POKE -12531,64
```

or, from machine language:

```
LDA  #0
STA  $CF0C
LDA  #$40
STA  $CF0D
```

To change the default address of the Macro table to $2080, you would enter:

```
POKE -12530,128
POKE -12529,32
```

or

```
LDA  #$80
STA  $CF0E
LDA  #$20
STA  $CF0F
```

To make these pokes work, we must be sure that the Keyboard Filter is activated. One way to do this is simply to go through the initialization sequence. However, it can also be accomplished by direct poking. If your ROM Board is in slot number (SLOT), and the Keyboard Filter is in socket number (SOCKET), you can say:

```
POKE -16256 + 16 * SLOT, 136 + SOCKET.
```

Note that the Filter usually will be activated at the time you want to change parameters, so the above POKE normally will not be necessary. Also note that the slot of the ROM card can be found by using the "Slot Finder" program included with the system.

CONFLICTS WITH OTHER $C800 CHIPS

The Keyboard Filter occupies the address space from $C800 to $CFFF. This is nice because it does not interfere with any of the Apple's Basic ROM address space, but it does occasionally cause problems by conflicting with ROMs on other peripheral cards. This section will describe how to deal with such problems. The nature of the Apple's I/O structure demands that only one peripheral be active at any given time. However, there may be times when you would like to use the Keyboard Filter along with a modem, printer, etc. For this purpose, the Filter maintains its own set of peripheral addresses, in exactly the same fashion that the DOS does. It stores its own local output switch at $CF07 and $CF08 and its input hook at $CF09 and CF0A. Every time it prints a character it vectors through $CF07 and every time it reads a character it vectors through $CF09. For example, we can get it to output to a printer by storing the printer driver address in $CF07. Simple commands (Control/Q and Control/K) are provided for doing this in a convenient fashion so that most of the time you need not worry about how the hooks are managed.

Unfortunately, some peripheral cards also use the $C800 ROM address space. Examples include the Apple High Speed Serial Card, the Mountain Hardware Apple Clock, and the D.C. Hayes MicroModem. When control is transferred to such a card, it always disconnects all other $C800 ROMs by referencing address $CFFF. Herein lies the problem: When the Filter goes off to a card that uses a $C800 ROM, that card will reference $CFFF, causing the Filter to be disconnected, and your program will crash when it tries to return to the Filter.

Thus, the normal procedure for outputting to a printer will not work with the High Speed Serial Card, although it will work with the Communications Card and the Parallel Printer Card, which do not use the $C800 ROM space.

Currently, to use another peripheral device that uses a $C800 ROM, it will be necessary to disable the Keyboard Filter. This can be accomplished by entering "PR#(SLOT)" in the immediate mode, or PRINT D$;"PR#";SLOT from a program. (D$=CONTROL/D caracter) This will turn off the ROMPLUS+ board (and Keyboard Filter) and turn on the other peripheral in slot number SLOT. To continue use of the Keyboard Filter, it will be necessary to reactivate ROMPLUS+ ("IN#(SLOT)" or "PR#(SLOT)") and re-initialize the Keyboard Filter with the "B" or "D" option. See the section on Keyboard Filter re-initialization, Page 11.

PREVENTING MEMORY CONFLICTS: SETTING HIMEM & LOMEM

A "memory conflict" occurs when one program wants to use memory locations that are already used by another program. Since the Keyboard Filter has to be used with many other programs, one of its most important design goals was to keep memory conflicts to a minimum. Even so, such problems are unavoidable. This section lists most of the possible memory conflicts that are likely to arise and gives hints on how they can be circumvented.

The Keyboard Filter uses normal RAM for three purposes. They are:

1. High resolution screens,
2. Font tables, and
3. Macro tables.

If a memory conflict is to arise, it is because a program is using memory required by one of the above. We will discuss each of them in turn.

The Apple High Resolution Screens sit from $2000-$3FFF (page 1) and $4000-$5FFF (page 2). Page 2 is not available on a 16K system and will blow the DOS if used on a 32K system. If you are using a 16K system, a "HIMEM:8192" should be entered to protect your program from the high resolution graphics area. (Note that these problems apply to any programs using high resolution graphics; they are not unique to the Keyboard Filter.) Additionally, RAM Applesoft conflicts with high resolution page 1; see the section on RAM Applesoft for hints on overcoming this problem.

Another conflict can arise because of the font table. The font table default location is $6000, which immediately causes problems on a 32K system by conflicting with the DOS. See the

section on "Changing Default Parameters" for instructions on moving the default location. On a 32K system, we suggest moving the font table to $4000. Font table conflicts are not extremely important, as you may always rely on the on-chip font, which requires no RAM.

The most common conflicts will be those caused by the macro table, whose default location is at $800. Since $800 is the first available location after the primary text page, it is very valuable "real estate" - lots of programs want to use it. In most cases, the user will have to do something to protect it.

Integer Basic defaults to keeping its variables at $800. This can easily be changed by using a LOMEM. You should set LOMEM to 2048 + the size of the macro table after loading in the table, but before running any program. Alternatively, your program can perform the LOMEM for you. A "LOMEM:4096" should be adequate for all but the largest macro tables.

Applesoft presents a slightly more difficult problem. The default location for ROM Applesoft programs is $800; unfortunately, there is no intrinsic command to change this. It can, however, be changed by POKEing memory cells 66 and 67 with the desired program location. This MUST be done BEFORE any program is LOADED or ENTERED! For example, typing "POKE 66,0:POKE 67,16" BEFORE loading your Applesoft program will protect the keyboard macro table in exactly the same way as "LOMEM:4096" will for Integer Basic.

An even more serious problem is encountered with RAM Applesoft. Its code begins at $800, and there is no practical way to move it. Instead, you must move the macro table; this may be done using the Macro Editor program, or see the section headed "Changing Default Parameters". You will also have to do this

when using the Keyboard Filter with other programs that use $800. Note that this procedure is only necessary when keyboard macros are to be used - if no macro table is needed, then there is no problem.

ON-BOARD RAM USAGE

The Keyboard Filter uses 18 bytes of RAM to store important variables which keep track of its state and control its operation. To minimize memory conflicts, these variables are stored in the RAM on the ROMPLUS+ peripheral card. Note that these locations will be accessible only when the Filter is active. This section describes the function of each variable so that advanced programmers can manipulate the Filter at a low level.

ADDRESS CF04 - STATUS

This byte contains 6 single bit flags to keep track of the current state of the program. They are:

RAWBIT	$80 IF SET, FILTER IS IN RAW MODE (bit 7)
HIRESBIT	$50 IF SET, FILTER IS IN HIGH RESOLUTION GRAPHICS MODE (bit 6)
OSTRIKEBIT	$20 IF SET, FILTER IS IN OVERSTRIKE MODE (bit 5)
SHLOCKBIT	$08 IF SET, FILTER IS IN SHIFT LOCK MODE (bit 3)
SHMODEBIT	$04 IF SET, FILTER WILL READ SHIFT KEY FROM FROMPLUS TTL INPUT (bit 2)
LCBIT	$01 USED INTERNALLY TO HELP WITH LOWER CASE LETTERS (bit 0)

To re-set these parameters to their default values, you may POKE -12540,72 (for graphics mode) or POKE -12540,8 (for text mode).

ADDRESSES $CF05 & $CF06: NEXTIN & NEXTOUT

This is a 2-byte pointer used to manage the 64 byte input buffer. NEXTIN points to where we put the next input character while NEXTOUT points to where we get the next output character from. If they are equal, the buffer is empty.

ADDRESSES $CF07 & $CF08: OUTHOOK

These bytes hold the address of the routine the Filter goes to when outputing a character. It defaults to $FDF0. It is changed by the (Control/Q) command.

ADDRESSES $CF09 & $CF0A: INHOOK

These bytes hold the address of a routine the Filter goes to when it wants to input a character. It defaults to $FD1B. It is changed by the (Control/K) command.

ADDRESS $CF0B: NEXT

This variable is used internally to parse multi character control sequences.

ADDRESSES $CF0C & $CF0D: FONT

These bytes hold the address of the current font table. It defaults to $6000.

ADDRESSES $CF0E & CF0F: MTABLE

This is the address of the current Keyboard Macro table. Default is $800.

ADDRESS $CF10: HGR COLOR

This is a number from 0-4 specifying the current color.

0 = WHITE
1 = PURPLE
2 = GREEN
3 = BLUE
4 = ORANGE

Default for this variable is 0 (white). To set the color with a poke, use POKE -12528, (color).

ADDRESS $CF11: CURFONT

This is the number of the current font. Font 0 is the on-chip font. Font #1 is pointed to by $CF0C and $CF0D. Font #2 is 1024 bytes above font #1, and so forth. CURFONT defaults to 0. To set Curfont with a poke, use POKE -12527, (font).

ADDRESS $CF12: INVERSE

This byte is XORed with the data for each character and is used to get inverse letters. It should be 0 (normal) or 7F (inverse). You can get very strange effects by setting it to other values. To set the inverse mode with a poke, use POKE -12526, (0 or 127).

ADDRESS $CF13: HGRPAGE

This byte keeps track of the current high resolution graphics page being displayed. It should be either $20 (page 1) or $40 (page 2). Any other value will produce very strange results. HGRPAGE defaults to $20 (page 1).

ADDRESSES $CF14 & $CF15: SCRATCH

These two locations are used internally for miscellaneous functions.

ADDRESSES $CF16 & $CF55: INPUT BUFFER

A sixty-four byte buffer used for buffering the keyboard for keyboard macros, etc.

APPENDIX V - SHIFT KEY MODIFICATION

As mentioned earlier, the Keyboard Filter was designed to support a simple modification which would allow the Filter to read the Apple's shift key. With this modification in place, normal typewriter-style entry of both upper and lower case letters becomes possible. The modification is quite simple, and may be done by anyone with basic knowledge of soldering techniques. The only requirement is for a single wire to be run from an existing hole on the Apple's keyboard to a small plug which connects with Mountain Hardware's ROMPLUS+ board. The modification is non-destructive, i.e., it does not require the cutting of any traces, etc. To perform the shift key modification, simply follow the steps listed below.

NOTE: The plug and wire are available from Mountain Hardware. Order Part No. MHP-X021. Price $3.00

WARNING: DO NOT ATTEMPT THE FOLLOWING PROCEDURE UNLESS YOU HAVE SOME EXPERIENCE WITH SOLDERING TECHNIQUES. IF YOU HAVE DOUBTS ABOUT YOUR CAPABILITIES, WE SUGGEST YOU TAKE YOUR APPLE TO A QUALIFIED TECHNICIAN. THE FOLLOWING PROCEDURE MAY VOID YOUR APPLE'S WARRANTY.

1. Unplug your Apple!

2. Turn the computer over and remove the four screws at the front (below the keyboard), two screws along each side, and the two screws in the rear corners. Don't lose the screws!

3. The keyboard is connected to the Apple's main board by a short ribbon cable which plugs into a dip socket on the main board. Carefully lift off the bottom of the computer without putting any strain on this cable. Unplug the cable and put the bottom of the computer (with the main board attached) aside.

4. Notice that there are small numbers printed on the underside of the keyboard. Near the right hand edge of the keyboard, locate the number 42. (See Figure 2)

5. There will be three small solder pads in a row next to the number 42. The LEFT-MOST PAD is an empty plate-through hole. This is where you will attach your wire.

6. Prepare a 15 inch piece of light gauge insulated wire.

7. Carefully melt the solder in the hole and insert one end of the wire.

8. The other end of the wire goes to pin #1 of the auxiliary connector at the top of the ROMPLUS+.

9. Carefully plug the keyboard connector back into the Apple's main board and re-assemble the computer.

10. Connect the wire to the ROMPLUS+ board, plug the Apple back in, and you are ready to go!

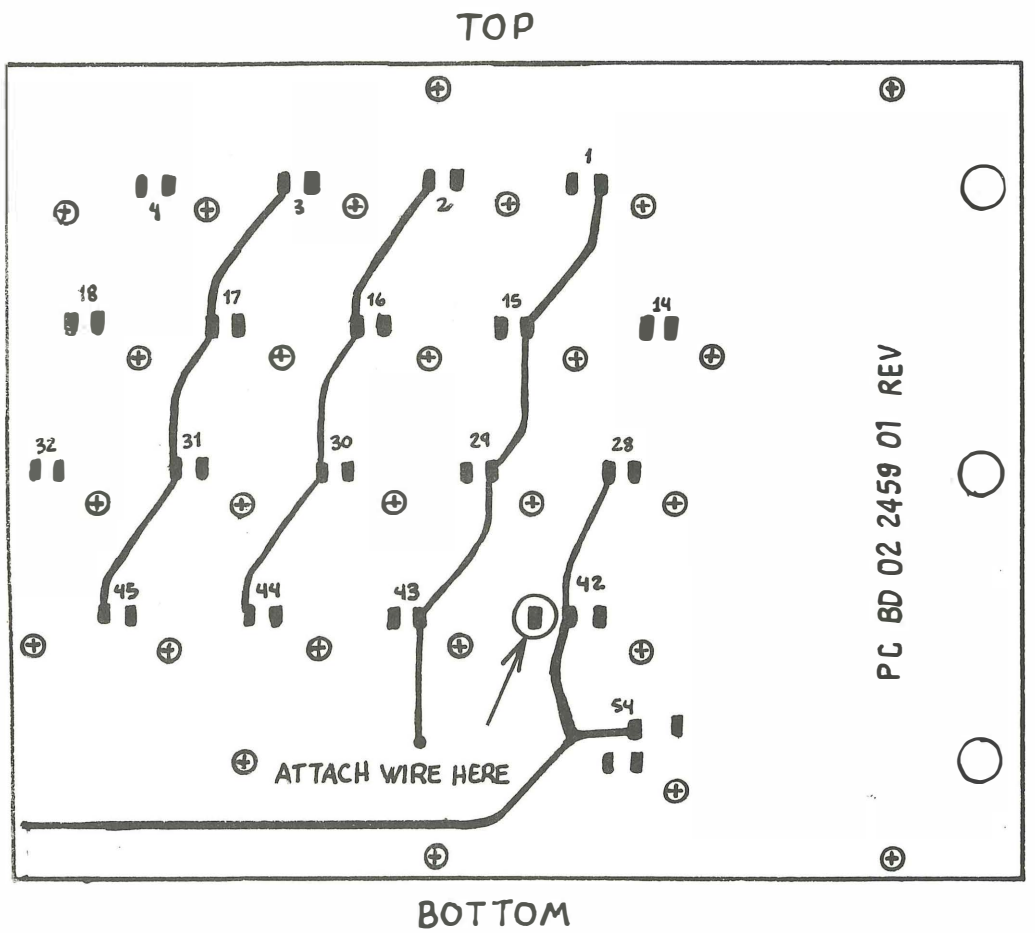


FIGURE 2

USING THE SHIFT KEY MODIFICATION

Once the shift key modification has been installed as described above, the Keyboard Filter can be told to read the regular shift keys with the (Control/V) command. Once a (Control/V) has been entered, upper and lower case letters may be typed normally. The Filter's (Control/L) shift lock will continue to work as usual. A second (Control/V) will turn off the shift key mode.

As usual, there is one slight complication. The Apple normally uses the shift key for the entry of three punctuation symbols: @,↑, and]. The "at" sign is a shift P, the up arrow a shift N, and the right bracket (which unlike the others is not displayed on the key-top) is a shift M. When the Keyboard Filter is interpreting the shift key for capitalization, it has no way of knowing whether you want a "@" or a capital P, and so forth. So the Filter assumes you want the capital letter (the usual case). If you want the other symbol, you will have to either go into the Raw Mode (in Raw Mode the keyboard is read as though the Filter were not there) or turn off the shift key modification (Control/V). In either case you may then type the character as you would normally, then switch back to your regular entry mode.

QUICK REFERENCE CHART

The following is a list of control characters and their functions:

A: Not used by Keyboard Filter

B: Not used by Keyboard Filter

C: Normal usage - stops programs

D: Used by DOS

E: Turns on CURSOR MOVEMENT MODE

F: FONT SWITCHING character - follow with number of desired font (0=font on Keyboard Filter ROM)

G: Normal usage - rings bell

H: Normal usage - back space

I: Toggles INVERSE MODE

J: Not used by Keyboard Filter

K: Used to select INPUT FROM PERIPHERALS. Follow with slot number of peripheral desired.

L: Toggles SHIFT LOCK mode

M: Normal usage - carriage return

N: Not used by Keyboard Filter

O: Toggles OVERSTRIKE MODE

P: SWITCHES PAGE being displayed

Q: Diverts OUTPUT to printer, etc. Follow with slot number of output device.

R: Toggles RAW MODE

S: Prints a KEYBOARD MACRO. Follow with key for desired macro. Also used for STOPLIST & ENDLIST.

T: Switches COLOR of characters. Follow with number of desired color (0 - 4)

U: Not used by Keyboard Filter

V: Toggles normal SHIFT KEY USAGE in modified Apples

W: COPY TO END OF LINE

X: Normal usage - terminates current line

Y: Not used by Keyboard Filter

Z: CLEARS currently displayed PAGE. From within program use CALL -13376 (must be last command on line).

INITIALIZATION PROCEDURE

1. From Basic, enter PR#(#of ROMPLUS+ slot)
2. Enter (Shift/Control/M)1A,B,C, or D
3. Initialization options are as follows:

   Option A = First time initial, graphics mode
   Option B = Re-initial, graphics mode
   Option C = First time initial, text mode
   Option D = Re-initial, text mode

USE WITH RAM APPLESOFT

1. Boot DOS (Disk Operating System)
2. Initialize Keyboard Filter from Integer Basic
3. Use (Control/P) to switch display to Page 2
4. Type FP to load Applesoft II
5. Type POKE -16304,0

COLORS

0 = WHITE
1 = PURPLE
2 = GREEN
3 = BLUE
4 = ORANGE

Mountain Hardware

Located in the Santa Cruz Mountains of Northern California, Mountain Hardware, Inc. is a computer peripheral manufacturer dedicated to the production of use-oriented high technology products for the microcomputer. On-going research and development projects are geared to the continual supply of unique, innovative products that are easy to use and highly complementary in a broad variety of applications.

300 Harvey West Boulevard
Santa Cruz, CA 95060
(408) 429-8600