®

UNLOCKED and UNPROTECTED

Compatible with
- Apple IIe®
- Apple II+
- Apple II

Beagle Bros Micro Software Inc.

FLEX TYPE

FORMERLY "FLEX TEXT"

VARIABLE-WIDTH HI-RES TEXT UTILITY
by MARK SIMONSEN

VARIABLE-WIDTH TEXT
Flex Type allows your Applesoft programs to produce normal 40-column text as well as Expanded and Condensed characters ON THE SAME SCREEN, giving you a wide range of 20- to 70-characters per text line. No extra hardware is required.

ADD GRAPHICS TO TEXT.
Or Text to Graphics! In effect, you are able to Hplot hi-res lines on your text screen layouts or type on your hi-res drawings.

USES NORMAL COMMANDS
Flex Type understands normal Applesoft commands, including Home, Inverse, Normal, Htab 1-70, and Vtab 1-24. Text window poke-commands are also supported. Flex Type is completely compatible with GPLE, Double-Take and Beagle Basic.

FRIENDLY AND COMPATIBLE
Your existing Applesoft programs will run under Flex Type control. Just boot Flex Type, and run your program.
(Note: Some large programs may overwrite the hi-res screen and require extra manipulation; complete instructions are included in this manual.)

RE-DEFINABLE CHARACTERS
Any text character may be redrawn with Flex Type's Apple CHARACTER EDITOR, and stored on disk and/or in memory, giving you full control over previously fixed-format letters, numbers and symbols.

UPPER & LOWER CASE
No extra hardware is required to produce attractive upper and lower case on the screen, in any width of course. Shift key modifications for non-IIe's are supported.

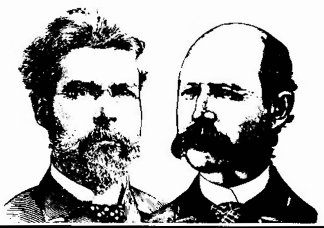
TWO-WAY SCROLLING
Hi-res pictures and text may be scrolled smoothly in both directions, adding flexibility to your hi-res screen displays.

EASY TO USE
Just boot Flex Type and go. Switch between Apple's "normal" text screen and hi-res Pages 1 and 2 whenever you want.

Free "PEEKS and POKES" 11 x 17 Wall Chart Enclosed
Apple's PEEKS, POKES, POINTERS and CALLS on one heavy-duty poster. An indispensable Apple programming tool.

(70-COLUMN TEXT REQUIRES MONOCHROME MONITOR.)

INCLUDES PEEKS & POKES CHART



Beagle Bros™
INDOOR SPORTS

FLEX TYPE

Copyright © 1982, Mark S. Simonsen

Published by BEAGLE BROS
4315 Sierra Vista
San Diego, California 92103

TABLE OF CONTENTS

Flex Type Requirements

FLEX TYPE requires a 48K Apple with Applesoft BASIC (in ROM on the motherboard, on a ROMcard, or loaded in the Language Card) and a disk drive.

The Flex Type Diskette

The Flex Type Master Disk is DOS 3.3 compatible. All programs on it however may be easily De-muffined to DOS 3.2 format. (NOTE: Do not modify any programs on the master disk unless you have made a backup copy.)

Using Flex Type in Software to be Resold

Flex Type is licensed only for your personal use. If you are interested in marketing software that uses Flex Type please contact Beagle Bros regarding a third party licensing agreement.

Copyright Information

IMPORTANT

For a quick reference to Flex Type's commands, see the COMMAND SUMMARY on page 29 of this book.

1.0 INTRODUCTION

1.1 What Flex Type Does

Flex Type is an Assembly Language program that automatically converts all of your input and output (I/O) into a format that allows you to choose between 70, 56, 40 and 20 column text width.

1.2 Advantages of Using Flex Type

Not only does Flex Type offer you four different screen widths (actually it offers three different font widths and a double wide mode), it also gives you access to upper and lower case letters, all special characters, true mixed graphics and text, and user definable character sets. However, the biggest advantage of using Flex Type is the ability it gives you to display 70 columns of text; almost twice that of a normal Apple!

1.3 Disadvantages of Using Flex Type

Although not required in 20, 40, or 56 column format, it is recommended that you use this program with a high resolution monitor when in 70 column mode. This is because of the fact that the picture on most televisions is not clear enough to allow you to distinguish very small characters when they are placed close together. Also, very large programs (those that nearly fill all available memory) may not be able to use Flex Type because at least one Hi-Res Page must be preserved for display. See Appendix B for further information.

1.4 Features of Flex Type

- Enter, list, edit and run any program in up to 70 columns of upper and lower case characters.

- Supports all Applesoft BASIC and Apple Monitor commands along with all escape sequences, editing functions (arrow keys), and Zero Page window pokes.

- Several selectable character sets are included. User definable character sets for custom type fonts and graphics characters are easily generated with Apple Character Editor (A.C.E.) which is included on the Flex Type Master Disk.

- Flex Type is stingy with memory and uses only 5% of your available RAM (random access memory) for the main program. (Character sets require an additional 3/4 K each.)

- Flex Type is completely compatible with Neil Konzen's Program Line Editor (P.L.E.) and Global Program Line Editor (G.P.L.E.).

2.0 USING FLEX TYPE

2.1 Invisible Flex Type

Flex Type has been designed so that its operation is transparent to you, the user. That means your Apple will behave normally, the way you expect it to behave. Whatever your Apple could do before, it can now do at least as well or better.

2.2 Demonstrating Flex Type

To better understand your Apple's new abilities, take a minute now to view the demonstration program on your master disk. If you haven't already done so, boot the Flex Type Master Disk by inserting it into disk drive #1 and turning on the power to your Apple][plus. Apple][owners without the Autostart ROM will additionally need to enter:

```
6 <ctrl-P> <return>  (from the monitor)

     or

PR#6 <return>        (from BASIC)
```

To see the demo, type:

```
RUN FLEX TYPE DEMO <return>
```

NOTE: A '<return>' means to press the RETURN key, just as '<ctrl-P>' means to hold down the CTRL key and press P. This notation will be used throughout the tutorial.

2.3 Commands

Now that you've seen the demo, you are undoubtedly anxious to begin using Flex Type. In order to make the most of Flex Type, you first need to know the commands that invoke its special features.

Flex Type is easy to use. If you know how to use an Apple, you practically know how to use Flex Type. The only difference you will notice right away is that your display is now 30 characters wider than before. That is because you are in 70 column mode. You will see later how to switch modes. In this mode you can type in longer lines and when you list your programs they are automatically displayed in a 70 column format.

Go ahead. Load a program and list it, or type something in. It's kind of nice to have so much information on the screen at one time isn't it? Commands like TAB, HTAB, and SPC also work on a 70 column basis rather than a 40 column one. For instance without Flex Type, if you were to type:

```
HTAB 69 : PRINT "HI" <return>
```

your Apple would tab over 40 columns, "realize" it had no more room on that line, output a carriage return and tab over another 29 columns for a total of 69 and then print HI. With Flex Type (in 70 column mode), if you type:

```
HTAB 69 : PRINT "HI" <return>
```

your Apple will tab over 69 columns and print HI. The upper limit on HTAB, TAB, SPC and other similar commands will now depend on the text mode that you are in (e.g. 20, 40, 56, or 70).

2.3.1 Alternating Character Sets

Alternate character sets are available in immediate mode (with a BASIC or monitor prompt) and deferred mode (during program execution). In immediate mode there are several ways to switch character sets. The first is simply typing the Toggle Text command (control-T) followed by the number of the desired character set (being sure not to type any spaces or any other characters between the control-T and the number) and a carriage return. Type:

```
<ctrl-T> 2 <return>
```

You got a ?syntax error message and the prompt and the cursor are slightly larger now because the character set you selected is a 56 column one. If you LIST a program you will see that it is formatted accordingly. The syntax error message came about because neither DOS nor Applesoft recognizes 'control-T 2' as a legitimate command. The second method is better. Type:

```
<ctrl-T> 3 <ctrl-X>
```

Voila! Notice the absence of the syntax error. Control-X cancelled the input line so 'control-T 3' was never seen by DOS or Applesoft, but because Flex Type intercepts each character individually the command was executed. Control-X can and should be used with all Flex Type commands in the immediate mode of execution. By the way, you should now be looking at a standard-sized character set (40 column mode). But as you will see in the discussion on upper and lower case, it is far from standard.

The third method of entering Flex Type commands is to use the Applesoft CHR$ function. You can select the next character set by typing:

```
PRINT CHR$ (20) "1" <return>

     or

PRINT CHR$ (20) 1 <return>
```

Twenty (20) is the ASCII equivalent of control-T. The fourth method is similar to the third. Type:

```
PRINT "<ctrl-T>3" <return>
```

Now to see double wide characters type the Wide command:

```
<ctrl-W> <ctrl-X>
```

Wow, there's a big difference between a twenty column character set and a seventy column one! To turn off the double wide mode type control-W again. Wide mode is intended to be used only with 40 column character sets but may be used with any size. You should be aware though that regardless of the font width, Wide mode only displays 20 columns of text per line.

The HELLO program loaded three different character sets of three different sizes for you to use. Up to nine (9) different character sets may be loaded at one time. And naturally you may choose to have as few as one. To change the number of character sets to be loaded when Flex Type is booted requires only a few changes to the HELLO program. Beginning with line number 100 assign the character set names to the CS$ array in the order that they are to be loaded. (Start with CS$(1), then CS$(2), CS$(3), etc.).

NOTE: Because Flex Type supports three different font widths it is suggested that you use the following naming convention for your character sets. This should keep you from forgetting the width of a text font. Also when editing text fonts, A.C.E. tries to determine the size of a character set when it is loaded. If it has been named in accordance with the naming convention, A.C.E. will recognize that fact. If not, it will assume a 40 column text width.

Font Names

1. 7Ø column text fonts must be suffixed by .7Ø (e.g. ASCII.7Ø).

2. 56 column text fonts must be suffixed by .56 (e.g. ASCII.56).

3. 4Ø column text fonts may be suffixed by either .4Ø or by anything else such as .Set, .Font, etc.

The character sets once loaded are selected by a number which indicates the relative position in which they were loaded. Numbering starts with one (1) as you may have noticed from the earlier examples, and has a maximum of nine (9).

Because each character set requires an extra 768 bytes (3/4 K) of memory it is wise to only load those that you plan to use. That will allow as much memory as possible for your programs.

As mentioned previously, these commands are recognized in programs (deferred execution mode) too. Four ways to incorporate Flex Type commands into programs are shown below.

```
10 PRINT "<ctrl-T>1THIS WILL BE IN 70 COLUMN MODE."
```

or

```
10 T56$ = "<ctrl-T>" + "2" : PRINT T56$"THIS WILL BE IN 56 COLUMN
   MODE."
```

or

```
10 PRINT CHR$ (20)"3THIS WILL BE IN 40 COLUMN MODE."
```

or

```
10 T70$ = CHR$ (20) + "1" : PRINT T70$"THIS WILL BE IN 70 COLUMN
   MODE."
```

The first two methods shown above are not recommended because they usually cause listings (to the screen and printer) to appear goofed up.

Changing character widths (modes) will force the cursor to the leftmost position on the screen. However, selecting a different character set of the same width will not affect the cursor position.

The relative position of the cursor on the screen is also affected by character type. For instance, HTAB 10 in 20 column mode (double wide mode) is not the same screen location as HTAB 10 in 70 column mode. In fact, it would be HTAB 35 in 70 column mode that equals HTAB 10 in 20 column mode. Remember this when you want to mix character types on the same line.

2.3.2 Upper and Lower Case (not applicable to Apple IIe)

Entering lower case text is easy with Flex Type. Type control-A once to enter lower case mode. In lower case mode, typing control-A once will shift the next character to upper case and any subsequent characters will be lower case. Typing control-A twice returns you to upper case shift lock. If you have made the shift modification to your Apple, it will work in lower case mode.

NOTE: Unlike most other Flex Type commands, control-A does not need to be followed by a control-X.

2.3.3 Special Characters (not applicable to Apple IIe)

For access to the special characters in the ASCII set the Zap command (control-Z) is provided. The Zap command when executed will only affect the character immediately following it. Five of the ten special characters are available in upper case and the other five are naturally available only from lower case. Below is a list of the ten special characters and the keys to type (on the keyboard) or print (from a program) to access them.

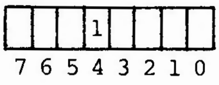
UPPER CASE

```
<control-Z> K  =  [  (left bracket)
<control-Z> L  =  \  (back slash)
<control-Z> M  =  ]  (right bracket)
<control-Z> N  =  ^  (up arrow or caret)
<control-Z> O  =  _  (underscore)
```

LOWER CASE

```
<control-Z> k  =  {  (left brace)
<control-Z> l  =  |  (vertical bar)
<control-Z> m  =  }  (right brace)
<control-Z> n  =  ~  (tilde)
<control-Z> o  =  □  (rub)
```

If it seems funny to use K, L, M, N & O to generate the special characters, it would probably be helpful to understand how the Zap command works. When Flex Type sees a control-Z it sets a flag telling itself to set the sixteens bit of the next character typed or printed. For those of you not familiar with HEX or binary, the sixteens bit is the fifth bit from the right in a byte. (It is actually the fourth bit because they are numbered from 0 - 7, right to left.) The sixteens bit is this one:



Why is it called the sixteens bit? If all the bits in a byte were zero (off) and the sixteens bit was a 1 (on) then the decimal value of the byte would be 16 (10 in hex and 0001 0000 in binary). For more understanding on this, refer to APPENDIX C on the ASCII character set.

2.3.4 Mixed Graphics and Text

Another useful feature of Flex Type is the capability to combine text and graphics on the same screen. With Flex Type you can place text anywhere on graphics, not just the bottom four lines of the screen.

Let's sidetrack for a moment. For flexibility Flex Type provides you with three commands that let you decide how the hi-res screens should be used. They are control-A, control-B and control-P. Control-A causes all output to go to hi-res page one. Control-B will cause all output to be placed on hi-res page two. And control-P causes the page currently being written on to be displayed. Because of the way these commands are set up, you have a lot of freedom with the hi-res screens. This versatility is not at first obvious. Type:

```
PRINT CHR$(2)"THIS WILL BE PRINTED ON PAGE 2."CHR$(1) <return>
```

Since you are looking at page 1 you don't see what was printed. To see page 2 type:

```
PRINT CHR$(2) CHR$(16) <return>
```

You are now looking at the same page your output went to. (If there's garbage all over the screen, type HOME to clear things up, then return to page 1 and repeat the above steps.) The same kind of trick can be performed from page 2. Type:

```
PRINT CHR$(1)"THIS WILL BE PRINTED ON PAGE 1."CHR$(2) <return>
```

To return to page 1 type:

```
PRINT CHR$(1) CHR$(16) <return>
```

These three commands (ctrl-A, B, and P) can be used regardless of the contents of the two screens. They can be filled with text, graphics or a combination of the two. Getting back on track, combining text and graphics on the same screen with Flex Type is easier than you might suspect.

To combine text with an already existing hi-res picture, just BLOAD the picture to the desired screen (see APPENDIX B for addresses). After loading the picture, it should automatically be displayed. (If not, you have either accidentally or purposely loaded it into the other hi-res page, i.e. the one you are not looking at.) With the picture loaded, you can put text anywhere on the screen by using VTAB, HTAB, TAB, SPC, PRINT, etc.

To go one step further and plot on the hi-res screens requires that you do nothing extra. You may however execute HGR (for page 1) or HGR2 (for page 2). These two commands function exactly as they would without Flex Type (i.e. initialize graphics and clear the screen to black) except that you can also print text anywhere on the screen. You're not limited to only printing on the bottom four lines of the screen. Should you desire to limit your text to the bottom four lines, see the miscellaneous info at the end of APPENDIX D. Now, whether you've executed one of the HGR's or not, you may specify the HCOLOR you want and HPLOT, DRAW or XDRAW as normally performed on the Apple][. (See the Applesoft Tutorial, Applesoft][BASIC Programming Reference Manual, and Apple][Reference Manual for further information on these graphics commands.)

2.3.5 Miscellaneous Commands

Three other Flex Type commands that are provided mostly for your convenience are control-E, control-F, and control-N. Control-E is the same as typing ESC E or CALL -868. Control-F performs the same function as ESC F and CALL -958. Both of these commands make clearing part of the screen as easy in a program as it is in immediate mode. Control-N will invert the entire screen or any windowed portion of it. The above commands just as with all Flex Type commands may be used from within programs (BASIC or Assembly Language) or in immediate mode (with either the BASIC or monitor prompt).

Another Flex Type command is control-O. Control-O toggles Overstrike mode on and off. For example, to underline a word use one of the following methods.

```
10 PRINT CHR$(15) "THE" CHR$(8) CHR$(8) CHR$(8) "___" CHR$(15)
```

or

```
10 VTAB 12: HTAB 10: PRINT "HELLO"
20 VTAB 12: HTAB 10: PRINT CHR$(15) "_____" CHR$(15)
```

The following commands were added to Flex Type so that it could more easily emulate (or simulate) an 80 column board. They also make it possible to interface Flex Type with programs like GPLE and PLE.

control-K	Clear from cursor to end of window.
control-L	Move cursor to upper-left corner of window and clear window.
control-S	Set scroll mode; followed by 0 = no scroll, 1 = normal scrolling, and 2 = fast scrolling.
control-U	Deactivate Flex Type and return to normal text screen.
control-V	Scroll display down one line, leaving cursor in the current position.
control-Y	Move cursor to upper-left corner of window without clearing screen.
control-\	Move cursor one space to the right.
control-]	Clear line from cursor to right edge of window.
control-^	Scroll display up one line, leaving cursor in the current position.
control-_	Move cursor up to next line (reverse linefeed).

3.0 ADVANCED TECHNIQUES

3.1 Relocating the Code

Included on the master disk is a binary file (B type) named RELOCATOR. This assembly language routine, in cooperation with the BASIC program named HELLO, performs the code relocation of Flex Type. HELLO loads Flex Type into memory (the default location is between DOS and its buffers), appends the character set(s) to the end of it and then calls RELOCATOR to adjust the addresses within Flex Type so that it will run properly at that location. This two step process is actually quite simple.

NOTE: RELOCATOR is not a general purpose code relocator nor is Flex Type actually a true relocatable file. It cannot be loaded by a relocating loader and is not an R type file. By making RELOCATOR specialized for Flex Type, initialization is made simpler, quicker and less wasteful of memory and disk space.

To load Flex Type below hi-res page 1 at $0800 (2048) rather than at the default location, RUN HELLO 2. (NOTE: To have HELLO 2 run when you boot, RENAME HELLO to HELLO 1, and RENAME HELLO 2 to HELLO. You will also need to change the program name in line number 220 of HELLO 2 to HELLO.)

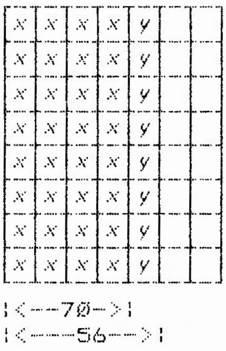
3.2 Alternate Character Sets

To select a different character set(s) as Flex Type's default(s), follow the instructions in section 2.3.1. Loading a different character set(s) after Flex Type is up and running requires that you know the starting address of the present character set(s). This address is usually $9400 (37888 in decimal) unless you loaded Flex Type at $0800 in which case it's $1000 (4096 decimal). (You may not load more character sets than the quantity loaded when Flex Type was booted. For example, if three different character sets were loaded by the HELLO program then you may only replace those three. If you need to have four or more sets available you will need to reboot. This does not apply if Flex Type was loaded at $0800.) To determine the starting address of the character set(s) type:

```
PRINT PEEK (974) + PEEK (975) * 256 + 2048
```

Once the address is known BLOAD the new character set at that address plus 768 ($300) times its position. Then POKE ADDRESS - 10 + POSITION, with WIDTHTYPE. (See the program called OPTCHAR on the master disk for an example.)

To define your own character sets you may use ANIMATRIX from the Apple DOS Tool Kit, a similar program, or the supplied character editor (A.C.E.). If you use a program like ANIMATRIX be aware that you may only use the entire 7x8 array of pixels (picture elements) that it provides for each character when you are defining a 40 column character set. Because the 56 and 70 column characters in Flex Type are not that wide, restrict those character definitions to the leftmost 5 and 4 pixels (respectively) of each row in the character block. Below is an illustration.



There are a couple of things that you ought to know if you plan on using A.C.E. First, to use it type RUN A.C.E. Second, when A.C.E. is run it will (re)load Flex Type (whether or not it is currently in memory) with room for only two character sets; one to be edited and one to be used for reference. When exiting A.C.E. you will be left within Flex Type with access to only those two character sets. (As always if you wish to restart Flex Type without rebooting, first RUN REMOVE FLEX TYPE.) The instructions for using A.C.E. are very simple, easy to learn and are included in the program itself. So...happy editing.

3.3 Impersonating a Flasher

Usually on an Apple the characters are displayed in NORMAL, INVERSE and FLASHing with hardware. As you may have noticed, Flex Type recognizes INVERSE and NORMAL and outputs characters accordingly (something that most other hi-res character generators and hi-res text programs can't do). But if you've tried FLASH you no doubt noticed that your output was only displayed as inverse characters. That is because there is no hardware to support flashing characters on the hi-res screens. All is not lost however. With a special software routine it is usually (but not always) possible to simulate FLASH. And in case you've forgotten, just such a routine was used in part of the Demo program. The trick is simply to print out the text string normally, wait a specified period of time, print it again in inverse and repeat the process. An example would be:

```
10 TIMES = 100 : VP = 11 : HP = 1
20 X$ = "THIS TEXT IS FLASHING."
30 GOSUB 1000
      .
      .
      .

1000 PAUSE = 50
1010 FOR I = 1 TO TIMES
1020 TGGLE = NOT TGGLE : NORMAL
1030 IF (TGGLE) THEN INVERSE
1040 VTAB VP : HTAB HP : PRINT X$;
1050 FOR J = 1 TO PAUSE : NEXT J
1060 NEXT I
1070 RETURN
```

The above method works well only if your program doesn't need to perform any other tasks while you are simulating FLASH. What if you want to do something else at the same time? Well, with only minor modificatons to the subroutine you can perform several operations at the same time. Below is an example:

```
10 VP = 11 : HP = 1
20 X$ = "THIS FLASHES WHILE YOU DO SOMETHING ELSE."
30 GOSUB 1000
40 ...SOMETHING...
50 GOSUB 1000
60 ...SOMETHING...
70 GOSUB 1000
      .
      .
      .

1000 TGGLE = NOT TGGLE : NORMAL
1010 IF (TGGLE) THEN INVERSE
1020 VTAB VP : HTAB HP : PRINT X$;
1030 RETURN
```

Once the parameters VP, HP and X$ are set up in the main program they need not be redefined. All you need to do is intersperse a few GOSUBs within your processing and the text will flash while you perform calculations, print, etc. For an example of FLASH simulation see the program entitled FLASHER on the master disk.

4.0 IMPORTANT ODDS AND ENDS

4.1 Exiting and Re-entering Flex Type

A brief discussion of some ways to leave and get back into Flex Type is given here to increase your awareness of your new environment in the Apple.

If you ever exit Flex Type (for instance with the TEXT command or the reset key) you may re-enter it without rebooting the master disk or running the HELLO program by typing a "&" and a return. Upon returning you will find yourself on the hi-res page you last used with the last character set you used. It isn't important to know Flex Type's entry point. It seems appropriate here to mention that the only way to cleanly alternate between Flex Type's hi-res text and normal text screen text is with the TEXT and & commands. (These may also be used from within a program.) Similarly the only way to completely exit Flex Type and free up all memory it occupies is to RUN REMOVE FLEX TYPE.

Suppose you accidentally hit reset and exit Flex Type unintentionally. If you just typed in a long program, don't worry you haven't lost it. You're just in normal text mode. To re-enter Flex Type type:

```
& <return>              (from BASIC)
```

or, if you are using the & command for something else

```
CALL 973 <return>       (from BASIC)
```

or

```
3CDG <return>           (from the monitor)
```

If you have a newer Apple with the encoder board under the keyboard, you may want to set the switch so accidentally hitting reset will do nothing. And for those of you with Autostart ROM's, defeating the reset entirely by intercepting it is possible. (See Apple][Reference Manual.)

4.2 The Window

As mentioned previously, Flex Type supports Zero Page Window pokes. The screen may be windowed in any of the modes (2Ø, 4Ø, 56, and 7Ø columns). However, because of the way that Flex Type works, it is only possible to change the top and bottom edges of the window (WNDTOP and WNDBTM respectively). The location of WNDTOP is 34 ($22) and WNDBTM is 35 ($23). DO NOT, repeat DO NOT, ever poke locations 32 ($2Ø, WNDLFT) or 33 ($21, WNDWDTH) while using Flex Type. Also, the warnings in the Applesoft][BASIC Programming Reference Manual and the Apple][Reference Manual are valid with Flex Type too (i.e. WNDTOP should always be less than WNDBTM, just as WNDBTM should always be greater than WNDTOP).

4.3 More Do's and Don'ts

Don't use CALL -936, -958, -868, -922, or -912.
Do use HOME, ctrl F, ctrl E, ctrl J, and VTAB 24:PRINT as replacements respectively.

Don't use ctrl B <return> to re-enter BASIC from the monitor.
Do use ctrl C <return>. (If you accidently use ctrl B, execute an FP command to prevent a crash.)

Don't use ESC ctrl-Q or ESC ctrl-L to remove GPLE.
Do use REMOVE FLEX TYPE.

Don't use PLE/GPLE's Edit function in INVERSE mode. (Because it interprets inverse characters as control characters.)
Do use PLE/GPLE's Edit function in NORMAL mode.

Don't run ESCAPE CREATE, ESCAPE SAVE or ESCAPE PRINTER while GPLE is interfaced with Flex Type.
Do run them at any other time.

Don't run REMOVE PLE while in hi-res text.
Do run REMOVE PLE in normal text screen text.

4.4 Little Known Facts

1. To use Flex Type in combination with GPLE (or PLE) do the following:

Boot your GPLE (or PLE) disk.
Run HELLO or HELLO 2 from the Flex Type disk.
Run GPLE.48 PATCH (or GPLE.LC PATCH, GPLE.DM PATCH or PLE PATCH depending on which version of GPLE (or PLE) you loaded).

Control-V will now switch between normal and hi-res text with full access to GPLEs features in either mode. (This command doesn't exist in PLE).

If you have version 4.Ø or later of GPLE (from Beagle Bros), you should not use the above method. Instead, do the following:

Brun CONFIG GPLE from the GPLE disk.
Select 7 (other) when prompted for the video driver and use FLEXTYPE.VID from the Flex Type disk.
Answer YES to the 4Ø column default question.
Boot your GPLE disk.
Run HELLO or HELLO 2 from the Flex Type disk.
Type TEXT <return> and then & <return>.
Control-V works as above.

2. The sample programs on the disk will only work as intended under the default font configuration (i.e. three fonts: ASCII.7Ø, ASCII.56, & ASCII.4Ø).

3. Sending the "rub" character, CHR$(127), to a printer will cause the character preceding it to be deleted. It doesn't have any effect on the screen however and may be used for almost any purpose.

4. Flex Type loaded below Hi-res page 1 only has sufficient room for 5 character sets. However, if Hi-res page 1 is not used then there is plenty of room for the maximum 9 character sets below Hi-res page 2.

5. Flex Type doesn't write on the normal Text screen.

6. CALL -936 will clear the normal Text screen but will not affect the Hi-res screens.

7. Flex Type uses the following Zero Page locations: $EC, $ED, $EE, $EF, $FA, $FB, $FC, $FD, $FE, and $FF.

8. Most of page 3 in memory ($3ØØ-3FF) is available for you to use. Flex Type uses only three bytes, $3CD-3CF, and DOS and the System Monitor use $3DØ-3FF.

9. You can use the character sets from Apple's DOS Tool Kit (and other similar hi-res character programs) with Flex Type.

1Ø. When PLE is interfaced with Flex Type, its commands are only available in hi-res text.

11. Because GPLE/PLE will intercept some of Flex Type's commands (e.g. ctrl-E), you may sometimes have to use PRINT CHR$(5), etc. in immediate mode.

12. Change these GPLE Escape functions to what is printed below. (Note: Lower case letters represent control characters.)

```
Esc T- TTEXT:POKE-163ØØ,Øm
```

Change the 8 after "POKE 1Ø4," in the ctrl-shift-N Escape function to a 96 (or 64) if you booted with HELLO 2.

13. Only use Esc H (display control characters) in normal text and not when using hi-res text. This does not apply if the version of GPLE you are using is 4.Ø or later (from Beagle Bros).

APPENDIX A

Compatibility with BASIC and Monitor Commands

All Applesoft Floating Point BASIC commands are supported, but some have been altered slightly. A reference list and explanations follow.

TOKEN		KEYWORD	TOKEN		KEYWORD	TOKEN		KEYWORD
HEX	DEC		HEX	DEC		HEX	DEC	
80	128	END	A4	164	LOMEM:	C8	200	+
81	129	FOR	A5	165	ONERR	C9	201	-
82	130	NEXT	A6	166	RESUME	CA	202	*
83	131	DATA	A7	167	RECALL	CB	203	/
84	132	INPUT	A8	168	STORE	CC	204	^
85	133	DEL	A9	169	SPEED=	CD	205	AND
86	134	DIM	AA	170	LET	CE	206	OR
87	135	READ	AB	171	GOTO	CF	207	>
* 88	136	GR	AC	172	RUN	D0	208	=
* 89	137	TEXT	AD	173	IF	D1	209	<
8A	138	PR#	AE	174	RESTORE	D2	210	SGN
8B	139	IN#	AF	175	&	D3	211	INT
8C	140	CALL	B0	176	GOSUB	D4	212	ABS
8D	141	PLOT	B1	177	RETURN	D5	213	USR
8E	142	HLIN	B2	178	REM	D6	214	FRE
8F	143	VLIN	B3	179	STOP	* D7	215	SCRN(
90	144	HGR2	B4	180	ON	D8	216	PDL
* 91	145	HGR	B5	181	WAIT	* D9	217	POS
92	146	HCOLOR=	B6	182	LOAD	DA	218	SQR
93	147	HPLOT	B7	183	SAVE	DB	219	RND
94	148	DRAW	B8	184	DEF	DC	220	LOG
95	149	XDRAW	B9	185	POKE	DD	221	EXP
96	150	HTAB	BA	186	PRINT	DE	222	COS
97	151	HOME	BB	187	CONT	DF	223	SIN
98	152	ROT=	BC	188	LIST	E0	224	TAN
99	153	SCALE=	BD	189	CLEAR	E1	225	ATN
9A	154	SHLOAD	BE	190	GET	E2	226	PEEK
9B	155	TRACE	BF	191	NEW	E3	227	LEN
9C	156	NOTRACE	C0	192	TAB(	E4	228	STR$
9D	157	NORMAL	C1	193	TO	E5	229	VAL
9E	158	INVERSE	C2	194	FN	E6	230	ASC
* 9F	159	FLASH	C3	195	SPC(	E7	231	CHR$
A0	160	COLOR=	C4	196	THEN	E8	232	LEFT$
A1	161	POP	C5	197	AT	E9	233	RIGHT$
A2	162	VTAB	C6	198	NOT	EA	234	MID$
A3	163	HIMEM:	C7	199	STEP			

GR - set to low resolution graphics with four lines of normal text screen text at the bottom. Flex Type will not display hi-res text in combination with low resolution graphics. To return to hi-res text execute the & command.

TEXT - reset full text window, VTAB to row 23 and return to normal text screen text. (& returns to hi-res text.)

HGR - set to full screen high resolution graphics with ability to include text anywhere.

FLASH - cause output to be displayed as inverse (see section 4.0 for flashing character simulation).

SCRN(- only works in normal text mode. To simulate SCRN in hi-res text use:

```
ADRS = PEEK (974) + PEEK (975) * 256
VTAB row: HTAB column: CALL ADRS+12: A = PEEK(234) - 128
```

(The variable A will equal the decimal value of the character that was found on the screen, e.g. the letter "A" is a 65).

POS - only works in normal text mode. To simulate POS in hi-res text use:

```
PEEK (252)
```

Use of PR# and IN# should be avoided while in hi-res text. This is because almost all peripheral interface cards send output to the normal text screen making it impossible for Flex Type to intercept it. It should be obvious then that PR# and IN# may be used freely while in normal text mode. If you accidently or purposely issue one of those two commands while in hi-res text mode, execute an & command to return control to Flex Type.

To LIST in 7Ø columns to a printer in slot 1:

```
PRINT CHR$(4)"PR#1": PRINT CHR$(9)"7ØN": LIST: & <return>
```

To RUN a program in 7Ø columns on a printer:

```
(first program line)
1Ø PRINT CHR$(4)"PR#1: PRINT CHR$(9)"7ØN"

(last program line)
1ØØØØ &: END
```

Flex Type also supports all monitor commands including I and N. (See Apple][Reference Manual for other monitor commands.)

APPENDIX B

Preserving the Hi-Res Screens

It is essential to avoid overwriting the areas of memory that Flex Type uses for display. Assembly language programmers can easily comply with this requirement by assembling their code to run and be loaded at addresses other than those used by Flex Type. BASIC programmers however are in a different boat, as Applesoft programs load from the lowest memory up and the variables work toward the hi-res screens (see figure 1). Protecting at least one screen is necessary for Flex Type to function properly. However, depending on your application, you may want to protect both.

There is only 6K (1K = 1024 bytes) of memory between the end of text page 1 and hi-res page 1 that is allotted for BASIC programs. If you need to use hi-res page 1 and your programs are 6K or less in length, you're O.K. To determine the exact length of a program, LOAD it and then type:

```
PRINT (PEEK(175) + PEEK(176) * 256) - (PEEK(103) + PEEK(104)
   * 256) <return>
```

```
$FFFF -->  ------------------------------------  <-- 65535
                          ROM
$C000 -->  ------------------------------------  <-- 49152
              Disk Operating System
$9D00 -->  ------------------------------------  <-- 40192
                One Character Set
$9A00 -->  ------------------------------------  <-- 39424
                    FLEX TYPE
$9200 -->  ------------------------------------  <-- 37376
                   DOS Buffers
$8B00 -->  ------------------------------------  <-- 35584
             Applesoft strings start      |
             at HIMEM and build down      v
           ....................................
$6000 -->  ------------------------------------  <-- 24576
             High-resolution graphics,
                      Page 2
$4000 -->  ------------------------------------  <-- 16384
             High-resolution graphics,
                      Page 1
$2000 -->  ------------------------------------  <-- 8192
           ....................................
             Variables start at           ^
             LOMEM and build up           |
           ....................................
             Applesoft program            ^
             lines push LOMEM up          |
$0800 -->  ------------------------------------  <-- 2048
             BASIC System use:
             text screen, input buffer,
             stack, zero page, etc.
$0000 -->  ------------------------------------  <-- 0000
```

FIGURE 1.

If the number is smaller than 6144 the program will fit. To estimate how long a program is without having to load it into memory, take the number of disk sectors it occupies, subtract 1 and divide by 4. Here's an example. Say the program in question occupies 19 disk sectors. Subtracting 1 (for the track sector list sector) leaves 18. Divide that by 4 (since there are 256 bytes per sector, and it takes 4 sectors to equal 1K) and you get 4.5. Your program is approximately 4.5K long. It would safely fit, but because Applesoft variables build from the end of the program up, it would be necessary to have the first statement in the program set LOMEM at least above hi-res page 1. It should look like this:

```
10 LOMEM: 16384          (sets it above hi-res page 1)

     or

10 LOMEM: 24576          (sets it above hi-res page 2)
```

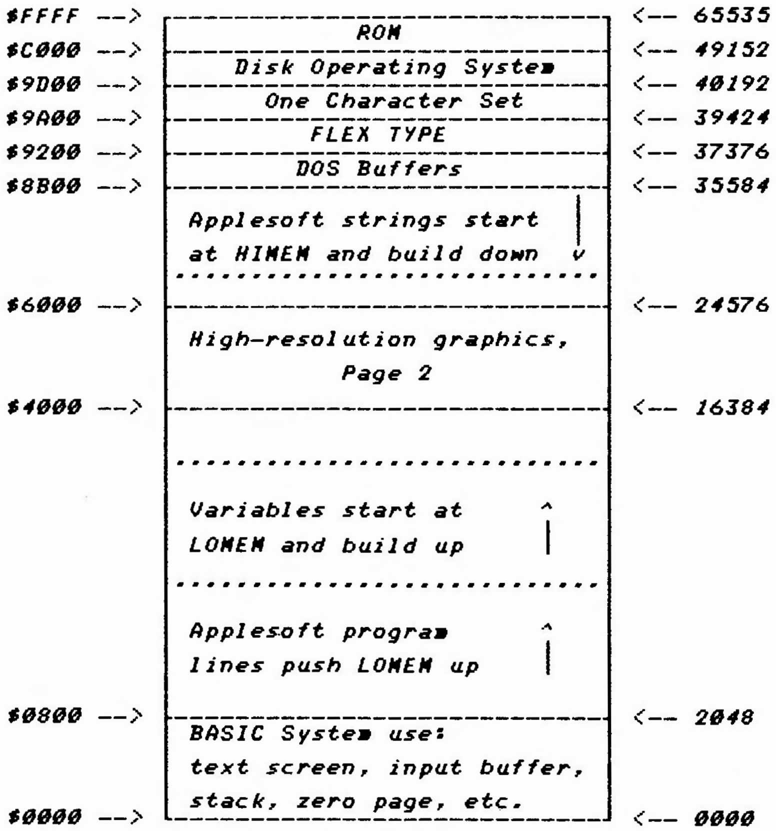


FIGURE 2.

This action will also prevent Applesoft strings (which build down from HIMEM) from overwriting the graphic screens.

Programs larger than 6K have several alternatives for solutions. First, if you only need to use one screen, use hi-res page 2. This frees up the 8K in hi-res page 1. You may now run programs that are up to 14K (57 disk sectors) long (see figure 2).

Second, set the program start to $4000 (16384 decimal) or $6000 (24576 decimal) depending on whether or not you need both screens. Do this with a POKE 104,X : POKE X * 256,0 (where X = 64 or 96). See the Load Flex Type subroutine in A.C.E. that is called in line number 20 of that program for an example. This yields 18.75K and 10.75K (76 and 44 disk sectors respectively) for BASIC programs. And if in combination with this you load Flex Type below hi-res page 1 you get an extra 2.75K of memory (see figures 3 & 4 and section 3.1 on loading Flex Type below hi-res page 1).

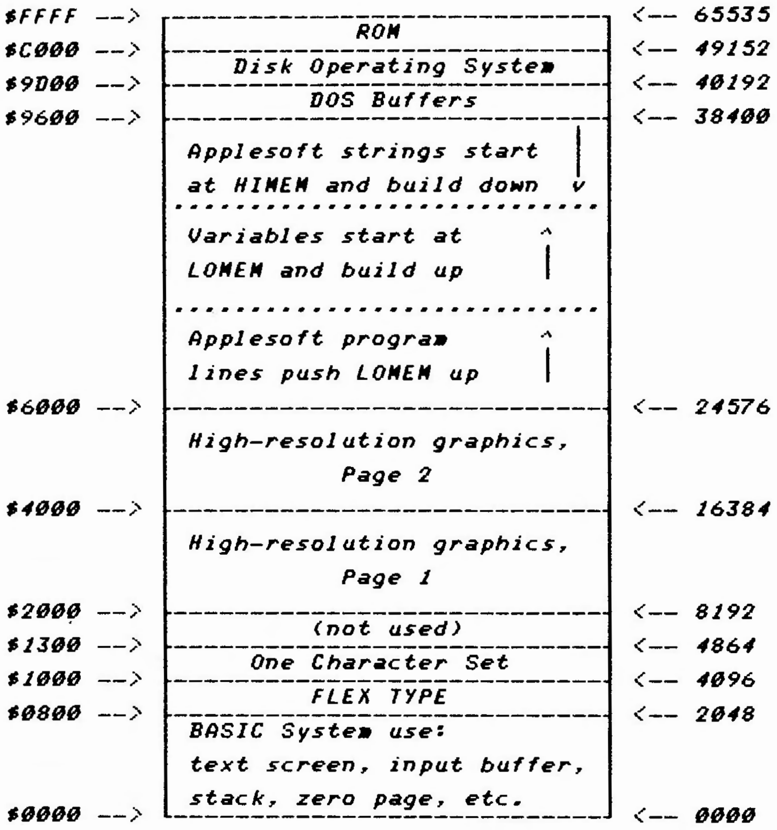

$FFFF --> ROM <-- 65535
$C000 --> Disk Operating System <-- 49152
$9D00 --> DOS Buffers <-- 40192
$9600 --> <-- 38400
Applesoft strings start at HIMEM and build down
Variables start at LOMEM and build up
Applesoft program lines push LOMEM up
$6000 --> <-- 24576
High-resolution graphics, Page 2
$4000 --> <-- 16384
High-resolution graphics, Page 1
$2000 --> <-- 8192
(not used)
$1300 --> <-- 4864
One Character Set
$1000 --> <-- 4096
FLEX TYPE
$0800 --> <-- 2048
BASIC System use: text screen, input buffer, stack, zero page, etc.
$0000 --> <-- 0000

FIGURE 3.

```
$FFFF -->  ---------------------------------  <-- 65535
          |              ROM                |
$C000 -->  ---------------------------------  <-- 49152
          |     Disk Operating System       |
$9D00 -->  ---------------------------------  <-- 40192
          |          DOS Buffers            |
$9600 -->  ---------------------------------  <-- 38400
          |  Applesoft strings start   |    |
          |  at HIMEM and build down   v    |
          | ............................... |
          |                                 |
          |                                 |
          |                                 |
          | ............................... |
          |  Variables start at        ^    |
          |  LOMEM and build up        |    |
          | ............................... |
          |  Applesoft program         ^    |
          |  lines push LOMEM up       |    |
$4000 -->  ---------------------------------  <-- 16384
          |  High-resolution graphics,      |
          |          Page 1                 |
$2000 -->  ---------------------------------  <-- 8192
          |          (not used)             |
$1300 -->  ---------------------------------  <-- 4864
          |      One Character Set          |
$1000 -->  ---------------------------------  <-- 4096
          |          FLEX TYPE              |
$0800 -->  ---------------------------------  <-- 2048
          | BASIC System use:               |
          | text screen, input buffer,      |
          | stack, zero page, etc.          |
$0000 -->  ---------------------------------  <-- 0000
```

FIGURE 4.

Last but not least, if you are short on memory, you can chain programs or break your program into parts and have each part call (RUN) the next part. See The DOS Manual.

APPENDIX C

The ASCII Character Set

The Ascii character set is given here for your reference. It has the decimal, hex, and binary equivalents for each character.

	DEC	HEX	BINARY	CHAR	WHAT TO TYPE
GROUP 1:	0	00	00000000	NULL	ctrl @
	1	01	00000001	SOH	ctrl A
	2	02	00000010	STX	ctrl B
	3	03	00000011	ETX	ctrl C
	4	04	00000100	ET	ctrl D
	5	05	00000101	ENQ	ctrl E
	6	06	00000110	ACK	ctrl F
	7	07	00000111	BEL	ctrl G
	8	08	00001000	BS	ctrl H
	9	09	00001001	HT	ctrl I
	10	0A	00001010	LF	ctrl J
	11	0B	00001011	VT	ctrl K
	12	0C	00001100	FF	ctrl L
	13	0D	00001101	CR	ctrl M
	14	0E	00001110	SO	ctrl N
	15	0F	00001111	SI	ctrl O
	16	10	00010000	DLE	ctrl P
	17	11	00010001	DC1	ctrl Q
	18	12	00010010	DC2	ctrl R
	19	13	00010011	DC3	ctrl S
	20	14	00010100	DC4	ctrl T
	21	15	00010101	NAK	ctrl U
	22	16	00010110	SYN	ctrl V
	23	17	00010111	ETB	ctrl W
	24	18	00011000	CAN	ctrl X
	25	19	00011001	EM	ctrl Y
	26	1A	00011010	SUB	ctrl Z
	27	1B	00011011	ESCAPE	ESC
	28	1C	00011100	FS	ctrl \ (or ctrl Z, ctrl L)
	29	1D	00011101	GS	ctrl] (or ctrl shift M)
	30	1E	00011110	RS	ctrl ^ (or ctrl shift N)
	31	1F	00011111	US	ctrl _ (or ctrl Z, ctrl O)
GROUP 2:	32	20	00100000	SPACE	space
	33	21	00100001	!	!
	34	22	00100010	"	"
	35	23	00100011	#	#
	36	24	00100100	$	$
	37	25	00100101	%	%
	38	26	00100110	&	&
	39	27	00100111	'	'
	40	28	00101000	(	(
	41	29	00101001	)	)
	42	2A	00101010	*	*
	43	2B	00101011	+	+
	44	2C	00101100	,	,

	DEC	HEX	BINARY	CHAR	WHAT TO TYPE
	45	2D	00101101	-	-
	46	2E	00101110	.	.
	47	2F	00101111	/	/
	48	30	00110000	0	0
	49	31	00110001	1	1
	50	32	00110010	2	2
	51	33	00110011	3	3
	52	34	00110100	4	4
	53	35	00110101	5	5
	54	36	00110110	6	6
	55	37	00110111	7	7
	56	38	00111000	8	8
	57	39	00111001	9	9
	58	3A	00111010	:	:
	59	3B	00111011	;	;
	60	3C	00111100	<	<
	61	3D	00111101	=	=
	62	3E	00111110	>	>
	63	3F	00111111	?	?
GROUP 3:	64	40	01000000	@	@
	65	41	01000001	A	A
	66	42	01000010	B	B
	67	43	01000011	C	C
	68	44	01000100	D	D
	69	45	01000101	E	E
	70	46	01000110	F	F
	71	47	01000111	G	G
	72	48	01001000	H	H
	73	49	01001001	I	I
	74	4A	01001010	J	J
	75	4B	01001011	K	K
	76	4C	01001100	L	L
	77	4D	01001101	M	M
	78	4E	01001110	N	N
	79	4F	01001111	O	O
	80	50	01010000	P	P
	81	51	01010001	Q	Q
	82	52	01010010	R	R
	83	53	01010011	S	S
	84	54	01010100	T	T
	85	55	01010101	U	U
	86	56	01010110	V	V
	87	57	01010111	W	W
	88	58	01011000	X	X
	89	59	01011001	Y	Y
	90	5A	01011010	Z	Z
	91	5B	01011011	[	[(or ctrl Z, K)
	92	5C	01011100	\	\ (or ctrl Z, L)
	93	5D	01011101	]	] (or ctrl Z, M)
	94	5E	01011110	^	^ (or ctrl Z, N)
	95	5F	01011111	_	_ (or ctrl Z, O)

The following characters require lower case mode to be set with the control A command.

	DEC	HEX	BINARY	CHAR	WHAT TO TYPE
GROUP 4:	96	60	01100000	`	@
	97	61	01100001	a	A
	98	62	01100010	b	B
	99	63	01100011	c	C
	100	64	01100100	d	D
	101	65	01100101	e	E
	102	66	01100110	f	F
	103	67	01100111	g	G
	104	68	01101000	h	H
	105	69	01101001	i	I
	106	6A	01101010	j	J
	107	6B	01101011	k	K
	108	6C	01101100	l	L
	109	6D	01101101	m	M
	110	6E	01101110	n	N
	111	6F	01101111	o	O
	112	70	01110000	p	P
	113	71	01110001	q	Q
	114	72	01110010	r	R
	115	73	01110011	s	S
	116	74	01110100	t	T
	117	75	01110101	u	U
	118	76	01110110	v	V
	119	77	01110111	w	W
	120	78	01111000	x	X
	121	79	01111001	y	Y
	122	7A	01111010	z	Z
	123	7B	01111011	{	{ (or ctrl Z, K)
	124	7C	01111100	|	| (or ctrl Z, L)
	125	7D	01111101	}	} (or ctrl Z, M)
	126	7E	01111110	~	~ (or ctrl Z, N)
	127	7F	01111111	□	□ (or ctrl Z, O)

APPENDIX D

Flex Type Command Summary

Boot the Flex Type disk or RUN HELLO to start.
(See page 18 if you are using PLE or GPLE.)

SCREEN CONTROL

& (or CALL 973)	Re-enter Flex Type from normal-text mode
TEXT (or Reset)	Enter normal-text mode from Flex Type
ctrl-A	Print on Hi-Res Page 1
ctrl-B	Print on Hi-Res Page 2
ctrl-P	Display current print page
ctrl-N	Normal/Inverse (entire screen) toggle
ctrl-O	Overstrike/Normal toggle

TYPE WIDTH

ctrl-T 1	70-column text (or 1st font loaded)
ctrl-T 2	56-column text (or 2nd font loaded)
ctrl-T 3	40-column text (or 3rd font loaded)
ctrl-W	20-column text (double-Wide characters)

SPECIAL CHARACTERS

ctrl-A	Capital letters
ctrl-A	Lower-case letters
ctrl-A	Shift (capitalize next letter only)

ctrl-Z K	Left Bracket ([ )	**ctrl-Z k**	Left Brace ({)
ctrl-Z L	Backslash (\ )	**ctrl-Z l**	Vertical Bar (|)
ctrl-Z M	Right Bracket ( ])	**ctrl-Z m**	Right Brace (})
ctrl-Z N	Up Arrow (^)	**ctrl-Z n**	Tilde (~)
ctrl-Z O	Underscore (_)	**ctrl-Z o**	Rub (□)

MISCELLANEOUS

HOME	Clear screen being viewed
ctrl-E	Clear text from cursor to end of line
ctrl-F	Clear text from cursor to end of screen

ALTERNATE CHARACTER SETS

TO CREATE a new character set—
Run **A.C.E.** (the "Apple Character Editor" on the Flex Type disk).

TO LOAD new character sets—
Change the HELLO program's first lines to assign Character Set names; **CS$(1)**, **CS$(2)**, **CS$(3)**...; in the order they are to be loaded. Then Save and Run HELLO.

Here's some miscellaneous information about the hi-res screens that may be useful.

To load pictures to hi-res screen 1, type:

```
BLOAD <picture name>,A$2000 <return>

          or

BLOAD <picture name>,A8192 <return>
```

and to load pictures to hi-res screen 2, type:

```
BLOAD <picture name>,A$4000 <return>

          or

BLOAD <picture name>,A16386 <return>
```

To simulate normal HGR with only 4 lines of text at the bottom of the screen, type:

```
POKE 34,20 <return>
```

To undo this use the TEXT command or POKE 34,0.

NOTE: If you type HGR from page 2 or HGR2 from page 1 you will not see the prompt or the cursor. To remedy this situation use the control A, B and P commands. If you typed HGR from page 2, PRINT CHR$(16) to return to page 2 or PRINT CHR$(1) CHR$(16) to go to page 1. If you typed HGR2 from page 1, PRINT CHR$(16) to return to page 1 or PRINT CHR$(2) CHR$(16) to go to page 2.

INDEX

Warranties and Limitations of Liability

Beagle Bros®
INDOOR SPORTS

Copyright © 1982, Mark Simonsen
Published by BEAGLE BROS INC.
4315 Sierra Vista, San Diego, California 92103
619-296-6400