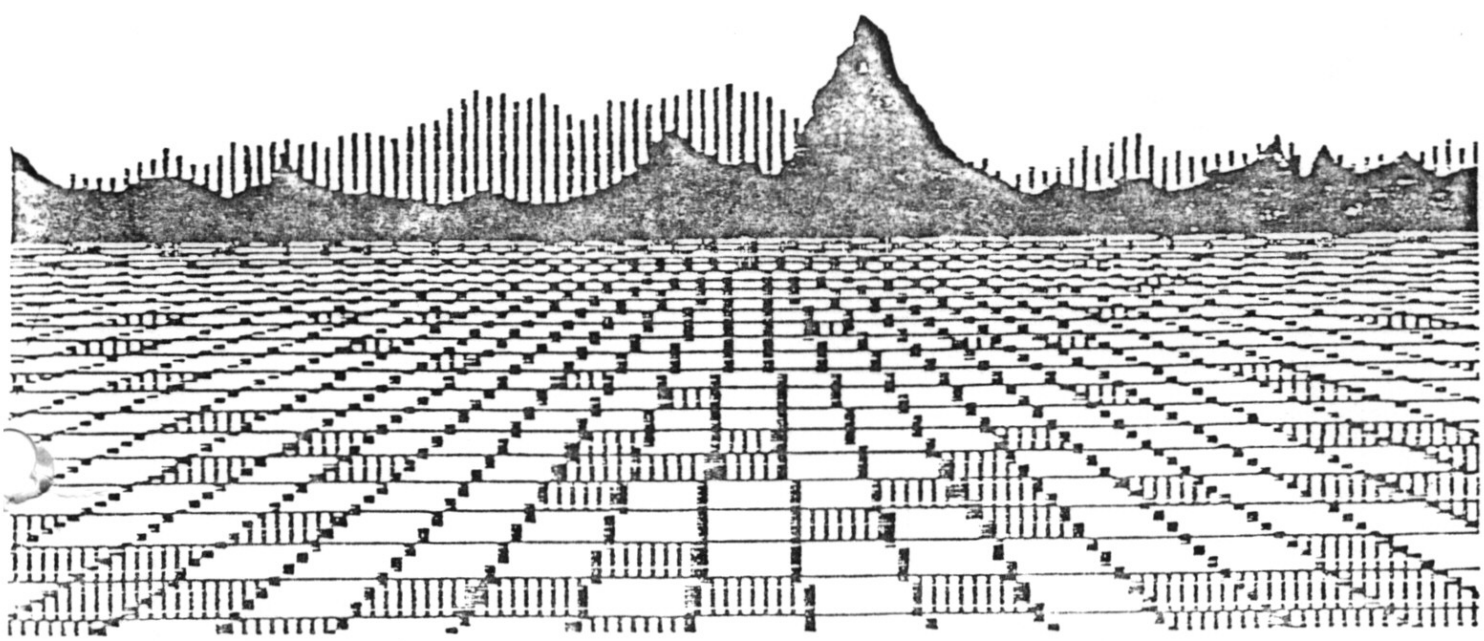
# COLORKIT REFERENCE MANUAL

## ARIZIN INC.

PRICKLY-PEAR SOFTWARE
9234 E. 30th ST.
TUCSON, AZ. 85710



# THE COLOR COMPUTER TOOLKIT ( COLORKIT )

Program 1.2
Manual 1.5


PRICKLY-PEAR SOFTWARE
9234 E. 30th Street
Tucson, Arizona 85710
(602) 886-1505

COLORKIT, 27500, 32740, 27500 in 32K
&H6B6C,&H7FE4,&H6B6C

(11100, 16340, 11100 in 16K)

The COLORKIT is a machine language program that provides the user with several changes and enhancements to BASIC, along with several powerful utilities useful in the development of BASIC programs, all available with just a few keystrokes. The KIT is usable for running continuously in the system in a transparent fashion, or for loading only when a Tool is required, used, then removed.

COLORKIT is 5.2 K bytes long. It is fully relocatable, so you are free to put it anywhere, but if you put it in the lower 32K, you should ensure that it is safe from getting written over by BASIC with the appropriate CLEAR command:

| Memory | KIT start | CLEAR parameter |
|---|---|---|
| 32K | 27500 | CLEAR 300,27000 |
| 16K | 11100 | CLEAR 300,11000 |

Note: The 300 in the CLEAR is arbitrary. Replace it with whatever amount of string space you need.

The COLORKIT is runnable on both TAPE & DISK systems without modification. To Backup:

(C)SAVEM"COLORKIT",27500,32740,27500
((C)SAVEM"COLORKIT",11100,16340,11100 on 16K)

NOTE: Loading the KIT off DISK without an appropriate CLEAR may garble the DISK!!!

The COLORKIT is written in position independent code, relocatable on LOAD with:

(C)LOADM"COLORKIT",(New Address-Current Address)

If (New Address-Current Address) is negative then ADD 65536.

The offset for loading a 16K version into 32K is 16,400:

CLEAR300,27000
CLOADM"COLORKIT",16400

There are about 27 free bytes at the end.
Loading 32K into 16K is done with a 49136 offset, leaving about 43 bytes free.

The COLORKIT modifies and uses only the BASIC ROM 'hooks', which can be reset to their original condition by the .BYE command, the total EXIT. Thus any BASIC program, variable data or top of memory CLEAR remains undisturbed.

And a RETURN to a previous UTILITY program can be made (like another enhancement package that may use the hooks). The rule for compatability between the KIT and another utility: always EXEC the other program first, then you can go into the KIT with an EXEC 27500, and go out with a .BYE back to the first utility. The KIT's storage/replacement of modified hooks makes it work, but don't try it the other way, EXECuting from the KIT to the other utility. This may seem to work, but might violate the other programs system. In other words, the KIT can handle it, the other may not.

The COLORKIT assumes you have CLEARed to at least 27500 (or 11100 in 16K). Any top of memory CLEAR lower than this will reserve the extra portion of memory for the programmable FUNCTION keys. The key buffer runs from the CLEAR Memory location up to the beginning of the KIT.

A couple of hundred bytes for the keys are plenty. Any time you have less than 255 bytes left in the buffer, you are told the number of bytes available for programming, and can never write outside of the Buffer.

The COLORKIT can be loaded into memory above 32767 in a 64K RAM mode system. However, since the function key buffer is normally between CLEAR and the COLORKIT, the KIT will treat all the ROM space as it's own. So modification is required; consult the 64K modification section for more information.

The COLORKIT cannot be saved properly while running, (due to a cold/warm start flag) so only save it after you (C)LOADM it and before you EXEC it, or after you type '.BYE'. If you would like it to be 'initialized' on (C)LOADM to your specifications, with function keys setup, LITE/DARK/SNLF/DBLF/KLON/KLOF, etc; then EXEC, set up the functions/attributes desired, type .BYE, and type :

```
(C)SAVEM"FUNKIT",27000,32740,27500
                 11000,16340,11100
```

This will store both the key functions and KIT together. Or you can save and load the function buffer seperately:

```
(C)SAVEM"FUNCTION",27000,27499,27500
                   11000,11099,11100
```

You can store any number of buffers and load them in when needed. Just ensure that the beginning of the Buffer (CLEAR Value) and KIT addresses are the same when loading as when the buffers were saved.

NOTE: It is illegal and plain wrong to copy this or any other copyrighted program for anything other than BACKU purposes. Considering that this program is a good product, and that the price is reasonable, purchasing this progr will insure having the very latest version of the program and manual, and open up a line of communication between are open for suggestions on how to improve this program, either by adding, removing, or modifying certain features Already, two updates/improvements have been made to the program, and five to the manual.

Ask yourself, "Is copying worth sleepless nights, recurrent nightmares, and blown processor chips?"

NOTE: If you do happen to 'find one on the ground', or accidentally hit (RECORD) on your tape recorder while 'reviewing' a friends KIT, or something, and find it is to your liking, then please send a donation (to quiet tha guilty conscience!) along with any comments (most important), even if anonymously. This will at least give us mor feedback than normal, with which to add to our base of user requests, for use in the design of the major 2.0 upda the KIT.

## HOW THE COLORKIT WORKS

Built into many BASIC ROM Routines is a jump down to a TRAP or HOOK table in memory at page 1 (from 350/&H15E to 424/&H1A8).

Normally, a RETURN instruction in the table simply returned processing to the ROM. But if a JUMP to [ADDRESS] instruction was in the table instead of a RETURN, then another routine can be run, altering the normal course of the ROM, and allowing for additions to or replacements of the BASIC ROM routines.

On power up, both the EXTENDED and DISK BASIC ROM'S initialize the RAM to RETURN instructions but change some ROM Traps like the General Print Trap at 359, that adds device numbers in DISK EXTENDED, or the Interpretor Trap at 410 to add TRON/TROFF, CSAVEM/CLOADM etc. in EXTENDED BASIC.

The COLORKIT (on initial EXEC) stores the current state of five RAM hooks/traps and replaces them with jumps to KIT routines which perform various run time effects, such as Keyclick or Dark Screen; then RETURNs to the ROM.

The LINE INPUT TRAP searches for a .KIT command being input while in the 'idle loop' or direct immediate mode, and also adds Screen Editing.

The GENERAL PRINT TRAP handles Screen Effects like Dark Screen, Function Key printing, Print Delay, Echo and Text Screen control.

The KEY INPUT Trap adds Keyclick.

The LIST Line Trap adds a beginning of line marker to every line listed, to clearly identify each line, and for use with the Full Screen Editor.

The RUN Time Interpretor Trap handles RUN Delay, (BREAK) key control, and TRON/TROFF line number printing, along with techniques to speed up BASIC.

Some KIT commands call up stand alone machine code routines (like .DATA or .BLOK), while other commands simply turn on or off a flag used during run time trap processing (like .KLON, .DARK, .PDLY).

A .BYE command resets the hooks to their original state and effectively turns off the KIT.

A COLD/WARM flag prevents restorage of traps if an EXEC to the KIT is performed while running the KIT, thus once turned on, any number of subsequent reEXEC's can be done, which will just print the LOGO, and set up keyclick frequency. Without the flag, if you ever EXECuted the KIT while already running it, you could never turn it off.

## EXCLUSIONS

This Software is sold on an 'as is' basis. There shall be no liability or responsibility to the user with respect to any problems encountered.

We welcome reports of any bugs or problems should they occur. Just let us know.

If you have trouble loading and suspect a bad tape or disk, it will be replaced free for a period of sixty days after purchase with return of the bad copy.

We believe the software to be free of fatal bugs but with a program of this size and flexibility and its inherent linking in of BASIC and the user program, we may not have covered every BASIC programs possible side effects. If your program does things like CLEAR above the Kit or modify the traps or execute exotic Machine Code, it would be best to .BYE before running the user program.

## ADDRESS INPUT IN THE KIT

Certain commands in the KIT require the entry of addresses. In all cases the following conventions apply:

Default entry is in HEX, with automatic entry after 4 keys are entered (ex: FF22 ).
Decimal numbers can be input by first preceding the number with a . (period).
When Decimal numbers are input, the automatic entry is after 5 digits are entered (ex: .65314 ).
Addresses with less than 4 HEX or 5 Decimal digits may be input with (ENTER) (ex: 600(ENTER), .25(ENTER))
Backspacing for corrections is available anytime by pressing the left arrow key.
In default HEX mode, number keys 0-9, letters A-F, and (BREAK) are allowed.
When in Decimal mode (first character a . ), only keys 0-9 and (BREAK) are available for input.
If only (ENTER) is input, or (BREAK) is input anytime, then a 0 is returned.
In .DUMP and .MEM, you may restart and enter new addresses by pressing the left arrow key.

## USING THE COLORKIT IN HIGH MEMORY

The COLORKIT can be loaded and run in high memory in a 64K RAM mode system. However, to properly use the programmable function keys, several changes must be made.
To facilitate the conversion, use the .MEM tool to convert a normal COLORKIT, then .BLOK copy the KIT up after the alterations.

$ 7368  to 8E 7F FF 12

$ 73C5  to 8E 7F FF 12

$ 7481  to 8E 7F FF 12

$ 74A5  to CE 7F FF 12

$ 7858  to 18 8E 7F FF

You have now told the KIT to end the key buffer at $7FFF. If you desire the top of the buffer to be at a different address, in order to reserve the memory for other routines, then replace the 7FFF's with the address (+1) of the desired top of buffer. (or the address of the first reserved byte.)

You now have all the features and protection that the normal KIT's programmed keys had, and the start of the buffer is still at the CLEAR value. CLEAR 388,32888 should give you plenty of room.

In 64K RAM mode, if you hit RESET while running the KIT, you are put back in RAM mode. Since the KIT in high memory is now 'invisible', the system will crash when "OK" is printed since the print hook into the KIT is still active, but the routine isn't available.

To RESET:

1. Enter .BYE
2. Hit RESET

To restart the KIT:

1. Enter POKE &HFFDF,8 (to put you in RAM mode)

.DATA - DATA PACKER - The KIT offers two ways to store machine code, memory or other data within BASIC itself. .MMRG (described elsewhere) stores code invisibly behind BASIC. .DATA will convert the code to 'visible' BASIC DATA statements that are created if no program is resident or appended to the current BASIC pogram, storing up to 80 bytes per line stored in HEX format. You can then READ and POKE it back into memory by using this short one line routine:

```
FOR ADD=(START ADD)TO(END ADD):READ A$:POKE ADD,VAL("&H"+A$):NEXT
```

(ADD and A$ are any dummy variables)

If there isn't enough room to convert all of the code to DATA (you need approx. 3 times the memory), then the conversion will abort where it ended, and an .OUT OF ROOM error will occur. You should DELete the converted DATA lines, modify CLEAR, and try converting smaller pieces of code.

To use:

1. (C)LOAD the BASIC program with the one liner built in (unless already in memory)
2. Enter .ADD if required (or if not CLOADMed)
   2a. Enter start address and end address
   2b. Hit (ENTER) to not modify EXECUTE address
3. Enter .DATA

The BASIC and machine code parts are now 'merged' together, and can be (C)SAVEd or (C)LOADed as one ordinary BASIC program. When RUN, line 32 puts the machine code data back into memory, after which you reference it with EXEC or USR(n), or whatever your application calls for.

.DELR - DELETE REMS - Deletes all REM's, ' type REM'S, text after the REM's, and any and all colons immediately prior to the REM's. Should the REM occur at the beginning of the line, then the entire line will be deleted, with a "DELETED LINE LN" message displayed.

It is poor programming practice to GOTO or GOSUB to a REM line, (though I do it myself). This should be checked for by using .GBL to search for a tokenized GO LN ( reverse-slash; letter G; letter O; wildcard space [for TO or SUB]; and the line number deleted ); or just look for the line number itself without the GO's.

The number of bytes deleted will be displayed.

.DELS - DELETE SPACES - Deletes all spaces in a BASIC program except those occurring in a Print string, after a REM statement, or in a DATA line.

The number of bytes deleted will be displayed.

.GBL - GLOBAL SEARCH - This command will search the current BASIC Program for a particular string of up to 11 characters and print the line containing the string if found.

Entering the command . ( one dot ) will search again starting at the line following the last search line. If the string pattern was not found then no line is printed but the global pointer is reset back to the beginning of the Basic program.

If you need to find a BASIC command then you must tokenize the string first by putting a reverse slash ( shift-clear ) as the first character in the string to be searched for.

The KIT comes up ready to search for a CLEAR command. (CLEAR & POKE commands are both good words to look for in an unknown program to avoid possible conflicts with the KIT when the BASIC program is RUN).

A space in the Search string is a 'wildcard' character, which will match against any character. If the search string contains only a space wildcard character, then it will print every line in the program sequentially by just hitting '.' (ENTER).

If no BASIC program exists, .GBL will abort.

NOTE: The .GBL will print all matching lines but may also print out additional lines when looking for BASIC keywords because some double token commands share token values with single token ones.

. - NEXT .GBL - A single dot with (ENTER) will print out the next sequential ocurrance of the .GBL string if one exists. If no line matches, then an .OK will appear and the search line number is reset to the beginning of the program. See .GBL.

.MMRG - MACHINE CODE MERGE - Merges the present BASIC program with a block of memory pointed to by .ADD, or with the most recent program read in by a CLOADM command, and returns the length of the machine code (+3).

To use:

1. (C)LOAD the BASIC program (unless already in memory)
2. (C)LOADM the machine code (or point to memory with .ADD (required if LOADed off DISK))
3. Enter .MMRG
4. Use the value returned when defining the EXEC or USR entry point in the Basic program :
   ML=PEEK(27)*256+PEEK(28)-value
   EXEC ML (or DEFUSRn=ML:Z=USRn(n))

If the EXECute address of the machine code is different from the START address, (or multiple entry points or USR's are needed) then add the difference(s) to ML.

These are the steps required to create a BASIC/Machine Code hybrid program.
Enter this BASIC Program:

```
10 ML=PEEK(27)*256+PEEK(28)-'value'
20 EXEC ML
```

Then use the KIT to handle the Machine Code portion:

1. Load machine code or data (if not in memory)
2. Enter .ADD if needed (or if not CLOADed)
   1a. Enter start address and end address
   1b. Hit (ENTER) to not modify EXECUTE address
3. Enter .MMRG, which responds with 'value'
4. Use 'value' in line 20 of the BASIC program:
   20 ML=PEEK(27)*256+PEEK(28)-'value'

When the program is run, the machine code program that was pointed to by .ADD (or by a CLOADM) when you entered .MMRG (which had been appended to the end of the BASIC program), will now be executed.

The machine code merged will not LIST and will only go away if you enter NEW or a DELn-.

You can still edit the BASIC program (insert/delete), and the Machine code will ride up and down safely at the tail of BASIC, always being pointed to properly when RUN, by the setup command.

Since the two parts are merged together, they can be (C)SAVEd or (C)LOADed as one ordinary BASIC program.

The Machine code must of course be relocatable.

The 'value' returned is three higher than the actual size of the code you specified, because .MMRG adds 3 0's to the end of BASIC (+ code), to ensure compatibility with DISK.

So if you point to a 100 byte block of memory, then .MMRG will return a 103, and your setup command should look like this:

```
ML=PEEK(27)*256+PEEK(28)-103
```

NOTE:Don't append a BASIC program to one that has code/data MMRG'ed to it. .MPRG first.

Due to an oversite, no 'crash-proofing' was implemented in this tool. Make sure you have the correct addresses with .SAV (modify using .ADD if necessary), and that you have enough memory with .VAR, or ?MEM. You need (size of code + 3) free memory bytes before .MMRGing.

.MPRG - MACHINE CODE PURGE - Removes any and all hidden Machine Code, Data, etc, at the end of BASIC. (Put there by ML MERGE)

.OLD - RECOVER LOST PROGRAMS - If you accidentally entered NEW and lost your program, then this can recover it. You must enter .OLD before any variables are written to, PCLEAR's are attempted, and no new BASIC lines are entered.

Under these conditions, .OLD should recover a LOST BASIC program, one that was masked during NEW, BACKUP, or DSKINI.

Recovering a lost program without the Kit resident:

1. CLEAR if necessary.
2. (C)LOADM the KIT.
3. EXEC the KIT, type .OLD, and .BYE if desired.
4. LIST to confirm recovery.

NOTE: The KIT will not recover any machine code MMRG'd to BASIC. And in some instances an .OLD will have no real program to try and recover, and will do its best to make variable space or free memory look like BASIC on a LIST. Just enter NEW to get rid of it, even if you have a BASIC program hidden with .PROT, it will not be modified.

.PROT - PROTECT BASIC - Hide the Basic Program that's in memory now, so CLOAD will not destroy it, DEL, EDIT, LIST, RUN, etc, cannot access it. The program is invisible for all practical purposes.

This is used in merging BASIC programs, or in covering up the current program so another one can be (C)LOADed, RUN or worked on, then removed with a NEW command followed by a .REST to uncover the original program.

.REST - RESTORE BASIC - Finds BASIC "hidden" by .PROTect BASIC. If a Basic program was CLOADed or typed in while another program is .PROTected, then the two are merged together. The current BASIC program should have line numbers higher than those in the protected program (RENUM if necessary). If you do get line numbers out of sequence, just RENUM from the last line number in sequence. (RENUM last LN, last LN)

## MERGING BASIC PROGRAMS

1. Make note of the highest line number in the current BASIC program.
2. Enter .PROT
3. (C)LOAD any BASIC program. (or even type in a program )
4. RENUM the program to fit anywhere above the protected program's highest line number.
5. Enter .REST
6. The two programs are now merged together.

NOTE: Do not execute any different PCLEAR instructions while a BASIC program is protected.

.VAR - VARIABLE LIST - Lists all Numeric (V.) and String ($V.) variables currently defined in memory (by a running BASIC program), and all user definable BASIC functions (F-FN.).

Also dislayed are the number of string bytes currently used and the amount of string space reserved (to aid in minimizing the CLEAR command's first value).

The current top of memory designation is displayed, and the maximum range of FREE MEMory, available for loading in Machine Code for example.

Note: Due to the way BASIC works, the first two letters of any ?SN ERROR producing input line will also be entered in the Variable list.

.DARK - DARK SCREEN BACKGROUND - The entire screen is set to light characters on a dark background (Lowercase) when printing to Column 8. This mode 'highlights' the current line being typed, while darkening the rest of the screen. If the Screen Editor is enabled, screen data is converted to uppercase when the input line is processed, allowing for programming in lowercase. All program data is stored in memory in uppercase. If Lowercase characters are desired for use in the Printing related commands, then set the screen to .LITE, input the line of BASIC switching UPPER/LOWER case freely, then re-enable the screen to .DARK.

NOTE: The dark screen doesn't look very good with a Lower Case Mod Board since Dark Screen Data is really lowercase characters. Disable hardware lowercase for best results.

.DBLF - DOUBLE SPACE - Sends an extra linefeed character when a carriage return is output to the printer, for printers that don't produce their own automatic Linefeeds.

In Normal printers, this will double space all output to the printer.

.ECON - PRINT ECHO ON - Printer set up as slave to print everything that goes on the Text Screen as it is displayed. Available only if the printer is on line.

NOTE: Some printing may not be ECHO'ed depending on the how the machine code doing the printing was written.

.ECOF - PRINT ECHO OFF - Normal silent printer, no ECHO.

.LITE - LITE SCREEN BACKGROUND - Normal Screen. (Upper/Lowercase)

.PDLY - PRINT DELAY - Allows either no delay (8) or nine printing speeds (1-9) when print is routed to the screen. Holding down the space bar or certain Keys will override the delay, with shift 2 pause still available. Useful for slowing down LISTings of BASIC programs, and DISK DIRECTORY's.

Also available is Single Step mode which will only print while a Key is held down. (This mode is useful for LISTing, but can be confusing)

.PDLY can accept the Delay value following the .PDLY command (i.e. .PDLY2) or will prompt you for the value:

```
.PRINT DELAY:8-9;S=SNG STP;CR;2?
```

You may enter:

- 8 - Turn off PRINT Delay
- 1-9 - Enter New PRINT Delay
- S - Single Step Mode

The current delay is the last value on the computer's prompt line.

The Print Delay is checked on a character by character basis.

If the delays don't seem to work, unplug your joysticks.

NOTE: During CLOAD's, watch out for long print delays when the file name is printed at the top of the screen, or it might miss the beginning of the DATA if the inter-record gap is short.

.SNLF - SINGLE SPACE - Normal BASIC Printing. (no extra linefeeds to printer)

.TXON - AUTO TEXT SCR - Normal BASIC setting of the TEXT screen when PRINT or INPUT commands are encountered.

.TXOF - MAN TEXT SCR - No change of DISPLAY SCREENS, with PRINT or INPUT commands. Allows you to view the Graphics screen while in the immediate mode ( SCREEN1,n ), or Examine Page 8 ( POKE65488,8 ), or use print commands in a program while remaining in graphics, etc.

Hitting RESET or typing SCREEN8 (or .TXON) will move the TEXT screen back into view.

# KEYBOARD TOOLS

The following commands provide certain idle and/or run-time features with respect to inputting from the keyboard.

.FN - FUNCTION KEYS - There are TEN user definable function keys, or temporary storage locations for screen data, available in Screen Edit Mode. Any number key (0-9) can store and/or print a sequence of up to 250 keystrokes or screen characters.
If you want function key capability, you will have to clear some room under the KIT, and enter .FN (once) to initialize the storage buffer (unless you loaded an already initialized buffer). .FN can be used anytime to clear out all previously programmed functions.
(If the buffer hadn't been initialize once, expect the unexpected)

To PROGRAM a KEY: Hit the ? key, an equal sign "=", then the number 0-9. The KIT will respond with "PROGRAM" followed by the beginning of line marker. Type in the line or use the Screen Editor to compile a screen line, then (ENTER). The function key is now programmed with the input string.
It is possible to print other programmed keys while programming, or program other keys while already programming (nesting), though it can get confusing.

To PRINT a PROGRAMMED KEY: Hit the ? key, then any number from 0-9 (any other key exits). If the key was programmed, then the programmed text will be printed on the screen starting at the current cursor position.

If you need the "?" sign in your program for PRINT?, etc; just hit "?" twice.

PROGRAM EXAMPLE: programming key 0 with "?#-2,CHR$(":

1. Enter .FN (if necessary)
2. Press Key '?', then '=', then '0'
3. The KIT says 'PROGRAM'
4. Enter "?#-2,CHR$("
5. KIT says ".OK"
6. Check by hitting '?', then '0'

Some uses for the keys:

1. LIST 1000-1999 For refresh listings of a subroutine or part of a program you are currently screen editing.

2. ?#-2,CHR$(; JOYSTK(0); etc. Commonly used and moderately long commands.

3. ?PEEK( )*256+PEEK( +1) To get a 16 bit address from memory, print the function, fill in the blanks, and enter.

4. ?#-2,CHR$(27)+CHR$( Output to printer the (ESCAPE) character plus an additional character.

5. Storing a large portion of the screen for later reference or special editing purposes.

6. Personal notes, scratch pads, reminders, etc.

If you find that you use a standard set of programmed functions repetitively, then you can (C)SAVEM the function buffer from the CLEAR to the beginning of the KIT and then (C)LOADM it later seperately or even save it with the KIT so the functions are always ready with just a (C)LOADM of the KIT.

(C)SAVEM"FUNKIT",27000,32740,27500

NOTE: Don't use the function storage area if you intend to use the clear space below the KIT for other code or data, since this space is the Program Key buffer.
Put the code or data above the KIT if possible, by moving the KIT down with a .BLOK move or during (C)LOADM with an offset of (New Address - Normal Address. If the offset value is negative then add 65536)

.KLON - KEYCLICK ON - Audible tone on keypress to verify key acceptance, and indicate keybounce (if any).

To change the frequency (normally #234), execute a SOUND F,1 command where F is the desired frequency value. (1-255)

NOTE: Since .KLON uses the sound generator, a SOUND command in a program may change the key tone. And an AUDIO ON command will be cancelled by a keypress. Use .KLOF to enable AUDIO ON.

.KLOF - KEYCLICK OFF - Normal silent keys.

.SCON - SCREEN EDITOR ON - Enable Full Screen Editor, with full cursor control, insert, delete, etc.

SEVEN control keys are reserved by the Editor:

1. UP, DOWN, LEFT and RIGHT Keys move the cursor in the corresponding direction. The longer the key is held down, the greater the speed of movement.

2. Shift-right arrow opens up space(s) from the cursor to the end of the screen (bottom right).

3. Shift-left arrow closes up space(s) from the cursor to the end of the screen.

4. The (BREAK) key opens up one space at the cursor, depositing a white graphics character block in its place, which is the beginning of LINE MARKER.

The concept of the screen editor is this: treat the screen as a 'scrabble board', moving characters around; adding, deleting or changing them until you have constructed an input line that will then be passed to BASIC as a command or program line.

The Screen Editor will try to accept an input line from the Text screen, by starting at the (ENTER) and working backwards until it finds either a beginning of line marker or the beginning of the screen.

The line in between these two points will be fed to BASIC as if you typed the entire line in as is.

The line must have less than 250 characters, otherwise a .LINE TOO LONG error will result. (break up the line into smaller pieces and enter each piece seperately)

If .DARK screen is enabled, and since dark is really lowercase, all screen characters are converted from lower to upper case before they are passed to BASIC. If you need to preserve lower case characters for use in PRINT strings then select the .LITE screen (does not modify input strings).

If you need to input a key that is a cursor control character (arrows, etc), then type .SCOF, and type the line in with EXTENDED BASIC's line input routine, or you can at any time use the BASIC EDIT command.

.SCOF - SCREEN EDITOR OFF - Disable Screen Editor, use normal cursor, etc.

# USING THE SCREEN EDITOR

## MERGING PROGRAM LINES:

1. LIST (line#1)
2. Move the cursor just past the end of the listed line, hit (BREAK) to Mark a new line, enter LIST (line#2)'
3. After the line lists, go back to the Mark, type a ":" over it, and then close up space with the shift-left arrow key until the two lines are brought together (deleting line#2's line number).
4. Go to the end of the new whole line and hit ENTER.
5. The lines are now merged together and input to BASIC. (if the total length was less than 250)
6. Delete line #2, since it is now part of line #1.

(You can also use the Function keys to facilitate merging.)

## RENUMBER/COPY/MOVE PROGRAM LINES:

1. LIST(line number)
2. Move the cursor to type over and change the line number of the listed line.
3. (ENTER) the end of the line.
4. Delete the original line if renumbering or moving.

## REPEATING DIRECT COMMANDS

1. Enter a DIRECT command.
2. After it executes, position the cursor at the end of the command line and (ENTER) again.

You can repeat direct commands as many times as needed.
This makes multiple CSAVEM's (or any other long direct command line) much easier.

## DEBUGGING PROGRAM LINES

If you get a ?SN ERROR in a line of BASIC and can't find the error, then list the line, start breaking the line in half, and entering each half in turn in direct mode (if possible) until the error is found, repeating the 'halving process' until you track the error down.

## FORMATTING MENU SCREENS

If you need a menu or message screen in a program, you can use the screen editors capabilities to directly edit the screen with the desired message, adding and moving characters around until you are satisfied, then break up the screen and add the line number, "?" character, quote, etc., for ex.:

Move the cursor and type on the screen,

```
WELCOME TO
 TOOLKIT LAND.
  AN ADVENTURE
   OF ENORMOUS
    PROPORTIONS
```

then convert it to:

```
10 ?"WELCOME TO
     TOOLKIT LAND"
20 ?"  AN ADVENTURE
       OF ENORMOUS"
30 ?"    PROPORTIONS"
```

## INPUTTING BASIC OUTPUT

You can have BASIC produce an output to the screen after which you re-enter it to BASIC.
For example, the program can compute some values based on a comlex formula, which it then prints out in a DATA like print line. You then enter the line using the screen editor into a BASIC program as a DATA line.

# RUN TIME TOOLS

These tools are available if the BASIC program is started with a RUN LN command, where LN is the FIRST line number in the program. Or the BASIC program's first line can contain the command RUN LN with LN being the SECOND line number, so a plain RUN direct command will enable these options:

```
10 RUN 20
20 'start of program
30 ...etc.
```

Regardless of how the program is started, the COLORKIT system lets BASIC run faster than is normally possible (from 5-35%), because of the way the keyboard is checked for <BREAK> or <PAUSE> in between every BASIC command.

Normally, BASIC scans the entire keyboard (52 keys) to find if one <BREAK> or two <SHIFT+2> of only three keys are pressed. COLORKIT replaces this routine with one that directly checks the <BREAK> and <SHIFT+2> keys only, so the time delay in between every command is less.

.BROF - BREAK KEY OFF - <BREAK> Key has no effect in programs that are running unless the <BREAK> was entered during an INPUT or LINE INPUT. RESET is the only external EXIT.

The Shift 2 PAUSE function remains available.

Having the <BREAK> key off will increase the speed of BASIC slightly.

<BREAK> off is useful to prevent "little fingers" from halting programs.

.BRON - BREAK KEY ON - Normal <BREAK> Key Function.

.RDLY - RUN DELAY - Allows either no delay (0) or nine (1-9) running speeds for running BASIC programs. <SPACE> or most keys held down will override the delay during program execution if necessary.

Also available is Single Stepping (S) a BASIC program (BASIC running only while a key is depressed). When in Single Step mode, the current line number is displayed non-destructively in the upper right hand corner of the screen. Single Step can be useful for debugging purposes.

.RDLY can accept the Delay value following the .RDLY command (i.e. .RDLY5) or will prompt you for the value:

```
.RUN DELAY:0-9;S=SNG STP;CR;0?
```

You may enter:

```
0    - Turn off Run Delay
1-9  - New RUN Delay
S    - Single Step Mode
```

The current RUN delay is the last value on the computer's prompt line (normally 0).

Run Delay is performed at the start of every new BASIC line.

NOTE: If the delays don't seem to work, unplug your joysticks.

.LN. - TRACE Line Number Display Modification.

Not a command, but anytime the TRACE function is invoked, the normal current line number display [LN] is replaced with .LN. for extra clarity.

# MEMORY TOOLS

The following tools perform operations on or with memory.

.ADD - MACHINE ADDRESS - Certain tools (.DATA, .MMRG, .SAV) expect the start and end address of machine code/Data to have been setup by a CLOADM command. The .ADD command allows you to manually set these addresses up for special situations or if the Machine code was loaded off DISK, which does not set the addresses up like a cassette load.

Just input the addresses in Hex or Decimal when prompted. You can change the EXEC address if desired, though this is not normally necessary. Hitting <ENTER> in response to the EXEC prompt will not modify it.

The .ADD does an automatic .SAV to verify your entry.

.BLOK - BLOCK MOVE - This tool lets you move or copy any portion of memory (machine code, Variable space, ROM to RAM, etc.) to any place in memory. You tell the KIT the start and end addresses of the block, and the KIT will move it anywhere (careful!) up or down.

After the addresses are entered, a "C" key will make a copy of the Block, leaving the original memory unmodified. If an "M" key is input, then the code is moved, with the original bytes cleared to 0. Any other key will abort the move/copy entirely.

.DUMP - DUMP MEMORY - It is sometimes handy to look at memory or have a hard copy of it.

.DUMP will print out to the screen or printer in ASCII or HEX any portion of memory. Any key pressed will toggle dumping on/off. If print delay is enabled but you want to overide it, hit any key to pause dumping, then any key to continue but hold the key down for as long as you want speeded up dumps.

Entering a left arrow while DUMPing will return to the start of the DUMP routine to enable DUMPs from a new address.

Break during the DUMP is the exit.

Default <ENTER> responses to the prompts will set up an ASCII Dump to Screen from 0000 to $FFFF.

.DUMP COMMAND SET:

LEFT Arrow - Enter new DUMP Addresses, etc.
BREAK - During normal dumping, EXIT's .DUMP tool
Any other key- Toggles PAUSE ON/OFF
Any two key presses with the second key held down overides print delay if selected.

.MEM - MODIFY MEMORY - For a better look into memory than .DUMP provides, .MEM allows a more complete examination and modification of memory.

Memory addresses are displayed in Hex and Decimal, with memory contents displayed in Hex, ASCII, Decimal, and double Decimal (16 bit value equal to the current memory cell's contents * 256 plus the next memory cell's contents).

The up and down arrows serve to scroll up or down through memory (with auto repeat).

New hex data may be entered into the current memory cell at any time when a flashing "H" is visible at the end of the line, while ASCII string data may be entered directly by first pressing <shift-CLEAR>, which will then display a flashing "S" at the end of the line. (Hex is the initial mode, and can be returned to from ASCII input mode by pressing <CLEAR>).

If no initial address is specified or the BREAK key was hit, then 0 is used as the current address.

.MEM COMMAND SET:

UP Arrow - List previous memory address
DOWN Arrow - List next memory address
LEFT Arrow - Enter new Memory Address
SHIFT-CLEAR- Enable ASCII string memory modify mode "S"
CLEAR - Enable Hex memory modify mode "H"
BREAK - During hex memory modify, aborts entry if BREAK is second hex digit input
BREAK - During normal examining, EXIT's .MEM tool

.SAV - CSAVEM/CLOADM - This command will display the last CLOADM/CSAVEM'ed Machine Code File's Name, Start, End, & Execute Address, in a format suitable for Direct entry using the Screen Editor, allowing for easy backup copies of Machine Code tapes.

After a CLOADM (not LOADM), the following addresses contain the following data:

| | |
|---|---|
| 474-481 | 8 character NAME |
| 487,488 | START ADDRESS |
| 126,127 | END ADDRESS (+1) |
| 157,158 | EXECUTE ADDRESS |

The .SAV command displays this information automatically.

## BACKING UP MACHINE CODE PROGRAMS

1. CLOADM the machine code program.

2. Enter .SAV

3. COLORKIT will respond with the following line preceded with the Screen Editor's beginning of line marker.

```
*CSAVEM"FILENAME",(start),(end),(exec)
```

4. Use the arrow keys to position the cursor at the end of the CSAVEM line, then (ENTER). (remove the 'C' in the CSAVEM for DISK saves)

5. If you just want the tape information without a full load (to avoid autostart or possibly wiping out the KIT or other program when the tape CLOADMs in memory), then CLOADM like normal, but stop the tape just after the header block (when the tape name gets printed at the top of the screen), hit RESET, and enter .SAV. The tape name, start and execute addresses will be valid, but the end address will not since that is determined by actually counting each byte as it gets read in during a full CLOADM.

## TOOLKIT CONTROL COMMANDS

These two commands concern the KIT itself.

.BYE - REMOVE TOOLS - Unhook all Traps, turn off the startup flag, return all modified addresses to their previous state (this means that you can be in some other utility system, EXEC the KIT and run it, then .BYE and return to the other utility).

To fully remove the KIT, you must enter an extra single (ENTER) in addition to the .BYE(ENTER) (i.e. .BYE (ENTER)(ENTER)).

After the first (ENTER), the KIT is effectively turned off, but you can optionally still type in any number of KIT commands, until the second plain (ENTER) is input.

The reason for this is that certain tools can be used on the KIT itself, like a .BLOK copy of the KIT to a different place in memory. The copy of the KIT will start properly since it was copied with the proper off state of the startup flag.

(If a copy of the KIT is moved or copied to memory or tape while the KIT is running, it will not run properly because of the startup flag. EXEC the KIT 4 bytes higher than the normal start address to perform a cold start, then EXEC again at the beginning to reset the EXECUTE pointer, or use .ADD to redefine it)

After the COLORKIT is turned off, it can be safely written over, or saved. Removal is required in order to (C)SAVEM this program properly because of the startup flag. The options in effect when the COLORKIT is copied will also be in effect in the new copy when it EXECutes.

.HELP - COMMAND SYNTAX SCREEN - List all KIT commands.

The command names in the .HELP screen as displayed are used in syntax checking the commands, so any changes to this table will alter the command names themselves. See the end of the manual for information on altering command names.

The address where the COLORKIT is currently located is displayed on the last line.

.SCOF temporarily before (C)LOADing a BASIC program that was saved in ASCII format, (or in DISK MERGE commands) otherwise the load will abort after the first line is input.

BUG: In the normal line editor, to bypass a line of BASIC that was typed in direct mode, you can hit shift-left arrow to wipe out the whole line, or hit BREAK, which moves the cursor to the next line without wiping out or executing the line. However, in the .SCOF mode, BREAK acts like an ENTER key, so only use shift-left.

Typing in SCREEN,1 will enable the ORANGE TEXT screen for LISTing or EDITing, since the Color Set parameter is now used to determine the screen color. Normal BASIC does not allow this. To disable this feature, (to keep the orange screen from coming up if you use SCREENn,1 in a program) change 75E2 (.30178/13778) from $F7 (247) to $8C (140).

Remember to start the RUNning of a BASIC program with RUN(LN), where LN is the first line in the program, to enable RUN DELAY, BREAK OFF, etc. (if selected) or Modified TRON printing. Or have the first line of BASIC do a RUN (LN) to the second line of BASIC so a plain RUN will allow these options.

When using the DELETE SPACE tool, keep in mind that any subsequent lines edited must have the spaces put back after Variable names immediately followed by BASIC commands, before being entered.

After a .BYE command you are still allowed to enter additional KIT commands, so you can properly .BLOK move the KIT itself, crunch it to .DATA statements, etc. To fully REMOVE the system, type in anything other than a KIT Command (or just hit (ENTER)).

Due to the increased speed of BASIC programs running in the KIT, critical timing loops set up in a timing program may have to be changed.

Since the keyclick uses the sound generator, an AUDIO ON command will be cancelled by a keypress. Type .XLOF to allow the AUDIO ON.

In case of any problems, first .BYE to see if things are OK. Then EXEC again. (You can EXEC the KIT any number of times)

There are three distinct EXECute Modes of the KIT:

1. The first EXEC, which does a complete start including saving of the current trap state, keyclick frequency setup, and the LOGO.

2. Any subsequent EXEC's, which only set keyclick frequency and display the LOGO.

3. EXEC 4 bytes higher than the ordinary start address of the KIT, which forces a hard start in improperly saved KIT's.

If you find that a backup copy EXEC's with the sign-on LOGO, but doesn't work properly, the Kit may have been saved improperly while running. In this case EXEC the Kit 4 bytes higher than Normal (EXEC 27504/11104). Then EXEC 27500/11100 to reset the EXECUTE pointer, or use .ADD. Be sure to then .BYE and (C)SAVEM the KIT properly.

# MISCELLANY/MODIFICATION

. The difference between a 16K and a 32K system is 16400. Some addresses referred to in the manual are for 32K, so subtract 16400 for determining the 16K address.

. The one copy of the Kit will run AS IS unmodified on Tape or Disk, and in 16 or 32K.

. Expect the unexpected when loading the Kit in uncleared areas. Do this normally:
CLEAR 200,27000/11000 (500/100 bytes of Function space)
(C)LOADM"COLORKIT":EXEC

. The keyclick pitch uses the last SOUND frequency parameter. To Modify the Keyclick pitch at the EXEC of the Kit (normally #234):
POKE 27509/11109,(freq)

. To Modify the Command Set, simply enter .MEM at address 7DE6 (.32230/15830), enable ASCII input mode with a shift-clear, then locate and modify the command desired. Then BREAK to EXIT. Confirm the command change with a .HELP. Each command must begin with a dot, have up to six characters, and end with a space. Don't mess with .NEXT or anything after it.

. To change the Wildcard character in .GBL from a space to another character, ("." for example) POKE 70BA (.28858/12458) with the ASCII value of the new character. Don't use tokenizable characters. ( ?, =, /, etc. )

. The Cursor Variable Speed Ramp Value is in 7F65 (.32613/16213). Put in a 1,2,3, or 4 to obtain different ramps/speeds. 3 is normal.

. The Delay value for the cursor's color cycling (16 bits) is at 7F5C (.32604/16204). $0080 is normal.

. To change the color of the beginning of line marker, (to reduce contrast on B/W TV's, etc.) change locations $7A74 (31248), $79F7 (31123), $7899 (30773), and $6C83 (27679) to the Value desired. Use the .MEM tool to do this easily, but stay in the .MEM tool and change all four locations before exiting. Use values ) $CF. Also change $7496 (29846) & $7FDD (32733)

. The overall cursor movement speed can be modified slightly by altering $7F6A. ( 8 bit ) $0C is normal. This will also affect the choppy look of the moving cursor.

. The Unit Print Delay Value (16 bit) is in 7F5E (.32606/16206). ( normally $0040 )

. The Unit Run Delay Value (16 bit) is in 7F60 (.32608/16208). ( normally $00A0 )

. If you don't want to use the "=" sign to program the Function Keys, then modify the ASCII value in 7359 (.29529/13129) to the value of the key desired. The key has to have an ASCII value higher than "1" ( $31 )

. Though the KIT wasn't set up for it, you can enable a fair auto key repeat with the following pokes:
$78AB, from $59 to $5A
$7985, from $FF to $FE
$7986, from $34 to $EB
$7F64, from $0A to $07

. The Kit will be upgraded time to time to add improvements, modifications, bug fixes, etc; any minor updates possible on older systems that can be patched or poked will be sent free of charge to users. Major change updates can be purchased for half price. Newer Manuals are $2.00.

# KIT COMMAND SUMMARY

.ADD - MACHINE ADDRESS - Set addresses for .DATA, .MMRG. <ENTER> on EXEC prompt will not modify EXEC ADDRESS.
.BLOK - BLOCK MOVE - "C" key copies Block, source memory unmodified. "M" key moves code, source cleared to 0.
.BROF - BREAK KEY OFF - <BREAK> Key has no effect in running programs
.BRON - BREAK KEY ON - Normal <BREAK> Key Function. Must RUN "LN" to begin BASIC.
.BYE - REMOVE TOOLS - Unhook Traps, turn off startup flag, return modified addresses to previous state. ( .BYE <ENTER><ENTER> ).
.DARK - DARK SCREEN BACKGROUND - Screen set to light characters on dark background. For Lowercase, enter .LITE, input line, enter .DARK.
.DATA - DATA PACKER - Converts machine code to BASIC DATA Statements. Readable with:
FOR ADDR=<START ADDR) TO <END ADDR):READ A$:POKE ADDR,VAL("&H"+A$):NEXT (ADDR and A$ are any dummy variables)
.DBLF - DOUBLE SPACE - Sends extra linefeed with carriage return to printer.
.DELR - DELETE REMS - Delete all REMs, ' REMs, all text after REMs, and any semi-colons in front of REMs.
.DELS - DELETE SPACES - Delete all spaces except in PRINT strings, DATA and REM lines. After use, reinsert spaces (if re-editing) after variable names if required.
.DUMP - DUMP MEMORY - Dumps to screen/printer in ASCII/HEX any portion of memory. Any key toggles dumping on/off. Left arrow while DUMPing restarts DUMP routine, <BREAK) during DUMP to exit.
.ECOF - PRINT ECHO OFF - Normal silent printer, no ECHO.
.ECON - PRINT ECHO ON - Printer set as slave to print everything on the Text Screen as it is displayed.
.FN - FUNCTION KEYS - TEN function keys, in Screen Edit Mode. Any number key (0-9) can store/print up to 258 keystrokes or screen characters. CLEAR room, and .FN.
To PROGRAM: 2, =, number 0-9.
To PRINT: 2, number 0-9.
For "2" sign, hit "2" twice.
To save KIT + KEYS: (C)SAVEM"FUNKIT",27888,32677,27588
.GBL - GLOBAL SEARCH - Search for and list line containing string. Shift-clear to tokenize search string, space in search string is 'wildcard'.
.HELP - COMMAND SYNTAX SCREEN - List all KIT commands. Current COLORKIT address on last line.
.KLON - KEYCLICK ON - Audible tone on keypress. SOUNDn,n to change frequency (1-255). Use .KLOF to allow AUDIO ON.
.KLOF - KEYCLICK OFF - Normal silent keys.
.LITE - LITE SCREEN BACKGROUND - Normal Screen. (Upper/Lowercase)
.MEM - MODIFY MEMORY - Memory addresses in Hex and Decimal, memory contents in Hex, ASCII, Decimal, and double Decimal (16 bit)
UP Arrow - list previous memory address
DOWN Arrow - list next memory address
LEFT Arrow - Enter to list new Memory Address
SHIFT-CLEAR- Enable ASCII memory modify mode "S"
CLEAR - Enable Hex memory modify mode "H"
BREAK - During hex memory modify, aborts entry if BREAK is second hex digit input
BREAK - During normal examining, EXIT's .MEM tool
.MMRG - MACHINE CODE MERGE - Appends machine code to current BASIC program.
1. (C)LOAD BASIC
2. (C)LOADM machine code (or .ADD)
3. Enter .MMRG
4. DEFUSR=PEEK(27)*256+PEEK(28)-value returned
.MPRG - MACHINE CODE PURGE - Removes .MMRGed code
. - NEXT .GBL - A single dot with <ENTER) to list next ocurrance of .GBL string.
.OLD - RECOVER LOST PROGRAMS - NEW, BACKUP, or DSKINI BASIC Recovery.
.PDLY - PRINT DELAY - No delay (0) or (1-9) printing speeds when printing to screen. Keys override delay. Single Step mode (S) prints only while key held down. Enter .PDLYn or .PDLY will prompt for value:
.PRINT DELAY:0-9;S=SNG STP;CR;2?
Current delay is last value on line. If the delays don't work, unplug your joysticks.
.PROT - PROTECT BASIC - Merging, hiding BASIC programs. RENUM higher than hidden program if appending.
.RDLY - RUN DELAY - No delay (0) or nine (1-9) running speeds for running BASIC programs. Keys overide delay. Single Stepping (S) BASIC (RUN only while key pressed. .RDLY can accept Delay value following .RDLY command (i.e. .RDLY5) or will prompt for value:
.RUN DELAY:0-9;S=SNG STP;CR;0?
Current delay is last value on line. <ENTER) will not modify delay. If the delays don't work, unplug your joysticks.
.REST - RESTORE BASIC - Finds BASIC "hidden" by .PROT. If program lines out of sequence, RENUM (last in-sequence line),(last in-sequence line)
.SAV - CSAVEM/CLOADM - Display last CLOADM/CSAVEM'ed Machine Code File's Name, Start, End, & Execute Address.
474-481 8 character NAME
487,488 START ADDRESS
126,127 END ADDRESS (+1)
157,158 EXECUTE ADDRESS
.SCON - SCREEN EDITOR ON - Up, down, left, right cursor control with arrow keys. Shift-right arrow opens up space from cursor to end of screen. Shift-left arrow closes up space from cursor to end of screen. <BREAK) opens up one space at cursor, leaving white graphics block (beginning of LINE MARKER).
Screen Editor accepts input line from Text screen, starting at <ENTER) and working backwards until beginning of line marker/beginning of screen found. If .DARK screen enabled, screen characters converted from lower to upper case. If you need lower case, select .LITE screen.
If you need a cursor control character (arrows, etc), enter .SCOF, type line with EXTENDED BASIC's line input routine, or use EDIT command.
.SCOF - SCREEN EDITOR OFF - Disable Screen Editor, use normal cursor, etc.
.SNLF - SINGLE SPACE - Normal BASIC Printing. (no extra linefeeds to printer)
.TXOF - MAN TEXT SCR - No change of DISPLAY SCREENS, with PRINT or INPUT commands. RESET or type SCREEN0 for TEXT screen
.TXON - AUTO TEXT SCR - Normal BASIC setting of TEXT screen when PRINT or INPUT commands processed.
.VAR - VARIABLE LIST - Lists Numeric (V.) and String ($V.) variables, String bytes currently used/reserved, Current top of memory designation, Maximum range of FREE MEMory.