Micro Technology Unlimited
~~841 Galaxy Way~~ P.O. Box 12106
~~Manchester, NH 03103~~ Raleigh, NC 27605
~~(603) 627-1464~~ (919) 833-1458

August 20,1980

Dear Keith,

Here is a preliminary copy of the KGP (Keyword Graphics Package). The cassette contains a copy which loads in a 32K PET with "new" ROMs, and a copy which loads in a 16K PET with "new" ROMs. To load in the 32K version enter the following commands:

```
POKE 53,100:NEW
LOAD "KGP32K"
```

When the program has loaded, the command SYS 256*100 will link in the KGP. To load the 16K version enter these commands:

```
POKE 53,36:NEW
LOAD "KGP16K"
```

When the program has loaded, the command SYS 256*36 will link in the KGP.

If you wish to transfer the software to disk, the 32K version resides from $6400 to $7FFF, and the 16K version resides in $2400 to $3FFF. Also, the KGP must not be linked in when it is saved!

There are plans to make one more addition to the package. This will add two more control bytes for character definitions. One control byte will be for a relative move, and the other for a relative draw. The relative distance moved or drawn will be from +127 to -128 in the X and Y directions. The move or draw will also be scaled and rotated according to the current scaling and orientation parameters.

The accompanying documentation is intended to be the first part of the final manual. It is our intention for this part to provide sufficient information so that the majority of applications can be programmed without frequent references to the full descriptions of the commands. I welcome your comments on how well we meet this goal, and how we might do better. I look forward to hearing your comments.

Sincerely yours,

Larry Isaacs

Larry Isaacs

PS. Here is some additional information which is not included in the documemtation. First, the default value for VMPAGE is $90 for KGP.32K, and $40 for KGP.16K. Also, the character table used for displaying characters occupies $7E00 to $7FFF in KGP.32K, and $3E00 to $3FFF in KGP.32K.

Since the default character set had been built before the code was saved, this table contains valid pointers to the default character set. This means that when the KGP is linked in, characters may be printed without executing CSETUP. However, after linking in the KGP, the character definition pointer (CHDLOC) will be pointing to the beginning of the default definition. You should therefore execute CSETUP before adding any definitions, so you don't overwrite the default definitions. Also, the default definitions are at the end of the KGP's code. There is not much space between the end of these definitions, and the beginning of the character table. It might be a good idea to set CHDLOC to other unused memory before adding definitions.

ADDITIONAL NOTES ON THE K-1008-6 PET GRAPHIC INTERFACE BOARD

1 - Writing the VIDEO and ENABLE CONTROL REGISTERS

The addresses for the VIDEO and ENABLE CONTROL REGISTERS are only partially decoded. Thus, the VIDEO CONTROL REGISTER may be written by writing to any address in the range 48896 to 49151 (BF00 to BFFF). The ENABLE CONTROL REGISTER may be written by writing to any address in the range 48640 to 48895 (BE00 to BEFF). Also, it should be possible to put a ROM in the B000 - BFFF address space on the K-1008-6, since reading from from ROM won't affect the control registers.

2 - THE LIGHT PEN JUMPER (U31-4 to U31-13)

When the LIGHT PEN REGISTER IC's are not installed on the K-1008-6, the LIGHT PEN JUMPER should also not be installed. Presence LIGHT PEN JUMPER in this case would prevent the reading of data from the video memory. If you install the LIGHT PEN REGISTER IC's, you must also install the LIGHT PEN JUMPER for proper operation of the board.

3 - THE SEMI-RANDOM DOT PATTERN ON POWER UP

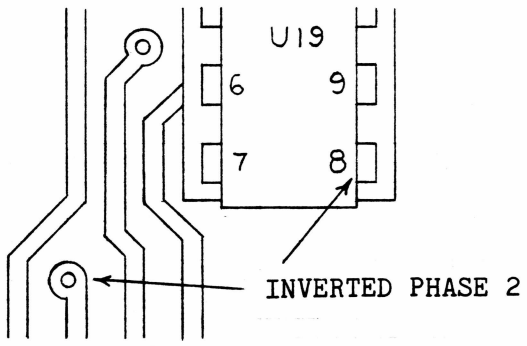
On some K-1008-6 boards, vertical bars instead of a semi-random dot pattern will be seen when the video memory is displayed after power up.

4 - USING 2716'S (Single 5 Volt Type)

The slight differences in pin assignmemts between the 2716 and 2332 normally don't cause a problem. However, because the K-1008-6 must use the first half of certain machine cycles to fetch display data, these differences do cause a problem. To use a 2716, the following must be accomplished:

1. PHASE 2 must be removed from pin 21 of the 2716 and pin 21 tied high.
2. CHIP ENABLE must be removed from pin 20 and INVERTED PHASE 2 attached to pin 20 instead.
3. ADDR 11 must be removed from pin 18 and CHIP ENABLE attached to pin 18 instead.

The recommended way of accomplishing this is to build an adapter using a 24 pin header and 24 pin socket. First solder all the pins on the header to the corresponding pins on the socket, except for pins 18, 20, and 21. Then solder pin 24 on the header to pin 21 of the socket. Next, solder pin 20 on the header to pin 18 on the socket. Now connect INVERTED PHASE 2 to pin 20 of the socket by one of the following two ways. You can solder a MICRO-CLIP to pin 20 of the socket, and clip the other end to pin 8 of U19. Or, you can solder a wire from pin 20 of the socket to the platted-through hole shown in the diagram below. This adapter will allow the 2716 to occupy the first 2K of the address space of the socket in which it is plugged.



INTRODUCTION TO THE KEYWORD GRAPHICS PACKAGE

The Keyword Graphics Package provides a means of creating sophisticated graphics images with the ease of using a higher level language, namely BASIC. When linked to the PET's operating system via a SYS command, 50 new commands are made available for your use. They may be executed as direct commands, or as statememts in a BASIC program. All of the commands are presented in tables are at the end of this section. The commands are divided into 7 tables which are entitled DISPLAY CONTROL COMMANDS, GRAPHICS COMMANDS, TEXT COMMANDS, CURSOR COMMANDS, CHARACTER DEFINITION COMMANDS, BOUNDARY COMMANDS, and MISCELLANEOUS COMMANDS. Before presenting a discussion of the command tables, this section explains some general facts and features the KGP.

The first of these features is the KGP's ability to perform a transform on the coordinates used for drawing. Transforming a coordinate first involves halving or doubling the coordinate a specified number of times. Then an offset is added to give the coordinate actually used. This means that even though the Visible Memory is limited to displaying a 320 X 200 dot matrix, your effective drawing area is $\pm$ 32767 by $\pm$ 32767. You can select a portion of that area for viewing by specifying amounts for X and Y to be shifted, along with X and Y scale factors to control how much you see.

A related feature which should be mentioned involves the cursor coordinates. There are a number of commands which use the pair of coordinates last referenced or drawn to. Because of the transform feature, it is necessary to keep two pairs of coordinates, one to save the coordinates in untransformed state, and another to save cordinates which may have been transformed. The untransformed coordinates are saved in the drawing cursor coordinates. The coordinates which may have been transformed are saved in the internal cursor coordinates. Naturally, when no transforms are enabled, these two cursor coordinates will contain the same values.

The KGP also has the ability to accept a set of boundaries and perform boundary checking. Specifying a set of boundaries involves specifying minimum and maximum values for X and Y. When boundary checking is turned on, no point outside this region will be written. Since boundary checking requires each point be checked, it will cause the drawing to slow down somewhat. If greater speed is required, boundary checking may be turned off. No matter what the state of the boundary checking, the KGP is designed not write into memory outside the visible portion of the Visible Memory.

Another feature associated with boundaries is the ability to define up to four windows. A window is simply a set of boundaries which have be associated with a window number. Also, when you switch from one window to another, the current internal cursor coordinates will be saved for later recall with that previous window. The internal cursor coordinates will then be set to the coordinates which were saved for the new window. This means when you switch windows, you can continue where you left off.

One last feature to be discussed here refers to the short commands available for a subset of the commands. When the KGP encounters a statement like A=B+C, it has to scan the whole command table to see if "A" is a KGP command or not. Naturally this slows down execution of a program. To speed execution when drawing or displaying characters, a short form of the necessary commands and a short command mode are provided. The short form command consists of a " " character followed by a letter. When you turn on the short command mode, only short commands are scaned for. This means that the KGP can tell that "A" isn't a command after checking the first character, resulting in faster execution.

As for error detection, the KGP can detect when certain values are out ot range, and in some cases, when parameters are missing. However, the majority of errors which occur during normal programming will likely be detected by the PET operating system. If only the Visible Memory is being displayed when an error is detected by the PET operating system, you will not see the error message, nor any direct indication that your program has stopped. Though it may seem like the PET has crashed, you can still enter commands at this point. However, you will not see what you type. The best procedure to follow when you suspect that an error has occured is to first hit the STOP key to make sure your program is stopped. Then execute the command " P" to bring back the PET display. If the last message is "BREAK IN LINE ...", then your program was still running when the STOP key was hit. Otherwise, you can now see the error message. If the KGP detects an error, it will enable the PET display before printing the error message.

And one final note. This manual will refer to the dots on the PET screen as being white or black. For those with green phosphor screens, references to white dots should be interpreted as green.

INTRODUCTION TO THE COMMAND TABLES

This section discusses the seven command tables, and provides supplementary information where needed. You may wish to review each of the tables as it is discussed in this section.

The first table covers the DISPLAY CONTROL COMMANDS. The first four commands, PETMEM, VISMEM, PVMEM, and NOMEM, control what will be displayed on your PET screen. You should note that the PETMEM command also has a short form. The P command will turn off the short command mode in addition to selecting the normal PET display. This will prove handy if you have to stop a program while the short command mode is on. This is also the quickest way to return to the normal PET display when an error occurs.

The next three commands, NRMDSP, RVSDSP, and FLPDSP, are used to set the display mode. By setting the display mode, you control whether "on" means white or black. Changing the display mode in no way affects the current contents of the display, only how future commands affect the display.

The GMODE command is used to set the graphics mode. The graphics mode determines what happens to the graphics display when drawing occurs. Drawing can turn points on, off, or flip their state. Changing the graphics mode also in no way affects the current display, only how future commands affect the display. To complement the GMODE command, the RDGM command will read the current value of the graphics mode into the specified variable.

The next table contains the DISPLAY CLEARING COMMANDS. Clearing, in this case, means turn the specified dots off. This can turn the screen white or black depending on the display mode. The SCFLIP command was included in this table because of its similarity to the SCLEAR command. The SCFLIP command will flip the state of each point in the specified region. Also, it should be noted that the WCLEAR command performs the function of recalling a window as well as performing a "home-clear" on that window.

Next comes the table of GRAPHICS COMMANDS. The first two commands, WRPIX and RDPIX, refenence a single point. It is important to note that the point referenced is determined by the internal cursor coordinates. Also note, if you save the current state of a point using the RDPIX command, you can later return that point this state by setting the graphics mode to the value read, then writing the point with the WRPIX command. This assumes the point was within the display area, and within the current boundaries if boundary checking is on.

The commands LINE, MOVE, and DRAW are used for drawing lines. If you have transforms enabled in some way, i.e. XFFLG = 1,2, or 3, you should make note of how each of these commands affect the cursor coordinates. If you wish to have dotted lines, you can use the DOTL line command to set the dotted line parameters.

The remaining commands in this table are involved with the coordinate transformations. Specifying negative values for XSCALE and YSCALE will cause halving of the coordinate, while positive values cause doubling. Before you use the transform capability for the first time, you should refer to Section 5.0 for further details about using the transform capabilities.

The next table covers the TEXT COMMANDS. The CHAR and AUTEXT commands are normally used for displaying text. The characters will be drawn according to the current character scale factor, and orientation parameter. These parameters may be set using the CHSCALE and CHROT commands, respectively. Minus values for these commands will cause halving of the character size and counter-clockwise rotation of the character orientation, respectively.

The ASCII character definitions supplied with the KGP are intended for displaying at normal size, or greater, and oriented on even increments of 45 degrees. Displaying them at less than normal size, or on odd increments of 45 degrees will give unsatisfactory results. Also, the AUTEXT command is intended only for printing characters horizontally. If you wish to obtain similar operation in a different orientation, you should use the CHAR command and do the testing yourself to determine when a "CR-LF" and possibly scrolling should be performed. The "CR-LF" function can be performed with the MOVE command, and the scrolling done with the SCROLL command.

Next is the table of CURSOR COMMANDS. The GRACSR command, which displays a cross-hair, is designed to always indicate where a DRAW command would begin the line. Note that if XFFLG is 1 or 2, the GRACSR command will be setting the internal cursor coordinates to new values. Using the TEXCSR command, which displays an underline, will have no effect on either of the cursor coordinates. It is important to note that both the graphics and text cursors are drawn in flip mode, so drawing it a second time at the same coordinates will erase the cursor. To determine the current contents of either cursor coordinates, the RDCSR and RDXY commands are available.

Next is the table of CHARACTER DEFINITION COMMANDS. Using these command requires familiarity with how character definitions work. Briefly, a definition consists of a byte which identifies the character being defined, followed by a series of bytes which control how the character is to be drawn, the last of which is zero byte indicates the end of a defintion. When a character definition is build, the first byte is used as the character being defined; and based on this byte, a pointer to the remaining definition is placed in a character table. Later, when that character is printed, the pointer is looked up in the current character table and the associated definition is drawn. Also, once started, the building will continue until a zero byte is encountered as the character to be defined. This zero byte normally follows the zero byte which ends the previous definition.

With these commands you can add definitions to a current set, or build separate sets of definitions with separate character tables. Also, the definitions don't have to draw letters. They may draw whatever figure you wish.

The next table contains the BOUNDARY COMMANDS. Note that using the SETWIN command will not disturb the current boundaries if they are associated with a window, unless it is the same as the window being set. The WINDOW command allows you to recall a window and continue drawing or printing where you left off. The SETBND, SETWIN, and WINDOW each turn on boundary checking when executed. The NOCHK command can be executed if you wish to turn the boundary checking off, and BNDCHK if you wish to turn it back on.

The last command table contains the MISCELLANEOUS COMMANDS. To speed up the drawing of figures, the GRSHRT command can be executed to select scanning of only the short form commands. Also, it is important that the KGP know where the Visible Memory is located. If it is at some other location than the default, the VMPAGE command may be used to set the starting page of the Visible Memory.

The command tables show the parameters required by each command. Any parameter may be either a constant or variable, including array elements, except those indicated by "var", "xvar", and "yvar". These must be a variable. The KGP will accept values within the range $\pm$32767 for all of the parameters except those indicated by "i", "bval", "bval1", and "bval2". If the parameter is indicated by an "i", only those values indicated in parenthesis will be accepted. If the parameter is indicated by "bval", "bval1", or "bval2", only values in the range 0-255 (i.e. a value which will occupy 1 byte) will be accepted. If a value is outside the valid range, an ILLEGAL QUANTITY ERROR will occur. If a parameter is missing, a SYNTAX ERROR will occur.

There are a couple of tables following the command tables. The first of these is a table of Special Function Characters available for the CHAR and AUTEXT commands. The three special functions for the CLR, HOME, and RETURN characters are always and only available to the AUTEXT command. The other function characters are a part of the character definitions that comes with the KGP, and are available only as part of the whole set.

Next is a chart which shows how to specify the vector bytes used in character definitions to control the drawing of the characters. By adding the direction number to the length number you get the number which specifies a line segment to be drawn. By specifying the proper sequence off vector bytes, you can draw any character you wish. Please note that 15 is not allowed as a length number. This case is used to indicate a control byte. Also, a vector byte of zero is used to indicate the end of a definition. These vector bytes placed in a definition using the CHDFC command.

The last table contains a list of the control bytes and their function. These control bytes are like special subroutine calls, with the byte(s) following the control byte being the parameters. These control byte functions may be used at any point in a definition, and may be nested as well. After the function has been performed, the drawing of the definition will continue with the next byte following those of required for the function. The bytes required for these special functions are also placed in a definition using the CHDFC command.

TABLE OF DISPLAY CONTROL COMMANDS

Section	Command	SC	Parameters	Description
3.1.1	PETMEM	JP	none	Displays PET screen memory contents. The short command also exits short command mode.
3.1.2	VISMEM		none	Displays visible Memory contents.
3.1.3	PVMEM		none	Displays both PET screen memory and Visible Memory contents with the K-1008-6; displays just Visible Memory contents with the K-1008.
3.1.4	NOMEM		none	Blanks the screen with the K-1008-6; displays PET screen memory with the K-1008.
3.2.1	NRMDSP		none	Selects normal display mode. Turning points on displays white dots; and turning points off displays black dots.
3.2.2	RVSDSP		none	Selects reverse display mode. Turning points on displays black dots, and turning points off displays white dots.
3.2.3	FLPDSP		none	Flips the display mode.
3.2.4	GMODE		i (0,1,or 2)	Sets the graphics mode. This controls how drawing affects the display. If set to 1, drawing turns dots on; if set to 2, drawing turns dots off; if set to 0 drawing flips the dots.
3.2.5	RDGM		var	Reads the current graphics mode into variable var.

TABLE OF SCREEN CLEARING COMMANDS

Section	Command	SC	Parameters	Description
4.3.1	CLEAR	]E	none	Clears the Visible Memory contents, i.e. turn all the dots off.
4.3.2	SCLEAR		x1,y1,x2,y2	Clears the specified region of the Visible Memory. When clearing, x1 and x2 are rounded to include the whole byte which the coordinate occupies. The coordinates x1,y1,x2,y2 must specify diagonal corners of the region. Also, the coordinates will be forced within the range 0-319 for x1 and x2, and 0-199 for y1 and y2. After execution, the drawing cursor coordinates will contain x2,y2. The internal cursor coordintes will contain the rounded value of x1 or x2, whichever is smaller, and y1.
4.3.3	WCLEAR		i (0,1,2,or 3)	Sets the current boundaries to those of window i and performs a "home-clear" for this window. The internal cursor coordinates are saved for later recall with the previous window, and then set to the values previously saved for the new window. The drawing cursor coordinates are left unchanged. When the window is cleared, the xnm and xmx boundaries are rounded to include the whole byte which the boundary occupies. This command also turns boundary checking on.
4.3.4	SCFLIP		x1,y1,x2,y2	Flip the specified region of the Visible Memory. This turns off dots that are on, and turns on dots that are off. This command operates exactly the same as the SCLEAR except the region is flipped instead of cleared.

TABLE OF GRAPHICS COMMANDS

Section	Command	SC	Parameters	Description
5.1.1	WRPIX	]W	none	Write the pixel, i.e. dot, at the internal cursor coordinates. What happens to the point is determined by the graphics mode.
5.1.2	RDPIX		var	Reads the state of the pixel, i.e. dot, at the internal cursor coordinates. The variable var will be set to 1 if the point is white, to 2 if point is black, to 255 if point does not lie within the Visible Memory display, or within the current boundaries if boundary checking is turned on.
5.2.1	LINE	]L	x1,y1,x2,y2	Draw a line from x1,y1 to x2,y2. The coordinates will be transformed according to XFFLG. After the line is drawn the drawing cursor coordinates will contain x2,y2. The internal cursor coordinates will also contain x2,y2 if XFFLG = 0 or 1. The internal cursor will contain transformed x2,y2 if XFFLG = 2 or 3.
5.2.2	MOVE	]M	x1,y1	Sets the drawing and internal cursor coordinates to x1,y1. The internal cursor coordinates will be transformed if XFFLG = 1 or 3.
5.2.3	DRAW	]D	x2,y2	Draw a line from the drawing cursor coordinates to x2,y2. The coordinates are transformed according to XFFLG with the drawing cursor coordinates being treated as x1,y1. After the line is drawn, the cursor coordinates will contain values as described for the LINE command.
5.2.4	DOTL		bval1,bval2	Sets the dotted line parameters. Lines will be drawn by drawing bval1 dots according to the drawing mode, followed by bval2 dots drawn invisibly. To disable dotted lines, make bval2 zero.
5.3.1	XFFLG	]F	i (0,1,2,or 3)	Enables and disables coordinate tranformations. 0 = no transformations; 1 = transform x1,y1 coordinates; 2 = transform x2,y2 coordinates; 3 = transform both x1,y1 and x2,y2 coordinates. The transformation formulas are X=xorg+X*2**xrep and Y=yorg+Y*2**yrep.
5.3.2	XSHFT		val	Sets xorg to val.
5.3.3	YSHFT		val	Sets yorg to val.
5.3.4	XSCALE		val	Sets xrep to val.
5.3.5	YSCALE		val	Sets yrep to val.

TABLE OF TEXT COMMANDS

Section	Command	SC	Parameters	Description
6.1.1	CHAR	]C	bval or str	Displays character whose value is bval. Or, displays contents of string str.
6.1.2	AUTEXT	]A	bval or str	Displays character whose value is bval. Or, displays contents of string str. This command is intended for horizontal printing of text, when it will also perform "CR-LFs" and verticle scrolling as needed to keep the displayed characters within the current boundaries.
6.2.1	CHSCALE		val	Sets the character scaling parameter, chrep, to val. This parameter will also affect when "CR-LFs" and scrolls are performed. Character size = normal size*2**chrep.
6.2.2	CHROT		val	Sets the character rotation parameter, chrot, to val. Character rotation = chrot*45 degrees.
6.3.1	SCROLL		xt,yt,x1,y1,x2,y2	Move x1,y1 corner of the region x1,y1,x2, y2 to xt,yt. The coordinates x1,x2, and xt will be rounded to include the whole byte which the coordinate occupies. The coordinates x1,y1,x2,y2 must specify diagonal corners of the region. Also, the x1 and x2 coordinates will be forced within the 0-319, and y1 and y2 will be forced within the range 0-199. After execution, the drawing cursor will contain x2,y2. The contents of the internal cursor will vary, refer to Section 6.3.1 for details.

TABLE OF CURSOR COMMANDS

Section	Command	SC	Parameters	Description
7.1.1	GRACSR	]G	none	Sets the internal cursor coordinates equal to the drawing cursor coordinates, and then draws the graphics cursor at the internal cursor coordinates. If XFFLG = 1 or 3, the internal cursor coordinates will be transformed prior to displaying the graphics cursor. The cursor will always be drawn in flip mode, so drawing it a second time makes it dissappear.
7.1.2	TEXCSR	]T	none	Draws the text cursor at the internal cursor coordinates. The cursor will always be draw in flip mode, so drawing it a second time makes it dissappear.
7.2.1	RDCSR		xvar,yvar	Reads the internal cursor coordinates into variables xvar and yvar.
7.2.2	RDXY		xvar,yvar	Reads the drawing cursor coordinates into variables xvar and yvar.

TABLE OF CHARACTER DEFINITION COMMANDS

Section	Command	SC	Parameters	Description
8.1.1	CSETUP		none	Builds a set of character definitions. This set of characters is built from the definitions which begin at the location pointed to by the character definition pointer (CHDLOC). The building stops when a "0" is encountered as the character to be defined.
8.1.2	CHRESET		none	Resets, or unbuilds, all character definitions by setting the contents of the current character table to all zeros. It leaves the character definition pointer (CHDLOC) unchanged.
8.2.1	CHDLOC		val	Sets the character definition pointer to val. New definitions will be built beginning at this location.
8.2.2	RDDLOC		var	Reads the current value of the character definition pointer into variable var.
8.2.3	CHTLOC		val	Sets the beginning location of the character table. The table occupies 512 bytes and contains a pointer to the definition of each character that has been built.
8.2.4	RDTLOC		var	Reads the current location of the character table into variable var.
8.3.1	CHINIT		bval	Prepares to receive a definition for the character whose value is bval. The parameter bval is stored at the location pointed to by the character definition pointer and the pointer is then incremented. If bval is in the range 1 through 255, then definition mode is turned on. If bval = 0, then definition mode is not turned on. CHINIT 0 is used to terminate a set of definitions.
8.3.2	CHDFC		bval	If the definition mode is turned on, bval is stored at the location pointed to by the character definition pointer and the pointer is incremented.
8.3.3	CHBLD		none	If the definition mode is turned on, the definition from the most recent CHINIT command is built into the character table for the character value specified in that CHINIT command.

TABLE OF BOUNDARY COMMANDS

Section	Command	SC	Parameters	Description
9.2.1	SETBND		xmn,ymn,xmx,ymx	Sets the current boundaries to the specified values and turns on boundary checking. If xmn or xmx is outside the range 0-319, it will be forced to the nearer of the two limits. If ymn or ymx lies outside the range 0-199, it will be forced to the nearer of these two limits. The cursor cordinates are changed by this command. The drawing cursor will contain xmx,ymx; and the internal cursor will contain xmn,ymn.
9.2.2	SETWIN		i,xmn,ymn,xmx,ymx (i=0,1,2,or 3)	Sets the boundaries of window i to the specified values. These values will be forced to the same range of values as in the BNDSET command. The newly set window will also become the current boundaries if the current boundaries aren't associated with a window number, or if the current boundaries are associated with the same window number as the one being set. Also turns boundary checking on. This command will affect the cursor coordinates the same as the SETBND command.
9.2.3	WINDOW		i (0,1,2,or 3)	Set the current boundaries to those of window i. Saves the current internal cursor coordinates for recall with the previous window, and restores the internal cursor coodinates to those previously saved for window i. Also turns boundary checking on.
9.3.1	BNDCHK		none	Turns boundary checking on.
9.3.2	NOCHK		none	Turns boundary checking off.

MISCELLANEOUS COMMANDS

Section	Command	SC	Parameters	Description
10.1.1	GRSHRT		none	Enables short command mode where only the short form of the commands are recognised. This speeds program execution.
10.1.2		]X	none	Turns off the short command mode.
10.2.1	VMPAGE		bval	Tells the KGP that the starting page of the Visible Memory is bval.
10.2.2	GKILL		none	Unlinks the Keyword Graphics Package from BASIC.

TABLE OF SPECIAL FUNCTION CHARACTERS

DEL (20) - Erase the character at the current character position.

Up-arrow (145) - Move up one line.

Down-arrow (17) - Move down one line.

Right-arrow (29) - Move right one character position.

Left-arrow (157) - Move left one character position.

Shift-1 to Shift-9 (177-185) - Move one dot position in direction the digit is from 5.

Shift-D (196) - Add one to CHREP, i.e. double the current character size.

Shift-E (197) - Subtract one from CHREP, i.e. halve the current character size.

Shift-F (198) - Add one to CHROT, i.e. rotate orientation 45 degrees counterclockwise.

Shift-G (199) - Subtract one from CHROT, i.e. rotate orientation 45 degrees clockwise.

Shift-$ (164) - Draw the definition contained in GR$.

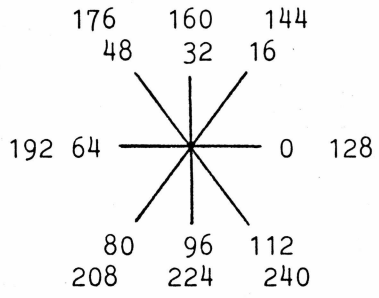
CLR (147) - AUTEXT only. Clears the region specified by the current boundaries, and then homes the internal cursor coordinates.

HOME (19) - AUTEXT only. Homes the internal cursor coordinates.

RETURN (13) - AUTEXT only. Performs a carriage return followed by a line feed.

VECTOR BYTE CHART

Direction Number



0 to 112 = MOVE
128 to 240 = DRAW

Length Number

Specifies length of move or draw. This number may be from 0 to 14. Number 15 is reserved for indicating control bytes.

CONTROL BYTE TABLE

HEX VALUE	DECIMAL VALUE	PET CHAR.	MEANING
$2F	47	/	Display the definition for the character in the next byte.
$3F	63	?	Display the string of characters in the following bytes. The string must be terminated by a zero byte.
$4F	79	O	Display the contents of the BASIC string variable whose name is contained in the following two bytes.
$5F	95		Add the next byte to character orientation parameter, chrot.
$AF	175	Shift-/	Add the next byte to the character scale factor, chrep.
$BF	191	Shift-?	Display the definition which starts at the address in the next two bytes.
$CF	207	Shift-O	Sets the graphics mode to the contents of the next byte if it is 0,1, or 2 . If it is $FF, the graphics mode is flipped. If it is some other value, it is ignored.