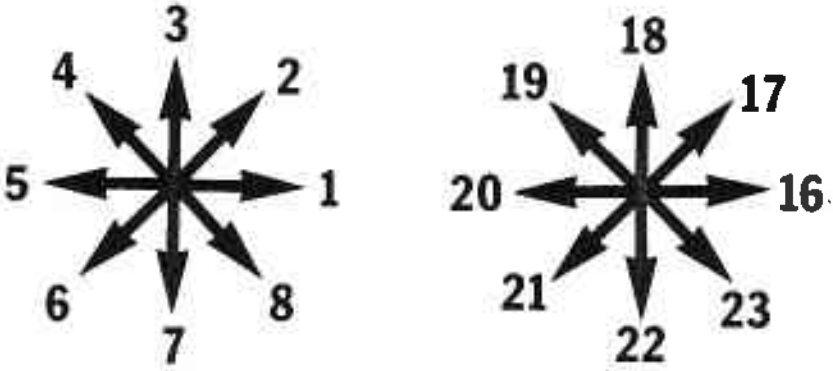
GRBASIC

GRAPHICS ENHANCEMENT PACKAGE

BY

SIMON SMITH



TRS-80 MODEL I LEVEL II
TRS-80 MODEL III LEVEL II
MED SYSTEMS SOFTWARE

GRAPHIC BASIC

By Simon Smith
Med Systems Software

Graphic Basic provides Level II and Disk Basic with a complete set of high speed graphics commands. Since it occupies only 1400 bytes of memory, Graphic Basic may be used in any TRS-80 computer with 16K or more of memory. Graphic Basic is capable of instantly drawing a line between any two pixels on the screen, drawing a shape anywhere on the screen, magnifying the object, and even rotating it at 45 degree increments. Although mastering all the functions of Graphic Basic may take some time, they are very easy to use once understood. Please read the instructions before using this program. Especially do not attempt to draw shapes until you have studied how to do it. If you make a mistake in defining the shape table, your computer may hang up, and have to be RESET.

TERMS AND CONDITIONS
LIMITED WARRANTY

MED SYSTEMS SOFTWARE shall have no liability or responsibility to purchaser or any other person or entity with respect to any liability, loss, or damage caused or alleged to be caused directly or indirectly by this product, included but not limited to any interruption of service, loss of business and anticipatory profits or consequential damages resulting from the use or operation of this product. This product will be exchanged if defective in manufacture, labeling or packaging, but except for such replacement, the sale or subsequent use of this program material is without warranty or liability. Magnetic media may not be copyable on user's system using standard copying procedures. All media are warranted to load and run. If defective, return original media for free replacement within 90 days of receipt of order.

LOADING

Read Appendix B (page 13) and Appendix C (page 14) before loading! Several copies are on each side of the cassette. A special sketch program is also included in both tape and DISK versions.

Cassette: If you have a Model III, select low cassette speed.
Type 'SYSTEM' and press ENTER.
Type 'GRBAS' and ready the cassette.
Press ENTER.
When loading is complete, type '/' and press ENTER.
'READY' will be immediately displayed.
Graphic Basic can now be used.

Disk – Boot DOS.
Make sure that 'BASIC/CMD' is on a disk in one of the drives.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

For the first loading of GRBASIC only:
Look at the directory of the GRBASIC disk.
Locate the version of GRBASIC that you need:
- TRS32K= TRS-DOS version for 32K
- TRS48K= TRS-DOS version for 48K
- NEW32K= NEWDOS-80 version for 32K
- NEW48K= NEWDOS-80 version for 48K

Type 'RENAME XXXXXX/CMD TO GRBASIC/CMD' (where XXXXXX is the name of the version you require) and press ENTER.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Type 'GRBASIC' and press ENTER.
GRBASIC will automatically load Disk Basic and begin execution.
If a 'PROGRAM NOT FOUND' error occurs, check to see that both GRBASIC/CMD and BASIC/CMD are on the disks in your drives.

Graphic Basic loads into high memory and protects itself.

LINE DRAWING

The syntax used to draw a line is -

```
LINE =S X1,Y1 TO X2,Y2 TO X3,Y3 . . . . .
```

'R' should be used in place of 'S' to erase (or RESET) the line, thus:

```
LINE =R X1,Y1 TO X2,Y2 TO X3,Y3 . . . . .
```

X1,Y1,X2,Y2,X3,Y3 can be any legal Level II expressions. The X coordinate must remain between 0 and 127, and the Y coordinate between 0 and 47, or an error will occur. This corresponds to the standard graphic coordinate limits imposed by the TRS-80.

Examples -

```
LINE =S 0,0 TO 127,47 ; or, LINE =R 0,0 TO 127,47
LINE =S 0,0 TO 127,0 TO 127,47 TO 0,47 TO 0,0
LINE =S X1,Y1 TO X2,Y2 TO X3,Y3
LINE =S TO X2,Y2  ; Draws line from end of last line
   to X2,Y2.
LINE =S RND(127), RND(47) TO INT(X/Y), Z↑2
```

Illegal -

`LINE 0,0 TO 127,47`	Syntax Error (No 'S' or 'R')
`LINE =S 0,0`	Missing Operand Error ('TO' Missing)
`LINE =S 0,1 TO 130,6`	Illegal Function Call Error (130 is too large an X coordinate)

Note - A line erased in the opposite direction from which it was drawn may not completely erase the line, since the two lines are slightly different.

DEFINING SHAPES

A shape, or shape table, is a series of numbers or vectors that tell Graphic Basic how to draw an object. The task of drawing shapes can be broken down into two distinct phases - defining a shape, and drawing it on the screen. Defining a shape and telling Graphic Basic where it is in memory will be discussed first.

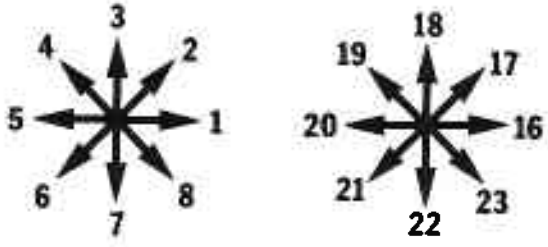
The procedure of making a shape table is outlined in the following steps:

1.) Draw the shape on graph paper (Radio Shack video worksheets are best).

2.) Select a point near the center of the object.

3.) Draw in a series of arrows which trace the perimeter of the object.

4.) Starting with the center point you chose, convert the arrows to numbers using the vector directions below



(1-8 move and draw, 16-23 only move).

5.) POKE the numbers into memory.

6.) POKE the number 99 (63H) into the memory location directly after the shape table (This tells Graphic Basic where the table ends).

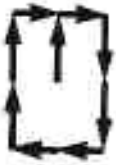
Note - Starting the drawing process at a central point is useful because the magnification and rotation commands operate around the first point in the shape table.

The following example illustrates how to use the steps above in drawing a rectangle.

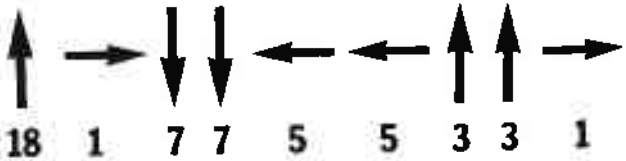
1.) Draw the shape.

2.) Find an appropriate center.

3.) Draw in arrows.



4.) Convert the arrows to numbers.
Since we don't want to draw from the center to the side, we begin with an 18 (i.e. move up but don't draw). Our table will be - 18,1,7,7,5,5,3,3,1



5.-6.) Poke the numbers into memory
You can put the table anywhere, as long as it is not on top of Graphic Basic, Disk Basic or Level II, and will not be wiped out by their operation. When Graphic Basic loads and is activated, it sets Basic's memory size pointer to 523 bytes below where Graphic Basic loads into high memory. These 522 bytes are reserved for shape tables. The chart on the following page gives important addresses for POKEing shape tables into memory.

System Memory	Shape Table Storage	GRBASIC loads	Shape Pointer Location (SPL)
16K	7850H-7A95H 30800D-31381D	7A96H 31382D	7A9FH 31391D
32K	B850H-BA95H 47184D-47765D	BA96H 47766D	BA9FH 47775D
48K	F850H-FA95H 63568D-64149D	FA96H 64150D	FA9FH 64159D

The Shape Pointer Location (SPL) is composed of two bytes. These two bytes contain an address which points to the beginning of the current shape. By default, it points to the beginning of GRBASIC's shape table storage area. The SPL need only be changed when you have more than one shape or when a shape is stored in a non-standard location, or both. This is described later in the text.

There are several ways to get a shape's data into a specified block of memory.

The easiest method is simply POKEing the numbers from a Basic DATA line into memory. In our example we could use the following program :

```
10 FOR I=30800 TO 30800+10
20 READ N
30 POKE I,N
40 NEXT
50 DATA 18,1,7,7,5,5,3,3,1,1,99
   (99 indicates end of table)
```

When the program is RUN, our shape table will be POKEd into memory at the begininng of the shape table storage area. Since the shape pointer location contains 30800 when Graphic Basic is loaded (16K version), Graphic Basic already knows where to find our table. If we wanted to store the table somewhere else in memory, we would have to POKE the SPL (Shape Pointer Location) with the table's starting address. To do this, calculate the least significant byte of the address and put it into SPL, then put the most significant byte into SPL+1 (see Appendix A: 'Calculating LSB and MSB').

Another way to get a table into memory is to store the data in a string. Example -

```
10 FOR I=1 TO 10
20 READ N
30 A$=A$+CHR$(N)
40 NEXT
50 DATA 18,1,7,7,5,5,3,3,1,1,99
```

Since this technique does not make use of the shape table storage area, we must POKE SPL with the address of where the variable is stored. This is easy -

```
60 SPL=31391
   (location in 16K version, see chart on page 3)
70 POKE SPL,PEEK(VARPTR(A$)+1)
80 POKE SPL+1,PEEK(VARPTR(A$)+2)
```

If you are not familiar with the VARPTR command in Basic, refer to the Level II manual.

Since Basic often moves the position of strings around in memory, it is a good idea to re-POKE the SPL everytime you draw the shape when using this method. If you define a dummy string constant within the program, such as 100 A$ = "XXXXXXXXXXX" and put your shape table into the string as shown below, this will not be a problem.

```
100 A$="XXXXXXXXXXX"
110 ST=PEEK(VARPTR(A$)+1)+PEEK(VARPTR(A$)+2)*256
120 FOR I=ST TO ST+10
130 READ N
140 POKE I,N
150 NEXT
160 DATA 18,1,7,7,5,5,3,3,1,1,99
```

You must have at least as many characters in the dummy string as in the shape table before you poke the values in. Simply type as many X's as you need. It is always a good idea to recalculate the SPL at the start of a program, but that is all that is necessary if you are using a "packed" dummy string.

Dummy strings can be saved to DISK or tape without destroying the shape. That is, once packed, that line of programming is permanent, and lines 110-160 above could be deleted and never used again to draw this shape. Only the SPL need be recalculated each time, as shown in lines 60-80 above. Note that the SPL cannot be calculated from VARPTR until the line containing the dummy string graphic shape is executed.

Those who possess the Editor/Assembler and/or disks have yet another way of getting shape tables into memory. The Editor/Assembler (EDTASM) can be used to create a shape file (on either cassette or disk) that can be directly loaded into memory. This saves memory (no DATA statements or POKEing routines are required), and time (since the table is loaded into memory, it does not need to be POKEd). To create a shape file, first figure out all the numbers in the table. When you have done that, simply enter EDTASM, define an ORiGin for the file (preferably in the shape storage area for simplicity) and type all the shape data in as DEFB statements. When the text is Assembled and dumped to either tape or disk, you will have a shape file that can be loaded into memory. Although typing in all the DEFB's may be tedious, think of the time and memory you save by avoiding DATA and POKEing! Disk users can also use DEBUG to type the data into memory, so that it could then be DUMPed to disk as a shape file. Remember, however, to always set Graphic Basic's SPL to point to your shape.

DRAWING SHAPES

Graphic Basic provides four commands for drawing and manipulating shapes. These are:

```
LINE SHAPEDRAW AT X1,Y1
LINE SHAPEXDRAW AT X1,Y1
LINE MAG=M
LINE ROT=R
```

LINE SHAPEDRAW AT draws the shape currently pointed to by SPL at the coordinates X1,Y1. SHAPEDRAW must be all one word, and AT must be one word, but spaces may separate the words. LINE SHAPEXDRAW AT erases the shape currently pointed to by SPL. X1 and Y1 may be any legal Level II expressions. If X1 and/or Y1 assume values greater than 127 and 47 (respectively), the shape appears to have been drawn off the edge of the screen, but some parts of the shape may still be visible. Interesting effects can be generated using this 'trick'. If a shape is drawn with too large a value it will appear on the opposite side of the screen from which it disappeared. If an illegal vector move, 9 for example, (see diagram on page 4) is encountered in the shape table, an ILLEGAL FUNCTION CALL ERROR will result.

LINE MAG=M sets the magnification factor for all shapes equal to M. M may be any legal Level II expression, and may assume a value from 1-255. A value of 255 is the maximum magnification available (and is too large for any normal purpose), and a value of 1 is the minimum (making the shape the same size as it was initially drawn.) The LINE MAG command enlarges an object by multiplying every vector move in the shape table by M.

LINE ROT=R rotates the shape currently pointed to by SPL. R has the same limits as M. A ROT of 1 rotates the shape 45 degrees, a ROT of 2, 90 degrees, and so on. A ROT of 8 will rotate the shape right back to its initial orientation. Because the TRS-80's graphic pixels (SET squares) are twice as tall as they are wide, the ROT command may severely distort a shape.

A rotation of 90 degrees will stretch an object in the Y direction, and compress it in the X direction. A rotation of 180 degrees (ROT=4) is very useful, however, because it produces a mirror image of the shape.

Neither LINE ROT nor LINE MAG commands are permanent; you can 'restore' a shape to its normal appearance by merely specifying a magnification of 1 and rotating the object until it is correctly oriented. Also, all shape commands, except LINE MAG, only affect the shape currently defined by the Shape Pointer Location. To draw or manipulate a different shape, simply POKE SPL and SPL+1 with the starting address of the new shape table (refer to the section on DEFINING SHAPES). By POKEing SPL and SPL+1 with the different values, a program can selectively draw and manipulate several different shapes. Always remember to terminate a shape table with 99D (63H) so that Graphic Basic doesn't draw all the shapes you have entered into the shape storage area one on top of another!

MULTIPLE SHAPES

A program can make use of several different shape tables by POKEing SPL and SPL+1 with different addresses. For example, suppose the programmer wishes to draw shape #1 at coordinates 100,30 and shape #2 at 10,5 and suppose shape table #1 starts at address 30800 and shape table #2 starts at 30900. This program could draw the two shapes in the correct locations -

```
10 SPL=31391                        '16K version
20 POKE SPL,80                      'LSB of 30800
30 POKE SPL+1,120                   'MSB of 30800
40 LINE SHAPEDRAW AT 100,30
50 POKE SPL,180                     'LSB of 30900
60 POKE SPL+1,120                   'MSB of 30900
70 LINE SHAPEDRAW AT 10,5
```

Remember that LINE ROT only affects the current shape!

ANIMATION

It should be clear by now that once a shape is in memory, it can easily be drawn anywhere on the screen. This makes the task of creating animated 'movies' very simple. To make a shape move across the screen, define a FOR NEXT loop in which the X coordinate goes from 0-127, SHAPEDRAW the shape 'AT X,23' (or whatever Y coordinate you wish), SHAPEXDRAW the shape at those same coordinates, and then go on to the next X value. The following program demonstrates this -

```
10 CLS
20 FOR X=0 TO 127 STEP 4        'STEP 4 is just for speed
30 LINE SHAPEDRAW AT X,23
40 LINE SHAPEXDRAW AT X,23
50 NEXT
```

By drawing an object with a MAG of 1, erasing it, and then drawing it again with larger MAG, one can create a depth or 3D effect. When attemping such graphic displays, it is a good idea to also make the shape move across the screen, since this enhances the 3D appearance.

CHANGING MEMORY SIZE

If a program requires Graphic Basic but needs more memory, the 522 bytes reserved for shape tables may be reduced. To do this, the user must decide the highest address that Basic needs to use. If this address is above the point where Graphic Basic loads, (see chart on page 3) Graphic Basic will have to be eliminated in order to provide sufficient memory for Basic. If it is below the loading address, POKE 40B1H (16561D) and 40B2H (16562D) with the least and most significant bytes, in this order, of the highest address you want Basic to use. (see 'Calculating LSB and MSB') Type 'CLEAR 50' and press ENTER. Basic will now have those extra bytes for your program, but you can no longer store a shape table there. These steps can also be used to increase the space reserved for shape tables.

APPENDIX A: CALCULATING LSB AND MSB

When POKEing an address into SPL, or when changing memory size, the LSB (Least Significant Byte) and the MSB (Most Significant Byte) of the address are required. To find these two numbers, first take the address and divide it by 256. The integer portion of the resulting number will be the MSB of that address. To get the LSB, multiply the MSB by 256 and subtract the result from the address. This process is demonstrated in a Basic program -

```
10 MSB=INT(ADDRESS/256)
20 LSB=ADDRESS-MSB*256
```

Once these two numbers have been calculated, the LSB can be POKEd into SPL (or the lower byte of the memory size pointer) and the MSB into SPL+1 (or the high byte of the memory size pointer)

Note: When POKEing numbers greater than 32767, don't forget to subtract 65536 from the address.

APPENDIX B: LOADING INFORMATION

CASSETTES: Each side of the cassette contains four programs. The first program is the 16K GRBASIC, the second is the 32K GRBASIC, and the third is the 48K GRBASIC. You will have to find the correct version by listening to portions of the tape. When you find the version you wish to use, mark down the tape counter number for future reference.

EVERYONE: The last program on the cassette is a BASIC sketch program. On disk, it is called SKETCH/BAS. From cassette, use CLOAD. This program is a very useful GRBASIC utility which will allow you to sketch shapes on the screen directly. The vector or shape table can then be dumped to the screen or a printer for use in creating the in-memory shape table. Read this manual before using SKETCH. SKETCH contains complete instructions, but some understanding of this manual's contents is assumed.

APPENDIX C: NOTES FOR DISK USERS

'GRBASIC *' should be used in place of 'BASIC *'.

'LINE INPUT' functions normally.

Additional programs on the diskette, written in GRBASIC, are:

PRODEM/BAS
MOONBATT/BAS
SKETCH/BAS

ONE DRIVE USERS: The GRBASIC disk does not contain an operating system. Therefore, the proper GRBASIC file must be transferred to a disk containing an operating system. TRS-DOS and NEWDOS vary in their ability to do this.

TRS-DOS provides no way to transfer a file from one diskette to another. Unless you have two drives, you will not be able to run GRBASIC. Med Systems will place GRBASIC on a TRS-DOS disk if it is sent to us. The disk must contain a complete TRS-DOS system. Please enclose $2.00 to cover first class return postage. We regret the inconvenience, but TANDY is not conducive to our use of their system without considerable expense.

NEWDOS 80 makes it easy to transfer the file. Once NEWDOS is booted, but before the GRBASIC disk is in the drive, type:

COPY :0 XXXXXX/CMD TO GRBASIC/CMD

where XXXXXX is the file name as described on page 2. Then follow the directions NEWDOS gives you.