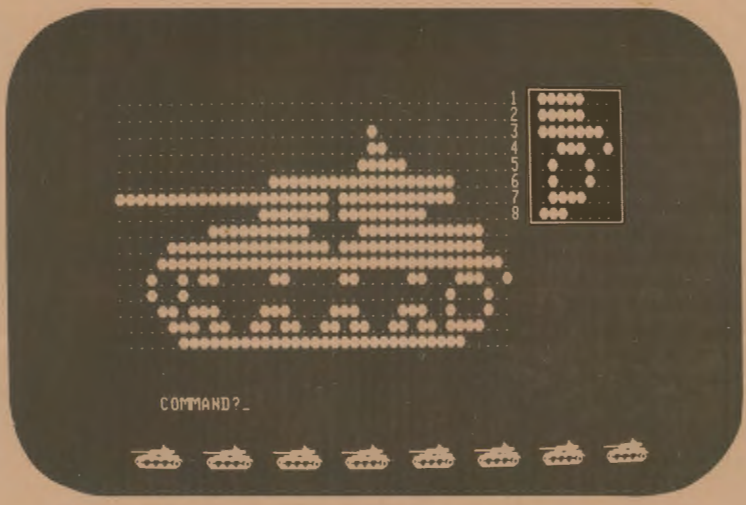

SHAPE MAKER™

AN ON-SCREEN EDITOR FOR DEFINING
GRAPHICS FOR THE EXIDY SORCERER

by Don Ursem

A product of QS QUALITY SOFTWARE

Published by Quality Software

LOADING INSTRUCTIONS

SHAPE MAKER is a BASIC program and is written to work on any memory size Sorcerer. To load the tape, follow these steps:

1. Be sure the BASIC ROM PAC is inserted before turning on the Sorcerer.
2. Ready the tape recorder. High tone settings and about 75% volume are usually required for the Sorcerer.
3. Turn on the Sorcerer and load and execute the program as follows (information from the computer is in italics):

   READY

   CLOAD SHAPE

   FOUND SHAPE —ETC.

   LOADING —ETC.

   READY

   RUN
4. We recommend that you make a working copy of SHAPE MAKER on your own recorder and use the original copy as a backup.

COMMAND USAGE

Commands are entered when the "COMMAND?" prompt appears on the screen. The exceptions to this are the "#" command and the cursor control commands. All commands are one-word commands and are transmitted by pressing the RETURN key, except in the case of cursor control commands when no RETURN is required. Both upper and lower case letters are accepted. Many commands can be abbreviated to one-letter commands. If at any time an incorrect command is input, it will be erased and the "COMMAND?" prompt will reappear.

The commands have been divided into three groups:

1. Single Character Commands - these commands affect only the single-character work area.
2. On-Screen Buffer Commands - these commands affect at least one of the ten on-screen character buffers.
3. Cursor Control Commands - these commands are for use when in the cursor control mode and affect the on-screen buffer area.

Typing a CTRL-C when the "COMMAND?" prompt is on the screen will exit out of the program but won't disturb the symbols in character memory. BYE then exits to the Monitor for dumping memory or saving to tape. PP returns control to BASIC without disturbing either SHAPE MAKER or character memory. Do not press RESET or you will lose the program and half of the programmable characters. The CLEAR key also destroys half of the programmable characters (#128—#191), but if you don't need these characters the CLEAR key is an acceptable way to clear the buffer area and workspace (type CLEAR, RETURN).

SINGLE CHARACTER COMMANDS

n[pattern] example: 3xx xx or
311011

This command modifies one line (line n) of the workspace area. Type the line number followed immediately by a pattern of zero to eight characters. **[pattern]** can be made up of 0's, 1's, x's, or blanks (spaces) where 1 or x means this is where a white dot is desired and 0 or blank means this is where no light (a black dot) is desired. If less than eight characters are input, the pattern will be automatically filled in with blanks (black). A null pattern will clear the line (all black).

Any single special character can be built quickly using the n[pattern] command. Enter the following patterns to build the Greek letter Omega:

100011100
200100010
301000001
401000001
500100010
600010100
701100011
800000000

SAVE example: S

This command places the workspace contents into a programmable character that you choose. Type S and RETURN. The character will be scanned, converted into hexadecimal, and displayed as a normal size character. Then you will be prompted to select a graphic or shift-graphic key. Now type shift-graphic 1. The character in the workspace is stored in the location where the shift-graphic 1 character is defined, and that letter will print every time you press that key. Try typing a few shift-graphic 1 characters, and you will see this. When you press RETURN, these characters will be ignored and the "COMMAND?" prompt will return.

You may also respond to the "SELECT A KEY" prompt with the "#" command, which is described below. The "#" command allows you to assign the decimal character code (the code that is used in the CHR$ function).

RETRIEVE example: R

This command retrieves a character already defined and places it in the workspace. It is the opposite of the SAVE command. You will be prompted to select the key you wish to have the program retrieve from memory. Any key may be selected, including the CTRL key characters which cannot be printed with the CHR$ function.

You may also respond to the "RETRIEVE WHICH KEY" prompt with the "#" command, which is described below. The "#" command allows you to select the character by its decimal character code (the code that is used in the CHR$ function).

DISPLAY example: D

This command displays, in decimal format, the eight bytes that define a selected character that has already been created. It also gives, in decimal format, the addresses where the definition of the selected character is stored and the decimal character code for the selected character. You will be prompted to select the key that you wish this information for. Try typing shift-graphic 1 following a D command. The current character for this key is shown along with its CHR$ number (192), its address locations (–512 to –505) and the eight bytes that define the character. The primary use of the DISPLAY command is to obtain the information necessary to create user-defined characters in a BASIC program. See the section of this booklet entitled "Constructing Shapes With BASIC Data Statements".

You may also respond to the "DISPLAY WHICH KEY?" prompt with the "#" command, which is described below. The "#" command allows you to select the character by its decimal character code (the code that is used in the CHR$ function).

INVERT example: I

This command inverts or "reverses out" the pattern currently in the workspace. That is, white dots are changed to black dots and black dots are changed to white dots. Use the RETRIEVE command to pull the letter "A" into the workspace, then type I and press RETURN. A black letter A on a white background will be created in the workspace.

This command is also useful for two-player game programs where you need equivalent symbols for the "white" and "black" players. Once you build one set of symbols, the INVERT command quickly generates the other.

FLIP example: F

This command flips the pattern in the workspace upside down. For an application of the FLIP command, see the description of the TURN command.

TURN example: T

This command reverses the pattern in the workspace right to left. The leftmost column becomes the rightmost column, etc. The TURN and FLIP commands can be used to save time in forming characters, as demonstrated by the following example:

Type these values into the workspace:

```
1xxxx
2xxx
3x  xx
4    xx
5      xxxx
6      xx
7      x
8
```

This forms an arrow pointing northwest. Arrows in the three other diagonal directions can be formed quickly. Enter the FLIP command and an arrow point-

ing southwest is formed. Then enter TURN and an arrow pointing southeast is formed. Enter the FLIP command again and an arrow pointing northeast is formed.

NEW example: N

This command clears the workspace (makes all dots black).

example:

This command can be entered only when one of three prompts is on the screen: "SELECT A KEY", "RETRIEVE WHICH KEY?" or "DISPLAY WHICH KEY?". It is used when you prefer to enter the decimal character code (the code that is used in the CHR$ function) instead of selecting a key on the keyboard. Entering # will cause the prompt to be modified to ask for the ASCII number. Enter the decimal character code. If you attempt to SAVE into a non-programmable character code, you will get the message, "USE ONLY GRAPHIC OR GRAPHIC SHIFT CHARACTERS", i.e., codes greater than 127. The RETRIEVE and DISPLAY commands can access all character codes.

ON-SCREEN BUFFER COMMANDS

There is an on-screen buffer area (the upper left-hand portion of the screen) where the representation of ten characters can be displayed simultaneously. In data processing terms a "buffer" is a temporary "hold" area for storing data that is still to be worked on. The ten buffers allow you to store one or more characters on the screen while building another character in the workspace. The characters are displayed in two rows of five characters each. Each character representation is called a buffer and is identified by a number as follows:

1	2	3	4	5
6	7	8	9	10

There is a special set of commands that allow characters to be moved in and out of buffers and in and out of memory, and to modify the entire buffer area. Before studying the On-screen Buffer Commands, exit the program and enter RUN 8000 to see a shape displayed in the buffer area.

PUTb example: P3

This command puts the current workspace into buffer number b. Whatever was previously in buffer b is lost. Enter the command P1 and the workspace will be reproduced on the screen in buffer number one.

GETb example: G10

This command gets the character representation in buffer b and loads it into the workspace. Whatever was previously in the workspace is lost.

SAVE* example: S*

This command saves all ten characters represented in the buffer area into the character codes #245 through #254 (note: the Single Character SAVE Command uses character code #255, and the last character SAVEd will always be stored in character #255 as well as wherever it was supposed to be SAVEd). The keys that are tied to codes #245 to #254 are shift-graphic keys on the numeric keypad (÷, 4, 6, x, 1 and 2, 3, +, 0, .).

If you have a shape in the buffers, enter S* and watch as the program loads each buffer, one by one, into the workspace and then stores it in memory. A counter is displayed to indicate which buffer SHAPE MAKER is working on.

This command leaves an extraneous "SELECT A KEY" prompt on the screen. Simply press RETURN to get rid of this prompt and return to the "COMMAND?" prompt.

FLIP* example: F*

This command flips the entire buffer area upside down. The top of buffer #1 becomes the bottom of buffer #6, etc. The workspace is not affected.

TURN* example: T*

This command turns the entire buffer area from left to right. A left facing figure will turn and face right. The workspace is not affected. The usefulness of this command can be seen by exiting the program, entering RUN 8000, and then entering the T* command.

INVERT* example: I*

This command inverts or "reverses out" the entire buffer area. All white dots become black dots and black dots become white dots. The workspace is is not affected.

P* example: P*

This command puts the workspace contents into all ten buffers. The primary use of this command is for clearing the buffers by first clearing the workspace with a NEW command and then entering P*. Or the buffers can be set to all white by successive N, I, and P* commands.

RETRIEVE* example: R*

This command retrieves the ten characters represented by character codes

#245–#254 and stores them in the ten buffers. It is the opposite of the SAVE* command. Whatever was in the on-screen buffers before the command was executed is lost.

CURSOR MODE COMMANDS

If you have learned how to use SHAPE MAKER's Single Character Commands to construct multi-character shapes – great! But building your own multi-character shapes character-by-character gets tricky if you want to change and experiment. SHAPE MAKER's cursor control mode makes it much easier. With this, you can move the cursor around right on the screen, drawing or correcting as you go. The whole ten-character buffer area becomes one large sketch pad area for true graphic editing.

CTRL-P example: hold down CTRL, press P

This command works like an on/off switch to go in and out of the cursor control mode. When in the cursor control mode, only Cursor Mode Commands are recognized. A CTRL-P when in the cursor control mode causes exit from that mode and returns the "COMMAND?" prompt to the screen. The RETURN key is not needed to enter Cursor Mode Commands, except following the CTRL-P to get into the cursor control mode. After entering the cursor mode, the next command must be a direction (U, D, L, R) or the cursor mode will automatically exit.

UP example: U

This command sets the direction for the cursor to travel to up. The cursor becomes an up arrow but does not move. No dots are affected. The up arrow key may be used instead of a U.

DOWN example: D

This command sets the direction for the cursor to travel to down. The cursor becomes a down arrow but does not move. No dots are affected. The down arrow key may be used instead of a D.

RIGHT example: R

This command sets the direction for the cursor to travel to right. The cursor becomes a right arrow but does not move. No dots are affected. The right arrow key may be used instead of an R.

LEFT example: L

This command sets the direction for the cursor to travel to left. The cursor becomes a left arrow but does not move. No dots are affected. The left arrow key may be used instead of an L.

MOVE example: space bar

This command moves the cursor one position in the direction of the arrow but does not modify the dots in the pattern.

ON example: X

This command puts a white dot at the cursor position and moves the cursor one position in the direction of the arrow. The 1 key may be used instead of an X.

OFF example: RUB

This command puts a blank (black dot) at the cursor position and moves the cursor one position in the direction of the arrow. This is the RUB key, not shift-RUB. It is used to erase white dots.

REPEAT example: REPEAT key

This command is used to repeat the last command and is convenient because the key does not have to be released. It can be used for continuous spacing or line drawing.

CONSTRUCTING LARGE SHAPES

By now you have discovered how a shape that is two bytes high and five wide can be formed using SHAPE MAKER. In particular, if you saw the tank being built by executing RUN 8000, then you are aware of the graphic detail possible using the Sorcerer. Our tank is not a crude version of a tank that you might find on some computers. Ours is recognizable as a German Panzerkampfwagon IV Ausf F2 (1942), the main battle tank of Rommel's Afrika Korps.

What about shapes larger than 2x5? You may want to design a chess piece three bytes high or a playing card eight bytes high. Given the 128 bytes available with the Sorcerer, it is possible to construct a shape with dimensions as great as 8x16 or 12x10. For these larger pictures, the following is recommended:

1. Use the cursor control mode to build one ten-byte section.
2. GET and SAVE these ten characters one at a time into the top row of GRAPHIC keys 1, 2, 3, ..., 0.
3. Use GET and PUT to move the content of buffers 6-10 into buffers 1-5.
4. Use cursor control to build the next five characters in buffers 6-10.
5. GET and SAVE these in the next five available GRAPHIC keys.
6. Go back to step 3 and repeat the cycle until the entire shape is defined.

For very large shapes, it helps to sketch the full pattern out on graph paper, to serve as a guide.

SAVING SHAPES ON TAPE

Dumping character sets to tape is a recommended procedure if you have many characters, an entire special alphabet, very large shapes, etc. To do this, stop the program with a CTRL-C and enter BYE to get into the Monitor. Set up your tape recorder and start the recording. Then type

SAVE CHSET FC00 FFFF

and press ENTER to save the 128 programmable character definitions on tape. After the SAVE is complete, type PP to get back to BASIC. The SHAPE MAKER program will still be there.

Later, when you are ready to write your BASIC program that will use the characters you saved, get into the Monitor and load in the tape with the command LOAD CHSET. When the load is completed, reenter BASIC with PP. You are ready to write your BASIC program and test it using the previously defined graphics.

When you have finished writing your BASIC program, we recommend that you make it the second file on the same tape cassette as your special character set. This way you can conveniently load both without having to leave BASIC. Just type

CLOAD CHSET

then when that load is complete, type

CLOAD PROG

where PROG is whatever your BASIC program is called. This works because the Sorcerer keeps track of what kind of file you're loading. The first file is not a BASIC file, so the CLOAD command loads it exactly as the Monitor LOAD command would do. And it gets loaded into non-BASIC memory, so the second CLOAD doesn't wipe it out. When you are reloading character sets from tape and then running a BASIC program, be careful that you don't clear the screen anywhere in the program, because this automatically resets the first 64 graphic characters back to their power-on default values.

CREATING SHAPES WITH BASIC DATA STATEMENTS

For BASIC programs that involve only a few symbols or small shapes, it may be more convenient to have the program itself POKE the character definitions into memory rather than loading them in separately. The procedure used to do this is to include DATA statements in your BASIC program that contain the character definitions, and then design a FOR NEXT loop that POKEs the definitions into the correct spot in memory.

As an example, review the description of the SAVE command on page 3, where we created a special character – the Greek letter Omega. Using the DISPLAY command for this character, which we stored in the graphic-shift 1 key, we find that the decimal character code is #192, the memory location of the character definition is 23,34,65,65,34,20,99,0. From this information it is easy to write the BASIC statements necessary to reconstruct this character.

```
10 FOR LOC=–512 to –505
20 READ N : POKE LOC,N : NEXT LOC
30 DATA 28,34,65,65,34,20,99,0
40 PRINT CHR$(192)
```

EXAMPLE BASIC PROGRAMS USING SHAPES

The following two BASIC programs are given as examples of interesting ways in which the output of SHAPE MAKER can be used in your own BASIC programs.

PROGRAM #1 - REVERSE GRAPHICS

Although the Sorcerer does not come with reverse graphics (black letters on a white background), you can, with SHAPE MAKER's help, create a reverse graphics capability.

1. Load SHAPE MAKER and use the R, I, and S commands to create inverse ASCII. Begin by retrieving character #32 (space), inverting it, and storing it as character #160. Continue through the retrieval of character #127, storing it in #255. This will take some time, even with SHAPE MAKER, but when you are done you can save your work and use it for all your programs.
2. Exit SHAPE MAKER with CTRL-C and BYE, then save the graphics characters on tape as described in the previous section, "Saving Shapes on Tape".
3. Return to BASIC, type NEW, and type in the following program:

```
 0 CLEAR 128:REM TO AVOID OUT OF STRING SPACE ERRORS
10 INPUT W$
20 GOSUB 1000:PRINT W$,B$
30 GOTO 10
1000 REM SUBROUTINE TO CONVERT W$ TO B$
1010 B$="":REM INITIALIZE B$ TO NULL STRING
1020 FOR X=1 TO LEN (W$):Y=ASC(MID$(W$,X))
1030 IF Y>31 and Y<128 THEN Y=Y+128
1040 B$=B$+CHR$(Y):NEXT X
1050 RETURN
```

   When you RUN this program, you will be prompted to enter a string, and the string will then be printed out in regular and inverse graphics.
4. Notice that statements 1000 through 1050 make up a subroutine that can be used in any BASIC program. This subroutine, along with the character set that you saved on tape, gives you reverse graphics capability for all of your BASIC programs.

PROGRAM #2 - TANKITY TANK TANK

RUN this program and our now familiar Panzer tank will come rolling on the screen and, with the help of its trusty cannon, clear the screen for you. This short routine will display the tank, move it, and fire its cannon, given that the set of bytes that define the tank's shape are loaded into the memory area for characters #245 to #254.

```
9020 ST=245
9030 FOR P=—2116 to —3900 STEP —1          :REM SCREEN LOCATION
9040 FOR C=0 to 4                          :REM FIVE COLUMNS
9050 POKE P+C,ST+C:POKE P+C+64,ST+C+5      :REM UPPER AND LOWER ROW
9060 NEXT C
9070 POKE P+C,32:POKE P+C+64,32            :REM BLANK TRAILING BYTES
9080 IF RND (7) > .15 THEN 9100            :REM FIRE CANNON RANDOMLY
9090 FOR I=P—1 TO P—20 STEP —1:POKE I,45:POKE I,32: NEXT I
9100 NEXT P
9110 END
```

If you study this program and understand how it works, then you will be well on your way to making effective use of the Sorcerer's graphics capability in your BASIC programs. By the way, this routine is included in the SHAPE MAKER program. Just exit the regular program with CTRL-C and type RUN 9000.

COMMAND SUMMARY

	COMMAND	EXAMPLE	WHAT IT DOES	FULL DESCRIPTION PAGE
SINGLE CHARACTER COMMANDS	n[pattern]	3xx xx or 311011	puts [pattern] on line n of workspace	3
	SAVE	S	stores workspace into a graphic or shift-graphic key	3
	RETRIEVE	R	retrieves a key into workspace	3
	DISPLAY	D	displays key content as BASIC data	4
	INVERT	I	inverts workspace (reverse field)	4
	FLIP	F	flips workspace upside down	4
	TURN	T	turns workspace right to left	4
	NEW	N	clears the workspace	5
	#	#	ASCII reply to 'which key?' prompt	5
ON-SCREEN BUFFER COMMANDS	PUTb	P3	puts workspace contents into buffer b	6
	GETb	G10	gets workspace from buffer b	6
	S*	S*	saves all buffers into characters 245-254	6
	F*	F*	flips all buffers as one	6
	T*	T*	turns all buffers as one	6
	I*	I*	inverts all buffers	6
	P*	P*	puts workspace into all buffers	6
	R*	R*	retrieves characters 245-254 into buffers	6
CURSOR CONTROL MODE COMMANDS	CTRL-P		enter/exit cursor control mode	7
	UP	U or ↑	set travel direction to up	7
	DOWN	D or ↓	set travel direction to down	7
	RIGHT	R or →	set travel direction to right	7
	LEFT	L or ←	set travel direction to left	7
	MOVE	space	move cursor in current direction without modifying pattern	8
	ON	X or 1	puts a dot at the cursor position and moves cursor in current direction	8
	OFF	RUB key	erases a dot at the cursor position and moves cursor in current direction	8
	REPEAT	REPEAT key	repeats last ON, OFF or MOVE command	8

QUALITY SOFTWARE
6660 Reseda Blvd., Suite 103, Reseda, Ca. 91335
(213) 344-6599