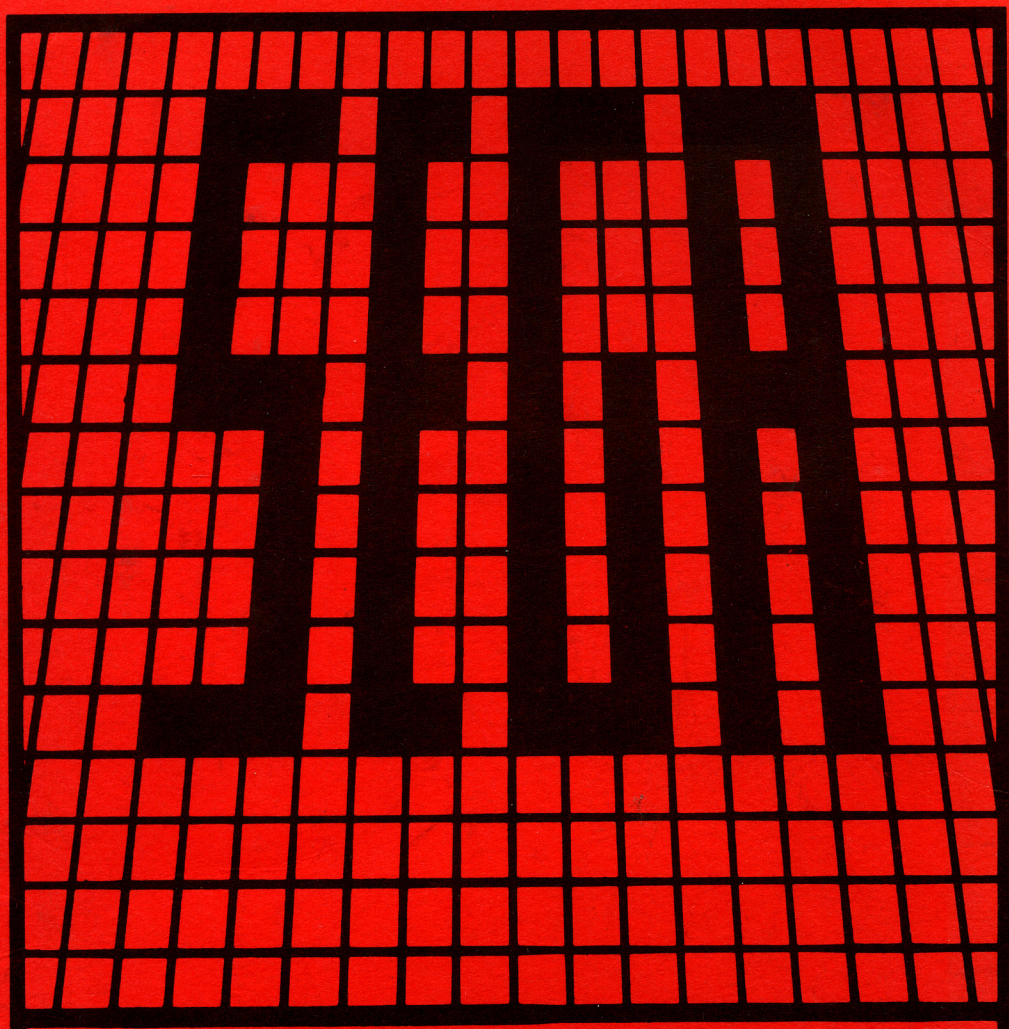


PROGRAMMERS MANUAL

SC-3000

SEGA

Programming Manual

By B. BROWN

Edited and produced for HI-TECH Programming Ltd
by NOMAC Publishing Ltd,
6 Como st,Takapuna,Auckland,NEW ZEALAND.

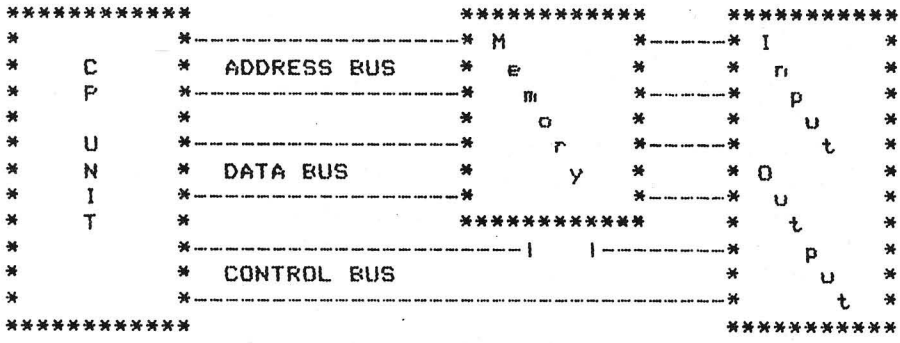
CONTENTS

INTRODUCTION

The SEGA SC3000 home computer is a late addition to the computer scene. It has good graphics and sound, with the promise of greater things to come in the near future with the release of disc attatchments. This book seeks to add to the growing knowledge of the internal workings of theSEGA, and in so doing, help others in their search for better and quicker ways of programming.

BASIC OPERATION PRINCIPLES

The SEGA computer can be represented as three main components,



CENTRAL PROCESSING UNIT (CPU)

This device communicates with all the devices connected to it, and transfers information between the devices as required. (This may involve the manipulation of the data internally within the CPU.)

MEMORY

There are two types of memory used, Read Only and Random Access memory (ROM and RAM). The ROM contains the BASIC language (beginners all-purpose symbolic instruction code), and the necessary programs which enable the CPU to communicate with all the other devices. The contents of the ROM are retained when the power is turned off. ROM can only be Read by the CPU, and is a sort of text book from which the CPU gets the necessary instructions informing it of what to do. RAM is used for temporary program storage, and its contents disappear when the power is turned off. This explains why you must transfer your program to cassette tape. RAM can be thought of as a blackboard. Information can be both written onto it and erased.

INPUT/OUTPUT DEVICES

These devices allow the user to communicate with the CPU and allows feedback from the CPU to the user. An example of an input/output device is the keyboard and Video Display.

COMMUNICATION BETWEEN DEVICES

Each device connected to the CPU is given a unique box number (ADDRESS). The CPU can communicate with the specific device by placing its box number (ADDRESS) on the ADDRESS BUS. A bus is a common highway which allows communication between devices. Having placed the right address on the bus, (ie selected the correct box number), the CPU can then read from or write to the selected device. The CPU transfers information between devices in BINARY format. The smallest element in binary is a BIT. A bit is represented as having one of two possible states, ON or OFF. The ON state is normally designated '1' whilst the OFF state is designated a '0'. The CPU however, can work with eight bits at a time. This group of eight bits is called a BYTE. A byte can be thought of as eight buckets, where each bucket could be full or empty. It thus follows that the maximum number of combinations possible with eight bits is 256. Each address (box) is capable of storing eight bits, thus any box can have as its contents a value of between 0 and 255. The CPU moves the bytes around via the DATA BUS. In this case the DATA BUS is bidirectional, ie information can travel from the CPU to a device or from a device to the CPU. Each device is connected to the address bus which is used by the CPU to tell the device that the CPU is talking to it. The address bus is sixteen bits wide, thus the CPU can access any one of 65536 possible locations (or boxes which hold 8 bits each). To inform the devices as to which way the information is travelling on the data bus, a CONTROL BUS is used. This control bus informs the device if it should expect to receive data (ie a write) or whether it should present data so that the CPU can read it (ie a read). The CPU has temporary storage boxes inside it called REGISTERS. When the CPU wishes to transfer information from one address to another, the CPU carries out the following sequences,

1) Places the correct address (box number) on the address bus
2) Reads the contents of the selected address via the data bus
3) Transfers the information to one of its registers
4) Places the destination address on the address bus
5) Transfers the contents of its register onto the data bus

6) Informs the device at that address to get the new contents for that address, which is appearing on the data bus

INPUT/OUTPUT PORTS

The CPU can have up to 256 seperate ports. These are selected by an eight bit value on the address bus, and the use of a special signal on the control bus. This special signal is activated when you use the command OUT or INP in basic. These ports can each hold an eight bit value. Not all of the ports are used, so refer to chapter one and the section dealing with the memory mapping arrangements for further information.

This covers the sequence of operations in a relatively simple manner, and has served to introduce the reader to some of the more technical terms which will be used shortly.

BINARY & DECIMAL

A byte of eight bits has already been introduced. These eight bits can be either on or off, so a byte in binary could be represented as follows,

B7	B6	B5	B4	B3	B2	B1	B0
1	1	1	0	1	0	1	1

Bit seven is the bit which has the greatest value, while bit zero has the least value. Bit seven is thus called the MOST SIGNIFICANT BIT (MSB) while bit zero is called the LEAST SIGNIFICANT BIT (LSB). In terms of the decimal value of each bit, the following example should help,

Decimal Value	128	64	32	16	8	4	2	1
Binary digit	B7	B6	B5	B4	B3	B2	B1	B0

thus a byte of 11000000 will have a decimal value of 192, because bit 7 and bit 6 are both '1', so the decimal result is 128+64. Where a '1' occurs, the decimal value is added, while all '0's are ignored.

HEXADECIMAL NOTATION

Binary numbers of eight bits are sometimes tedious to write down, so a method was devised in which the binary numbers are represented in another form. This form is known as HEXIDECIMAL (hex). It has a number base of 16 digits (decimal has 10, binary has two). The equilavent decimal, binary, and hex values are listed below,

BINARY	DECIMAL	HEXIDECIMAL
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

As shown, hex ranges from0'F. When the hex number is larger, ie 16 in decimal then the hex number becomes 10. This is exactly the same as in decimal when you go from 9 to 10. Looking at a byte (eight bits), the four least significant bits are called the LOWER NIBBLE, while the four most significant bits are called the UPPER NIBBLE. (A nibble is 4 bits).

Upper Nibble				Lower Nibble				
B7	B6	B5	B4	B3	B2	B1	B0	
1	1	0	1	0	1	1	1	Binary value of each bit

To represent this in hex requires two hex digits, as each hex digit can only represent four bits. The upper and lower nibbles are converted to hex digits, with the resultant hex digits being written with the most significant one first. In the example above,

1101 in binary is '13' decimal so thats 'D' in hexidecimal
0111 in binary is ' 7' decimal so thats '7' in hexidecimal

so the corresponding hex digits which represent the byte 11010111 is 'D7'. Hexidecimal digits are prefixed with &H in SEGA basic, and the hexidecimal value of any decimal number can be found by using HEX$.

CHAPTER 1

The SEGA computer has two main sections, HARDWARE and SOFTWARE. Hardware refers to the physical reality or components, whilst software refers to the programs which control the hardware.

HARDWARE: The hardware can be split into several main sections.

1) CENTRAL PROCESSOR: This is a Z80 8bit processor. It has a maximum address range of 65535 bytes. The first 32K is occupied by the Basic ROM or Games ROMs, while the other 32K is for RAM.

2) VIDEO DISPLAY: The Video Chip is a Texas Instruments TMM9929A. This provides up to four display modes, 32 sprites, 20 millisecond interrupt generator, and 16Kbytes of dedicated RAM. The Video Ram has no connection to the central processor, and is updated by writing to the VDP. The VDP is port mapped at &HBE and &HBF. The internal structure of the VDP and its programming is detailed in chapter 2.

3) SOUND GENERATOR: This is an SN76487AN chip. It has three sound channels and a noise generator. Each channel has its own programmable attenuator for controlling the output volume. It is IC4 on the main PC board, and its programming is discussed in chapter 3.

4) SYSTEM RAM: This is a 2Kbyte chip 8212 (IC3). It is memory mapped at address's &H0000 ' &H07FF. It is used for stack and data storage by plug-in cartridges.

5) INPUT/OUTPUT DEVICES: These include the keyboard, printer, joysticks and cassette. The devices are connected to the computer system via a programmable interface chip, a 8255 PIA (IC5). This PIA has threeports and a control register. The information sent to the control register determines whether the ports will be read or write or both.
The ports are labelled as follows;

PORT A	located at address	&HDC	Keyboard Matrix
PORT B	located at address	&HDD	Keyboard Matrix
PORT C	located at address	&HDE	Keyboard Control
PORT D	located at address	&HDF	Control Register

The actual programming of this PIA will be covered in chapter 5.

6) ADDRESS DECODER: This is achieved by IC2. A logic level of zero on the appropiate CS lead will enable that particular chip. Only one device may be enabled at any time. The CPU can only talk/listen to one device at a time, so it is the function of the address decoder to prevent more than one device interacting with the CPU at any moment.

INTERRUPTS: The SEGA computer operates with two interrupts. An interrupt is a halting of the process being carried out by the processor, a jump is then made to a specific program in memory, and when this program ends the original program is resumed.

NON-MASKABLE INTERRUPT (NMI): The interrupt causes the processor to jump to address &H0066. This occurs whenever the RESET button is pushed. A check is made of location &H97E2 which stores whether a program resides in memory, then the start-up routines are executed. The NMI cannot be disabled.

INTERRUPT (INT): This is used for TIME$ and is generated by the VDP chip every 50 milli-seconds. It can be disabled by a DI (disable interrupts) command using machine-code. It must also be noted that the SEGA computer also uses Interrupt Mode 1, which forces INT to address &H0038.

THE SEGA MEMORY: The SEGA uses a Z80 microprocessor, thus has a maximum address range of 64K.

```
0000  ************  All Basic programming packs
      *  BASIC   *  occupy 0000'7FFF, and comprise
      *  ROM     *  not only ROM but also RAM.
      *    or    *  The 2K of system RAM is located
      *  GAME    *  at C000'C7FF. There is NO onboard
      * CARTRIDGE* ROM! The Video RAM, keyboard,
7FFF  ************  sound generator, and printer
      *  RAM     *  are all bank-selected using
      *  AREA    *  I/O ports. Game Cartridges
      *          *  use the system RAM chip located
      *          *  at C000'C7FF which is the only
      *          *  memory which is on-board. RAM is
FFFF  ************  always located in 7FFF'FFFF.
```

INPUT/OUTPUT PORTS:

7F	* SOUND *	SN74689AN Sound Generator.
DC	* PORT A *	Keyboard Matrix.
DD	* PORT B *	Keyboard Matrix
DE	* PORT C *	Keyboard Control.
DF	* CONTROL *	PIA Intel 8255.
BE	* VDP *	TMM9929A VDG.(+16K VRAM)
BF	* VDP *	Other part of VDG.

FOUR COLOR PRINTER/PLOTTER: The printer is run by a dedicated 8bit micro-computer, type 6805. This CPU has the ROM built inside the actual chip, and thus, if it goes faulty, it must be thrown away. The mechanism is standard, and is used in a wide range of printers, eg, Sharp, Commodore, Casio, etc. Some parts are thus interchangeable.

SOFTWARE:

BASIC CARTRIDGES: The Basic cartridges (LVIIIA/B) contain a 32K ROM chip and also RAM chips. The Basic operating system must use some of the RAM space for the storage of variables etc, (ie reserved Ram areas), thus this explains why only 26620 bytes are available to the user when using the level IIIB cartridge.
THE LEVELIIIB CARTRIDGE: This contains a 32K ROM, 4 16Kx4bit RAM chips, and a few support chips.
GAMES: The games cartridges usually contain a single ROM chip. The on-board system RAM located at &HC000 is used for temporary storage of variables and the system stack. Some cartridges do use two ROM chips.

SEGA BASIC ROM: It occupies the first 32K of memory space. This leaves only 32K left for RAM. The Basic ROM contains the Basic Language, and allows the user to program the computer using english type statements. The necessary routines to manage the keyboard, printer sound generator etc are all part of the Basic Language. These routines may be called independently so that a programmer can use them as part of his own program. This is achieved by use of the CALL statement from basic.

RESERVED RAM AREAS: &H8000 " &H97FF

In order for Basic to convert data from one form to another, and to execute commands or run programs, it must reserve storage space for this purpose. The reserved Ram is also used to store pointers which hold the address or location of the program in memory, the data being used, variables and their values, what line number is being executed, the color and cursor information, the character and sprite patterns, etc. Table XXX1 lists some relevant reserved locations.
BASIC PROGRAM POINTERS: Whenever a Basic program is typed in or RUN, the Basic Language in ROM must know where to locate the program, whereabouts the program ends, where the variables are and what their names are, etc. Basic thus stores all this information in a Reserved RAM area, reserved because if this information is lost or destroyed, then the program will fail to execute properly, if at all. Each location in the Reserved RAM area holds a specific value, eg, memory locations &H8160 and &H8161 store the address of the start of a Basic program. To determine the start address in hexadecimal, type the following

```
PRINT HEX$(PEEK(&H8161));HEX$(PEEK(&H8160))
```

The other pointers associated with the Basic program are listed in Table XXX1. Manipulation of these pointers can result in Merge programs, the ability to save and load machine-code blocks of memory or string storage areas (ie data) etc. For an example of this, refer to the auto-load routine in chapter 4.

BASIC LINE STORAGE FORMAT: When a line of Basic program is typed into the computer, it is stored in an area of designated free RAM. The way that each line is stored in memory is as follows,

Byte 1	Number of Bytes in the Line
Byte 2	Least significant Byte of Line number
Byte 3	Most significant Byte of the Line number
Byte 4	Zero
Byte 5	Zero
Byte 6 "Byte N-1	Basic line contents
Byte N	Always a carriage return &HOD

The end of each Basic line is terminated by a carriage return (CR). If this occured before it should, the Basic Language would erase the rest of the line contents. Occasions where this might happen are explained in the section on String Packing.

TOKENISED BASIC KEYWORDS: Basic keywords are stored in memory as a single hex byte. This saves memory space. When programs are listed or printed, the keywords are expanded into their full meaning. Table XXX2 has a listing of the hex bytes and their equivalent Basic keyword. When counting the number of bytes in a line, keywords are counted as a single byte only.

STRING PACKING: String packing refers to the imbedding within REM statements of a machine-code routine. Because Sega Basic always starts at the same address in memory (&H9800) then this becomes relatively easy. It must be remembered that the machine-code routine cannot have&HOD or 13 decimal in it, else Basic will think that the line has actually finished, and the remaining machine-code will be lost. Refer to the program listed in Table XX21 for an example of this. Once the program has been RUN, press break and list line 5. The machine-code data statements and poke routine can then be deleted, and the code can be saved as part of a normal program.

COLOR BYTES: Locations &H9339 and &H933A hold the color information for the text and graphics screens respectively. The byte is split up into two halves, the first half controls the writing color, and the other half the background color. Refer to Table XXX7 for the values which determine each color. If a Red text on Yellow background is required in the text mode, POKE &H9339,&H8B (8=Red, B=Yellow).

INKEY$ STORAGE AREAS: Locations &H9460 onwards store the value received from the keyboard during an INKEY$ statement. Table XXX4 lists the appropiate key, value and location for each key press. Note that each key pressed returns a different value, and that several locations are used to store the returned values.

USING INKEY$ WITH HYBRID PROGRAMS: A hybrid program is a mixture of machine code and Basic. This technique allows fast speed and ease of programming. A typical layout follows,

```
 5 REM machine-code program poked into here
10 A$=INKEY$: CALL &H9808 : GOTO 10
20 REM &H9808 is start address of mcode
30 REM and tests key value returned in
40 REM locations &H9460-, then moves the
50 REM ship left, right, fires etc
```

ERRORS MESSAGES: The Basic Error messages are stored at &H73E8 " &H7676. The routine at &H73B7 is used to determine the actual error, and then print it to the screen. The code of the error is passed to the routine, which searches a table for the error code, then loads the text message that follows the error code. The following program lists the various errors and their appropiate code.

```
 5 REM MMMMMMMMMMMMMMMMMM
10 SCREEN 1,1: CLS
20 FOR X=&H9808 TO &H980E
30 READ A: POKE X,A: NEXT
40 FOR Y=0 TO 70: POKE &H9809,Y
50 PRINT "Y=";Y;" ";: CALL &H9808
60 PRINT: NEXT Y
70 DATA &H3E,0,&H4F,&HCD,&HB7,&H73
80 DATA &HC9
```

POWER-UP DIAGNOSTICS ROUTINES: The Sega computer, on power-up, carries out a self-check on the various internal compnents. Should a failure occur, a jump is made to the fault indication routine, and an audible indication is given to the user. These indications are,

```
Single Beep = RAM  Failure   &H6809
Double Beep = ROM  Failure   &H680D
Triple Beep = VRAM Failure   &H6811
```

ROM ROUTINES: These routines are used by the cpu when it communicates with the devices connected to it. These routines can be called independently by the programmer, using a CALL statement. Table XXX3 lists some important ROM routines.

Table XXX1. RESERVED RAM AREAS.

Hex Address	Purpose
8160/8161	Start of Basic program
8162/8163	End of Basic program
8164/8165	String Storage pointer
8166/8167	Top of String Storage
8168/8169	Top of Memory pointer
82A2	Program found flag, 0=found
82A3	Filename being loaded (16 bytes)
83A3	Filename being saved (16 bytes)
8B30	Basic Stack Area
8B36	&H80 bytes. Write to VRAM &H1800+
9336	Screen control
9339	Color text screen byte
933A	Color graphics screen byte
9364	&H80 bytes VRAM stores &H1800+ here
9411	Top range of cursor
9412	Bottom range of cursor
9413	8 bytes for storage of PATTERN command
9420	&H28 bytes for storage of VRAM data
9460 " 9480	INKEY$ Storage area
9484	Cursor, 0=normal, 2=graphics
9485	1=lowercase, 0=uppercase
9486	keybeep, 0=beep, 1=nobeep
9489	Cursor position X value
948A	Cursor position Y value
948E	Time$ seconds
948F	Time$ minutes
9490	Time$ hours
9744/9745	Address of DATA byte

Table XXX2. BASIC KEYWORDS.

82	LIST	A0	ON	C1	*	BC	MOTOR
83	LLIST	A1	RETURN	C2	/	E0	FN
84	AUTO	A2	ERASE	C3	MOD	E1	TO
85	DELETE	A3	CURSOR	C4	+	E2	STEP
86	RUN	A4	IF	C5	-	E3	THEN
87	CONT	A5	RESTORE	C6	< > or > <	E4	TAB
88	LOAD	A6	SCREEN	C7	> = or = >	E5	SPC
89	SAVE	A7	COLOR	C8	< = or = <	CC	NOT
8A	VERIFY	A8	LINE	C9	>	CD	AND
8B	NEW	A9	SOUND	CA	<	CE	OR
8C	RENUM	AA	BEEP	CB	=	CF	XOR
90	REM	AB	CONSOLE	8080	ABS	8081	RND
91	PRINT or ?	AC	CLS	8082	SIN	8083	COS
92	LPRINT or L?	AD	OUT	8084	TAN	8085	ASN
93	DATA	AE	CALL	8086	ACS	8087	ATN
94	DEF	AF	POKE	8088	LOG	8089	LGT
95	INPUT	B0	PSET	808A	LTW	808B	EXP
96	READ	B1	PRESET	808C	RAD	808D	DEG
97	STOP	B2	PAINT	808E	PI	808F	SQR
98	END	B3	BLINE	8090	INT	8091	SGN
99	LET	B4	POSITION	8092	ASC	8093	LEN
9A	DIM	B5	HCOPY	8094	VAL	8095	PEEK
9B	FOR	B6	SPRITE	8096	INP	8097	FRE
9C	NEXT	B7	PATTERN	8098	VPEEK	8099	STICK
9D	GOTO	B8	CIRCLE	809A	STRIG	80A0	CHR$
9E	GOSUB	B9	BCIRCLE	80A1	HEX$	80A2	INKEY$
9F	GO	BA	MAG	80A3	LEFT$	80A4	RIGHT$
CO	^	BB	VPOKE	80A5	MID$	80A6	STR$
				80A7	TIME$		

TABLE XXX3. ROM ROUTINES.

Hex Address	Nature of Routine
1000 " 17BF	Character table (8x8) for VDP
1700 " 19FF	Basic keywords
1CB1	Determination of free bytes
2310	Get next character into DE
2400	Write character in A to video screen
2BD4 (2BD1)	Read 80 bytes data from VRAM (&H1800) to &H9364, write 80 bytes from &H8B36 to VRAM (&H1800), move 80 bytes at &H9364 to &H8B36
2C2A (2BCE)	Read data from VRAM
2C32 (2BCB)	Write address in HL to VDP for VRAM read
2C3D (2BC8)	Write data to VRAM
2C44 (2BC5)	Write address in HL to VDP for VRAM write
2C51 (2BC2)	Read VDP Status register
2C54 (2BBF)	Write to a VDP register. Data in A, Register in C.
3604	Hex conversion routines
3A03	Delay using the BC register
3A0F	Write Sync bytes to tape
3A12	Write byte to tape
3B33	Write 8 bytes from &H9413 to VRAM
3D32	SCREEN 1,1
3D90	SCREEN 2,2
3DEE	Initialise Text and Graphic screens
3FA0 " 411F	Keyboard characters arranged in matrix form
4120 " 4258	Basic keyboard symbol table
4590	Reset TIME$ to "00:00:00"
4756	Change cursor to graph
475E	Change cursor to normal
4766	Change input to lowercase
476E	Change input to uppercase
4918	INKEY$
4A6F	Write text pointed to HL to current screen position
6800	Restart 00H (Power)
6803	Restart 38H (VDG)
6806	NMI Entry (Reset)
6AB5	Print FRE routine
6C37	RUN
779F	VERIFY
78EF	LOAD
7A40	SAVE

TABLE XXX4. INKEY$ STORAGE AREAS.

Memory Location	Keys Monitored	Values Returned
&H9460	1QAZ,KI	1'8,32'128
&H9461	2WSXspc.LO	1'128
&H9462	3EDCclr/;PO	1'128
&H9463	4RFVdelpi:@	1'128
&H9464	5TGBcd][	1'8,32'128
&H9465	6YHNclcr	1'8,32,64
&H9466	7UJMcrtcup	1'8,32,64
&H9467	Joysticks	
&H9468	eng,fnc,ctr, sht,spc	1'8

NOTE:

spc= space	eng= eng/diers
clr= clear	fnc= function
del= delete/ins	ctr= control
cd = cursor down	sht= shift
cl = cursor left	cup= cursor up
crt= cursor right	cr = carriage return

CHAPTER 2

THE VISUAL DISPLAY PROCESSOR:

The VDP is a Texas Instruments 9929A chip. This has several important features, such as sprites and interrupt capabilities. In the SEGA computer, the VDP is mapped at two port locations, &HBE and &HBF. These ports are the means by which the central processor communicates with the VDP chip and the Video Ram.

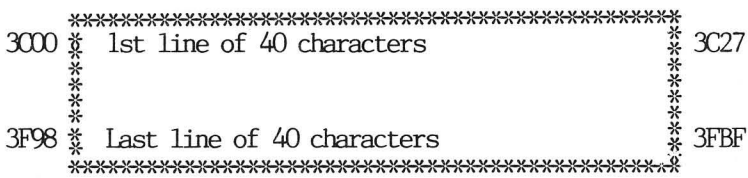
THE VISUAL DISPLAY MODES:

The VDP has four seperate display modes. The four modes are,

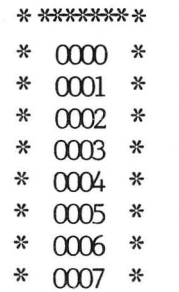
1) Graphics Mode I
2) Graphics Mode II
3) Text Mode
4) Multicolor mode

Only the two used in the SEGA will be explained here, but a program which allows the user to program the multicolor mode is appended at the end of this chapter.

THE TEXT MODE: The text mode provides for 40 characters wide by 24 lines of text. Only two colors may be present on the screen at any time. Basic only allows the use of 38 characters per line, this is done t allow for older television sets who might chop off the 1st two characters. The two colors are referred to as the writing or foreground color, and the background color. These colors are specified by the COLOR command, or may be altered by poking location &H9339 with the appropriate value. The address of Video Ram (VRAM) used to store the characters is as shown,



THE GRAPHICS MODE II: The graphics mode allows all 16 colors to be used simultaneously, and the display is arranged as 256 by 192 pixels, where a PIXEL is a single dot on the screen. A seperate area in VRAM is used to store the color attribute of each pattern on the screen. The patterns are stored as follows, (displaying the 1st character in line 1 only)



The characters are normally made up out of 8x8 pixel blocks. This shows the makeup of the first character of the first line on the graphics screen. The eight bytes that make the character are arranged as shown, with the address inclosed. The second character will thus use address's 0008 " 000F, the third character will use address's 0010 " 0017 etc. The color byte for each character is located at &H2000 ", ie, the color attribute address for the 1st byte is &H2000, for the 2nd byte it is &H2001, for the 1st byte of the 2nd character it is &H2008.

The following program illustrates the colors avaiable in the graphics mode

```
10 SCREEN 2,2:CLS:B=0
20 FOR X=&H0000 TO &H17FF
30 B=B+1:IF B=7 THEN RESTORE:B=0
40 READ A:VPOKE X,&HF0
50 VPOKE X+&H2000,A
60 NEXT X
70 GOTO 70
80 DATA &H01,&H24,&H35,&H6A,&H7B
90 DATA &H8C,&H9D,&HEF
```

ARCHITECHURE OF THE TMS9929A:

The VDP chip comprises eight (8 bit) write only registers, a read only (8 bit) status register, and an autoincrementing (14 bit) address register. The registers hold the necessary address's or data for the VDP chip to be able to find the required patterns in VRAM and determine the location, color, size etc of sprites or the text. The eight register functions in turn are;

Register 0: Register 0 controls the external VDP input, as well as mode select. The external VDP input allows the image from another VDP to appear in the background. In the case of the SEGA this is disabled. MODE SELECT(M3) controls the format of the display screen. This is combined with M2 and M1 of register 1 to select the desired screen layout. (see Table XXX5)

Bits	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	M3	EV

Register 1: Register 1 controls the Video Ram type selection, the blanking out of the active display area, interrupt enable, M1, M2 and the size and magnification factor of any sprites. The SEGA computer has the following, VRAM bit = 1 for 4116 type, Blank bit = 1, Interrupt enabled(50Hz) = 1, Screen mode = text, Size and Mag are 0.
SIZE: This bit determines whether 8 x 8 sprites or 16 x 16 sprites are used.
MAG: This doubles the size of the sprites if a 1, else if a 0 then the size is that set by the size bit. (Table XXX6 gives the combinations equal to the MAG command on the SEGA)

Bits	7	6	5	4	3	2	1	0
	16K	BL	IE	M1	M2	0	SIZ	MAG

Register 2: Register 2 holds the NAME TABLE address for the text or graphic screen, this being &H3000(text) or &H0000(graphics).

Bits	7	6	5	4	3	2	1	0
	0	0	0	0	4 bit Address			

Actual Address = 4 bit address * &H400

Register 3: Register 3 holds the COLOUR ATTRIBUTE TABLE address for the graphics screen, this equal to &H2000 for the SEGA.

```
Bits    7   6   5   4   3   2   1   0
        ***************************************
        *           8 bit Address             *  Actual Address =
        ***************************************  8 bit address *
                                                   &H40
```

Register 4: Register 4 holds the PATTERN GENERATOR address for the text or graphic screen, being &H1800(text) or &H3800(graphics).

```
Bits    7   6   5   4   3   2   1   0
        ***************************************
        * 0 * 0 * 0 * 0 * 0 * 3 bit Add    *  Actual Address =
        ***************************************  3 bit address *
                                                   &H800
```

Register 5: Register 5 holds the SPRITE ATTRIBUTE address (&H3B00).

```
Bits    7   6   5   4   3   2   1   0
        ***************************************
        * 0 *         6 bit Address         *  Actual Address =
        ***************************************  6 bit address *
                                                   &H80
```

Register 6: Register 6 holds the SPRITE PATTERN address (&H1800).

```
Bits    7   6   5   4   3   2   1   0
        ***************************************
        * 0 * 0 * 0 * 0 * 0 * 3 bit Add    *  Actual Address =
        ***************************************  3 bit address *
                                                   &H800
```

Register 7: Register 7 holds the COLOR for the writing/background combination.

```
Bits    7   6   5   4   3   2   1   0
        ***************************************
        * Writing Color    * Background Color  *
        ***************************************
```

STATUS Register: The status register holds the interrupt flag, the fifth sprite flag and number, and the sprite collision flag.

```
Bits    7   6   5   4   3   2   1   0
        ***************************************
        * F * 5S * C *  Fifth Sprite Num    *
        ***************************************
```

HOW TO WRITE/UPDATE A VDP REGISTER:

Two bytes are required to update or write to a register.

Byte 1 is the required data

Byte 2 is the required register

The composition of byte 2 is

10000 + RS0 + RS1 + RS2 (Where RS0-2 are 1 bit each)

	RS0	RS1	RS2
Register 0	0	0	0
Register 1	0	0	1
Register 2	0	1	0
Register 3	0	1	1
Register 4	1	0	0
Register 5	1	0	1
Register 6	1	1	0
Register 7	1	1	1

NOTE: IT IS IMPORTANT THAT THE STATUS REGISTER IS READ AT PORT &HBF BEFORE YOU UPDATE ANY VDP REGISTER.

There is a ROM routine at &H2C54 which provides this facility. Load Register C with the register number (0"7), Register A with the Data byte before calling.

EXAMPLE: Change the color information of the text screen by directly writing to VDP register7.

```
 10 SCREEN 1,1:CLS
 20 PRINT "This is actually black writing"
 30 PRINT "on a green background."
 40 FOR X = &HA000 TO &HA00C
 50 READ AA : POKE X,AA : NEXT X
 60 FOR DE = 1 TO 500 : NEXT DE
 70 CALL &HA000 : PRINT "But is it really?"
 80 GOTO 80
 90 DATA 243,219,191,62,33,211,191
100 DATA 62,135,211,191,251,201
110 REM Disable interrupts, read status register
120 REM LD A with green/black(&H21), Out(&HBF) A
130 REM LD A with register destination
140 REM Out(&HBF) A, Enable int's, Return
```

NOTE: On return to Basic, ie after pressing break, you will notice that the screen reverts to black on green. This is because Basic gets the color information from address &H9339.

WRITING TO VRAM: Load the HL register with the screen address then call &H2C44, and output the value to port &HBE. The address is autoincremented by one location after each write, eg,

```
ENTRY:  A000 F3      DI              ; Disable Interrupts
        A001 D3BF    IN(BF),A        ; Clear Status register
BEGIN:  A003 21003C  LD HL,3C00      ; Text screen
        A006 CD442C  CALL 2C44       ; Write address
        A009 0610    LD B,10         ; 16 times
        A00B 3E32    LD A,32         ; Character "2"
LOOP:   A00D D3BE    OUT(BE),A       ; Print it
        A00F 10FC    DJNZ LOOP       ; 16 times
        A010 C9      RET             ; Back to Basic
```

READING FROM VRAM: Load the HL register with the screen address, call &H2C32, then input the value from port &HBE. The address is auto-incremented after each read.

*** REMEMBER ***: Disable interrupts, then read the status register at port &HBF before you do what you want, or you will get strange results.

ALTERING THE CURSOR POSITION: If using machine-code then the above procedures dealing with reading/writing to Video Ram are required to set up the 14 bit address pointer. However, if using a hybrid program, ie a mixture of machine-code and Basic, especially when calling the print routine at &H4A6F, then the cursor position may be altered by poking the appropiate X and Y values into locations &H9489 and &H948A respectively before calling the print routine. An example of this is given in the next section.

WRITING TEXT OR CHARACTERS TO VRAM: There is a routine in ROM which allows the user to move data to the Video Ram. The following program illustrates this. The text is hidden in the data statements, and a machine code subroutine is used to point the HL register to the text, then the ROM routine at &H4A6F is called. This writes the text out to Video Ram at the current cursor position. Note that the text must end in &H0D or 13 decimal, and you can also clear the screen etc, by the use of control codes (cls=12 decimal).

```
10 SCREEN 1,1
20 FOR X=&HA000 TO &HA010 : REM the machine code
30 READ A: POKE X,A: NEXTX
40 FOR Z=&HB000 TO &HB00D : REM the text string
50 READ S: POKE Z,S: NEXT Z
60 CALL &HA000
70 REM Change cursor x,y positions
80 DATA &H3E,&H0F,&H32,&H89,&H94
90 DATA &H3E,&H0A,&H32,&H8A,&H94
100 REM Machine-code routine
110 DATA &H21,&H00,&HB0,&HCD,&H6F,&H4A,&HC9
120 REM Text message follows
130 DATA 78,111,116,32,66,97,100,32,101
140 DATA 104,33,33,33,13
```

```
ENTRY:   A000  3E0F      LD   A,0F
         A002  328994    LD   (9489),A ; X position = 15
         A005  3E0A      LD   A,0A
         A007  328A94    LD   (948A),A ; Y position = 10
         A00A  2100B0    LD   HL,B000  ; Point to text
         A00D  CD6F4A    CALL 4A6F     ; Call print routine
         A010  C9        RET           ; Back to Basic
TEXT:    B000  " "                     ; 'Not Bad eh!(0D)'
```

There is also another routine used for writing a string of characters to the video screen. This routine is at &H2400 and may be used in the following way,

```
 10 SCREEN 1,1: CLS
 20 FOR X=&HA000 TO &HA009
 30 READ A: POKE X,A: NEXT
 40 CALL &HA000
 50 STOP
 60 DATA &H3E,&H32,&H06,&H20
 70 DATA &HCD,&H00,&H24,&H20
 80 DATA &HFB,&HC9
 90 REM LD A with "2"
100 REM LD B with number of times to be printed
110 REM Call routine at &H2400
120 REM Dec B and Jp not zero to print routine
130 REM Return when B is zero.
```

SPRITES: A sprite is a predefined graphic character. This can be one of four possible sizes, eight by eight pixels, sixteen by sixteen pixels, sixteen by sixteen pixels (double the first), or thirty-two by thirty-two pixels (double the second). The sprite may be moved pixel by pixel around the screen, and a test may be made to see if any two sprites overlap by a single pixel element. The sprite size is controlled by the MAG command in Sega Basic, and the actual shape of the sprite is defined by the PATTERN command. The position of the sprite is controlled by the SPRITE command.

SPRITE PLANES: The Sega uses thirty-two planes, where each plane can be thought of as a transparent screen each behind the other. Only one sprite can be present on a sprite plane at any one time, but as the planes are stacked behind each other, sprites appearing on the closest plane have the highest display priority, ie, they appear in front of the sprites on the planes behind it. Sprites can thus appear to move in front of, or behind other sprites, depending upon which planes are used.

The pattern plane, or the plane on which ordinary text is written to, is the lowest priority, thus sprites will always appear in front of written text.

SPRITE COLLISION DETECTION: Sprite collisions may be detected by reading the Status register located at port &HBF. If any two sprites overlap by a single pixel, bit 5 will be set to logic 1. A basic program to test this would be

```
 10  SCREEN 2,2: CLS
 20  PRINT "Sprite collision demo."
 30  FOR DE=1 TO 1600: NEXT DE
 40  PATTERNS#1,"FFFFFFFFFFFFFFFF"
 50  PATTERNS#2,"FFFFFFFFFFFFFFFF"
 60  SPRITE 0,(120,20),1,14: C=1
 70  FOR X= 0 TO 255
 80  B=INP(&HBF): IF (B AND 32)=32 THEN
     GOSUB 120
 90  SPRITE 1,(X,20),2,C
100  IF INKEY$="" THEN GOTO 100
110  NEXT X: STOP
120  CURSOR 20,10: PRINT CHR$(5);"Collision"
130  BEEP: C=4: RETURN
```

Machine code programs may look something like,

```
A000  DBBF    INP(&HBF),A
A002  E620    AND 20
A004  FE20    CP  20
A006  28??    JR Z Collision
A008          Continue with main program
```

SPRITE ATTRIBUTES TABLE: Starting at address &H3B00 are four bytes for each sprite. These groups of four bytes control the position, color and number of each sprite. Sprite 0 has the first four locations, sprite 1 the second group of four bytes, etc. Refer to Table XXX7 for the relative locations. Table XX26 lists a machine code program which creates sprites, moves them on the screen, checks for sprite collision, changes their color, beeps, and gets a response fromthe keyboard (all using mcode!).

SPRITE PATTERN GENERATOR TABLE: Located at address &H1800 are eight bytes for each sprite. These locations hold the pattern for the sprites, as defined by the Basic command PATTERN. This area also contains the eight by eight patterns for the text screen. They are swapped over as needed by the routine at &H2BD4. The following Basic program illustrates the creation of a sprite, and its movement by poking the attribute area of VRAM.

```
 10 SCREEN 2,2: CLS: PRINT" Sprite Demo"
 20 B=&H1800:REM Create the Sprite
 30 FOR X=0 TO 7: READ A
 40 POKE B+X,A: NEXT
 50 B=&H3B00: REM Create attributes
 60 FOR X=0 TO 3: READ A
 70 POKE B+X,A: NEXT
 80 FOR X=0 TO 255
 90 POKE &H3B01,X
100 NEXT X:POKE &H3B03,4
110 GOTO 110
120 DATA &HFF,&HFF,&HFF,&HFF,&HFF,&HFF,&HFF,&HFF
130 DATA 32,0,0,15
140 REM Y=32,X=0,SPRITE0,COLOR15
```

PATTERN GENERATOR TABLES: These address's store the eight bytes that are needed to compose the character. For the Text mode, the patterns are loaded from ROM address &H1000 into the VRAM area when the computer is turned on or reset.

ALTERING THE CONTENTS OF THE TEXT PATTERN GENERATOR TABLE: In the text mode, the 8 x 8 patterns which make up the character are stored at address &H1800 onwards. Only the characters from &H20 to &HFF are defined in the pattern table, thus the pattern for each character is obtained by using the following formula,

address = &H1800 + character value*8

This gives the address of the first byte that makes up the character. The other seven bytes follow the address determined by the formula. This information can now be used to alter the contents of the existing characters so as to provide both normal and inverse video characters on the text screen at the same time. Basically, the following program replaces the eng/diers characters with the equivalent inverse video alphanumeric set.

```
  10 SCREEN 1,1:AZ$="":FOR A=1 TO 14
  20 READ A$:AZ$=AZ$+CHR$(A$):NEXT
  30 DATA &HA9,&HAE,&HB6,&HA5,&HB2,&HB3,
     &HA5,229,&HB6,&HA9,&HA4,&HA5,&HAF,46
  40 GOSUB 2000:CLS
  50 PRINT" Welcome to ";AZ$:PRINT
  60 PRINT" Try printing out the ";CHR$(&HC5);
     CHR$(&HCE);CHR$(&HC7);"/dier's"
  70 PRINT" characters.": PRINT
  80 STOP
2000 B=&H1800+&H40*8:C=&H1800+&H7F*8
2010 DC=(C+&H20*8)+8
2020 FOR X=B TO C STEP 8
2030 FOR A=X TO X+7
2040 DA=VPEEK(A)
2050 DB=DA XOR &HFF
2060 VPOKE(DC),DB:DC=DC+1
2070 NEXT:NEXT:RETURN
```

By manipulating the contents of the pattern tables, it would be easy to create upside down and reverse characters as well. Table XX27 lists such a program.

NAME TABLE ADDRESS'S: These are eight bit pointers which point to the specific pattern required. If using the Text mode, it represents the ASCII equivalent of the character.

MULTI-COLOR MODE: Table XXX9 lists a program which experiments with the multi-color screen mode. A machine code routine is poked into memory and when called, it switches over to the multi-color mode. Be sure to try this program with a color television set, as it is quite impressive. The color attributes for the multi-color mode are stored at &H3800 to &H3B00. Poking these areas with different values in the range 0 to 255 can result in very colorful displays.

SWAPPING THE CONTENTS OF THE TEXT SCREEN: Utilising the large memory available with the 32K RAM cartridge, it is possible to create a screen swap routine. This involves reading the entire contents of the text screen into a buffer, and then carrying on as per normal. When the old screen is required, a routine is called which rewrites the buffer back to the screen. The following program illustrates this. A machine code routine is poked into line 5 of the program.

```
  5 REM AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
        AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
        AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
 10 SCREEN 1,1:CLS:PRINT" Text Screen Swap"
 20 FOR X=&H9808 TO &H983B
 30 READ A:POKE X,A:NEXT
 40 PRINT" This is the original screen."
 50 FOR DE=1 TO 350:NEXT DE:CALL &H9808
 60 CLS:PRINT" This is a new screen."
 70 FOR DE=1 TO 350:NEXT DE
 80 CALL &H9822
 90 PRINT" Hows that!"
100 STOP
110 DATA &HF3,&HDB,&HBF,&H21,&H00,&H3C
    &HCD,&H32,&H2C,&HF3,&H21,&H00,&HA0
    &H06,&H05,&HC5,6,193,&H0E,&HBE,&HED
    &HB2,&HC1,&H10,&HF6,&HC9
110 DATA &HF3,&HDB,&HBF,&H21,&H00,&H3C
    &HCD,&H44,&H2C,&HF3,&H21,&H00,&HA0
    &H06,&H05,&HC5,6,193,&H0E,&HBE,&HED
    &HB3,&HC1,&H10,&HF6,&HC9
```

The routine at &H9809 saves the text screen contents into main RAM starting at location &HA000 onwards, while the routine at &H9822 writes the buffer at location &HA000 to VRAM. Refinement of this could result in simple animation. In machine-code the program is,

```
9808  F3       DI           ; Disable interrupts
9809  DBBF     IN A,(BF)    ; Clear status register
980B  21003C   LD HL,3C00   ; Text screen address
980E  CD322C   CALL 2C32    ; Set up VDP for read
9811  F3       DI
9812  2100A0   LD HL,A000   ; Buffer area
9815  0605     LD B,05
9817  C5       PUSH BC      ; Read
9818  0600     LD B,00
981A  0EBE     LD C,BE      ; C= Port BE
981C  EDB2     INIR         ; Read until B=0
981E  C1       POP BC
981F  10F6     DJNZ Read    ; Complete screen?
9821  C9       RET
9822  F3       DI
```

```
9823  DBBF      IN A,(BF)      ; Clear status register
9825  21003C    LD HL,3C00     ; Text screen address
9828  CD442C    CALL 2C44      ; Set up VDP for write
982B  F3        DI
982C  2100A0    LD HL,A000     ; Buffer area
982F  0605      LD B,05
9831  C5        PUSH BC        ; Write
9832  0600      LD B,00
9834  0EBE      LD C,BE        ; C = VDP
9836  EDB2      OUTR           : Do until B=0
9838  C1        POP BC
9839  10F6      DJNZ Write     ; All the screen?
983B  C9        RET
```

Table XXX5: MODE SELECT BITS.

MS1	MS2	MS3	Screen type	
0	0	0	Graphics mode I	(32 x 24)
0	0	1	Graphics mode II	(256x192)
0	1	0	Multicolor mode	(64 x 48)
1	0	0	Text mode	(40 x 24)

Table XXX6: SIZE & MAG BITS.

Mag	Size	Bit size	Sega manual	
0	0	8 x 8	MAG 0	(single sprite)
0	1	16 x 16	MAG 1	(single sprite)
1	0	16 x 16	MAG 2	(double mag0)
1	1	32 x 32	MAG 3	(double mag1)

Table XXX7: SPRITE ATTRIBUTE TABLE.

```
 ----------------
|  Y POSITION    |
|----------------|
|  X POSITION    |
|----------------|
|  SPRITE NAME   |
|----------------|
|EC|0|0|0|COLOR  |
 ----------------
```

EC. If a logic one, it shifts the sprites to the left by 32 pixels.

COLOUR. The 4 bits make up the color of the sprite. Refer to Table XXX8 for the color values.

Table XXX8: COLOR VALUES.

0	Transparent	8	Red
1	Black	9	Light Red
2	Green	A	Deep Yellow
3	Light Green	B	Light Yellow
4	Dark Blue	C	Dark Green
5	Light Blue	D	Magenta
6	Dark Red	E	Gray
7	Cyan	F	White

Table XXX9: MULTI-COLOR MODE PROGRAM.

```
 5  DEFFNA(R) = INT(RND(1)*R) + &H3800
10  SCREEN 2,2 : CLS
20  FOR X = &HA000 TO &HA011
30  READ A : POKE X,A : NEXT A
40  DATA &HF3,&H3E,&H00,&HD3,&HBF
50  DATA &H3E,&H80,&HD3,&HBF
60  DATA &H3E,&HC8,&HD3,&HBF
70  DATA &H3E,&H84,&HD3,&HBF,&HC9
80  DH=&H11: DF=&H3800: DG=&H3B00
90  FOR DE=DF TO DG: VPOKE DE,DH
100 NEXT
110 CALL &HA000
120 X = FNA(&H300)
130 VPOKE X,RND(1)*&HFF
140 GOTO 120
```

In machine-code,

```
A000    F3      DI          ; Disable interrupts
A001    3E00    LD A,00     ; Select multi-mode
A003    D3BF    OUT (BF),A
A005    3E80    LD A,80     ; Register 0
A007    D3BF    OUT (BF),A
A009    3EC8    LD A,C8     ; Multi-mode
A00B    D3BF    OUT (BF),A
A00D    3E84    LD A,84     ; Register 1
A00F    D3BF    OUT (BF),A
A011    C9      RET
```

CHAPTER 3

THE SOUND GENERATOR.

The sound chip is a SN76489AN device. It requires 32 clock cycles for the transfer of data from the CPU to be latched internally. This involves the use of the Ready line being tied to the WAIT input of the Z80 CPU.
This means that when loading the sound generator chip with data, theCPU is actually slowed down. The SG contains three programmable tone generators and a noise source, the output of each controlled by a programmable attenuator. The SG chip is port mapped at &H7F. The frequency and register is selected by a two-byte combination, while only one byte is necessary for attenuation control.

FREQUENCY SELECTION.

To determine how to program the SG chip the following information is necessary,

Clock speed = 3.84Mhz

N = Clock speed / (32 * Required frequency)

where N is converted to a 10 digit binary number.
Thus, to generate a tone of 1000Hz;

N = 3840000 / 32 * 1000

= 120 (N is always rounded to an integer)

Now convert N to Binary = 0001111000 (Most significant bit first)

REGISTER SELECTION.

To determine which sound register Table XX10 is used.

WRITING THE FREQUENCY AND REGISTER TO THE SGC.

In the above example of a 1000Hz tone, N was derived into a 10 digit binary number of 0001111000. These ten bits, along with the register code from Table XX10 are used to form the two bytes required to program the desired frequency and sound channel. Thus the format of the two bytes is,

Byte One: 1 + Register Code + last 4 bits of N
Byte Two: 00 + first 6 bits of N

Thus for our example of a 1000Hz tone using register one,

Byte One = 10001000 (or &H88)
Byte Two = 00000111 (or &H07)

The tone is produced by outputting the two values to port &H7F, thus

```
OUT &H7F,&H88 : OUT &H7F,&H07
```

will produce the desired result.

ATTENUATION CONTROL.

Control of the programmable attenuators can be achieved by a single byte update. The format of this byte is as follows,

Single Byte = 1 + attenuation register + attenuation value

The attenuation register is three bits and is shown in Table XX11. The attenuation value is shown in Table XX12 and comprises four bits. Thus to attenuate tone register one to a value of 10db using Basic,

Single byte = 10010101 (&H95) so OUT &H7F,&H95

THE NOISE GENERATOR.

Updating the noise register and attenuator requires a single byte transfer. This byte is 11100 + FB + SR

FEEDBACK CONTROL (FB): If FB=1 then noise is "periodic" else if FB=1 then the noise is set to "white" noise.

SHIFT RATE (SR): Refer to Table XX13 for the values of the two SR bits.

ATTENUATION CONTROL OF NOISE REGISTER: This is the same as described earlier, only the register code is 111.

SAMPLE EXPLOSION: To generate an explosion, use "white noise" then slowly increase the attenuation from 0db to OFF. Thus the frequency control byte is,

11100 + 1 + 00 = 11100100 (or &HE4)

The attenuation bytes range from 0db to OFF thus the range is,

1111 + 0000 to 1111 + 1111 (or &HF0 to &HFF)

thus the program in Basic is,

```
10  OUT &H7F,&HE4 : FOR X = &HF0 TO &HFF : OUT &H7F,X
20  FOR DE = 1 TO 20 : NEXT DE
30  NEXT X
```

CREATING MUSIC: Table XX14 is a Basic program which allows the user to input a series of notes (up to 255) and then play them back. The program calculates the various bytes necessary to program the sound generator chip.

TABLE XX10 REGISTER CODES

Register	Binary Code
Register 1	000
Register 2	010
Register 3	100
Register 4	110

TABLE XX11 ATTENUATOR CODES.

Attenuator	Binary Code
Tone reg 1	001
Tone reg 2	011
Tone reg 3	101
Noise reg	111

Table XX12 ATTENUATION TABLE.

Attenuation Value	Binary Code	Attenuation Value	Binary Cod
0db	0000	2db	0001
4db	0010	6db	0011
8db	0100	10db	0101
12db	0110	14db	0111
16db	1000	18db	1001
20db	1010	22db	1011
24db	1100	26db	1101
28db	1110	OFF	1111

Table XX13. SHIFT RATE BITS.

SR0	SR1	Desired Frequency of Noise.
0	0	Clock/512
0	1	Clock/1024
1	0	Clock/2048
1	1	Frequency is that specified by Register 3

Table XX14
MUSIC PROGRAM & SOUND EDITOR.

```
Music and Sound Editor.

5 PATTERNC#&HD0,"7884B4A4A4B48478"
10 PN=&H7F:DIM X1(255),X2(255),X3(255)
,W(255),T2(255)
20 FLAG=0
30 SCREEN1,1:CLS
40 PRINT"Music Editor.        By B.Brown
. ";CHR$(&HD0);" 84"
50 PRINT"--------------       ----------
-"
60 PRINT:PRINT"Options"
70 PRINT "1 - Play memory area"
80 PRINT "2 - Create music     "
90 PRINT "3 - Edit music array"
100 PRINT:PRINT "Select desired option
:"
110 AA$=INKEY$:IF AA$=""THEN GOTO 110
120 IF AA$="1" THEN GOTO 820
130 IF AA$="2" THEN GOTO 900
140 IF AA$="3" THEN GOTO 1000
150 GOTO 110
200 REM INPUT ROUTINES
210 B1$="10000000"
220 PRINT"Freq (118~3500) of tone #";2
B;" ";:INPUT FT:IF FT<118 OR FT>3500 T
HEN GOTO 220
230 BT=3840000/(32*FT)
240 DB=INT(BT+.5):GOSUB 430
```

```
250 B1$=LEFT$(B1$,4)+A1$
260 B2$=A2$
270 INPUT"Tone level (1~15) ";TL
280 IF(TL<1)OR(TL>15)THEN270
290 DB=TL:GOSUB 430
300 B4$="1001"
310 B3$=B4$+RIGHT$(A2$,4)
320 GOSUB 800
330 REM N1=Byte1,N2=Byte2,N3=Atten
340 GS$=B1$:GOSUB670:N1=OB
350 GS$=B2$:GOSUB670:N2=OB
360 GS$=B3$:GOSUB670:N3=OB
370 PRINT"Desired rest period "
380 PRINT"before next note.";:INPUT ZC
:RETURN
390 REM PLAY ROUTINE
400 OUT(PN),N3:OUT(PN),N1:OUT(PN),N2
410 FOR TP=1 TO ZC:NEXT
420 RETURN
430 REM DEC TO BIN
440 REM INPUT=DB,OUTPUT=A1$,A2$
450 FORZZ=1TO10:AA(ZZ)=0:NEXTZZ
460 DB=INT(DB)
470 FORT3=1TO10
480 T2=DB MOD 2
490 IFT2=1THENAA(T3)=1
500 DB=INT(DB/2)
510 NEXTT3
520 A1$="":A2$="":FORZZ=1TO10
530 A1$=A1$+STR$(AA(ZZ)):NEXT ZZ
540 GOSUB580:A1$=SB$
550 A2$="00"+LEFT$(A1$,6)
560 A1$=RIGHT$(A1$,4)
570 RETURN
580 SA$=""
590 FOR S=1TOLEN(A1$)
600 IFMID$(A1$,S,1)=" "THEN620
610 SA$=SA$+MID$(A1$,S,1)
620 NEXT S:SB$=""
630 FOR S=1 TO LEN(SA$)
```

```
640 SB$=SB$+MID$(SA$,LEN(SA$)+1-S,1)
650 NEXT S
660 RETURN
670 REM STRING TO DECIMAL
680 REM INPUT=GS$,OUTPUT=OB
690 OB=0
700 IFMID$(GS$,1,1)="1"THEN OB=OB+128
710 IFMID$(GS$,2,1)="1"THEN OB=OB+64
720 IFMID$(GS$,3,1)="1"THEN OB=OB+32
730 IFMID$(GS$,4,1)="1"THEN OB=OB+16
740 IFMID$(GS$,5,1)="1"THEN OB=OB+8
750 IFMID$(GS$,6,1)="1"THEN OB=OB+4
760 IFMID$(GS$,7,1)="1"THEN OB=OB+2
770 IFMID$(GS$,8,1)="1"THEN OB=OB+1
780 RETURN
790 REM RESET SOUND CAHNNELS
800 OUTPN,159:OUTPN,191:OUTPN,223
810 OUTPN,255:RETURN
820 REM PLAY MUSIC
830 CLS:PRINT"Playing music.":PRINT"--
----------"
840 IF FLAG=0 THEN PRINT:PRINT"Music a
rray is empty.":GOSUB 1140:GOTO 30
850 FOR ZB=1 TO 255
860 N1=X1(ZB):N2=X2(ZB):N3=X3(ZB):ZC=W
(ZB):IF N1=0 AND N2=0 AND N3=0 THEN ZB
=255:GOTO 880
870 GOSUB 390:SOUND 0
880 NEXT ZB
890 GOTO 30
900 REM Create music
910 CLS:PRINT "Create Music.":PRINT"--
----------":PRINT:GOSUB 1140
920 INPUT"How many notes to play.";ZA
930 IF ZA>255 THEN GOTO 920
940 FOR ZB=1 TO ZA
950 GOSUB 700
960 X1(ZB)=N1:X2(ZB)=N2:X3(ZB)=N3:W(ZB
)=ZC:TZ(ZB)=FT
970 NEXT:X1(ZB)=0:X2(ZB)=0:X3(ZB)=0
```

```
980 GOSUB 1140:FLAG=1:GOTO 30
990 STOP
1000 REM Edit music
1010 CLS:PRINT "Edit Music.":PRINT"---
--------":PRINT:IF FLAG=0 THEN PRINT "
Buffer is empty.":GOSUB 1140:GOTO 30
1020 PRINT "Freq bytes can only be cha
nged, not"
1030 PRINT "inserted. Use the ";CHR$(&
H8E);" key to change a ":PRINT "tone,
else ";CHR$(&H8F);" key to move to the
 next":PRINT "tone, and CR to abort."
1040 FOR ZB=1 TO 255
1050 PRINT "Tone ";ZB;" is ";TZ(ZB);"H
z"
1055 PRINT "Wait period is";W(ZB)
1060 TR$="":TR$=INKEY$
1090 IF TR$=CHR$(30) THEN GOSUB 1150:G
OTO 1050
1100 IF TR$=CHR$(29) THEN GOSUB 1140:N
EXT
1110 IF TR$=CHR$(13) THEN 1130
1120 GOTO 1060
1130 GOSUB 1140:GOTO 30
1140 FOR DE=1 TO 200:NEXTDE:RETURN
1150 GOSUB 1140:GOSUB 200:X1(ZB)=N1:X2
(ZB)=N2:X3(ZB)=N3:W(ZB)=ZC:TZ(ZB)=FT:R
ETURN
```

CHAPTER 4

CASSETTE ROUTINES.

MAJOR ENTRY POINTS: The major entry points for the cassette routines are,

VERIFY	&H779F
LOAD	&H78D5
SAVE	&H7A40

PROGRAM FORMAT: The programs are saved in two stages. The first part is the Header section. This comprises sync bytes, and the 16 character filename. The main program is saved next, this includes address's and the actual program, ie, line numbers etc.

VERIFY/LOAD: These routines are prefixed with a small routine which searches for the filename of the program. The filenames may be up to sixteen characters long, and for loading or Verifying, the filename is stored at location &H82A3 onwards. The filename from header section of the tape is loaded, then compared with that stored in memory. If no filename was specified, the program jumps to the Load main program section. If a filename was specified, and found to match with that readfrom tape, the program is loaded. If the program does not match, a jump is made to the skip portion of the program.

FILENAME STORAGE: Location &H82A2 is used as a Filename found flag, and if zero then the next program found on the cassette is loaded, else a Filename Found flag, if zero then the program is loaded, otherwise skip is made. When saving a program, the filename is taken from the keyboard input buffer, corresponding to &H83A3 [up to 16 bytes]. If the filename is less than 16 bytes, then the filename is padded with blanks.

ADDITIONAL INFO: Table XX15 lists the major entry points of all the cassette routines, and their function. Also listed are the sub-routines which are called also. Table XX16 and XX17 are Basic programs which load the Header and Program Bytes respectively to the video screen.

AUTOLOAD AND EXECUTE BASIC PROGRAMS: This may be achieved by poking a machine language routine into reserved memory. If the computer is then reset, the program will not be erased. The machine code routine calls the main entry point of LOAD, then calls &H6C37

which is the RUN entry point for Basic programs. However, location &H82A2 which holds the filename found flag must be zeroed to indicate that the next program found must be loaded. Table XX18 illustrates how this may be achieved.

MERGING BASIC PROGRAMS: A program to merge two Basic programs must use a machine-code routine to save the Ram pointers in memory, call the Load routine in Rom, reset the pointers and call the load routine a second time. The program listed below is a combination of most of that which has already been covered. It must be noted however, that the second Basic program's line numbers must be greater than the first or part of the program will be deleted.

```
 10 SCREEN 1,1: CLS : PRINT "BASIC MERGER"
 20 PRINT: PRINT "Loading Mcode."
 30 POKE &H8168,0 : FOR X=&HFF00 TO &HFF2F
 35 REM Reserve memory space at top of memory
 40 READ A: POKE X,A: NEXT
 70 PRINT "Press PLAY to load first program."
 80 POKE &H82A2,0: CALL &HFF00
 90 END
100 DATA &HCD,&HEF,&H78
110 DATA &H2A,&H62,&H81,&H2B,&H22,&H60,&H81
120 DATA &H3E,&H00,&H32,&HA2,&H82,&H21,&H1F
130 DATA &HFF,&HCD,&H6F,&H4A,&HCD,&HEF,&H78
140 DATA &H21,&H00,&H98,&H22,&H60,&H81,&HC9
150 DATA 76,111,97,100,32,50,110,100,32,112
160 DATA 114,111,103,114,97,109,&H0D
```

In machine-code the program is,

```
FF00  CALL 78EF          (LOAD prog1)
      LD HL (8162)       (Basic end pointer)
      DEC HL
      LD (8160),HL       (Store it into Basic start)
      LD A,00
      LD (82A2),A        (Filename found flag)
      LD HL,FF1F
      CALL 4A6F          (Print text message)
      CALL 78EF          (LOAD prog2)
      LD HL,9800
      LD (8160),HL       (Set pointer to prog1)
      RET
FF1F  'load 2nd program.'  (Text message)
```

TABLE XX15. CASSETTE ROUTINES IN ROM.

Rom Address (Hex)	Function
3A03	A Delay routine using the BC register
3A0F	Write sync bytes to tape
3A12	Write byte in A to tape
779F	Verifying Start
77F7	Skip
7822	Found
785D	Verifying End
788F	Verifying Error
78D5	Loading Start
78FD " 790E	Compare Filenames
792B	Skip
7956	Found
7982	Load Program
799AA	Loading End
79E9	Tape Read Error
7A40	Saving Start
7A59 " 7A85	Save Filename
7A94	Save number of bytes
7AB9	Save Sync Bytes
7AD2	Save Program
7AED	Saving End
7B07	Write HL to tape
7B13	Pad Filename with Blanks

Table XX16. LOAD HEADER TO VIDEO SCREEN.

```
10 SCREEN 1,1 : CLS : PRINT "Press Play
   to Load program." : B = 0
20 FOR X = &H78EF TO &H7923
30 POKE &HA8EF+B,PEEK(X) : B = B + 1 : NEXT X
40 POKE &HA90B,&HD3
50 POKE &HA90C,&HBE
60 POKE &HA924,&HC9
70 CALL &HA8EF
80 GOTO 70
```

Table XX17. LOAD PROGRAM BYTES TO VIDEO SCREEN.

```
10 SCREEN 1,1 : CLS : PRINT "Press Play
   to Load Program."
20 FOR X = &HA000 TO &HA022
30 READ A : POKE X,A : NEXT
40 CALL &HA000
50 STOP
60 DATA &HF3,&HCD,&H00,&H3A,&HCD,&H06
70 DATA &H3A,&HFE,&H17,&H20,&HF5,&H2A
80 DATA &H60,&H81,&H06,&H00,&HCD,&H0A
90 DATA &H7A,&HD3,&HBE,&H3E,&H3F,&HC4
100 DATA &H48,&H24,&H23,&H1B,&H7A,&HB3
110 DATA &H20,&HF0,&HC3,&HA9,&H79
```

Table XX18. AUTO LOAD AND RUN BASIC PROGRAMS.

```
10 SCREEN 1,1 : CLS : PRINT " Press Play
   to Load and Run Program."
20 DATA &HCD,&HD5,&H78,&HCD,&H37,&H6C
30 POKE &H8168,0
40 FOR X = &HF000 TO &HF005
50 READ A : POKE X,A : NEXT
60 POKE &H82A2,0
70 CALL &HF000
```

CHAPTER 5

THE KEYBOARD AND JOYSTICKS.

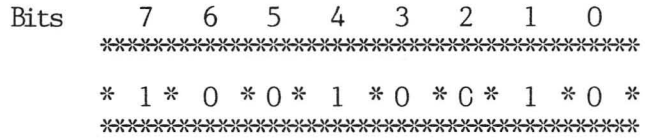
The keyboard, joysticks, cassette and printer are all controlled by an interface chip (8255). This interface chip allows the connection of the devices to the CPU, and the transfer of information between them. The interface is programmed by the cpu, ie it is instructed on what to look for and what it must do. This process is normally transparant to the user, ie the user is unaware of the process's being executed.

THE KEYBOARD: The keyboard is arranged in a matrix layout of eight columns by eleven rows. Only one column may be activated at one time, and the columns are controlled by a decoder chip. The keyboard rows are connected to two different ports, only one can be read by the cpu at any time. An intersection (which occurs due to a keypress) between the column and row of the matrix is detected by the cpu and is then interpreted to find out the actual key being pressed. Refer to Table XX20 for the key matrix layout.

THE 8255 PERIPHERAL INTERFACE CHIP: This is a programmable chip, and allows the connection of the keyboard, cassette, printer and joysticks to the cpu. The PIA has three ports, A,B, and C and a control register. The information written to the control register will determine the status of each port (ie inputs or outputs). When the ports are used as outputs, the written data is latched or held internally till the next update. In the SEGA the following is to be noted,

Port A is input, mapped at &HDC, connected to X columns of key matrix
Port B in input, mapped at &HDD, connected to X columns of key matrix
Port C is output, mapped at &HDE, connected to Y column of key matrix
Control register is mapped at &HDF

The data or words written to the control register to set up the specific ports as input or output are,



Bit 4 = Controls A
Bit 3 = Controls C upper
Bit 1 = Controls B
Bit 0 = Controls C lower

thus the byte to intialise the PIA is &H92 or 146 decimal.

ADDRESSING THE KEY-MATRIX.

The lower three bits (0,1,2) of Port C is used to address the Y columns of the keymatrix. The output of Port C is applied to a 74LS145 BCD decoder, which provides a one out of eight output to activate only one Y column at a time. The status of the three lower Port C bits will determine which output of the decoding chip is activated. Table XX18 lists the combinations of these three bits and the resultant activated output of the decoder. Table XX20 lists the keyboard matrix.

SCANNING THE KEYBOARD USING MACHINE-CODE: Table XX21 lists a Basic program which pokes a machine-code subroutine into memory. This routine intialises the PIA with &H92, then outputs a specified byte to port C, thus selecting the desired Y column of the key-matrix. This byte is specified in line 75 of the program, and refer to Table XX19 for the value of the byte and its appropriate column. It then loads the value of ports A and B, storing them in &HA000 and &HA001 respectively, before returning to Basic. By checking the returned code from port A or B, it is thus possible to search for a specified key press. Having assembled the routine into line 5 of the program, all data statements etc can be deleted from the final program. Table XX26 lists a program which scans the keyboard, and moves sprites etc, all using machine code.

MISCELLANEOUS CONNECTIONS: The remaining tables list the various connectrions of the SEGA and their appropriate function.

THE PRINTER PLOTTER: This relies on a single chip microprocessor, a 6805 up. Being a factory programmed device, it must be replaced in total, ie you haven't got access to the software controlling the 6805. Also note that the same mechanism is used by the ORIC, and COMMODORE printer plotters, and the spares are also the same, ie pens etc. Some SHARP printers are also identical, so shop around for pens, paper etc

Table XX19. THE 74LS145 DECODER COMBINATIONS.

PC2	PC1	PC0	Y Column	Hex Byte (outputted to &HDE)
0	0	0	Y0	00
0	0	1	Y1	01
0	1	0	Y2	02
0	1	1	Y3	03
1	0	0	Y4	04
1	0	1	Y5	05
1	1	0	Y6	06
1	1	1	Y7	07

Table XX21. BASIC KEY-SCAN PROGRAM.

```
5 REM AAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAA
7 REM Line 5 has about 100 A's in it.
10 SCREEN 1,1:CLS
20 FOR X=&H9808 TO &H981F
30 READ A:POKE X,A:NEXT X
40 CALL &H9808
50 PRINT"Port A =";PEEK(&HA000);
   "  Port B =";PEEK(&HA001)
60 GOTO 40
65 DATA &HF3
70 DATA &H3E,&H92,&HD3,&HDF
75 DATA &H3E,&H00
80 DATA &HD3,&HDE,&HDB,&HDC
85 DATA &H32,&H00,&HA0,&HDB,&HDD
90 DATA &H32,&H01,&HA0,&H3E,&H92
95 DATA &HD3,&HDF,&HC9
100 REM Y0=00,Y1=01,Y2=02,Y3=03
110 REM Y4=04,Y5=05,Y6=06,Y7=07
120 REM Change the 2nd byte in line 75
130 REM to scan a different row.
```

Table XX20. KEYBOARD MATRIX LAYOUT.

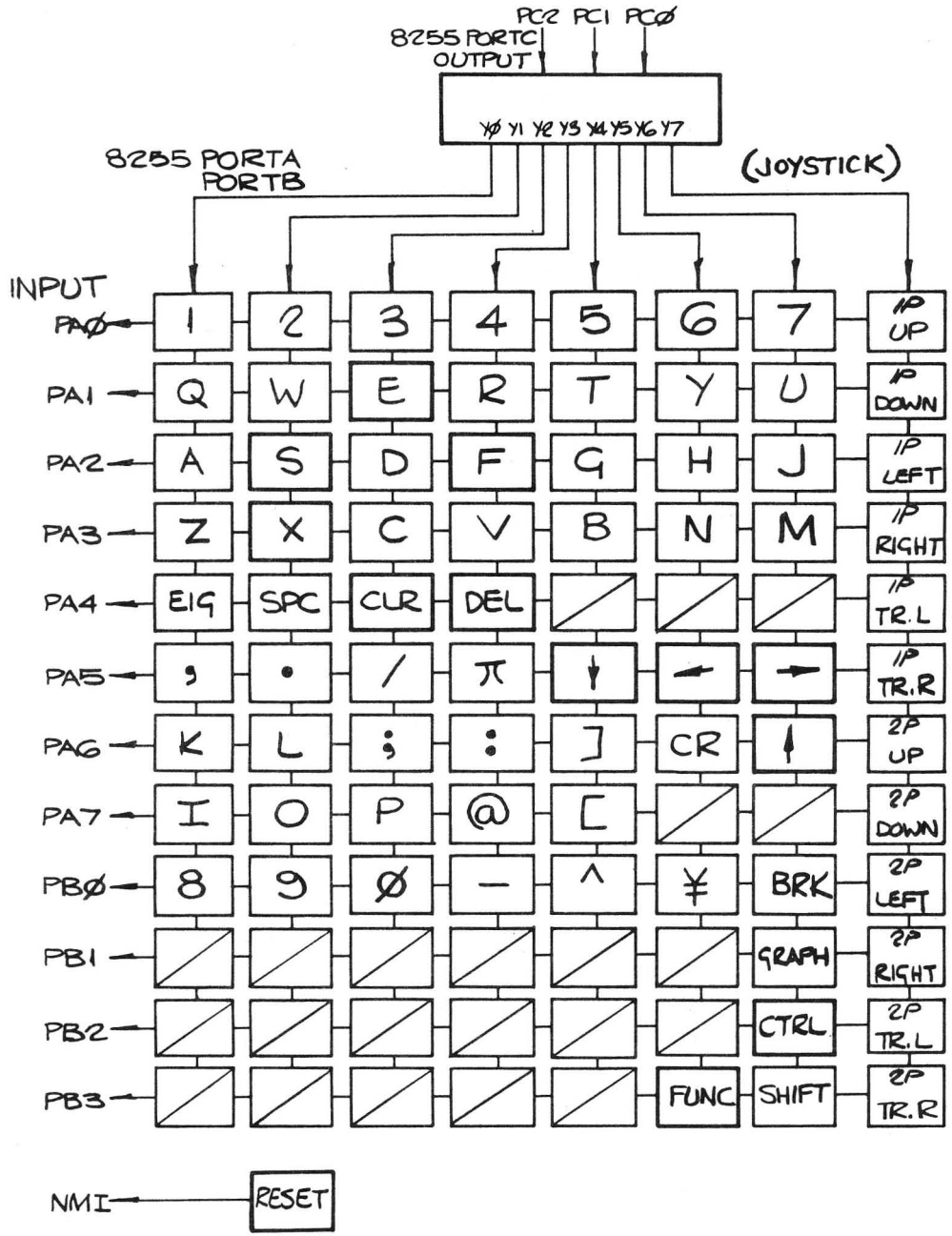


Table XX22. JOYSTICK PIN CONNECTIONS.

Pin Number	Function
1	Up
2	Down
3	Left
4	Right
5	No connection
6	Left fire
7	No connection
8	Common
9	Right fire

Table XX23. PRINTER PIN CONNECTIONS.

Pin Number	Function
1	Fault
2	Busy
3	Data
4	Reset
5	Feed
6	Gnd
7	No connection

Table XX24. PORT B & C CONNECTIONS.

PB0	Key Matrix	PC0	Key Matrix
PB1	" "	PC1	" "
PB2	" "	PC2	" "
PB3	" "	PC3	Not Used
PB4	Not Used	PC4	Cassette Output
PB5	Fault (Printer)	PC5	Data (Printer)
PB6	Busy (Printer)	PC6	Reset (Printer)
PB7	Cassette Input	PC7	Feed " "

Table XX25. VIDEO PORT CONNECTIONS.

Pin Number	Function
1	Audio
2	Gnd
3	Video
4	Gnd
5	Gnd

EXPANSION EDGE CONNECTOR.

Pin number		Pin number (Component side)	
1	A0	1	+5v
2	A1	2	+5v
3	A2	3	CSRAM *
4	A3	4	CEROM2 *
5	A4	5	MEMRD *
6	A5	6	MEMWR *
7	A6	7	I/ORD *
8	A7	8	I/OWR *
9	A8	9	No Connection
10	A9	10	MREQ *
11	A10	11	CON
12	A11	12	RAS1 *
13	A12	13	CAS1 *
14	A13	14	RAM A7
15	D0	15	RAS2 *
16	D1	16	CAS2 *
17	D2	17	MUX *
18	D3	18	A14
19	D4	19	A15
20	D5	20	No Connection
21	D6	21	GND
22	D7	22	GND

NOTE: * means active low

Table XX26. MCODE DEMONSTRATION.

```
9FFA  01E803      START:  LD BC 03E8
9FFD  CD033A              CALL  3A03   ;Delay routine
A000  3E92         MAIN:  LD A,92
A002  D3DF                OUT(DF),A
A004  3E00                LD A,00
A006  D3DE                OUT(DE),A
A008  DBDC                IN(DC),A
A00A  FEFE                CP FE        ;Check for key "1"
A00C  2806                JR Z LEFT
A00E  FEF7                CP F7        ;Check for key "Z"
A010  281A                JR Z RIGHT
A012  18E6                JR   START
A014  21013B       LEFT:  LD HL,3B01
A017  CD322C              CALL  2C32
A01A  DBBE                IN(BE),A
A01C  3D                  DEC A
A01D  FE00                CP 00
A01F  2807                JR Z  INC2
A021  CD442C      WRIT2:  CALL 2C44
A024  D3BE                OUT(BE),A
A026  181C                JR   DETECT
A028  3EFE         INC2:  LD A,FE
A02A  18F5                JR WRIT2
A02C  21013B      RIGHT:  LD HL,3B01
A02F  CD322C              CALL  2C32
A032  DBBE                IN(BE),A
A034  3C                  INC A
A035  FEFF                CP FF
A037  2807                JR Z INC3
A039  CD442C      WRIT1:  CALL 2C44
A03C  D3BE                OUT(BE),A
A03E  1804                JR  DETECT
A040  3EFE         INC1:  LD A,FE
A042  18F5                JR WRIT1
A044  F3          DETECT:  D1
A045  DBBF                IN(BF),A
A047  E620                AND 20
A049  FE20                CP 20
A04B  2803                JR Z COLL
A04D  C39A9F              JP  START
```

```
A050  21033B        COLL:  LD HL 3B03
A053  CD442C               CALL 2C44
A056  3E04                 LD A,04
A058  D3BE                 OUT(BE),A
A05A  0605                 LD B,05
A05C  CDA056        BEEP:  CALL  56A0
A05F  10FB                 DJNZ BEEP
A061  21033B               LD HL  3B03
A064  CD442C               CALL   2C44
A067  3E08                 LD A,08
A069  D3BE                 OUT(BE),A
A06B  C3FA9F               JP  START
A06E  219DA0        PATTERN: LD HL A09D   ;Set up sprite
                                           8 x 8 patterns
A071  E5                   PUSH HL
A072  210018               LD HL 1800     ;Write to pattern
                                           area Vram
A075  CD442C               CALL 2C44
A078  E1                   POP HL
A079  0610                 LD B,0F
A07B  7E            WRIT1: LD A,(HL)      ;Write the pattern
                                           bytes
A07C  D3BE                 OUT(BE),A
A07E  10FB                 DJNZ WRIT1
A080  C9                   RET
A081  21ADA0        ATTRIB: LD HL A0AD    ;Set up sprite
                                           attributes
A084  E5                   PUSH HL
A085  21003B               LD HL 3B00     ;Vram sprite attrib
                                           address
A088  CD442C               CALL 2C44
A08B  E1                   POP HL
A08C  0608                 LD B,7
A08E  7E            WRIT2: LD A,(HL)      ;Write attrib bytes
A08F  D3BE                 OUT(BE),A
A091  10FB                 DJNZ WRIT2
A093  C9                   RET
A094  CD6EA0        ENTRY: CALL A06E      ;Do sprite patterns
A097  CD81A0               CALL A081     ;Do sprite attributes
A09A  C3FA9F               JP   9FFA     ;Go do main routine
A09D  AAAAAAAAAAAAAAAA     SPRITE 0 PATTERN
A0A5  002070A8F8500000     SPRITE 1 PATTERN
A0AD  64500102             SPRITE 0 ATTRIBUTE
A0B1  64640004             SPRITE 1 ATTRIBUTE
```

MAIN ENTRY: A094 NO ERRORS DETECTED

IN BASIC

```
 10 SCREEN 2,2: CLS
 20 FOR X=&H9FFA TO &HA0B4
 30 READ A: POKE X,A: NEXT
 40 CALL &HA094: STOP
 50 DATA &H01,&HE8,&H03,&HCD,&H03,&H3A
 60 DATA &H3E,&H92,&HD3,&HDF,&H3E,&H00
 70 DATA &HD3,&HDE,&HDB,&HDC,&HFE,&HFE
 80 DATA &H28,&H06,&HFE,&HF7,&H28,&H1A
 90 DATA &H18,&HE6,&H21,&H01,&H3B,&HCD
100 DATA &H32,&H2C,&HDB,&HBE,&H3D,&HFE
110 DATA &H00,&H28,&H07,&HCD,&H44,&H2C
120 DATA &HD3,&HBE,&H18,&H1C,&H3E,&HFE
130 DATA &H18,&HF5,&H21,&H01,&H3B,&HCD
140 DATA &H32,&H2C,&HDB,&HBE,&H3C,&HFE
150 DATA &HFF,&H28,&H07,&HCD,&H44,&H2C
160 DATA &HD3,&HBE,&H18,&H04,&H3E,&H01
170 DATA &H18,&HF5,&HF3,&HDB,&HBF,&HE6
180 DATA &H20,&HFE,&H20,&H28,&H03,&HC3
190 DATA &HFA,&H9F,&H21,&H03,&H3B,&HCD
200 DATA &H44,&H2C,&H3E,&H04,&HD3,&HBE
210 DATA &H06,&H05,&HCD,&HA0,&H56,&H10
220 DATA &HFB,&H21,&H03,&H3B,&HCD,&H44
230 DATA &H2C,&H3E,&H08,&HD3,&HBE,&HC3
240 DATA &HFA,&H9F,&H21,&H9D,&HA0,&HE5
250 DATA &H21,&H00,&H18,&HCD,&H44,&H2C
260 DATA &HE1,&H06,&H10,&H7E,&HD3,&HBE
270 DATA &H10,&HFB,&HC9,&H21,&HAD,&HA0
280 DATA &HE5,&H21,&H00,&H3B,&HCD,&H44
290 DATA &H2C,&HE1,&H06,&H08,&H7E,&HD3
300 DATA &HBE,&H10,&HFB,&HC9,&HCD,&H6E
310 DATA &HA0,&HCD,&H81,&HA0,&HC3,&HFA
320 DATA &H9F,&HAA,&HAA,&HAA,&HAA,&HAA
330 DATA &HAA,&HAA,&HAA,&H00,&H20,&H70
340 DATA &HA8,&HF8,&H50,&H00,&H00,&H64
350 DATA &H50,&H01,&H02,&H64,&H64,&H00
360 DATA &H04
```

TABLE XX27 CHARACTER MANIPULATOR

```
10 SCREEN 1,1:CLS
20 DIM UD(8),P2(8)
30 CURSOR0,0
35 PRINT"CHARACTER MANIPULATOR"
40 PRINT :PRINT "OPTION."
50 PRINT " 1=INVERSE"
60 PRINT " 2=REVERSE"
70 PRINT " 3=UPSIDE DOWN"
80 PRINT CHR$(5)
90 A$=INKEY$:IF A$<"1"OR A$>"3"THEN GO
TO 90
100 FOR DE=1 TO 100:NEXT DE
110 A=VAL(A$):ON A GOSUB 130,200,330
120 GOTO 30
130 REM INVERSE
140 GOSUB 410
150 FOR A=0 TO 7
160 UD(A)=UD(A) XOR &HFF
170 NEXT
180 GOSUB 480:REM CALL UPDATE
190 GOSUB 530:RETURN
200 REM REVERSE
210 GOSUB 410
220 FOR C=0 TO 7:P2(C)=0:NEXT
230 FOR C=0 TO 7
240 IF (UD(C)AND 128)=128 THEN P2(C)=4

250 IF (UD(C)AND 64)=64 THEN P2(C)=P2(
C)+8
260 IF (UD(C)AND 32)=32 THEN P2(C)=P2(
C)+16
270 IF (UD(C)AND 16)=16 THEN P2(C)=P2(
C)+32
280 IF (UD(C)AND 8)=8 THEN P2(C)=P2(C)
+64
```

```
290 IF (UD(C)AND 4)=4 THEN P2(C)=P2(C)
+128
300 NEXT
310 FOR B=0 TO 7:UD(B)=P2(B):NEXT
320 GOSUB 480:GOSUB 530:RETURN
330 REM UPSIDE DOWN
340 GOSUB 410
350 B=7:FOR A=0 TO 7
360 P2(A)=UD(A):NEXT
370 FOR A=0 TO 7
380 UD(A)=P2(B):B=B-1
390 NEXT :GOSUB 480
400 GOSUB 530:RETURN
410 REM COMMON
420 CURSOR 0,6
430 INPUT "CHARACTER VALUE ?";X
440 AD=&H1800+X*8:Y=0
450 FOR B=AD TO AD+7
460 UD(Y)=VPEEK(B):Y=Y+1
470 NEXT:RETURN
480 REM UPDATE
490 AD=&H1800+X*8:Y=0
500 FOR B=AD TOAD+7
510 VPOKE(B),UD(Y):Y=Y+1
520 NEXT:RETURN
530 CURSOR 30,0:PRINT CHR$(X)
540 GOTO30

530 CURSOR 30,0:PRINT CHR$(X):RETURN
```

CHAPTER 6

INTERESTING BITS AND PIECES.

This chapter is dedicated to all those wives who spend endless hours trying to convince their husbands to give up that stupid toy, and spend more time with them. Gathered together in this chapter are the solutions to a wide range of problems, so now there is no excuse for husbands to spend all night trying all those various programming methods that don't work.

A SEGA PRINT USING STATEMENT:

Some people wish that the SEGA had a PRINT USING statement. Basically this allows you to format numbers which always appear in the same place, and with the same number of decimal places after the decimal point. So here is a routine which will always display numbers to two decimal places, and always place it so that the numbers line up with the decimal point always in the same column.

```
 10 INPUT A
 20 A=INT(A*100+.5)/100
 30 A$+STR$(A)
 40 L=LEN(A$)
 50 FOR I=1 TO L
 60 IF MID$(A$,I,1)="." THEN GOTO 100
 70 NEXT I
 80 A$=A$+".00"
 90 GOTO 110
100 IF I=L-1 THEN A$=A$+"0"
110 FOR K=1 TO 10-1
120 A$=" "+A$
130 NEXT K
140 PRINT A$
150 GOTO 10
```

The value of 10 in line 110 has been used to give a number with twelve characters long. The program would be used as a subroutine within your particular program, and accessed by a gosub statement.

A FAULTY RENUMBERER:

Not that you would want one anyway! No, just a note to say that the SEGA RENUM command does not work properly. To illustrate its major weakness, type in the following program.

```
 10 INPUT" String";A$
 20 IF LEN(A$)<7 THEN GOTO 500
 30 IF LEN(A$)>6 THEN 600
 40 GOTO 10
500 PRINT " A$<7": GOTO 10
600 PRINT " A$>6": GOTO 10
```

Then use the RENUM command. The program will be renumbered as follows,

```
10 INPUT" String";A$
20 IF LEN(A$)<7 THEN GOTO 500
30 IF LEN(A$)>6 THEN 600
40 GOTO 10
50 PRINT " A$<7": GOTO 10
60 PRINT " A$>6": GOTO 10
```

Notice that the line numbers in lines 20 and 30 have not been changed. Whenever a goto or line number follows a string manipulation, the renum feature will not work properly.

ERASING CHARACTERS ON THE GRAPHICS SCREEN:

Try the following program,

```
10 SCREEN 2,2:CLS
20 FOR X=1000 TO 1050
30 CURSOR 150,0:PRINT " Score:";X
40 NEXT
50 END
```

As you will have noticed, the characters written tend to overwrite each other. After a couple of prints, you can't read the score at all. The way to overcome this is by using a print CHR$(5) command. This erases everything to the right of the current cursor position. Modify the program to that below,

```
10 SCREEN 2,2:CLS
20 FOR X=1000 TO 1050
30 CURSOR 150,0:PRINT CHR$(5)
40 CURSOR 150,0:PRINT " Score:";X
50 NEXT
60 END
```

As you notice now, the print chr$(5) statement allows you print in the same position twice. However, note that the chr$(5) erases all information to the right of the cursor (except sprites). Its use must therefore be limited to close to the right hand edge, ie for displaying scores, etc, otherwise it could erase part of your pictures or graphic displays.

CONVERTING ORIC PROGRAMS TO THE SEGA:

Listed are the ORIC commands with the appropiate SEGA command;

ORIC	EQUIV SEGA [For use on Text screen only]
PLOT X,Y,"#"	CURSOR X,Y: PRINT"#" [for the Sega Y(23, the ORIC Y(27]
EXPLODE	OUT127,228 FOR X=240 TO 255 OUT 127,X FOR Y=1 TO 15 NEXT: NEXT
KEY$	INKEY$
IF SCRN(H,V)< >32	IF VPEEK ((V*40) +H+&H3C02)< >32
PAPER 0:INK 7	COLOR 7,0 [generally ignore]

FOR A=(46080+(ASC("#")*8)) TO .... (ASC("&")
This command sets up user-defined graphics. The equivalent command for the SEGA is

PATTERNC#ASC("#"),"whatever the 8 data bytes were"
all the way to
PATTERNC#ASC("&"),"etc"

[It is a good idea to map out the bit patterns used as the Sega allows only six of the eight columns to be used when defining the character patterns.]

```
CURSET 100,10      X1=100: Y1=10: LINE (X1,Y1)-(X1-10,Y1+20+P),1:
DRAW -10,20+P,1                BLINE  (X1,Y1)-(X1-20,Y1+25)
DRAW -20,25,0

WAIT 20            FOR DE=1 TO 25
                   NEXT DE

GET Z$             INPUT Z$
```

SOME NOTES ABOUT THE GRAPHICS:

There appear to be some strange things happening when using the graphics screen. This is due to the routines in ROM being designed with circles etc in mind. An example of this limitation follows,

```
10 SCREEN 2,2: CLS : COLOR 1,11,
   (0,0) - (255,191),12
20 LINE (57,50) - (100,100),15,BF
30 CURSOR 66,75: COLOR 1,4
40 PRINT "test"
50 GOTO 50
```

As you probably guessed, "test" is not printed and the background color is ignored. This is because the routine does not erase the previous contents of the video screen when writing new data to it. A possible solution is to add these lines to the previous program,

```
  5 ZX=&H2000: ZC=&H14
 25 GOSUB 100
 45 GOSUB 110
100 FOR Y=70 TO 90:BLINE(64,Y)-
    (95,Y): NEXT: RETURN
110 FOR X=64 TO 95 STEP 8
120 FOR Y=70 TO 90
130 VPOKE INT(Y/8)*256+INT(X/8)*8
    +YMOD8+ZX,ZC
140 NEXT: NEXT: RETURN
```

This demonstrates the writing to the color attribute area of the graphics screen. This technique should be used to add more color onto the screen, as the graphic chip does allow 16 colors to be used in a character block (ie 8 x 8). The computer is capable of generating color displays rivalling most computers today, and should be comparable to more expensive computers if programmed correctly.

LISTING PROGRAMS:

When listing Basic programs, pressing the SPACEBAR will pause the listing. Pressing it again, the listing will continue.

HALTING THE GAMES CARTRIDGES:

Pressing RESET will halt the game, while a further press will restart the game.

LOAD OR SAVE VARIABLES, MACHINE-CODE PROGRAMS, STRING ARRAYS ETC:

Well, we may as well go for broke on the last topic in this book. If you have survived to this point then congratulations are in order! By now, some of the concepts should be clicking together and so to finally put you off the deep end, lets get into saving or loading variables etc.
Basic Principle involved: We have already discovered that Basic uses locations in the Reserved RAM area in order to locate where to find the program, variables, strings etc. The LOAD and SAVE routines look up locations &H8160 to &H8165. These locations store the start and end address's of the Basic program and Variable storage areas. The area of memory between the start and end address of the Basic program is

saved to tape, but the storage area isn't. In a flash, we discover that if we replace these start and end address's of the Basic program with the address's for the variables, then call the SAVE routine, the computer will save the variables to tape for us. Having saved them to tape, if we reset the address's to what they were previously, all will be fine, and our program will continue on as usual. The same principle applies to the LOAD process. Okay, so the steps involved in designing this are,

1) Set up a machine-code routine to accomplish the task
2) Save the start/end address's somewhere safe
3) Get the variable address's and put them into where the start/end address's of the Basic program are stored
4) Call the LOAD or SAVE routine in ROM
5) Reset the original address's
6) Return back to Basic ....

Setting up the mcode routine. Lets hide the machine-code in a REM statement.

```
5 REM AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
```

Line 5 has as many 'A's as possible, about 250 of them. Now the first 'A' in line five is stored at address &H9808. Our machine-code routine can thus be poked into address &H9808 onwards (though the length of our routine cannot exceed 250). The pointers that we pick up from locations &H8160' must be saved somewhere safe, so we will store them as follows,

&H9808/9	Poke this with start address to be saved
&H980A/B	Poke this with end address to be saved
&H980C/D	Store &H8160/1 here
&H980E/F	Store &H8162/3 here
&H9810/1	Store &H8164/5 here
&H9812/3	Store &H8166/7 here
&H9814"	Machine code routine

The actual mcode routine written in machine code,

```
ENTRY   LD HL,(8160)
SAVE    LD (980C),HL        ;save Basic start
        LD HL,(8162)
        LD (980E),HL        ;save Basic end
        LD HL,(9808)
        LD (8160),HL        ;new start
        LD HL,(980A)
        LD (8162),HL        ;new end
        CALL 7A69           ;call save routine
        LD HL,(980C)
        LD (8160),HL        ;restore Basic start
        LD HL,(980E)
        LD (8162),HL        ;restore Basic end
        RET
```

```
ENTRY   LD A,00
LOAD    LD (82A2),A         ;zero filefound flag
        LD HL,(8160)
        LD (980C),HL        ;save Basic start
        LD HL,(8162)
        LD (980E),HL        ;save Basic end
        LD HL,(8164)
        LD (9818),HL        ;save string start
        LD HL,(8166)
        LD (9812),HL        ;save string end
        LD HL,(9808)
        LD (8160),HL        ;new start
        LD HL,(980A)
        LD (8162),HL        ;new end
        CALL 78EF           ;call load routine
        LD HL,(980C)
        LD (8160),HL        ;restore Basic start
        LD HL,(980E)
        LD (8162),HL        ;restore Basic end
        LD HL,(9810)
        LD (8164),HL        ;restore string start
        LD HL,(9812)
        LD (8166),HL        ;restore string end
        RET                 ;return to Basic
```

The LOAD part of the routine is slightly different, because the string pointers are altered by the load routine. Thus they are saved, and later restored after the load has executed. Location 82A2 is stored with zero this tells the load routine to load the first file it encounters.

The machinecode is now converted to DATA statements, and poked into the 'A's that make up line 5, eg

```
FOR X=&H9808 TO &H9808+number of data bytes
READ A:POKE X,A: NEXT
```

Once this is achieved, the routines can be called and executed. This has been used in the following three programs written by the author,

ACCOUNTS RECEIVABLE
ACCOUNTS PAYABLE
MAILING LIST

The major portion of the book is now over. Finally, the appendix lists some games and a utility program that you the reader can type into your SEGA computer. I hope that they provide you with hours of use, as well as the information in this book.

APPENDIX

This appendix lists three Basic games for a LVIIIA cartridge, and a very long STARTREK program for a LVIIIB. Appended is a PATTERN EDITOR utility program, which allows the user to create patterns on an 8 x 8 grid, any pixel can be set/reset, the hex values are worked out for you, and it shows you a normal size, and expanded size of the character you make up.

The Basic games are; CROSSROADS, ALIEN ATTACK, ONE ARMED BANDIT.

```
10 HI=0:SCREEN1,1:CLS:GOSUB 330:GOSUB
   460
20 GOSUB290
30 V=14:H=19:GOSUB 220
40 CURSORH,V:PRINT" ";
50 IFINKEY$=" "THENV=V-1
60 BEEP
70 CURSOR0,4:PRINTA$:CURSOR0,6:PRINTB$

80 CURSOR0,10:PRINTB$:CURSOR0,12:PRINT
   A$
90 IFVPEEK(V*40+H+&H3C00+2)<>32THEN160

100 CURSORH,V:PRINTCHR$(253);
110 L1$=LEFT$(A$,1):R1$=RIGHT$(A$,35)
120 L2$=LEFT$(B$,35):R2$=RIGHT$(B$,1)
130 A$=R1$+L1$:B$=R2$+L2$
140 IFV=2THEN210
150 GOTO40
160 FORX=1TO8:SOUND1,200,10:NEXTX:
    SOUND0:IFML>0THENML=ML-1:GOTO 30
170 IFSC>HITHENHI=SC
180 CURSOR6,20:PRINT"Press any key";:
    PRINT" to continue":FORX=1000TO500
    STEP-5:SOUND1,X,5:NEXTX:SOUND0
190 IF INKEY$=""THEN190
200 GOTO 20
210 BEEP:SC=SC+10:GOTO30
220 CLS:PRINT"High ";HI,"Score ";SC
230 CURSOR 0,4:PRINTCHR$(1)+A$
240 CURSOR 0,6:PRINTCHR$(2)+B$
250 CURSOR 0,10:PRINTCHR$(5)+B$
260 CURSOR 0,12:PRINTCHR$(6)+A$
270 CURSOR H,V:PRINTCHR$(253);
```

```
280 FORA=1 TO ML:CURSOR A,1:PRINT CHR$
    (253);:NEXT:RETURN
290 SC=0
300 A$="x?      x?      x?      x?  x?   x
?    "
310 B$="''      ''    ''        ''     ''
 ''    "
320 ML=5:RETURN
330 REM User defined graphics
340 PATTERNC#&H60,"000000070F3F0C00"
350 PATTERNC#&H27,"000000F0FEFE0600"
360 PATTERNC#&H3F,"000000F8FCFF0600"
370 PATTERNC#&H25,"000000031F3F1800"
380 SCREEN 2,2:CLS:CURSOR 60,95
390 COLOR1,2,(0,0)-(255,191),2:PRINT
    CHR$(17);
400 FORXX=1TO10:READC,L$
410 COLORC,2
420 PRINTL$;:NEXTXX
430 FORY=1 TO 3:FORX=800 TO 1000 STEP5
    :SOUND1,X,10:NEXTX:NEXTY
440 DATA 1,"C",14,"R",7,"O",10,"S",4,
    "S",15,"R",13,"O",12,"A",11,"D",9,
    "S"
450 SCREEN 1,1:RETURN
460 REM INSTRUCTIONS
470 CLS:PRINT"Welcome to CROSSROADS."
480 PRINT:PRINT" The object of the gam
e is to cross"
490 PRINT"the road without being knock
ed down"
500 PRINT"by a truck or car. You have
5 lifes"
510 PRINT"and move upwards by pressing
 the    "
520 PRINT"                SPACEBAR
        "
```

```
530 PRINT:PRINT"Good luck...(press any
 key to play)"
540 IF INKEY$="" THEN540
550 SOUND0:RETURN
```

ALIEN ATTACK LVIIIA/B

```
10 GOSUB 180:GOTO 170
20 IF FL=0 THEN FX=SX:FY=178
30 FL=1:FY=FY-4
40 SPRITE 2,(FX,FY),2,3
50 IFFY<25THEN70
60 GOTO90
70 IFFX=TRTHENGOSUB100
80 IFFY<15 THEN FL=0:SPRITE 2,(FX,192)
,2,3
90 RETURN
100 R=R+10:CURSORR+5,0:COLOR 13,1:PRIN
T CHR$(250);:IFR>80THENR=0
110 OUT127,228:FORXX=240TO255:OUT127,X
X:FORNP=1TO5:NEXT:NEXT:RETURN
120 IF INKEY$="X" THEN SX=SX+SZ:IF SX>
SU THEN SX=SU
130 IF INKEY$="Z" THEN SX=SX-SZ:IF SX<
SU THEN SX=SU
140 IF (INKEY$="S") AND (FL=0) THEN FX
=SX:GOSUB20
150 IF FL=1 THEN GOSUB20
160 VPOKE SA,SX:RETURN
170 FOR TR=TS TO TT STEP TU:GOSUB 120:
VPOKESB,TR:NEXT:GOTO 170
180 SCREEN 2,2:CLS:COLOR 6,1,,1
190 PATTERNS#1,"0000003C7E7EFF42"
200 PATTERNS#0,"0000001818187EFFFF"
210 PATTERNS#2,"0000004848480000"
```

```
220 SX=120:SPRITE 0,(SX,178),0,4
230 SPRITE 1,(0,20),1,11
240 TU=2:TS=0:TT=245:SZ=2:SV=245:SH=8
250 FL=0:SA=&H3B01:SB=&H3B05
260 CURSOR 110,0:PRINT "ALIEN";:RETURN
```

ONE ARMED BANDIT LVIIIA/B

```
10 SCREEN 1,1:CLS:SCREEN2,2:CLS
20 COLOR 1,14,(0,0)-(255,191),4
30 GOSUB 1030:GOSUB 1310
40 M0=100:CL$=CHR$(5):HL$=CHR$(230)
50 PRINT CHR$(17):DIM C1(6)
60 C1(0)=6:C1(1)=12:C1(2)=4:C1(3)=9
70 C1(4)=15:C1(5)=14
80 PRINT" One Armed Bandit."
90 PRINT CHR$(16)
100 PRINT" A simple game of chance.."
110 PRINT" Payoff..."
120 CURSOR 8,50:PRINT"One ";:SPRITE 1,
(28,48),0,6:CURSOR 35,50:PRINT" pays $
10.00"
130 CURSOR 8,60:PRINT"Two ";:SPRITE 0,
(28,58),1,12:CURSOR 35,60:PRINT" pays
$20.00"
140 CURSOR 8,70:PRINT"One ";:SPRITE 2,
(28,68),2,5:CURSOR 35,70:PRINT" pays $
200.00"
150 SPRITE 3,(8,78),3,15:SPRITE 4,(18,
78),3,15:SPRITE 5,(28,78),3,15:CURSOR
35,80:PRINT" JACKPOT  $1000.00":PRINT
160 COLOR 5:PRINT"Press any key to pla
y."
170 IF INKEY$="" THEN GOTO 170
180 GOSUB 1110
190 CURSOR 20,180:COLOR15:PRINTCHR$(5)
;"Money = $";CHR$(29);M0:COLOR 2
```

```
200 SPRITE 0,(50,43),5,C
210 SPRITE 1,(80,43),5,C
220 SPRITE 2,(110,43),5,C
230 COLOR 1
240 CURSOR 20,150:PRINT "Press ";:COLO
R 8:PRINT"SPACEBAR";:COLOR 1:PRINT" to
 Play":COLOR 2
250 IF INKEY$<>CHR$(32) THEN GOTO 250
260 M0=M0-10
270 CURSOR 20,180:COLOR 15:PRINTCHR$(5
);"Money = $";CHR$(29);M0:COLOR 2
280 GOSUB 710
290 GOSUB 500:GOSUB 570:GOSUB 640
300 IF R1=R2 AND R1=R3 THEN GOSUB 360:
GOTO 340:REM Jackpot
310 IF R1=1 AND R2=1 OR R1=1 AND R3=1
OR R2=1 AND R3=1 THEN GOSUB 430:GOTO 3
40
320 IF R1=0 OR R2=0 OR R3=0 THEN GOSUB
 400:GOTO 340
330 IF R1=2 OR R2=2 OR R3=2 THEN GOSUB
 460
340 REM End of loop
350 GOTO 190
360 REM Jackpot
370 IF R1=3 THEN M0=M0+1000:GOTO 390
380 RETURN
390 BEEP2:COLOR 4:CURSOR 180,40:PRINT"
Jackpot":BEEP2:OUT127,228:FOR DE=240 T
O 255:OUT127,DE:FOR DF=1 TO 15:NEXT:NE
XT:CURSOR 180,40:COLOR 14:PRINT CHR$(5
):COLOR 2:RETURN
400 REM One Cherry
410 M0=M0+10
420 BEEP2:COLOR 4:CURSOR 180,40:PRINT"
Cherry ":BEEP2:OUT127,228:FOR DE=240 T
O 255:OUT127,DE:FOR DF=1 TO 15:NEXT:NE
XT:COLOR 14:CURSOR 180,40:PRINT CHR$(5
):COLOR 2:RETURN
```

```
430 REM Two Apples
440 M0=M0+20
450 BEEP2:COLOR 4:CURSOR 180,40:PRINT"
Apples ":BEEP2:OUT127,228:FOR DE=240 T
O 255:OUT127,DE:FOR DF=1 TO 15:NEXT:NE
XT:COLOR 14:CURSOR 180,40:PRINT CHR$(5
):COLOR 2:RETURN
460 REM Mystery
470 MP=INT(RND(1)*100)+1
480 M0=M0+MP
490 BEEP2:COLOR 4:CURSOR 170,40:PRINT"
Mystery $";CHR$(29);MP:BEEP2:OUT127,22
4:FOR DE=240 TO 255:OUT127,DE:FOR DF=1
 TO 35:NEXT:NEXT:COLOR 14:CURSOR 170,4
0:PRINT CHR$(5):COLOR 2:RETURN
500 REM ROW 1 ROTATE
510 FOR X=1 TO 16
520 Y=INT(RND(1)*6)
530 C=C1(Y)
540 SPRITE 0,(50,43),Y,C
550 BEEP:NEXT:R1=Y
560 RETURN
570 REM ROW 2 ROTATE
580 FOR X=1 TO 16
590 Y=INT(RND(1)*6)
600 C=C1(Y)
610 SPRITE 1,(80,43),Y,C
620 BEEP:NEXT:R2=Y
630 RETURN
640 REM ROW 2 ROTATE
650 FOR X=1 TO 16
660 Y=INT(RND(1)*6)
670 C=C1(Y)
680 SPRITE 2,(110,43),Y,C
690 BEEP:NEXT:R3=Y
700 RETURN
710 REM Handle pull
720 COLOR 14
730 COLOR 14
```

```
740 CURSOR 146,38:PRINTHL$
750 CURSOR 146,46:PRINTHL$
760 CURSOR 146,54:PRINTHL$
770 COLOR 2
780 CURSOR 146,78:PRINTHL$
790 CURSOR 146,86:PRINTHL$
800 CURSOR 146,94:PRINTHL$
810 CURSOR 146,102:PRINTHL$
820 CURSOR 146,62:PRINTHL$
830 COLOR 2
840 CURSOR 146,78:PRINTHL$
850 CURSOR 146,86:PRINTHL$
860 CURSOR 146,94:PRINTHL$
870 CURSOR 146,102:PRINTHL$
880 COLOR 14
890 CURSOR 146,102:PRINTHL$
900 CURSOR 146,94:PRINTHL$
910 CURSOR 146,86:PRINTHL$
920 CURSOR 146,78:PRINTHL$
930 COLOR 2
940 CURSOR 146,62:PRINTHL$
950 CURSOR 146,54:PRINTHL$
960 CURSOR 146,46:PRINTHL$
970 CURSOR 146,38:PRINTHL$
980 OUT 127,224:FOR DE=240 TO 255
990 OUT 127,DE:FOR DF=1 TO 15
1000 NEXT:NEXT
1010 RETURN
1020 STOP
1030 PATTERNS#0,"000066FFFF7E3C18":REM
 Heart
1040 PATTERNS#1,"040876FFFF7E3C00":REM
 Apple
1050 PATTERNS#2,"3844440810100010":REM
 Mystery
1060 PATTERNS#3,"2473A5A47E25A57E":REM
 Dollars
1070 PATTERNS#4,"000000FFFF000000":REM
 Bar
```

```
1080 PATTERNS#5,"0000000000000000":REM
 Blank
1090 PATTERNS#6,"7E8199919199817E":REM
 Copyright
1100 RETURN
1110 CLS:COLOR 2,14,(0,0)-(255,191),7
1120 PRINT
1130 PRINT"     ";Z1$
1140 PRINT"     ";Z2$
1150 PRINT"     ";Z2$
1160 PRINT"     ";LEFT$(Z3$,17)
1170 PRINT"     ";Z4$
1180 PRINT"     ";Z4$
1190 PRINT"     ";Z3$
1200 PRINT"     ";Z5$
1210 PRINT"     ";Z6$;CHR$(229)+CHR$(23
6)+CHR$(230)
1220 PRINT"     ";Z7$
1230 SPRITE 6,(105,78),6,1
1240 PRINT"     ";Z6$
1250 PRINT"     ";CHR$(229)+CHR$(144)+C
HR$(144);" April 1984";CHR$(144)+CHR$(
144)+CHR$(229)
1260 PRINT"     ";LEFT$(Z5$,17)
1270 PRINT"     ";LEFT$(Z5$,17)
1280 PRINT"    ";CHR$(149)+LEFT$(Z3$,17
)+CHR$(150)
1290 COLOR 1:CURSOR 42,20:PRINT" Sega
Jackpot":COLOR 2
1300 RETURN
1310 REM Set up strings
1320 Z1$="":Z2$="":Z3$="":Z4$=""
1330 Z5$="":Z6$="":Z7$=""
1340 FOR AA=1 TO 17:READ AZ
1350 Z1$=Z1$+CHR$(AZ)
1360 NEXT
1370 FOR AA=1 TO 17:READ AZ
1380 Z2$=Z2$+CHR$(AZ)
1390 NEXT
```

```
1400 FOR AA=1 TO 20:READ AZ
1410 Z3$=Z3$+CHR$(AZ)
1420 NEXT
1430 FOR AA=1 TO 20:READ AZ
1440 Z4$=Z4$+CHR$(AZ)
1450 NEXT
1460 FOR AA=1 TO 20:READ AZ
1470 Z5$=Z5$+CHR$(AZ)
1480 NEXT
1490 FOR AA=1 TO 17:READ AZ
1500 Z6$=Z6$+CHR$(AZ)
1510 NEXT
1520 FOR AA=1 TO 18:READ AZ
1530 Z7$=Z7$+CHR$(AZ)
1540 NEXT
1550 RETURN
1560 DATA 149,229,229,229,229,229,229,
229,229,229,229,229,229,229,229,229,15
0
1570 DATA 229,32,32,32,32,32,32,32,32,
32,32,32,32,32,32,32,229
1580 DATA 229,229,229,229,229,229,229,
229,229,229,229,229,229,229,229,229,22
9,32,32,230
1590 DATA 229,229,32,32,32,229,229,32,
32,32,229,229,32,32,32,229,229,32,32,2
30
1600 DATA 229,144,144,144,144,144,144,
144,144,144,144,144,144,144,144,144,22
9,229,32,230
1610 DATA 229,144,144,32,32,32,32,32,3
2,32,32,32,32,32,144,144,229
1620 DATA 229,144,144,32,66,46,66,114,
111,119,110,32,32,32,144,144,229,229
```

```
5 REM AAAAAAAAAAAAAAAAAA
10 RESTORE 5020
30 PATTERNC#254,"0030FCCC7830CC00":PAT
TERNC#135,"1F1F1F0000000000"
40 DEF FNA(BY)=INT(RND(1)*BY)+1
50 E$=CHR$(250):K$=CHR$(254):B$=CHR$(2
47):S$="*":GOSUB 5000:RESTORE 70
60 DIMS(8,8),R(6),D(8,8),J(10),G(8,8),
CN$(4):CL$=CHR$(236):SR$=" ........":F
ORA=0TO4:READCN$(A):NEXT
65 SS$=".*"+CHR$(254)+CHR$(247)+CHR$(2
50)
70 DATA Green,Yellow,***RED***,Docked,
"           "
80 DV=6:FORA=1TO6:READDV$(A):NEXT
90 DATA Ion Drive,S.R Sensors,L.R Sens
ors,Pulsar Ctrl,Photon Tubes,Damage Ct
rl
100 CC$="CSLPTDGI":CC=8
110 GOSUB3040
120 K1=FNA(15)+10:S1=FNA(149)+150:B1=F
NA(9):C9=1:FORI=1TO8:FORJ=1TO8:G(I,J)=
-1:NEXT:NEXT:D1=FNA(1999)+2000/10*10:D
3=40:D2=D3+D1:GOSUB3060
130 FORI=1TO8:FORJ=1TO8:S(I,J)=0
140 NEXT:NEXT:L1=0:L2=0:GOSUB3130:DE=1
0
150 FORI=1TOINT(S1):DE=DE+1:IFDE>255TH
ENDE=10
160 SPRITE 10,(DE,180),8,2:A1=FNA(8):A
2=FNA(8)
170 IF S(A1,A2)>8 THEN 160
180 S(A1,A2)=S(A1,A2)+1
190 NEXTI
210 FORI=1TOK1:A1=FNA(8):A2=FNA(8):S(A
1,A2)=S(A1,A2)+100
```

```
215 NEXT:GOSUB3160
220 FORI=1TOB1:A1=FNA(8):A2=FNA(8):S(A
1,A2)=S(A1,A2)+10:NEXT
230 E1=FNA(8):E2=FNA(8):E7=FNA(8):E8=F
NA(8):P=3000:C1=0:T1=10:GOSUB790
240 SCREEN1,1:CLS:COLOR1,11
250 PRINT"Orders: Stardate =";D1:PRINT
:PRINT" As commander of the United Sta
rship  PEGASUS, your mission is to rid
 the   galaxy of the deadly Cygon's.":
PRINT" To do this, you must destroy th
e     Cygon invasion force of ";K1;" B
attle"
260 PRINT"cruisers. You have ";D3;" so
lar days to  complete your mission."
270 PRINT"The Pegasus is currently loc
ated at   Quadrant ";E2;"-";E1:PRINT"S
ector   ";E8;"-";E7:PRINT:GOSUB4000:PR
INT"Press I for instuctions."
280 U$=INKEY$:IFU$=""THEN280
290 IFU$="I"THEN 3170
300 SCREEN1,1:CLS:GOSUB2760
310 GOSUB1660
320 SOUND0:CURSOR0,22:PRINT"Your Comma
nd Captain ?";:Y=2
330 V$=INKEY$:IF V$="" THEN 330
340 IFV$=CHR$(13)THENGOSUB2760:GOTO320

350 IFCA=1THENCLS:GOSUB 2760
360 FORA=1TOCC:IF V$=MID$(CC$,A,1) THE
NC2=A-1:GOTO390
370 NEXT
380 GOTO 530
390 IFC2<>6THEN520
400 CLS:PRINTCL$;"Galaxy map.":PRINT"
————————"
410 CA=1:CB=0:PRINT" ";:FORI=1TO8:PRIN
T"  ";I;:NEXT:PRINT:PRINT"    ";:C2=31:
```

```
GOSUB780:FORI=1TO8:PRINTI;
420 FORJ=1TO8:IF G(I,J)<0 THENPRINT":
- ";:GOTO510
430 QW=G(I,J):IFQW<10THENQZ$="00"+STR$
(QW):GOTO460
440 IF QW>9 AND QW<100 THENQZ$="0"+STR
$(QW):GOTO460
450 QZ$=STR$(QW)
460 WZ$=""
470 FORQW=1TOLEN(QZ$):IFMID$(QZ$,QW,1)
=" "THEN490
480 WZ$=WZ$+MID$(QZ$,QW,1)
490 NEXT:QZ$=WZ$
500 PRINT":";QZ$;
510 NEXTJ:PRINT":":NEXTI:PRINT"    ";:G
OSUB780:PRINT:PRINT"Pegasus currently
at";E2;"-";E1:GOTO320
520 ONC2+1GOTO 1320,650,1040,2190,2370
,560,530,3170
530 CLS:CA=1:CB=0:PRINTCL$;"Your choic
es of command are:":PRINT"C - Course d
irective":PRINT"S - Short range sensor
 scan"
540 PRINT"L - Long range sensor scan":
PRINT"D - Damage control report":PRINT
"P - Fire Mega Pulsar's"
550 PRINT"T - Fire Photon torpedoes ":
PRINT"G - Galaxy map":GOTO320
560 CLS:PRINTCL$;"Damage Control Repor
t.":PRINT"————————————————————"
570 IF R(6)<0 THEN590
580 PRINT"Device          State of repai
r":FORC2=1TO6:PRINTDV$(C2),R(C2):NEXT:
GOTO 640
590 TP=FNA(3):CURSOR0,15:ONTPGOTO600,6
10,620
600 PRINT "Engineering reports,":GOTO6
30
```

```
610 PRINT "1st Officer reports,":GOTO6
30
620 PRINT "Navigation reports,"
630 PRINT"Damage control not available
"
640 FORQW=1TO250:NEXTQW:GOTO 310
650 GOSUB660:GOTO310
660 CLS:PRINTCL$;"Short Range Sensor S
can.":PRINT"----------"

670 IF R(2)<0 THENCB=0:FORA=0TO7:CURSO
R0,A+5:PRINTSR$;:NEXTA:PRINT:PRINT"Sho
rt range sensors damaged...":RETURN
680 IFPQ=1THENG(E1,E2)=S(E1,E2)
690 CB=1:GOSUB790
700 CURSOR0,4:PRINT" 12345678"
710 FORA=0TO7
720 CURSOR0,A+5:PRINTSR$;" ";A+1
730 NEXTA
740 FORI=1TO8
750 FORJ=1TO8
760 IF D(I,J)<>0 THENCURSORJ,I+4:PRINT
MID$(SS$,D(I,J)+1,1);
770 NEXTJ:NEXTI:CURSOR0,16:PRINT"Secto
r  ";E8;":";E7;"                ":RETURN

780 FORI1=1TOC2:PRINT"-";:NEXTI1:PRINT
:RETURN
790 IF L1=E1 THEN 810
800 GOTO 820
810 IF L2=E2 THEN RETURN
820 L1=E1:L2=E2:FORI=1TO8:FORJ=1TO8:D(
I,J)=0:NEXTJ:NEXTI:D(E7,E8)=4:PQ=1
830 IF E1<1 THEN E1=8
840 IF E1>8 THEN E1=1
850 IF E2<1 THEN E2=8
860 IF E2>8 THEN E2=1
870 IF S(E1,E2)-INT(S(E1,E2)/10)*10=0
```

```
THEN 920
880 FORI=1TOS(E1,E2)-INT(S(E1,E2)/10)*
10
890 E3=FNA(8):E4=FNA(8)
900 IF D(E3,E4)<>0 THEN 890
910 D(E3,E4)=1:NEXTI
920 IF INT(S(E1,E2)/10)-INT(S(E1,E2)/1
00)*10=0 THEN970
930 FORI=1TOINT(S(E1,E2)/10)-INT(S(E1,
E2)/100)*10
940 E3=FNA(8):E4=FNA(8)
950 IF D(E3,E4)<>0 THEN 940
960 D(E3,E4)=3:NEXTI
970 IF INT(S(E1,E2)/100)=0 THEN1020
980 FORI=1TOINT(S(E1,E2)/100):J(I)=300

990 E3=FNA(8):E4=FNA(8)
1000 IF D(E3,E4)<>0 THEN 990
1010 D(E3,E4)=2:NEXTI
1020 IF CB=1 THEN GOSUB 660
1030 RETURN
1040 CLS:PRINTCL$;"Long Range Sensor R
eport.":PRINT"————————————————————
——"
1050 IF R(3)<>0 THEN TP=FNA(3):GOTO 10
70
1060 GOTO 1120
1070 ONTPGOTO 1080,1090,1100
1080 PRINT"Navigation reports the Sens
ors are ":GOTO 1110
1090 PRINT"1st Officer reports the Sen
sors are ":GOTO 1110
1100 PRINT"Engineering reports the Sen
sors are "
1110 PRINT"out Captain ";VA$:GOTO 310
1120 PRINT"Long range scan on Quadrant
";E2;"-";E1
1130 PRINT:C2=13:GOSUB780:FORI=E1-1TOE
```

```
1+1:FORJ=E2-1TOE2+1
1140 IFI<1THEN1310
1150 IFI>8THEN1310
1160 IF J<1 THEN1300
1170 IF J>8 THEN1300
1180 G(I,J)=S(I,J)
1190 QW=S(I,J):IFQW<10THENQA$="00"+STR
$(QW):GOTO1220
1200 IF QW>9 AND QW<100 THENQA$="0"+ST
R$(QW):GOTO1220
1210 QA$=STR$(QW)
1220 QZ$=""
1230 FORQX=1TOLEN(QA$)
1240 IFMID$(QA$,QX,1)=" "THEN1260
1250 QZ$=QZ$+MID$(QA$,QX,1)
1260 NEXTQX:QA$=QZ$
1270 PRINT":";QA$;
1280 NEXTJ:PRINT": "
1290 NEXTI:GOSUB780:GOTO310
1300 PRINT": - ";:GOTO1280
1310 PRINT": - : - : - :":GOTO1290
1320 A=544:GOSUB2700
1330 CURSOR0,14:PRINT"Course";:GOSUB28
00:C2=N:IFDD=1THEN310
1340 P1=8:IF R(1)=0 THEN1370
1350 P1=.2:IF R(1)<-3 THEN1370
1360 P1=INT((4+R(1))*2)/10
1370 IFP1<1THENPZ$="0"+STR$(P1)
1380 PZ$=STR$(P1):PY$="":FORAZ=1TOLEN(
PZ$)
1390 IFMID$(PZ$,AZ,1)=" "THEN1410
1400 PY$=PY$+MID$(PZ$,AZ,1)
1410 NEXTAZ:PZ$=PY$
1420 CURSOR0,15:PRINT"Light Speed (0 -
 ";PZ$;")";:GOSUB2800:IFDD=1THEN310
1430 C3=N:IF C3<0 OR C3>8 THEN1370
1440 IF C3<=P1 THEN1460
1450 CURSOR0,16:PRINT"Engineering repo
rts":PRINT"Ion Drive is damaged...":PR
```

```
INT"Max Light speed=";P2$:GOTO1370
1460 IFC3<1THEN1480
1470 FOR XV=C3*100 TO C3*400 STEP 5:SO
UND1,XV+10,15:NEXTXV
1480 P=P-16*C3-5:N1=INT(8*C3):IFN1=0TH
EN1610
1490 N2=-COS(C2*.0174533):IF ABS(N2)<=
.01 THENN2=0
1500 N3=SIN(C2*.0174533):IF ABS(N3)<=.
01 THENN3=0
1510 AV=1:AW=N1
1520 E3=E7:E4=E8:P1=INT(E3+N2+.4):P2=I
NT(E4+N3+.4):E7=P1:E8=P2
1530 IF P1<1 THEN1970
1540 IF P1>8 THEN1970
1550 IF P2<1 THEN1990
1560 IF P2>8 THEN1990
1570 IF D(P1,P2)<>0 THEN2010
1580 D(E3,E4)=0:D(P1,P2)=4
1590 IFCB=1THENCURSOR0,16:PRINT"Sector
  ";E8;":";E7;"                    "
1600 AV=AV+1:IFAV<=AW THEN 1520
1610 Q=PP
1620 D1=D1+1:FORI=1TO6:IF R(I)=0 THEN1
650
1630 R(I)=R(I)+1
1640 IF R(I)>0 THENR(I)=0
1650 NEXTI:GOTO2020
1660 FORI=E7-1TOE7+1
1670 IF I<1 THEN1740
1680 IF I>8 THEN1740
1690 FORJ=E8-1TOE8+1
1700 IF J<1 THEN1730
1710 IF J>8 THEN1730
1720 IF D(I,J)=3 THEN1790
1730 NEXTJ
1740 NEXTI
1750 C1=0:IFPQ=0THEN1770
```

```
1760 IF S(E1,E2)>=100 THEN C1=2:GOTO18
00
1770 IFP<=500 THEN C1=1
1780 GOTO 1800
1790 C1=3:P=3000:T1=10:FORI=1TO6:R(I)=
0:NEXTI
1800 IF C1=0 THEN 1820
1810 GOTO 1840
1820 FOR I=1TO6:IF R(I)<0 THEN C1=1
1830 NEXTI
1840 RETURN
1850 C2=RND(1):IF C2<.25 THEN1910
1860 IFC2<.8 THEN1960
1870 CURSOR0,16:PRINT"Space Storm    "
:IFC1<3THEN1890
1880 PRINT"Starbase Shields protect th
e ship!":GOTO1960
1890 C2=FNA(DV):PRINTDV$(C2);" Damaged
      ":R(C2)=R(C2)-5*RND(1):IF C2<>
2THEN 1960
1900 CB=0:GOTO 1960
1910 FOR I=1 TO 6
1920 IFR(I)=0THEN NEXTI
1930 GOTO 1960
1940 PRINT"TRUCE ";:PRINTDV$(I);" Repa
ir status has improved":R(I)=R(I)+2*RN
D(1)
1950 IF R(I)>0 THEN R(I)=0
1960 GOSUB1660:RETURN
1970 S2=SGN(P1-1):E1=E1+S2:E7=INT(P1)-
8*S2:L1=E1+1
1980 IF P2>=1 AND P2<=8 THEN2000
1990 S2=SGN(P2-1):E2=E2+S2:E8=INT(P2)-
8*S2:L2=E2+1
2000 GOSUB790:CURSOR0,15:PRINT"Quadran
t";E2;":";E1;"                ":GOTO1590
2010 E7=E3:E8=E4:CURSOR0,17:PRINT"Pega
sus blocked at";INT(P2);"-";INT(P1):BE
```

```
EP:GOTO1620
2020 GOSUB1660:GOSUB1850:GOSUB2050:IFP
<0THEN2690
2030 IFD1>D2THEN2670
2040 GOTO310
2050 IFPQ=0THEN2180
2060 IF S(E1,E2)<100 THEN2180
2070 IFC1<>3THEN2090
2080 GOSUB1880
2090 G=1:H=0:FORI=1TOS(E1,E2)/100
2100 H=H+1:IFH<=8THEN2120
2110 H=1:G=G+1:IFG>8THENG=0
2120 IF D(G,H)<>2 THEN2100
2130 Q1=G-E7:Q2=H-E8:D4=SQR(ABS(Q1*Q1+
Q2+Q2))+.1:P5=INT((RND(1)*ABS(J(I)-1))
):J(I)=J(I)-P5:IFC1=3THEN2170
2140 P6=P5/D4:P=P-P6:CURSOR0,18:PRINT"
Pegasus Hit ":PRINT"CYGON at Sector";H
;"-";G:IFP6<10THEN2170
2150 A1=3-INT(P6/100):IF FNA(A1)<>1 TH
EN2170
2160 C2=FNA(DV):PRINTDV$(C2);" Has sus
tained DAMAGE":BEEP2:R(C2)=R(C2)-(P6/4
2)*RND(1):IFC2=2THENCB=0
2170 NEXTI:IFP<=0THEN2680
2180 RETURN
2190 CLS:PRINTCL$;"Mega Pulsar's.":PRI
NT"————————————"
2200 IF R(4)<>0 THENCURSOR0,15:PRINT"P
ulsar's are in need of repair...":GOTO
310
2210 CURSOR0,15:PRINT"Pulsar's 'LOCKED
' on target"
2220 PRINT"Energy available:";INT(P):P
RINT"Number of units to fire";:GOSUB28
00:IFDD=1THEN1620
2230 C2=N:IF C2>P OR C2<0 THEN2220
2240 GOSUB 2980
```

```
2250 P=P-C2:Q=PP:IF S(E1,E2)<100 THEN2
350
2260 P5=C2/INT(S(E1,E2)/100):G=0:H=1:F
ORI=1TOS(E1,E2)/100
2260 P5=C2/INT(S(E1,E2)/100):G=0:H=1:F
ORI=1TOS(E1,E2)/100
2270 H=H+1:IFH>=9THENH=1
2280 G=G+1:IFG>=9THENG=0
2290 IF D(G,H)>2 THEN2270
2300 Q1=G-E7:Q2=H-E8:IF D(G,H)<2 THEN2
270
2310 D4=SQR(Q1*Q1+Q2*Q2):P6=P5/D4:J(I)
=J(I)-P6:CURSOR0,19:PRINT"CYGON at";H;
"-";G;" hit":IFJ(I)>0THEN2340
2320 KD$=KD$+K$:GOSUB2900:CURSOR0,20:P
RINT"Destroyed ";KD$
2330 FORI2=I+1TOS(E1,E2)/100:J(I2-1)=J
(I2):NEXTI2:S(E1,E2)=S(E1,E2)-100:K1=K
1-1:D(G,H)=0:I=I+1:G(E1,E2)=S(E1,E2)
2340 NEXTI
2350 IFK1=0THEN2660
2360 GOSUB2050:GOTO310
2370 CLS:PRINT CL$;"Photon Torpedoes."
:PRINT"————————————————"
2380 IF R(5)<>0 THEN PRINT:PRINT" Out
of order":GOTO 310
2390 IFT1<=0THENPRINT:PRINT"Torpedoes
all fired.":GOTO 310
2400 GOSUB 2700:PRINT "Torpedo Course"
;:GOSUB 2800:IFDD=1THEN310
2410 C2=N:T1=T1-1:N2=-COS(C2*.0174533)
:IF ABS(N2)<=.01 THEN N2=0
2420 N3=SIN(C2*.0174533):IFABS(N3)<=.0
1THENN3=0
2430 GOSUB2920
2440 P1=E7:P2=E8
2450 P1=P1+N2:P2=P2+N3
2460 IF P1<.5 THEN 2650
```

```
2470 IF P1>8.5 THEN 2650
2480 IF P2<.5 THEN 2650
2490 IF P2>8.5 THEN 2650
2500 IFD(INT(P1+.4),INT(P2+.4))=0THEN2
450
2510 OND(INT(P1+.4),INT(P2+.4))GOTO253
0,2580,2610
2520 GOTO 2450
2530 CURSOR 0,15:PRINT"*** HIT STAR **
*":IFCB=1THENY=0
2540 IFFNA(4)<>1THENPRINT"Burned up":G
OTO2630
2550 S(E1,E2)=S(E1,E2)-1:IFFNA(10)=1TH
ENGOSUB2850:GOTO2680
2560 GOSUB2840:IFFNA(10)=1THEN2680
2570 GOTO 2620
2580 IFFNA(20)=1THENPRINT"Failed to de
tonate":GOTO 2630
2590 IFFNA(30)=1THENPRINT"Cygons shiel
ds have held.":GOTO2630
2600 KD$=KD$+K$:PRINT "Cygon Ship dest
royed ";KD$:S(E1,E2)=S(E1,E2)-100:K1=K
1-1:GOTO 2620
2610 PRINT "Starbase destroyed.":S(E1,
E2)=S(E1,E2)-10
2620 D(INT(P1+.4),INT(P2+.4))=0:IFCB=1
THENG(E1,E2)=S(E1,E2)
2630 IFK1=0THEN2660
2640 GOSUB2050:GOTO310
2650 PRINT"Torpedo missed":GOTO 2630
2660 PRINT"CYGONS DESTROYED":END
2670 CLS:PRINT"STARDATE * TIME RUN OUT
":END
2680 PRINT"PEGASUS DESTROYED":END
2690 END
2700 CLS:PRINTCL$;"Navigation Directiv
e.":PRINT"————————————————————":PRIN
T:PRINT
```

```
2710 PRINT"         0 ":PRINT
2720 PRINT"    315    45":PRINT
2730 PRINT"   270      90":PRINT
2740 PRINT"    225    135":PRINT
2750 PRINT"        180":PRINT:RETURN
2760 CA=0:GOSUB1660:CLS:PRINTCL$;"Stat
us Report.":PRINT"                  "
2770 PRINTCL$;"Stardate ";D1:PRINTCL$;
"Condition";" ";CN$(C1):PRINTCL$;"Quad
rant ";E2;"-";E1:PRINTCL$;"Sector    ";
:IF R(2)>=0 THENPRINTE8;"-";E7:GOTO279
0
2780 PRINT
2790 PRINTCL$;"Energy    ";INT(P):PRINT
CL$;"Torpedoes";T1:PRINTCL$;"Cygons
";K1:PRINTCL$;"Days left";D2-D1:RETURN

2800 C$="":INPUTV$:Y=2:DD=0
2810 IFV$=CHR$(13)THEN DD=-1:RETURN
2820 N=VAL(V$)
2830 RETURN
2840 PRINT"Went NOVA":FORDE=1TO200:NEX
TDE:IFCB<>1THENRETURN
2845 SP=50:GOSUB2880:RETURN
2850 CLS:PRINT"SUPERNOVA"
2860 FORDE=1TO200:NEXTDE:SCREEN2,2:CLS
:SP=200:GOSUB2880
2870 RETURN
2880 SCREEN2,2:CLS:CALL&H9808:FORDE=1T
OSP:DF=FNA(&H300)+&H3800:VPOKEDF,FNA(&
HFE):NEXTDE:SCREEN1,1
2890 GOSUB4000:RETURN
2900 REM EXPLOSION
2910 OUT127,228:FORSO=240TO255:OUT127,
SO:FORSP=1TO15:NEXTSP:NEXTSO:RETURN
2920 REM TORPEDOES
2930 OUT127,231:OUT127,240
2940 FORSO=0TO15:FORSP=192TO207
```

```
2950 OUT127,SP:OUT127,SO:NEXTSP
2960 OUT127,240+SO:NEXTSO
2970 RETURN
2980 REM PULSARS
2990 OUT127,228
3000 FORSP=240 TO255
3010 OUT127,SP:FORSQ=1TO3:NEXT
3020 NEXT
3030 SOUND0:RETURN
3040 SCREEN1,1:CLS:INPUT"Enter your na
me. ";UA$
3050 RETURN
3060 SCREEN2,2:CLS:COLOR5,11,(0,0)-(25
5,191),11:COLOR 1
3070 PRINTCHR$(17):PRINT" Quality Prog
rams"
3110 COLOR6,11:CURSOR80,80
3120 PRINT"Presents":RETURN
3130 COLOR 5,11:MAG2
3140 CURSOR40,142:PRINTCHR$(17);"SPACE
TREK"
3150 RETURN
3160 PRINTCHR$(16):COLOR13,11:CURSOR 2
0,130:PRINT"Starring Captain";:COLOR12
,11:PRINT CHR$(17);" ";UA$:RETURN
3170 SCREEN1,1:CLS:PRINT"Welcome to SP
ACETREK.":PRINT"————————————————————"

3180 PRINT:PRINT" The Galaxy is divive
d into 64 ":PRINT"Quadrants. Each Quad
rant is divided "
3190 PRINT"into 64 sectors. Co-ordinat
es 1-5 ":PRINT"means 1 across, 5 down.
 The "
3200 PRINT"galaxy has wrap around feat
ures for":PRINT"ease of travel."
3210 GOSUB3620
```

```
3220 CLS:PRINT"Course directives.":PRI
NT"————————————":PRINT
3230 PRINT"The PEGASUS can travel in a
ny of the":PRINT"eight directions as f
ollows,":PRINT
3240 PRINT"           0"
3250 PRINT"        315  45 "
3260 PRINT"     270       90"
3270 PRINT"        225 135"
3280 PRINT"          180"
3290 GOSUB3620
3300 CLS:PRINT"Ion Drive.":PRINT"———
———":PRINT
3310 PRINT" The PEGASUS is equipped wi
th the":PRINT"lastest Ion drive propul
sion system."
3320 PRINT" 1 sector   = .2 "
3330 PRINT" 4 sectors  = .5 "
3340 PRINT" 1 Quadrant = 1"
3350 PRINT"Use of the Ion drive requir
es a ":PRINT"single stardate."
3360 GOSUB3620
3370 CLS:PRINT"Short Range Sensors.":P
RINT"————————————————":PRINT
3380 PRINT" The short range sensors sc
an the":PRINT"present quadrant. The PE
GASUS looks"
3390 PRINT"like▲, the GYGONS◆, Bas
estars ✦,":PRINT"and Stars * .":GOSUB
3620
3400 CLS:PRINT"Long Range Sensors.":PR
INT"————————————":PRINT
3410 PRINT" The long range sensor scan
s the 9":PRINT"closest Quadrants."
3420 PRINT"The 1st digit = number of G
YGONS"
3430 PRINT"The 2nd digit = number of B
asestars"
```

```
3440 PRINT"The 3rd digit = number of S
tars."
3450 GOSUB3620
3460 CLS:PRINT"Galaxy Map.":PRINT"———
———":PRINT
3470 PRINT" Every time the LR sensors
are used":PRINT"the galaxy map is upda
ted.":GOSUB 3620
3480 CLS:PRINT"Mega Pulsars.":PRINT"—
————————":PRINT
3490 PRINT" The pulsars are very accur
ate due to":PRINT"modern guidance syst
ems. Any amount"
3500 PRINT"of available energy may be
fired. A":PRINT"CYGON ship has up to 3
000 units of"
3510 PRINT"energy.":GOSUB3620
3520 CLS:PRINT"Photon Torpedoes.":PRIN
T"———————————":PRINT
3530 PRINT" Torpedoes are limited to a
 single":PRINT"Quadrant. The couse is
given as per"
3540 PRINT"the Navigation directive. I
f a":PRINT"torpedo hits a star the sta
r can go"
3550 PRINT"SuperNova, thus destroying
the ship.":PRINT"Should the star go NO
VA, your chances"
3560 PRINT"are 90%.":GOSUB3620
3570 CLS:PRINT"Damage Control.":PRINT"
————————————":PRINT
3580 PRINT" This lists the state of re
pair of":PRINT"all devices. All repair
s are carried"
3590 PRINT"out during the game, but do
cking":PRINT"with a BASESTAR will effe
ct":PRINT"immediate repairs."
3600 PRINT"Docking is achieved by posi
```

```
tioning":PRINT"the PEGASUS alongside a
 Basestar.":GOSUB3620
3610 GOTO530
3620 PRINT:PRINT"Press any key to cont
inue."
3625 FORRT=1TO400:NEXTRT
3630 IFINKEY$=""THEN3630
3640 RETURN
4000 DH=&H11:DF=&H3800:DG=&H3B00:FORDE
=DFTODG:VPOKEDE,DH:NEXT:RETURN
5000 FORX=&H9808TO&H9819
5010 READA:POKEX,A:NEXT:RETURN
5020 DATA &HF3,&H3E,0,&HD3,&HBF,&H3E,&
H80,&HD3,&HBF,&H3E,&HC8,&HD3,&HBF,&H3E
```

PATTERN EDITOR LVIIIA/B

```
10 DIM PT(16,19),BD(8),BT(8)
20 PATTERNC#208,"001008FC08100000"
30 PATTERNC#211,"7884B4A4A4B48478"
40 PATTERNC#209,"20202020A8702000"
50 ZY$="----------------"
60 TP$="000000000000"
70 ZZ$=CHR$(142)+","+CHR$(143)+","+
   CHR$(208)+","+CHR$(209)
80 SCREEN 1,1:CLS:FOR X=1TO16:FORY=1TO
   19:PT(X,Y)=32:NEXT:NEXT
90 PRINT"Pattern Editor.":CURSOR25,0:
   PRINT"B.Brown ";:PRINTCHR$(211);:
   PRINT" 84":PRINT ZY$
100 PRINT:GOSUB 740
110 X=2:Y=5
120 CURSOR 26,2:PRINT"Expanded"
130 CURSOR 26,16:PRINT"Normal"
140 CURSOR 1,22:PRINT"(S)et,(Z)ero";:
```

```
    PRINT",(E)rase,";ZZ$;
150 CURSOR 1,23:PRINT"(P)rint shape";:
    PRINT",(V)alues in Hex";
160 CURSOR X,Y:PRINTCHR$(144);:FOR DE=
    1 TO 15:NEXT DE
170 A$=INKEY$:CURSOR X,Y:PRINT CHR$(PT
    (X,Y));:IFINKEY$="" THEN GOTO 160
180 IF A$=CHR$(28) THEN X=X+2: IF X>16
    THEN X=2:Y=Y+2:IF Y>19 THEN Y=5
190 IF A$=CHR$(29) THEN X=X-2: IF X<2
    THEN X=16:Y=Y-2:IF Y<5 THEN Y=19
200 IF A$=CHR$(30) THEN Y=Y-2: IF Y<5
    THEN Y=19
210 IF A$=CHR$(31) THEN Y=Y+2: IF Y>19
    THEN Y=5
220 IF A$="Z" THEN PT(X,Y)=32
230 IF A$="S" THEN PT(X,Y)=229
240 IF A$="V" THEN GOSUB 280
250 IF A$="P" THEN GOSUB 350
260 IF A$="E" THEN GOTO 80
270 GOTO 160
280 REM Print Hex values of each row
290 AY=4:AX=24
300 BX=2:BB=1:FOR BY=5 TO 19 STEP2:BA=
    0:GOSUB 640:BD(BB)=BA:BB=BB+1:NEXT

310 BB=1:FOR BE=5TO19 STEP2:CURSOR18,
    BE:BA$=HEX$(BD(BB)):IF LEN(BA$)< 2
    THEN BA$="0"+BA$
320 BB=BB+1:PRINT BA$;:NEXT
330 RETURN
340 REM Print Pattern on screen
350 AY=4:AX=26:FOR AE=5TO 19 STEP2:FOR
    AA=2 TO 16 STEP 2
360 IF PT(AA,AE)>32 THEN AD$=CHR$(229)
    :GOTO 380
370 AD$=CHR$(32)
380 CURSOR AX,AY:PRINT AD$;
```

```
390 AX=AX+1:IFAX>33THENAX=26:AY=AY+1
400 IF AY>11 THEN AY=4
410 NEXT:NEXT
420 GOSUB 280:AA$=""
430 FOR BA=1TO8:BB$=HEX$(BD(BA)):IF
    LEN(BB$)<2 THEN BB$="0"+BB$
440 AA$=AA$+BB$:NEXT:PATTERNC#&HD2,AA$

450 BZ=0:BS=1:FOR MA=2TOLEN(AA$)STEP2
460 BR$=MID$(AA$,MA,1)
470 IF BR$="A" THEN BR$="10"
480 IF BR$="B" THEN BR$="11"
490 IF BR$="C" THEN BR$="12"
500 IF BR$="D" THEN BR$="13"
510 IF BR$="E" THEN BR$="14"
520 IF BR$="F" THEN BR$="15"
530 BZ=VAL(BR$):BT(BS)=BZ AND 3
540 BS=BS+1:NEXT MA
550 AZ$="":FOR BA=1 TO 8:BZ=BT(BA)
560 IF BZ=1 THEN BZ=4
570 IF BZ=2 THEN BZ=8
580 IF BZ=3 THEN BZ=12
590 BR$=HEX$(BZ):IF LEN(BR$)<2 THEN
    BR$=BR$+"0":AZ$=AZ$+BR$:NEXT
600 PATTERNC#&HD4,AZ$
610 CURSOR 28,18:PRINT CHR$(&HD2);:
    PRINTCHR$(&HD4)
620 RETURN
630 REM Determine Value per row
640 IF PT(BX,BY)>32 THEN BA=BA+128
650 IF PT(BX+2,BY)>32 THEN BA=BA+64
660 IF PT(BX+4,BY)>32 THEN BA=BA+32
670 IF PT(BX+6,BY)>32 THEN BA=BA+16
680 IF PT(BX+8,BY)>32 THEN BA=BA+8
690 IF PT(BX+10,BY)>32 THEN BA=BA+4
700 IF PT(BX+12,BY)>32 THEN BA=BA+2
710 IF PT(BX+14,BY)>32 THEN BA=BA+1
720 RETURN
```

```
730 REM clear array
740 PRINT"  1 2 3 4 5 6 7 8 "
750 PRINT"  --------------- "
760 PRINT"1¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦"
770 PRINT" ¦---------------¦"
780 PRINT"2¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦"
790 PRINT" ¦---------------¦"
800 PRINT"3¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦"
810 PRINT" ¦---------------¦"
820 PRINT"4¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦"
830 PRINT" ¦---------------¦"
840 PRINT"5¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦"
850 PRINT" ¦---------------¦"
860 PRINT"6¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦"
870 PRINT" ¦---------------¦"
880 PRINT"7¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦"
890 PRINT" ¦---------------¦"
900 PRINT"8¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦"
910 PRINT"  --------------- "
920 RETURN
```