PEEKS AND POKES

FOR THE

DEANE WHITMORE

With compliments
D. G. Whitmore.

PEEKS AND POKES
FOR THE
SPECTRAVIDEO

Deane Whitmore

A TME PUBLICATION

Printed by

TARANAKI NEWSPAPERS LIMITED,

Currie Street, New Plymouth, New Zealand

To Denise,
my first love . . . . . really.

CONTENTS

ACKNOWLEDGEMENTS

In working on this book, I have come to realise that there will never be a computer that does exactly what I want it to do.

I wish to thank Spectravideo for producing a computer that comes close. It's taken a lot of research to appreciate this.

Special thanks to Barry and Martyn of Taranaki Micro Electronics for their support and their criticism. Without their encouragement this book may never have been written and certainly not published. They have MONITORed the production carefully.

I would like to thank Laurie Callender who provided support, tested the programs in the book for me and allowed me to include a section from his educational program ENGINE, which demonstrates the internal combustion engine. This program won Laurie a Disk Drive and Super Expander in a New Zealand wide software writing competition.

PREFACE

Spectravideo Computers, namely the SV-318 and the SV-328 are two of the most advanced computers available at an affordable price for everybody to own. They feature one of the most comprehensive Microsoft BASICs ever written and support some fairly revolutionary ideas for BASIC programming.

The range of peripherals available turn the Spectravideo into more than just a home computer, they open the door to small business and serious use as an office computer.

This book, although brief, has been written for the beginner to the semi-advanced user. It goes on where others have left off to indulge in some tricks and tips not extensively known, but, in fact should improve your programming skills and knowledge of the computer immensely.

Everything in this book has been checked thoroughly. However, knowing the research that has gone into it and the enthusiasm with which it was written, well. . . . . .

If there are any problems with the programs, then hopefully the text will help you solve them.

Only those aspects concerning common interest have been covered. The Spectravideo Computers are a fantastic graphic tool and should be used as such.

SPECTRAVIDEO BASIC

1. THE BASIC LANGUAGE FOR THE SPECTRAVIDEO.

2. RESERVED WORDS AND WHERE THEY ARE USED.

3. ERROR MESSAGES AND WHAT THEY MEAN.

4. ROM ENTRY ADDRESSES FOR THE ON BOARD BASIC.

5. INPUT/OUTPUT TABLE TO THE PERIPHERALS.

THE BASIC LANGUAGE

The Spectravideo Computers are provided with a very powerful BASIC language built right into the computer and available whenever the machine is switched on. The language was written by **MICROSOFT Corp.** and is called **SV Extended BASIC**, this is displayed when you switch the computer on.

BASIC is an abbreviated word which means **"BEGINNERS ALL-PURPOSE SYMBOLIC INSTRUCTION CODE"**. It is a high level language in that the language is problem orientated, it uses instructions directed towards the problem and not towards the computer.

The BASIC language for the Spectravideo is held in a 32k ROM (Read Only Memory) and contains well over 200 built in instructions that can be called simply by typing their names in an organised fashion.

The SV Extended BASIC has the following built in features that make it stand out as one of the most advanced low cost home computer languages available today.

- **A sophisticated and thorough screen editor for manipulating text and program instructions.**
- **Complete sound and graphic control using BASIC with some of the most advanced computer chips available today.**
- **The machine defaults to double precision calculation of 14 decimal places, making it one of the most accurate computing tools available.**
- **There are 10 user definable function keys that aid user friendliness in running programs and for extra editing power.**
- **Bank selectibility, which allows you to load two programs at the same time and gives you greater control of the computer and your programs.**
- **The computer provides a ROM cartridge port which uses premanufactured programs including games, word processors and other business software, leaving the RAM free for data storage etc.**

RESERVED WORDS

The SV Extended BASIC provides us with a large number of commands that are available to us for use in our programs.

Listed below are the reserved words available and alongside each are the following abbreviations which tell us which peripheral device they are related to. Some of the commands can only be used if that particular device is connected to the computer.

```
"B"  -  BASIC commands.
"G"  -  Graphic Screen commands.
"GT" -  Graphics Tablet commands.
"P"  -  Line Printer commands.
"D"  -  Disk Drive commands.
"T"  -  Tape Drive commands.
"M"  -  Modem commands.
"S"  -  Programmable Sound Generator commands.
"Ba" -  Bank Switching commands.
```

LIST OF RESERVED AND SPECIAL WORDS

ABS (B)	AND (B)	APPEND (D)	ASC (B)
ATN (B)	ATTR$ (D)	AUTO (B)	BEEP (B)
BIN$ (B)	BLOAD (T)(D)	BSAVE (T)(D)	CDBL
CHR$ (B)	CINT	CIRCLE (G)	CLEAR (B)
CLICK (B)(S)	CLOAD (T)	CLOK	CLOSE (T)(D)
CLS (B)	CMD	COLOR (B)(G)	COMMON
CONT (B)	COPY (D)	COS (B)	CSAVE (T)
CSNG (B)	CSRLIN	CVD (D)	CVI (D)
CVS (D)	DATA (B)	DEFDBL (B)	DEFFN (B)
DEFINT (B)	DEFSNG (B)	DEFSTR (B)	DEFUSR (B)
DELETE (B)	DIAL (M)	DIM (B)	DRAW (G)
DSKF (D)	DSKI$ (D)	DSKO$ (D)	ELSE (B)
END (B)	EOF (D)	EQV (B)	ERASE (B)
ERL (B)	ERR (B)	ERROR (B)	EXP (B)
FIELD (D)	FILES (D)	FIX (B)	FOR (B)
FPOS (D)	FRE (B)	FN (B)	GET (D)(T)(G)
GOSUB (B)	GOTO (B)	GO TO (B)	HEX$ (B)
IF (B)	IMP (B)	INT (B)	INKEY$ (B)
INP (B)	INPUT (B)(T)(D)	INSTR (B)	INTERVAL (B)
IPL (D)	KEY (B)	KILL (D)	LEFT$ (B)
LEN (B)	LET (B)	LFILES (D)(P)	LINE (G)
LIST (B)	LLIST (B)(P)	LOAD (T)(D)	LOC (D)
LOCATE (B)(G)	LOF (D)	LOG (B)	LPOS (P)
LPRINT (B)(P)	LSET (D)	MAX (D)	MKD$ (D)
MDM (M)	MERGE (B)(T)(D)	MID$ (B)	MKI$ (D)
MKS$ (D)	MOD (B)	MON	MOTOR (B)(T)
NAME (D)	NEW (D)	NEXT (B)	NOT (B)
OCT$ (B)	OFF (D)	ON (B)	OPEN (B)(T)(D)
OR (B)	OUT (B)	OUTPUT (B)(T)(D)	PAD (B)(GT)
PAINT (G)	PDL	PEEK (B)	PLAY (S)
POINT (G)	POKE (B)	POS	PRESET (G)
PRINT (B)(G)	PSET (G)	PUT (G)(T)(D)	READ (T)(D)
REM (B)	RENUM (B)	RESTORE (B)	RESUME (B)
RETURN (B)	RIGHT$ (B)	RND (B)	RSET (D)
RUN (B)	SAVE (T)(D)	SCREEN (G)	SET (D)
SGN (B)	SIN (B)	SOUND (S)	SPACE$ (B)
SPC (B)	SPRITE (G)	SQR (B)	STEP (B)
STICK (G)	STOP (B)	STRIG (B)	STR$ (B)
STRING$ (B)	SWAP (B)	SWITCH (Ba)	TAB (B)
TAN (B)	THEN (B)	TIME (B)	TO (B)
TROFF (B)	TRON (B)	USR (B)	USING (B)
VAL (B)	VARPTR (B)	VPEEK (B)	VPOKE (B)
WAIT (B)	WIDTH (B)	XOR (B)	

There are still a large number of commands that can be formed by combining some of the above. In fact, there are individual word commands which will give an error message if they are not used in combination with others.

Examples of these combined instructions are as follows:-

CLICK ON, CLICK OFF, IF - THEN, IF - THEN - ELSE, ON INTERVAL, INTERVAL ON, INTERVAL OFF, INTERVAL STOP, KEY ON, ON KEY - GOSUB, KEY OFF, LINE INPUT, PRINT USING, MAXFILES, MOTOR ON, MOTOR OFF, ON ERROR, ON - GOTO, ON - GOSUB, ON SPRITE, ON STOP, ON STRIG, PRINT#, PRINT# USING, SOUND ON, SOUND OFF, PUT SPRITE, SPRITE$, STOP ON, STRIG ON, STRIG OFF

The next small program displays some of the reserved words that are available from the BASIC ROM in the computer and may assist you in seeing what is available.

```
 5 SCREEN,0:CLS
 10 X = &H2C9:SL$="A"
 20 NM$ = SL$
 30 P = PEEK(X)
 40 IF P > 128 THEN 90
 50 IF P = 0 THEN 130
 60 NM$ = NM$ + CHR$(P)
 70 X = X + 1
 80 GOTO 30
 90 P = P - 128
100 NM$ = NM$ + CHR$(P)
110 PRINT NM$;" ";
120 X = X + 2: GOTO 20
130 SL$ = CHR$(ASC(SL$)+1):X = X + 1
131 IF SL$ > "Z" THEN END ELSE 20
```

ERROR MESSAGES

Error messages are printed on the screen whenever the computer does not either recognize an instruction you have given it or you have tried to use the instruction in its wrong context. These two examples are perhaps the most common mistakes the programmer makes and the error message that is displayed, which seems to be synonymous with all computers, is **SYNTAX ERROR.**

To the unwary, error messages seem to be rather frightening and often misleading. For example "Illegal function call", a Spectravideo error message, usually means that you are either using an instruction which acts on a peripheral device which is not plugged into the computer or you are using an instruction which is "illegal" in the present graphic or screen mode. It does not necessarily mean you have the wrong word.

The error messages associated with the BASIC of the Spectravideo are well covered in the owner's manual, but the point to be taken here is that not only are the error messages quite self descriptive, but they are also quite numerous and the understanding of them will only aid in the debugging of your programs.

For the interest of the error fanatic, below are the ROM entry addresses which display all the error messages to the screen.

```
10 FOR X = 1495 TO 2123
20 PRINT CHR$(PEEK(X));
30 NEXT X
```

Note that some of the error messages are related to peripheral devices that may not be connected to your computer. Some may be rather difficult to activate purposely, but they will occur should the corresponding error be encountered.

ROM ENTRY ADDRESSES

There are several reasons why the entry addresses of the built in functions are handy to know, and perhaps the prime reason is that once the name of the routine and its entry address are known they can be called from a machine code program and used as a premanufactured working sub-routine, thus eliminating errors in your programs and shortening the number of bytes the program uses.

Listed below are the ROM entry addresses for most of the BASIC commands available to us. The table gives the command and the hexa-decimal entry address for that command. Because the ROM is written in machine code, to use the entry address from BASIC the following instruction method must be used. First define your machine code function call; ie. **DEFUSR 0 = &H (entry address number)**; then call up your machine code routine as in integer variable; ie. **X = USR (0)**.

LIST OF ROM ENTRY ADDRESSES

AUTO	11F5	ABS	55B1	ATN	5139	ASC	6B10
ATTR$	1839	BSAVE	7624	BLOAD	7684	BEEP	40BE
BIN$	6904	CLICK	31AF	CLOSE	7375	COPY	34BF
CONT	671B	CLEAR	67A6	CLOAD	1EAA	CSAVE	1E15
CSRLIN	1A37	CINT	56B5	CSNG	56DD	CDBL	5765
CVI	732B	CVS	732E	CVD	7331	COS	50B8
CHR$	6B20	CIRCLE	2652	COLOR	4552	CLS	3777
CMD	34C4	DELETE	1C6C	DATA	109B	DIM	6061
DEFSTR	0F5C	DEFINT	0F5F	DEFSNG	0F62	DEFDBL	0F65
DSKI$	6B04	DSKO$	34A6	DEF	188A	DSKF	34C9
DRAW	29DA	DIAL	79C2	ELSE	109D	END	66CF
ERASE	676E	ERROR	11EA	ERL	50D1	ERR	5197
EXP	526B	EOF	74B0	FOR	0D65	FIELD	72CD
FILES	73B2	FN	55B1	FRE	6CF7	FIX	57E9
FPOS	74C6	GOTO	1028	GO TO	1028	GOSUB	0FF6
GET	2FB4	HEX$	68FF	INPUT	13D2	IF	1225
INT	57F8	INP	1A37	IPL	34BA	INSTR	5139
KILL	34B5	KEY	3120	LPRINT	125D	LLIST	1AB3
LPOS	1834	LET	10C0	LOCATE	2FD1	LINE	1374
LOAD	7121	LSET	7228	LIST	1AB8	LFILES	73AD
LOG	5197	LOC	7484	LEN	6B04	LEFT$	6B66
LOF	749A	MOTOR	2BE5	MERGE	7122	MKI$	7312
MKS$	7315	MKD$	7318	MID$	6B9F	MON	7B44
MAX	7CBA	MDM	3036	NEXT	6821	NAME	34B0
NEW	6556	NOT	5300	OPEN	7080	OUT	1A4C
ON	1124	OCT$	68FA	PRINT	1265	PUT	2FB1
POKE	1CAD	POS	1839	PEEK	1CA6	PSET	232D
PRESET	2328	PAINT	24FC	PDL	3280	PAD	32BD
PLAY	2C24	RETURN	1061	READ	1405	RUN	0FE2
RESTORE	66AE	REM	109D	RESUME	119D	RSET	7227
RIGHT$	6B96	RND	5300	RENUM	1CF0	SCREEN	459A
SWITCH	337F	STOP	66C8	SWAP	6735	SET	34AB
SAVE	7167	SGN	55C6	SQR	5222	SIN	50D1
STR$	6909	SPACE$	6B4D	SOUND	2BFD	STICK	3206
STRIG	3263	TRON	672F	TROFF	6730	TAN	5120
VAL	6BC0	VARPTR	66F7	VPOKE	46D8	VPEEK	46F2
WIDTH	1A6C	WAIT	1A52				

6CF7

Examples of ROM entry addresses used in BASIC, should you need to write your program this way, are given below. Not all the addresses will do something from BASIC, as most need additional commands for the computer to understand them. Single word commands, however, work fine with this method.

```
To clear the screen :- DEFUSR Ø = &H3777 : X = USR (Ø)
To activate BEEP    :- DEFUSR Ø = &H4ØBE : X = USR (Ø)
Load a program from cassette :- DEFUSR Ø = &H1EAA :  X = USR (Ø)
```

INPUT/OUTPUT TABLE

Below is a table of the INPUT/OUTPUT ports to the various peripheral devices that the Spectravideo can control. To use this information take the number listed in hexadecimal under I/O PORT and use the following format for your instructions:- **OUT &H80,0**

The number following the Port number is your selected byte that you may wish to read or write to that port.

INPUT/OUTPUT PORT LOCATIONS

I/O PORT (hex)	READ/WRITE	DESCRIPTION	REMARKS
1Ø	WRITE	Write Data Port	Printer
11	WRITE	Data Strobe	Printer
12	READ	Status (Bit Ø="Ø" for ready)	Printer
2Ø	READ	Receiver Buffer Register	Modem
	WRITE	Divisor Latch (Least Signific.)	Modem
	WRITE	Transmitter Holding Buffer Reg.	Modem
21	WRITE	Divisor Latch (Most Signific.)	Modem
	WRITE	Interrupt Enable Register	Modem
22	WRITE	Interrupt Idendification Reg.	Modem
23	WRITE	Line Control Register	Modem
24	WRITE	Read Modem Control Register	Modem
25	READ	Line Status Register	Modem
26	READ	Read Modem Status Register	Modem
28	READ	Receiver Buffer Reg.(Least Sig.)	RS-232
	WRITE	Divisor Buffer Register	RS-232
	WRITE	Transmitter Holding Buffer	RS-232
29	WRITE	Divisor Latch (Most Significant)	RS-232
	WRITE	Interrupt Enable Register	RS-232
2A	WRITE	Interrupt Identification Reg.	RS-232
2B	WRITE	Line Control Register	RS-232
2C	WRITE	Modem Control Register	RS-232
2D	READ	Line Status Register	RS-232
2E	READ	Modem Status Register	RS-232
3Ø	READ	FD-1793 Status Register	Disk
	WRITE	FD-1793 Command Register	Disk
31	READ/WRITE	FD-1793 Track Register	Disk
32	READ/WRITE	FD-1793 Sector Register	Disk

33	READ/WRITE	FD-1793 Data Register	Disk
34	READ	Read Intrq & Drq O/P Pins	Disk
	WRITE	Disk Select Register (Bit Ø="Ø" to select Disk 1 Bit 1="Ø" to select Disk 2)	Disk
38	WRITE	Density Select Register (Bit Ø="Ø" for Double Den. Bit Ø="1" for Single Den.)	Disk
5Ø	WRITE	Address Register Select	8Ø-Col.
51	WRITE	CRT Controller Register (RO-R17)	8Ø-Col.
58	WRITE	CRT Bank Control (=&HØFF Bank On, =&HØØ Bank Off)	8Ø-Col.
8Ø	WRITE	TMS-9918A Write mode = Ø	VDP
81	WRITE	TMS-9918A Write Mode = 1	VDP
84	READ	TMS-9918A Read Mode = Ø	VDP
85	READ	TMS-9918A Read Mode = 1	VDP
88	WRITE	AY-3-8910 Latch Address	PSG
8C	WRITE	AY-3-8910 Write	PSG
9Ø	READ	AY-3-8910 Read	PSG
96	WRITE	Write 8255 Port C (KYBD,CAS,PSG)	PPI
97	WRITE	WRITE 8255 Control Word Reg.	PPI
98	READ	Read 8255 Port A (PDL,JYSTK,CAS)	PPI
99	READ	Read 8255 Port B (KEYBOARD)	PPI

THE TEXT SCREEN

1. THE CHARACTER SET AND HOW TO USE IT.
2. CURSOR MANIPULATIONS.
3. CHANGING THE CHARACTER FONT.
4. VPOKING TEXT AND GRAPHIC CHARACTERS.
5. THE FUNCTION KEYS AND THEIR USE.
6. MEMORY LOCATIONS AND WHAT THEY CAN DO.
7. EXAMPLE PROGRAM.

248

156,313,400
1
490 576

THE CHARACTER SET

The Spectravideo Computers have the ability to display upper and lower case letters along with all the standard text symbols associated with the English language. In addition to these characters, there are over 50 graphic symbols directly available to the user. On the SV-318 computers these graphic symbols are visible on the keyboard but are not given on the SV-328 computer keyboard although they still exist. They are accessible through the use of the **LEFT** and **RIGHT GRPH** keys provided to the left of the space bar.

The Spectravideo Computers use the standard ASCII code character set and control codes. The standard characters for text are as follow.

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
          1234567890
!@#$%^&*()_+~}{"|?><[]\:',./
```

The character set can be displayed to the screen with the following program.

```
10 CLS
20 FOR X = 32 TO 126
30 PRINT CHR$(X);
40 NEXT X
```

The decimal numbers 32 to 126 correspond to the standard ASCII code numbers, which we convert to characters in the program.

There is one other instruction which will display the character set and it is:- **POKE 63397,83**. This will also include some other characters which are particular to Spectravideo only.

The standard characters are used by the computer to interpret the commands which you give it to form your programs. Words like LIST, GOSUB, GET etc are recognized as instructions from the on board BASIC interpreter.

The Text Screen is divided into 40 characters across by 24 characters down, although upon first turning the computer on the screen actually displays 39 x 24 characters with the 24th line reserved for the function key windows.

The characters can be used in almost any format that you may wish, simply by issuing commands to tell the computer where and how to display your text anywhere in these locations. Keywords like LOCATE, TAB, and PRINT USING are some of the BASIC instructions that can be used to achieve your desired results.

Examples of the use of Text Screen commands are given below, try them if you are unsure how to use them and practise some of your own. By the time we reach the end of this chapter, hopefully your attitude with regard to the Text Screen will have changed.

```
10 CLS
20 LOCATE 20,10: PRINT"SPECTRAVIDEO"
30 LOCATE 10,20: PRINT"COMPUTERS"
```

```
10 CLS
20 PRINT TAB(10)"SPECTRAVIDEO"
30 PRINT TAB(15)"COMPUTERS"


10 CLS
20 X=123.4567
30 B$="######.########"
40 PRINT USING B$;X
```

Try this last example with different values for X and see how the PRINT USING statement formats your output.

As you can see there are many combinations available and that any text format is possible.

Another utility that is available on the Spectravideo Computers, is the ability to use inverse characters wherever we want to on the Text Screen. Unfortunately we can only have inverse characters and not inverse keyboard graphic symbols, or at least not by using this method.

The usual method for switching to inverse characters is as follows:-

```
PRINT CHR$(27) + "p"
```

This method seems quite laborious, but once memorized you can recall it when it is needed. The instruction that returns you to normal text is:-

```
PRINT CHR$(27) +"q"
```

There is, however, one other method to switch to inverse characters which, if you have a head for numbers, may be easier to remember.

```
POKE 65077,1  turns inverse on

POKE 65077,0  turns inverse off
```

The ability to switch from one mode to another is quite simple in a running program as is demonstrated below.

```
10 CLS
20 POKE 65077,1:PRINT"SPECTRAVIDEO"
30 POKE 65077,0:PRINT"COMPUTERS"

or

10 CLS
20 LOCATE 5,10
30 POKE 65077,1:PRINT"******** "
40 LOCATE 15,10:POKE 65077,0
50 PRINT"SPECTRAVIDEO"
60 LOCATE 28,10
70 POKE 65077,1:PRINT" ********"
80 POKE 65077,0:END
```

Flashing displays can be created by changing the display of certain words from normal to inverse as you wish.

```
10 CLS
20 LOCATE 10,10
30 POKE 65077,1
40 PRINT " SPECTRAVIDEO "
50 FOR T = 1 TO 100:NEXT T
60 LOCATE 10,10
70 POKE 65077,0
80 PRINT " SPECTRAVIDEO "
90 FOR T = 1 TO 100:NEXT T
100 GOTO 20
```

CURSOR MANIPULATIONS

The cursor which is used for editing your instructions on the screen can be removed from the screen with the instruction **LOCATE,,0** and re-instated whenever it is needed with **LOCATE,,1**.

The following POKE commands do the same thing. **POKE 64005,0** removes the cursor and **POKE 64005,1** displays it again.

The insert prompt cursor can be forced onto the screen, and used for INPUT statements in your running programs. The command to do this is:- **POKE 65079,1**. Try the next program and notice the size of the cursor.

```
10 CLS
20 POKE 65079,1
30 INPUT"WHAT IS YOUR NAME";A$
40 CLS:PRINT"HELLO ";A$
50 END
```

Have you ever wanted a flashing cursor under BASIC, especially when input from the user should be high-lighted in some way? The next routines may be just what you are looking for.

```
10 CLS
20 PRINT"PRESS ANY KEY ON THE KEYBOARD"
30 LOCATE 33,0:LOCATE,,1
40 A$ = INKEY$
50 IF A$ = "" THEN 80
60 CLS:PRINT A$
70 END
80 LOCATE 33,0:LOCATE,,0
90 GOTO 30
```

```
10 CLS
20 LOCATE,,0
30 PRINT"PRESS ANY KEY ON THE KEYBOARD"
40 LOCATE 33,0:POKE 65077,0:PRINT" "
50 A$ = INKEY$
60 IF A$ = "" THEN 90
70 CLS:LOCATE,,1:PRINT A$
80 END
90 LOCATE 33,0:POKE 65077,1:PRINT" "
100 GOTO 40
```

CHANGING THE CHARACTER FONT

The shapes for the characters are held in Video RAM from address 2048 (dec.) to 4096 (dec.). They are grouped together in a series of eight numbers or bytes which the computer knows to turn into the characters when we press the keys on the keyboard.

These locations can be viewed along with the corresponding byte number they contain. With the following program we can verify the locations with the shape of the letters produced using binary numbers, where the 0's represent spaces and the 1's represent filled in pixels on the television screen.

```
10 CLS:Y=2047:FOR X = 2048 TO 4096
20 S = VAL (BIN$(VPEEK(X)))
30 B$ = "####    #########"
40 PRINT USING B$;X,S
50 IF X = Y+8 THEN PRINT ELSE GOTO 70
60 Y = X
70 NEXT X
```

Now that we know where these locations are, we can VPOKE new byte numbers into them and change the shape of our character font or create new graphic symbols.

The next program will demonstrate the use of these locations to change the letter "A" into a new graphic symbol.

```
10 CLS
20 VPOKE 2312,&B00110000
30 VPOKE 2313,&B11111100
40 VPOKE 2314,&B00110000
50 VPOKE 2315,&B11111100
60 VPOKE 2316,&B10000100
70 VPOKE 2317,&B10000100
80 VPOKE 2318,&B01001000
90 VPOKE 2319,&B00110000
100 PRINT "A"
```

Now every time you press the uppercase "A", the new symbol will be displayed in its place.

Notice that binary numbers were used for this program, but either decimal or hexadecimal numbers could have been used. Binary numbers serve to give you a visual image of the new character or graphic symbol you are creating.

We could change the graphic character back to normal by VPOKE[ing] the corresponding byte numbers into the letter "A" memory locations, but there is a quicker way if your program allows it, and that is to issue a **SCREEN 1** command. This resets the normal character font and graphic symbols then returns to the Text Screen.

The next program is a fun program you might like to try. It takes the memory locations of the character "A" and randomly VPOKE's byte numbers into them, creating an ever changing pattern on the text screen. The program uses a routine to check when CONTROL/STOP is pressed to exit the program and return the character "A" back to normal.

```
10 CLS
20 FOR X = 0 TO 895
30 PRINT "A";:NEXT X
40 FOR T = 2312 TO 2319
50 STOP ON:ON STOP GOSUB 100
60 BYTE = INT (RND(-TIME)*255)
70 VPOKE T,BYTE
80 NEXT T
90 GOTO 40
100 SCREEN 1
110 END
```

VPOKING CHARACTERS

The command VPOKE can be used to precisely display characters or graphic symbols anywhere on the screen once the layout or locations of the screen are known. Inverse characters can be VPOKED to the screen without having to change the print mode.

This form of control over the Text Screen must be one of the most advanced available on a home computer.

The following program will vpoke the complete character and graphic symbol set to the screen.

```
10 CLS
20 FOR X = 0 TO 255
30 Y = Y + 1
40 VPOKE Y,X
50 NEXT X
```

There are several applications for this method of printing and below are example programs that may improve your own use of the Text Screen. There is even a sophisticated adventure game written for the Spectravideo that relies on the ability to VPOKE characters and graphic symbols to the screen and surprisingly it runs very fast setting up graphical maps of your locations and dangers that await you.

The next routine will display the character that each number of possible bytes from 0 to 255 represents. It VPOKES the characters to location 500 on the Text screen as can be seen in line 40.

```
10 SCREEN,0:LOCATE,,0
20 FOR X = 0 TO 255
30 CLS:LOCATE 11,12
40 PRINT X;"-":VPOKE 500,X
50 FOR Y = 1 TO 300:NEXT Y
60 NEXT X
70 CLS:SCREEN,1:LOCATE,,1
```

The next program chooses random locations from 0 to 959 and VPOKES random characters to them to give interesting patterns on the screen.

```
10 CLS
20 SCREEN,0:LOCATE,,0
30 STOP ON: ON STOP GOSUB 80
40 X = INT(RND(-TIME)*959)
50 Y = INT(RND(-TIME)*255)
60 VPOKE X,Y
70 GOTO 20
80 CLS:SCREEN,1:LOCATE,,1
90 END
```

THE FUNCTION KEYS

At the bottom of the Text Screen are the function key windows. Wouldn't it be nice to remove these from time to time when running your programs?

The instruction that achieves just that is:- **SCREEN ,0** the instruction that re-instates the windows is:- **SCREEN ,1**. There is one other method that removes the function keys, try the following:- **DEFUSR = &H3B86 : X = USR (0)**, to return the function keys, use:- **DEFUSR = &H3B9F : X = USR (0)**.

The function keys and their contents are held in normal RAM from address 64030 (dec.) to 64178 (dec.). They can be viewed with the next small program which will slowly display the words that appear in the windows, by converting byte numbers held in these locations to characters.

```
10 CLS
20 FOR X = 64030 TO 64178
30 PRINT CHR$(PEEK(X));
40 FOR T = 1 TO 200: NEXT T
50 NEXT X
```

Spectravideo, however, has included an instruction, KEYLIST which will display all the words or commands assigned to the ten function keys.

To change any of the function keys to your own words or commands, the following format must be used. First call up the key you wish to assign you instruction to eg. KEY 3. the 3 could be any number from 1 to 10. Then enclose in quotation marks the command you wish assigned to that key. ie. **KEY 3, "COMPUTE"**.

Confirm that the command has been assigned by pressing the function 3 (F3) key, you should see COMPUTE displayed on the screen and the function key 3 window should also show the new command.

Control codes can be added to any command or word you wish to assign to the function keys. These control codes are well covered in the manual supplied with your computer and are repeated here for your reference. The method for using these control codes in a program follows. To clear the screen for example use:- **PRINT CHR$(12)**. Of course these control codes can be used directly from the keyboard as there is a CONTROL key included in the set. Hold down the CTRL key and press the letter that will activate the function you want. eg. **CTRL L** will clear the screen.

TABLE OF CONTROL CODES

NOTE:- "^" means CONTROL or CTRL

```
CHR$(2)    ^ B  -  Move the cursor back to the start of the next
                   word.
CHR$(3)    ^ C  -  Break and stop the program.
CHR$(5)    ^ E  -  Clear text from current cursor position to end
                   of line.
CHR$(6)    ^ F  -  Move the cursor to the start of the next word.
CHR$(7)    ^ G  -  Beep.
CHR$(8)    ^ H  -  Backspace and delete character to left of
                   cursor.
CHR$(9)    ^ I  -  Tabulate to the right 8 spaces.
CHR$(11)   ^ K  -  Home the cursor.
CHR$(12)   ^ L  -  Clear the screen and home the cursor.
CHR$(13)   ^ M  -  The same as ENTER, a carriage return.
CHR$(14)   ^ N  -  Move the cursor to the end of current line.
CHR$(18)   ^ R  -  Toggles the insert mode for inserting text.
CHR$(21)   ^ U  -  Clears the line the cursor is on.
CHR$(28)   ^ \  -  Moves the cursor to the right.
CHR$(29)   ^ ]  -  Moves the cursor to the left.
CHR$(30)   ^ [Shift][6] - Moves the cursor up.
CHR$(31)   ^ _  -  Moves the cursor down.
```

The method used to add control codes to keywords is as follows:-

```
KEY 3,"COMPUTE" + CHR$(13)
```

This will cause your keyword to be displayed followed by a carriage return or ENTER.

```
KEY 3,CHR$(12) + "COMPUTE" + CHR$(13)
```

This instruction has two control codes added to the keyword and from the control code table, it can be seen that the instruction clears the screen and displays your word which is followed by a carriage return.

The maximum number of characters you can have assigned to any one function key is 15, any others will not be displayed when that key is pressed, and will not be executed as instructions.

To return all the function keys to their original commands use the following instructions either directly or from a program.

```
DEFUSR = 89 : X = USR (0)
```

The function keys can be made to look quite interesting should you wish them to be displayed. Because the windows are printed in inverse characters, we can vpoke the inverse space locations of the character set and change its appearance. Try the next program that puts "racing stripes" on the cursor keys.

```
10 CLS
20 FOR X = 2816 TO 2823 STEP 2
30 VPOKE X,0
40 NEXT X
50 END
```

By changing the step increment in line 20 of the program you can vary the stripe spacing. Try changing the valued VPOKED into X and run the program again. If you want to display a totally different character then you could VPOKE each location from 2816 to 2823 with your own data, much the same as for changing the character set as described earlier.

MEMORY LOCATIONS

The Text screen uses the first part of the video RAM for storage of information on the screen. Each consecutive location from 0 to 959 represents a character position on the screen.

There are, however, a few locations in normal RAM that when POKED with the right byte number, give you even greater control over the Text screen.

Probably one of the best commands is **POKE &HF543,X** where X, is a number from 0 to 255. This particular location changes the width of text displayed depending on the number you used for X. If you use a number greater than 40, then some of your text will not be displayed, but is still actually held by the computer should you choose a lesser screen width.

The command that Spectravideo provides us with to alter the Text screen width is WIDTH, and you have the choice of either 39 or 40 characters using this command. WIDTH 80 can also be used when the 80 column board and a monitor are installed.

Try the next program, which will list itself with a screen width of only five characters. Try changing the byte value in line 20 to one of your choice, and re-run the program.

```
10 CLS
20 POKE &HF543,5
30 PRINT"SPECTRAVIDEO COMPUTERS"
40 LIST
50 END
```

The next series of locations can be used when you want the screen to appear in some eye catching form. It utilizes a series of locations that when VPOKED with 255 will fill the screen with horizontal lines. The addresses lie in video RAM from location 2048 to 2056.

The following program demonstrates the variations you can achieve by selecting these locations to **VPOKE** with information. It then slowly returns the screen to normal by **VPOKING** the same locations with 0.

```
10 CLS
20 FOR X = 2048 TO 2056
30 VPOKE X,255
40 FOR T = 1 TO 300:NEXT T
50 NEXT X
60 FOR X = 2048 TO 2056
70 VPOKE X,0
80 FOR T = 1 TO 300:NEXT T
90 NEXT X
100 GOTO 20
```

If you stop the program and list it, notice that the text is displayed in normal characters while the background appears to be in the inverse mode. The locations that we are actually using, contain the shape for the normal space character, and the program changes this character to inverse.

The program can be stopped anywhere, and to remove the lines on the screen, if you have any, use either **SCREEN** 1 or **VPOKE** the locations that are filled with byte number **255**.

The use of capital letters for input in a running program has always been a problem, because, if the user does not press the CAPS LOCK key before running the program, then it asks you to input again until it recognizes the characters. The problem of switching the CAPS LOCK key on in a running program can be done in several ways, with their advantages and disadvantages.

The shortest command to switch to upper case characters is **POKE** 65080,1, however, the light emitting diode on the CAPS LOCK key does not turn on, but it can be switched on with the next command embedded in your program; **OUT &H8C,255**. To return to lower case letters use **POKE** 65080,0. The next small program demonstrates its use.

```
10 CLS
20 POKE 65080,1
30 INPUT"WHAT IS YOUR NAME ";A$
40 POKE 65080,0
50 INPUT"SORRY, WHAT WAS YOUR NAME ";B$
60 PRINT"OH, ";B$
70 END
```

The next routine is perhaps the best available at this time. It checks to see if the CAPS LOCK key is on and if it is, it leaves it on, else if it is off it switches it on. This is quite difficult to say aloud and make it intelligible, but perhaps if you try the next routine you may understand.

```
10 CLS
20 INPUT"WHAT IS YOUR NAME ";A$
30 PRINT"THIS IS A LOWER CASE ";A$
40 IF PEEK(&HFE38) = 0 THEN DEFUSR = 16359 : Y=USR(0)
50 INPUT"WHAT IS YOUR NAME ";B$
60 PRINT"THIS IS AN UPPER CASE ";B$
70 END
```

Notice that the CAPS LOCK light is on as well, and I'm sure you didn't switch it on. Turn CAPS LOCK off and run the program again. This can be very handy in a program that requires the user to input single characters for options as you don't need instructions like **IF A$ = "A" OR A$ = "a" THEN GOTO** 1000, you only have to check for an upper case letter input.

There is one other instruction that will switch the computer to capital letters and light the diode on the CAPS LOCK key as well. Use the following:-

```
DEFUSR 0 = 16359 : X = USR (0)
```

By running this again you toggle the CAPS LOCK key from on to off.

EXAMPLE PROGRAM

The following program uses the Text Screen and demonstrates some of the things I have covered in this chapter.

MENU PROGRAM

This program utilizes the inverse characters and the keyboard graphic symbols to produce a professional looking selection menu that can be adapted very easily to your own application.

```
10 COLOR 15, 4: SCREEN 0: CLS: SCREEN,0: LOCATE,,0
20 LOCATE 8,1: PRINT"┌──────────────────────────┐"
30 LOCATE 8,2: PRINT"│ ":POKE 65077!,1: LOCATE 9,2: PRINT  "  MAIN
SELECTION MENU ": POKE 65077!,0: LOCATE 30,2: PRINT" │"
40 LOCATE 8,3: PRINT"└──────────────────────────┘"
50 GOSUB 180: GOSUB 190: GOSUB 200: GOSUB 210: GOSUB  220:  GOSUB
230: GOSUB 240: GOSUB 250: GOSUB 260
60 LOCATE 1,22: POKE 65077!,1: PRINT"  PRESS  THE  SPACE  BAR  -
THEN  ENTER ": POKE 65077!,0
70 X = 1
80 POKE 65077!,1: ON X GOSUB 180, 190, 200, 210, 220,  230,  240,
250, 260
90 S$ = INKEY$
100 IF S$ = CHR$(13) THEN 160 ELSE IF S$ = CHR$(32) THEN 120
110 IF S$ = "" THEN GOTO 80 ELSE GOTO 130
120 X = X + 1
130 Y = X: Y = Y - 1: POKE 65077!,0: ON Y GOSUB  180,  190,  200,
210, 220, 230, 240, 250, 260
140 IF X = 10 THEN 150 ELSE 80
150 GOTO 70
160 IF X = 1 THEN 270 ELSE IF X = 2 THEN 270 ELSE IF X =  3  THEN
270 ELSE IF X = 4 THEN 270 ELSE IF X = 5 THEN 270 ELSE IF X  =  6
THEN 270 ELSE IF X = 7 THEN 270 ELSE IF X = 8 THEN 270 ELSE IF  X
= 9 THEN 280
170 GOTO 90
180 LOCATE 8,6: PRINT " 1. SELECT OPTION MENU  ": RETURN
190 LOCATE 8,7: PRINT" 2. ENTER MACHINE CODE  ": RETURN
200 LOCATE 8,8: PRINT" 3. ASSEMBLE PROGRAM    ": RETURN
210 LOCATE 8,9: PRINT" 4. DISASSEMBLE PROGRAM ": RETURN
220 LOCATE 8,10: PRINT" 5. SELECT EDITOR MENU  ": RETURN
230 LOCATE 8,11: PRINT" 6. CALL MONITOR PROG.  ": RETURN
240 LOCATE 8,12: PRINT" 7. PEEKS AND POKES     ": RETURN
250 LOCATE 8,13: PRINT" 8. BOOT DISK DRIVE     ": RETURN
260 LOCATE 8,14: PRINT" 9. EXIT THIS PROGRAM   ": RETURN
270 GOTO 10
280 CLS: LOCATE,,1: POKE 65077!,0: SCREEN,1: END
```

THE GRAPHIC SCREENS

1. THE GRAPHIC SCREENS — INTRODUCTION.

2. THE VIDEO RAM MEMORY MAP.

3. USING THE VIDEO RAM.

4. SPRITES AND HOW TO USE THEM.

5. EXAMPLE PROGRAMS.

THE GRAPHIC SCREENS

The SV-318 and SV-328 computers are provided with a TMS-9918/9929 Video Display Processor (VDP) which generates all control, syncronization and composite signals. It also controls the storage, retrieval and refresh of the display data stored in the screen memory. The VDP uses a 10.738 Mhz crystal. It generates all required internal clock signals. The central processing unit (CPU) clock speed is supplied by the VDP and is obtained by dividing 10.738 by 3.

The VDP can be addressed through the normal RAM with the use of BASIC instructions. Commands such as VPOKE and VPEEK address the video RAM directly from BASIC and greatly enhance the control the programmer has over the video display processor.

The VDP includes such abilities as handling 32 sprites at one time and having four colour display modes.

```
GRAPHICS 1 MODE
GRAPHICS 2 MODE
MULTICOLOUR MODE
TEXT MODE
```

The GRAPHICS 1 and 2 modes allow the screen to be broken up into groups of 8 x 8 dots called pixels, these in turn are called pattern positions. The high resolution screen has a possible 256 across by 192 down pixel locations, therefore there are 32 x 24 pattern positions in the graphics mode.

GRAPHICS 1 mode allows 256 possible patterns to be defined for the 768 positions on the screen with two different colours for each line of a pattern. Therefore with 8 x 8 patterns, the full 15 colours including transparent may be used for a single position.

In the TEXT mode, the screen is broken up into 6 x 8 pixel positions. Sprites do not appear when in TEXT mode and only two colours can be defined for the entire screen.

The MULTICOLOUR mode or low resolution screen is broken into 64 x 48 positions, each being 4 x 4 pixels. Only one colour is allowed to each position.

Listed below are a few programs that display the various combinations of graphic screens available. They are mostly related to colour variations which should indicate the complexity of the graphic screens and how, with careful programming any graphic shape can be used in any colour.

```
 10 CLS
 20 FOR T = 1 TO 21
 30 PRINT"This is the Spectravideo Text Screen"
 40 NEXT T
 50 FOR T = 1 TO 15
 60 COLOR 15,T
 70 FOR X = 1 TO 200:NEXT X
 80 NEXT T
 90 COLOR 15,4
100 FOR T = 1 TO 15
110 COLOR T,4
120 FOR X = 1 TO 200:NEXT X
130 NEXT T
140 END
```

```
 10 SCREEN 2
 20 S = RND(-TIME)
 30 PRINT"    THE"
 40 PRINT"  LOW  RES"
 50 PRINT"   SCREEN"
 60 FOR T = 1 TO 1000:NEXT T
 70 FOR T = 1 TO 300
 80 X = INT(RND(S)*256)
 90 Y = INT(RND(S)*192)
100 C = INT(RND(S)*15)
110 LINE-(X,Y),C
120 NEXT T
130 FOR X = 1 TO 200:NEXT X
140 FOR S = 1 TO 10:FOR T = 1 TO 15
150 COLOR 15,4,T
160 FOR X = 1 TO 100:NEXT X
170 NEXT T:NEXT S
180 FOR T = 1 TO 1500:NEXT T
190 END
```

By changing the instruction in line 10 to read **SCREEN** 1 and running the program above again, you will have a demonstration of the high resolution screen. You can change line 40 to read **40 PRINT " HIGH RES."** if you like as well.

The next series of programs rely on VPOKING screen memory locations to give you an indication of the complexity of them and also a display of the individual locations available.

```
10 SCREEN 2
20 FOR T = 0 TO 1535
30 X = INT(RND(-TIME)*255)
40 VPOKE T,X
50 NEXT T
60 GOTO 60
```

```
10 SCREEN 1
20 LOCATE 50,180:PRINT"GRAPHICS 2 MODE"
30 FOR T = 0 TO 6144
40 X = INT(RND(-TIME)*255)
50 VPOKE T,X
60 NEXT T
70 CLS
80 LOCATE 50,180:PRINT"GRAPHICS 1 MODE"
90 FOR T = 8192 TO 14335
100 X = INT(RND(-TIME)*255)
110 VPOKE T,X
120 NEXT T
130 GOTO 130
```

VIDEO RAM MEMORY MAP

The Video RAM of 16,384 bytes contains all the information necessary to manipulate your graphics and text as you wish. It contains tables for the various information groups needed for the different screen types, eg. text characters, sprite pattern tables and sprite attribute tables. Listed below is the Video RAM memory Map as I could best ascertain at the time of publication, it may by no means be complete.

THE TEXT SCREEN.

Decimal Addresses	Hexa-decimal Address.	Description
Ø to 959	&HØ to &H3BF	Character positions on the screen.
2048 to 4096	&H8ØØ to &H1ØØØ	Character set shape table.

THE LOW RESOLUTION SCREEN.

Decimal Adresses	Hexa-decimal Address.	Description
Ø to 1535	&HØ to &H5FF	Screen Locations.
6912 to 7039	&H1BØØ to &H1B7F	Sprite Attribute Table.
14336 to 15360	&H38ØØ to &H3CØØ	Sprite Pattern Table.

HIGH RESOLUTION GRAPHIC SCREEN.

Decimal Addresses	Hexa-decimal Address.	Description
Ø to 6143	&HØ to &H18ØØ 17FF	GRAPHICS 2 Mode Screen Locations.
6912 to 7039	&H1BØØ to &H1B7F 1FFF	Sprite Attribute Table.
8192 to 14335	&H2ØØØ to &H37FF	GRAPHICS 1 Mode Screen Locations.
14336 to 15360	&H38ØØ to &H3CØØ	Sprite Pattern Table.

All of the above locations can be addressed with the use of VPOKE and VPEEK.

Of all the locations, the most useful are the sprite attribute and sprite pattern tables which can be worked upon under a program to give far greater control of sprites. These locations control the individual characteristics of each sprite that is created, controlling their X and Y locations on the screen, their colour and the sprite plane they may reside on at any one time.

Below are listed the sprite attribute and sprite pattern tables. Later in the chapter I will demonstrate the use of these tables to create animation as you may never have seen before.

SPRITE ATTRIBUTE TABLE

(giving individual addresses to the sprite attributes)
decimal (hexa-decimal)

SPRITE NUMBER	Y POSITION	X POSITION	SPRITE ~~PLANE~~ Pattern	COLOUR
SPRITE 0	6912(1B00)	6913(1B01)	6914(1B02)	6915(1B03)
SPRITE 1	6916(1B04)	6917(1B05)	6918(1B06)	6919(1B07)
SPRITE 2	6920(1B08)	6921(1B09)	6922(1B0A)	6923(1B0B)
SPRITE 3	6924(1B0C)	6925(1B0D)	6926(1B0E)	6927(1B0F)
SPRITE 4	6928(1B10)	6929(1B11)	6930(1B12)	6931(1B13)
SPRITE 5	6932(1B14)	6933(1B15)	6934(1B16)	6935(1B17)
SPRITE 6	6936(1B18)	6937(1B19)	6938(1B1A)	6939(1B1B)
SPRITE 7	6940(1B1C)	6941(1B1D)	6942(1B1E)	6943(1B1F)
SPRITE 8	6944(1B20)	6945(1B21)	6946(1B22)	6947(1B23)
SPRITE 9	6948(1B24)	6949(1B25)	6950(1B26)	6951(1B27)
SPRITE 10	6952(1B28)	6953(1B29)	6954(1B2A)	6955(1B2B)
SPRITE 11	6956(1B2C)	6957(1B2D)	6958(1B2E)	6959(1B2F)
SPRITE 12	6960(1B30)	6961(1B31)	6962(1B32)	6963(1B33)
SPRITE 13	6964(1B34)	6965(1B35)	6966(1B36)	6967(1B37)
SPRITE 14	6968(1B38)	6969(1B39)	6970(1B3A)	6971(1B3B)
SPRITE 15	6972(1B3C)	6973(1B3D)	6974(1B3E)	6975(1B3F)
SPRITE 16	6976(1B40)	6977(1B41)	6978(1B42)	6979(1B43)
SPRITE 17	6980(1B44)	6981(1B45)	6982(1B46)	6983(1B47)
SPRITE 18	6984(1B48)	6985(1B49)	6986(1B4A)	6987(1B4B)
SPRITE 19	6988(1B4C)	6989(1B4D)	6990(1B4E)	6991(1B4F)
SPRITE 20	6992(1B50)	6993(1B51)	6994(1B52)	6995(1B53)
SPRITE 21	6996(1B54)	6997(1B55)	6998(1B56)	6999(1B57)
SPRITE 22	7000(1B58)	7001(1B59)	7002(1B5A)	7003(1B5B)
SPRITE 23	7004(1B5C)	7005(1B5D)	7006(1B5E)	7007(1B5F)
SPRITE 24	7008(1B60)	7009(1B61)	7010(1B62)	7011(1B63)
SPRITE 25	7012(1B64)	7013(1B65)	7014(1B66)	7015(1B67)
SPRITE 26	7016(1B68)	7017(1B69)	7018(1B6A)	7019(1B6B)
SPRITE 27	7020(1B6C)	7021(1B6D)	7022(1B6E)	7023(1B6F)
SPRITE 28	7024(1B70)	7025(1B71)	7026(1B72)	7027(1B73)
SPRITE 29	7028(1B74)	7029(1B75)	7030(1B76)	7031(1B77)
SPRITE 30	7032(1B78)	7033(1B79)	7034(1B7A)	7035(1B7B)
SPRITE 31	7036(1B7C)	7037(1B7D)	7038(1B7E)	7039(1B7F)

This table will serve as a quick reference guide to any aspect of any sprite whenever you need it.

The following program gives a simple demonstration of the use of the sprite attribute table. It uses the memory locations for sprite 0, which you can confirm from the attribute table. The addresses used in the program are in decimal notation.

```
10 SCREEN 1
20 FOR T = 1 TO 8
30 READ A
40 A$ = A$ + CHR$(A)
50 NEXT T
60 SPRITE$(0) = A$
70 PUT SPRITE 0,(50,50),15,0
80 FOR X = 0 TO 255
90 VPOKE 6913,X
100 NEXT X
110 GOTO 80
120  DATA 224,248,62,255,62,248,224,0
```

If you change the instruction in line 80 to read **FOR X = 1 TO 15** and the instruction in line 90 to **VPOKE 6915,X** and run the program again, the sprite will stay still but the colour of the sprite will be ever changing.

SPRITE PATTERN TABLES

The sprite pattern tables are dependant upon the screen mode your program uses.

The tables listed below give the addresses to individual sprites depending on the graphic screen used. You can only use 8 x 8 pixel sprites on screen SCREEN 1,0, however, SCREEN 1,2 allows 8 x 8, 16 x 16 pixel sized sprites, but the sprite patterns are held in different locations for each size of sprite used. This is where the sprite pattern table begins to vary between screens. The tables below should make this quite clear.

SPRITE PATTERN TABLE

(8 x 8 pixel size sprites)

SPRITE No.	DECIMAL ADDR.	HEXA-DECIMAL ADDR.
0	14336 to 14343	&H3800 to &H3807
1	14344 to 14351	&H3808 to &H380F
2	14352 to 14359	&H3810 to &H3817
3	14360 to 14367	&H3818 to &H381F
4	14368 to 14375	&H3820 to &H3827
5	14376 to 14383	&H3828 to &H382F
6	14384 to 14391	&H3830 to &H3837
7	14392 to 14399	&H3838 to &H383F
8	14400 to 14407	&H3840 to &H3847
9	14408 to 14415	&H3848 to &H384F
10	14416 to 14423	&H3850 to &H3857
11	14424 to 14431	&H3858 to &H385F
12	14432 to 14439	&H3860 to &H3867
13	14440 to 14447	&H3868 to &H386F
14	14448 to 14455	&H3870 to &H3877
15	14456 to 14463	&H3878 to &H387F
16	14464 to 14471	&H3880 to &H3887
17	14472 to 14479	&H3888 to &H388F
18	14480 to 14587	&H3890 to &H3897
19	14488 to 14495	&H3898 to &H389F
20	14496 to 14503	&H38A0 to &H38A7
21	14504 to 14511	&H38A8 to &H38AF
22	14512 to 14519	&H38B0 to &H38B7
23	14520 to 14527	&H38B8 to &H38BF
24	14528 to 14535	&H38C0 to &H38C7
25	14536 to 14543	&H38C8 to &H38CF
26	14544 to 14551	&H38D0 to &H38D7
27	14552 to 14559	&H38D8 to &H38DF
28	14560 to 14567	&H38E0 to &H39E7
29	14568 to 14575	&H38E8 to &H38EF
30	14576 to 14583	&H38F0 to &H38F7
31	14584 to 14591	&H38F8 to &H38FF

SPRITE PATTERN TABLE

(16 x 8 pixel size sprites)

SPRITE No.	DECIMAL ADRR.	HEXA-DECIMAL ADDR.
0	14336 to 14351	&H3800 to &H380F
1	14368 to 14383	&H3820 to &H382F
2	14400 to 14415	&H3840 to &H384F
3	14432 to 14473	&H3860 to &H386F
4	14464 to 14479	&H3880 to &H388F
5	14496 to 14511	&H38A0 to &H38AF
6	14528 to 14543	&H38C0 to &H38CF
7	14560 to 14575	&H38E0 to &H38EF
8	14592 to 14607	&H3900 to &H390F
9	14624 to 14639	&H3920 to &H392F
10	14656 to 14671	&H3940 to &H394F
11	14688 to 14703	&H3960 to &H396F
12	14720 to 14735	&H3980 to &H398F
13	14752 to 14767	&H39A0 to &H39AF
14	14784 to 14799	&H39C0 to &H39CF
15	14816 to 14831	&H39E0 to &H39EF
16	14848 to 14863	&H3A00 to &H3A0F
17	14880 to 14895	&H3A20 to &H3A2F
18	14912 to 14927	&H3A40 to &H3A4F
19	14944 to 14959	&H3A60 to &H3A6F
20	14976 to 14991	&H3A80 to &H3A8F
21	15008 to 15023	&H3AA0 to &H3AAF
22	15040 to 15055	&H3AC0 to &H3ACF
23	15072 to 15087	&H3AE0 to &H3AEF
24	15104 to 15119	&H3B00 to &H3B0F
25	15136 to 15151	&H3B20 to &H3B2F
26	15168 to 15183	&H3B40 to &H3B4F
27	15200 to 15215	&H3B60 to &H3B6F
28	15232 to 15247	&H3B80 to &H3B8F
29	15264 to 15279	&H3BA0 to &H3BAF
30	15296 to 15311	&H3BC0 to &H3BCF
31	15328 to 15343	&H3BE0 to &H3BEF

SPRITE PATTERN TABLE

(16 x 16 pixel size sprites)

SPRITE No.	DECIMAL ADRR.	HEXA-DECIMAL ADDR.
0	14336 to 14367	&H3800 to &H381F
1	14368 to 14399	&H3820 to &H383F
2	14400 to 14431	&H3840 to &H385F
3	14432 to 14463	&H3860 to &H387F
4	14464 to 14495	&H3880 to &H389F
5	14496 to 14527	&H38A0 to &H38BF
6	14528 to 14559	&H38C0 to &H38DF
7	14560 to 14591	&H38E0 to &H39FF
8	14592 to 14623	&H3900 to &H391F
9	14624 to 14655	&H3920 to &H393F
10	14656 to 14687	&H3940 to &H395F
11	14688 to 14719	&H3960 to &H397F
12	14720 to 14751	&H3980 to &H399F
13	14752 to 14783	&H39A0 to &H39BF
14	14784 to 14815	&H39C0 to &H39DF
15	14816 to 14847	&H39E0 to &H39FF
16	14848 to 14879	&H3A00 to &H3A1F
17	14880 to 14911	&H3A20 to &H3A3F
18	14912 to 14943	&H3A40 to &H3A5F
19	14944 to 14975	&H3A60 to &H3A7F
20	14976 to 15007	&H3A80 to &H3A9F
21	15008 to 15039	&H3AA0 to &H3ABF
22	15040 to 15071	&H3AC0 to &H3ADF
23	15072 to 15103	&H3AE0 to &H3AFF
24	15104 to 15135	&H3B00 to &H3B1F
25	15136 to 15167	&H3B20 to &H3B3F
26	15168 to 15199	&H3B40 to &H3B5F
27	15200 to 15231	&H3B60 to &H3B7F
28	15232 to 15263	&H3B80 to &H3B9F
29	15264 to 15295	&H3BA0 to &H3BBF
30	15296 to 15327	&H3BC0 to &H3BDF
31	15328 to 15359	&H3BE0 to &H3BFF

USING THE VIDEO RAM

Now that we have an almost complete map of the video RAM, we can begin to use it to increase the visual effects of our programs. With the individual locations of the high resolution screen known, we can create block graphics with reasonable confidence using VPOKE and filling the locations with a colour number from 0 to 15. If you VPOKE a byte number from 16 to 255 the result is a default colour number based on the 16 colours available on the computer, so you are best to stick with the standard numbers for the colours, that way you will know the result of the number you VPOKE.

SPRITES AND HOW TO USE THEM

There are several ways of creating sprites on the Spectravideo Computers. The most common of which is with the use of binary numbers held in DATA statements, converted to characters and then assigned to a sprite string variable. **(SPRITE$)**

Binary numbers serve to give you a visual image of the sprite before it is converted and transferred to the Video RAM via your program. The program below creates a sprite and then moves it across the high resolution screen from left to right.

```
 10 SCREEN 1
 20 FOR X = 1 TO 8
 30 READ A$
 40 B$ = B$ +CHR$(VAL("&B"+A$))
 50 NEXT X
 60 SPRITE$ (1) = B$
 70 FOR T = 0 TO 255
 80 PUT SPRITE 1,(T,96),15,1
 90 NEXT T
100 GOTO 70
110 DATA 10000000
120 DATA 01000100
130 DATA 00100110
140 DATA 11111111
150 DATA 00100110
160 DATA 01000100
170 DATA 10000000
180 DATA 00000000
```

The DATA statements could have been decimal or hexa-decimal numbers, but the sprite pattern would not have been obvious. Note:- to change to decimal numbers, line 30 would have to read **30 READ A** and line 40 should read **40 B$ = B$ + CHR$(A)**. For hexa-decimal numbers the only change would be in line 40 where the **"&B"** would become **"&H"**. Of course the numbers held in the **DATA** lines would also change accordingly and they could be comfortably held in one DATA statement line.

There is one other method for creating sprites, and that is, using the **CHR$** instruction and accumulating them as a SPRITE$ variable. The following demonstrates the use of CHR$ to form the same sprite as was created in the above program.

```
SPRITE$(1) = CHR$(128) + CHR$(68) + CHR$(38) + CHR$(255) +
             CHR$(38) + CHR$(68) + CHR$(128) + CHR$(0)
```

Lines 20 to 50 can be deleted and there is no need for the DATA statement lines if this method is used in the preceding program.

Now that we have the locations for the sprite pattern table, we can begin to change the shape of our sprites with the use of VPOKE and new byte numbers representing the new data to create the change in the shape of the sprite. Sprite pattern data is held in consecutive addresses for the various sprites you may use in your programs, therefore, we can locate any row of data and change it. Reference can be made to the sprite pattern tables given earlier. The next program creates a sprite with decimal numbers in the DATA line, and displays the decimal numbers on the screen. It then changes the shape of the sprite and re-displays the new information held in the correct pattern locations.

```
10 SCREEN 1
20 FOR X = 1 TO 8:READ A
30 A$ = A$ + CHR$(A):NEXT X
40 SPRITE$(0) = A$
50 PUT SPRITE 0,(128,96),15,0
60 GOSUB 150
70 VPOKE 14336,0:VPOKE 14337,4
80 VPOKE 14338,6:VPOKE 14340,6
90 VPOKE 14341,4:VPOKE 14342,0
100 GOSUB 150
110 VPOKE 14336,128:VPOKE 14337,68
120 VPOKE 14338,38:VPOKE 14340,38
130 VPOKE 14341,68:VPOKE 14342,128
140 GOTO 60
150 LINE(0,0)-(80,80),4,BF: LOCATE 0,0
160 FOR T = 14336 TO 14344
170 PRINT"  ";T;VPEEK(T)
180 NEXT T
190 FOR Y = 1 TO 1000:NEXT Y
200 RETURN
210 DATA 128,68,38,255,38,68,128,0
```

The next program again uses the sprite pattern table for 32 x 32 pixel sprites and uses sprite number 2. The locations that are used in the program can be confirmed with the reference table given earlier. The program creates a recognizable shape and changes it to give real time animation. While the sprite is changing it also moves across the screen.

```
10 COLOR 15,4,4:SCREEN 1,2
20 FOR T = 1 TO 32:READ A:A$ = A$ + CHR$(A):NEXT T
30 SPRITE$(2) = A$
40 DATA 0,7,0,0,0,0,67,95,255,64,64,0,0,0,0,0,0,255
50 DATA 32,112,200,196,226,226,242,124,0,0,0,0,0,0
60 PUT SPRITE 2,(0,9),15,2
70 FOR X = 0 TO 255
80 VPOKE 6921,X
90 VPOKE &H3841,0:VPOKE &H3851,0:VPOKE &H3846,3
100 VPOKE &H3847,31:VPOKE &H3849,0:VPOKE &H384A,0
110 FOR S = 1 TO 5: NEXT S
120 VPOKE &H3841,7:VPOKE &H3851,255:VPOKE &H3846,67
130 VPOKE &H3847,95:VPOKE &H3849,64:VPOKE &H384A,64
140 NEXT X
150 GOTO 70
```

Notice that the sprite pattern addresses in the program are calling the respective hexa-decimal address this time and there is a new decimal location used in line 80. This address is the X co-ordinate of sprite number 2 acquired from the sprite attribute table and can be confirmed by checking the reference table given.

The sprite attribute table can also control our sprites Y co-ordinate, colour and sprite plane number with such dexterity that real time applications and true animation are possible under BASIC and not sophisticated machine code programming.

The following program demonstrates the use of the attribute table by creating a square sprite, displaying it's attributes and then changing the characteristics of the sprite to display the new data controlling it, namely X position, Y position and its' colour.

```
 10 COLOR 15,4,4:SCREEN 1,0:FOR T = 1 TO 8:RESTORE
 20 READ A:A$ = A$ +CHR$(A):NEXT T:SPRITE$(1) = A$
 30 DATA 255
 40 S=RND(-TIME)
 50 PRINT TAB(5)"Y position"
 60 PRINT TAB(5)"X position"
 70 PRINT TAB(5)"Sprite Pl."
 80 PRINT TAB(5)"Colour No."
 90 Y = INT(RND(S)*192)
100 X = INT(RND(S)*256)
110 C = INT(RND(S)*15)
120 VPOKE 6916,Y:VPOKE 6917,X:VPOKE 6919,C
130 LOCATE 0,0:PRINT TAB(15) VPEEK(6916)
140 PRINT TAB(15) VPEEK(6917)
150 PRINT TAB(15) VPEEK(6918)
160 PRINT TAB(15) VPEEK(6919)
170 FOR T = 1 TO 1000:NEXT T
180 LINE (90,0)-(130,50),4,BF
190 GOTO 90
```

Other useful applications of the sprite attribute table include the following:- removal of sprites instantaneously from the screen without trace, (good for ON SPRITE collisions), and removal of sprites from a selected sprite plane number and upwards to the highest plane of 31.

The removal of an individual sprite from the screen can be achieved in one of two ways. Either put the selected sprites' Y co-ordinate to **209** using PUT SPRITE [sprite plane],([X position],209),[sprite colour],[sprite pattern number], or by VPOKING 209 into the location that controls that particular sprite's Y position, eg. **VPOKE** 6920,209 for sprite number 2.

To bring the sprite back to the screen, just change the Y co-ordinate of the sprite to that of your choice with either of the above mentioned methods.

Run the following program which creates 5 sprites of the same shape and then removes the 3rd sprite across and down.

```
 10 SCREEN 1
 20 FOR T = 1 TO 8:RESTORE:READ A
 30 A$ = A$ + CHR$(A): NEXT T
 40 SPRITE$(1) = A$:SPRITE$(2) = A$:SPRITE$(3) = A$
 50 SPRITE$(4) = A$:SPRITE$(5) =A$
 60 DATA 255
 70 PUT SPRITE 1,(10,50),15,1:PUT SPRITE 2,(30,60),1,2
 80 PUT SPRITE 3,(50,70),13,3:PUT SPRITE 4,(70,80),12,4
 90 PUT SPRITE 5,(90,90),6,5
100 FOR T = 1 TO 1000:NEXT T
110 PUT SPRITE 3,(50,209),13,3      (or VPOKE 6924,209)
120 GOTO 120
```

To remove all sprites from a specified sprite plane number up to plane number 31 (the highest), change the Y position of the lowest sprite to remove to **208** using either PUT SPRITE or VPOKE as described earlier.

Change the last program so that line 100 reads **PUT SPRITE** 3, (50,208),13,3 or **VPOKE** 6924,208 and run the program again.

The sprites from the third one to the fifth should all disappear. To re-instate the sprites that are missing just change the Y co-ordinate of the lowest you removed or any sprite number you desire using PUT SPRITE or VPOKE and specifiy its new location.

Once in a while the program you just removed with the instruction NEW was a mistake, you really wanted to view all the sprite pattern data, well then, no problem. Just VPEEK into the corresponding video RAM locations and retrieve your sprite DATA and attributes. Typing NEW does not clear the video RAM of its DATA.

The movement of sprites can be achieved in various ways. The usual method is to use a FOR - NEXT loop with a PUT SPRITE command sandwiched between. eg.

```
10 FOR X = 0 TO 255
20 PUT SPRITE 1,(X,96),15,1
30 NEXT X
```

The next method is to use the VPOKE command and the value from a FOR - NEXT loop to change the value of the required co-ordinate location in video RAM to create the movement.

```
10 FOR X = 0 TO 255
20 VPOKE 6920,X
30 NEXT X
```

The last method is quite a complex instruction, but handled carefully it can be used to achieve sprite movements as never before imagined. The instruction relys on the program setting up the initial position of the sprite on the screen and then for each pass of the new instruction the program will move the sprite relative to its last position. Perhaps a small routine may make this a little clearer.

```
10 COLOR 15,4,4:SCREEN 1,0
20 FOR X = 1 TO 8:READ A:A$ = A$ + CHR$(A):NEXT X
30 SPRITE$(1) = A$
40 PUT SPRITE 1,(50,50),15,1
50 FOR X = 1 TO 10
60 PUT SPRITE 1,STEP(-1,1),15,1
70 NEXT X
80 FOR X = 1 TO 10
90 PUT SPRITE 1,STEP(-1,-1),15,1
100 NEXT X
110 FOR X = 1 TO 10
120 PUT SPRITE 1,STEP(1,1),15,1
130 NEXT X
140 GOTO 50
150 DATA 255,255,255,255,255,255,255,255
```

The new instruction is in fact a composite of various instructions available under BASIC. Line 40 is the first occurrence of it and is explained below:-

```
PUT SPRITE [Sprite Plane],STEP([X Offset],[Y Offset]),
Colour,Sprite No.
```

The numbers used for the X and Y offsets can be any that you wish, however, the larger the offset numbers, the faster the sprite will step across the screen.

More than one sprite's movement can be controlled this way. The following program which is a bit of fun, demonstrates multiple sprite movement using four sprites defined to almost the same location and moving together.

```
 10 COLOR 15,4,4:SCREEN 1,0
 20 FOR T = 1 TO 8:READ A:A$ =A$ +CHR$(A):NEXT T
 30 SPRITE$(1) = A$:SPRITE$(2) = A$:SPRITE$(3) = A$
 40 SPRITE$(4) = A$
 50 PUT SPRITE 1,(128,96),15,1
 60 PUT SPRITE 2,(128,97),15,2
 70 PUT SPRITE 3,(128,96),15,3
 80 PUT SPRITE 4,(128,97),15,4
 90 PUT SPRITE 1,STEP(0,-1),15,1
100 PUT SPRITE 2,STEP(1,-1),15,2
110 PUT SPRITE 3,STEP(1,0),15,3
120 PUT SPRITE 4,STEP(1,1),15,4
130 FOR T = 1 TO 5:NEXT T
140 GOTO 90
150 DATA 0,0,0,64,64,0,0,0
```

You would have noticed in the last program that I used only four sprites and no more. This is because one of the limitations of the TMS-9918/9929 video chip, is its inability to hold more than four sprites in the same horizontal line. Even if you offset the sprites position by one pixel, part of that sprite will not be displayed.

The joystick command STICK can be used with the PUT SPRITE - STEP command to good effect. The following program demonstrates the use of these commands to control the sprite on the screen either with the keyboard joystick, cursor keys or an external joystick.

```
 10 SCREEN 1
 20 FOR X = 1 TO 8: READ A: A$ = A$ + CHR$(A): NEXT X
 30 SPRITE$(1) = A$
 40 DATA 255,255,255,255,255,255,255,255
 50 X = 127: Y = 96
 60 PUT SPRITE 1,(X,Y),15,1
 70 GOTO 160
 80 PUT SPRITE 1, STEP(0,-2),15,1: RETURN
 90 PUT SPRITE 1, STEP(2,-2),15,1: RETURN
100 PUT SPRITE 1, STEP(2,0),15,1: RETURN
110 PUT SPRITE 1, STEP(2,2),15,1: RETURN
120 PUT SPRITE 1, STEP(0,2),15,1: RETURN
130 PUT SPRITE 1, STEP(-2,2),15,1: RETURN
140 PUT SPRITE 1, STEP(-2,0),15,1: RETURN
150 PUT SPRITE 1, STEP(-2,-2),15,1: RETURN
160 S = STICK (0) OR STICK (1)
170 ON S GOSUB 80,90,100,110,120,130,140,150
180 GOTO 160
```

EXAMPLE PROGRAMS

The following programs are examples of the use of the Graphics Screen, utilizing sprites and the video RAM to control movement and colour. Some of the listings may be used as sub-routines in your own programs, feel free to use them as you wish.

PATTERN

```
10 COLOR 15, 1, 1: SCREEN 1: CLS
20 FOR T = 2 TO 15
30 LINE (120,20)-(120,170), 1
40 X = 94: Z = 200
50 LINE (Z,95)-(120,X), T
60 X = X + 2: Z = Z - 2
70 IF Z = 120 THEN 90
80 GOTO 50
90 Z = 200: X = 94
100 LINE (Z,94)-(120,X), T
110 Z = Z - 2: X = X - 2
120 IF Z = 120 THEN 140
130 GOTO 100
140 Z = 40: X = 94
150 LINE (Z,94)-(120,X), T
160 Z = Z + 2: X = X - 2
170 IF Z = 120 THEN 190
180 GOTO 150
190 Z = 40: X = 95
200 LINE (Z,95)-(120,X), T
210 Z = Z + 2: X = X + 2
220 IF Z = 120 THEN 240
230 GOTO 200
240 FOR G = 1 TO 1000: NEXT: CLS: NEXT: GOTO 20
```

ORBIT

```
10 DEFINT A-Z: COLOR 15, 1, 1: SCREEN 1
20 FOR T = 1 TO 8: READ A: A$ = A$ + CHR$(A): NEXT: SPRITE$ (1) =
A$
30 CIRCLE (127,92), 10, 8,,, 1.3: PAINT (127,92), 8
40 D = D + 25: X = 90 * -SIN (D) + 127: Y = 20 * COS  (D)  +  92:
PUT SPRITE 1, (X,Y), 6, 1
50 GOTO 40
60 DATA 24, 60, 24, 0, 0, 0, 0, 0, 0
```

DESIGN

```
10 SCREEN 1
20 COLOR 15, 4, 4
30 X = 0: Y = 80
40 PSET (128,87)
50 FOR N = 0 TO 6.2832 STEP .1745
60 A = 128 + X * SIN (N): B = 87 + Y * COS (N)
70 LINE -(A,B)
80 NEXT N
90 X = X + 10: Y = Y - 10
100 IF Y = -10 THEN 120
110 GOTO 50
120 GOTO 120
```

HELICOPTER 1

```
10 COLOR 15, 4, 4: SCREEN 1,2: GOSUB 90
20 FOR T = 160 TO 10STEP -1: PUT SPRITE 1, (20,T), 15, 1: VPOKE
&H3821,0: VPOKE &H3831,0: FOR S = 1 TO 10: NEXT: VPOKE &H3821,63:
VPOKE &H3831, 248: NEXT: PUTSPRITE 1, (0,209), 15, 1
30 FOR T = 20 TO 220: PUT SPRITE 2, (T,10), 15, 2: VPOKE
&H3841,0: VPOKE &H3851,0: VPOKE &H3846,3: VPOKE &H3847,31: VPOKE
&H3849,0: VPOKE &H384A,0
40 FOR S = 1 TO 5: NEXT: VPOKE &H3841,7: VPOKE &H3851,255: VPOKE
&H3846,67: VPOKE &H3847,95: VPOKE &H3849,64: VPOKE &H384A,64:
NEXT: PUT SPRITE 2, (0,209), 15, 2
50 FOR T = 10 TO 160: PUT SPRITE 1, (220,T), 15, 1: VPOKE
&H3821,0: VPOKE &H3831,0: FOR S = 1 TO 10: NEXT: VPOKE &H3821,63:
VPOKE &H3831,248: NEXT: PUT SPRITE 1, (0,209), 15, 1
60 FOR T = 220 TO 20 STEP -1: PUT SPRITE 3, (T,160), 15, 3: VPOKE
&H3861,0: VPOKE &H3871,0: VPOKE &H3876,192: VPOKE &H3877,248:
VPOKE &H3879,0: VPOKE &H387A,0
70 FOR S = 1 TO 5: NEXT: VPOKE &H3861,255: VPOKE &H3871,224:
VPOKE &H3876,194: VPOKE &H3877,250: VPOKE &H3879,2: VPOKE
&H387A,2: NEXT
80 PUT SPRITE 3, (0,209), 15, 3: GOTO 20
90 FOR T = 1 TO 32: READ A: A$ = A$ + CHR$(A): NEXT: SPRITE$(1) =
A$
100 DATA 0, 63, 1, 3, 4, 8, 8, 8, 15, 7, 0, 0, 0, 0, 0, 0, 0,
248, 0, 128, 64, 32, 32, 32, 224, 192, 0, 0, 0, 0, 0, 0
110 FOR T = 1 TO 32: READ A: B$ = B$ + CHR$(A): NEXT: SPRITE$(2)
= B$
120 DATA 0, 7, 0, 0, 0, 0, 67, 95, 255, 64, 64, 0, 0, 0, 0, 0, 0,
255, 32, 112, 200, 196, 226, 226, 242, 124, 0, 0, 0, 0, 0, 0
130 FOR T = 1 TO 32: READ A: C$ = C$ + CHR$(A): NEXT: SPRITE$(3)
= C$
140 DATA 0, 255, 4, 14, 19, 35, 71, 71, 79, 62, 0, 0, 0, 0, 0, 0,
0, 224, 0, 0, 0, 0, 194, 250, 255, 2, 2, 0, 0, 0, 0, 0
150 RETURN
```

EXPLOSION

```
10 COLOR 15, 1, 1: SCREEN 1,2
20 SR$ = CHR$(0) + CHR$(18) + CHR$(9) + CHR$(105) + CHR$(23) +
CHR$(94) + CHR$(46) + CHR$(188) + STRING$(2,112) + CHR$(248) +
CHR$(121) + CHR$(159) + CHR$(41) + CHR$(36) + CHR$(0) + CHR$(8) +
CHR$(80) +CHR$(105) + CHR$(224) + CHR$(217)
30 SPRITE$ (1) = CHR$(2) + CHR$(30) + CHR$(30) + CHR$(59) +
CHR$(240) + CHR$(224) + CHR$(97) + CHR$(57) + CHR$(15) + CHR$(2)
+ STRING$(9,0) + STRING$(4,128)
40 SPRITE$ (2) = SR$ + CHR$(26) + CHR$(28) + CHR$(5) +
STRING$(3,6) + CHR$(14) + CHR$(13) + CHR$(217) + CHR$(118) +
CHR$(140)
50 SPRITE$ (3) = CHR$(32) + CHR$(120) + CHR$(248) + CHR$(240) +
CHR$(48)
60 X = 100: Y = 100
```

```
70 GOSUB 130: PUT SPRITE 3, (X+1,Y-1), 6, 3: FOR T = 1 TO 80:
NEXT: GOSUB 130: PUT SPRITE 1, (X-2,Y-4), 8, 1
80 FOR T = 1 TO 50: NEXT: PUT SPRITE 2, (X-6,Y-8), 10, 2: FOR T =
1 TO 50: NEXT: VPOKE 6916,209: FOR T = 1 TO 50: NEXT: PUT SPRITE
1, (X-2,Y-4), 8, 1
90 FOR T = 1 TO 80: NEXT
100 VPOKE 6924, 209: FOR T = 1 TO 80: NEXT: VPOKE 6916,209: FOR T
= 1 TO 50: NEXT: VPOKE 6920,209
110 FOR T = 1 TO 1000: NEXT
120 GOTO 70
130 SOUND 0,0: SOUND 6,30: SOUND 7,0: SOUND 8,16: SOUND 9,16:
SOUND 10,16: SOUND 11,0: SOUND 12,5: SOUND 13,0: SOUND 12,26:
RETURN
```

SPECTRAVIDEO LOGO

```
10 COLOR 15, 4, 4: SCREEN 1
20 DRAW "c15 bm 8,10 m 28,10 m 31,16 m 19,16 m 22,20 m 33,20 m
41,34 m 25,34 m 22,30 m 34,30 m 31,24 m 19,24 m 8,10": PAINT
(20,14), 15: DRAW "c15 bm 230,10 m 250,10 m 231,34 m 214,34 m
230,10 bm 230,16 m 240,16 m 230,30 m 221,30 m 230,16": PAINT
(240,14), 15
30 DRAW "c15 bm 31,10 m 47,10 m 54,24 m 45,24 m 50,34 m 44,34 m
31,10 bm 41,16 m 46,16 m 48,20 m 42,20 m 41,16": PAINT (40,14),
15: DRAW "c15 bm 206,10 m 225,10 m 221,16 m 208,16 m 206,20 m
214,20 m 212,24 m 204,24 m 201,30 m 213,30 m 210,34 m 195,34 m
206,10": PAINT (210,14), 15
40 DRAW "c15 bm 50,10 m 67,10 m 69,16 m 58,16 m 60,20 m 66,20 m
68,24 m 62,24 m 64,30 m 74,30 m 75,34 m 61,34 m 50,10": PAINT
(60,14), 15: DRAW "c15 bm 183,10 m 198,10 m 200,12 m 200,14 m
193,30 m 190,34 m 175,34 m 183,10 bm 185,16 m 194,16 m 189,30 m
180,30 m 185,16": PAINT (190,14), 15
50 DRAW "c15 bm 70,10 m 86,10 m 88,16 m 77,16 m 82,30 m 91,30 m
93,34 m 78,34 m 70,10": PAINT (75,14), 15: DRAW "c15 bm 174,10 m
179,10 m 167,34 m 163,34 m 174,10": PAINT (175,14), 15
60 DRAW "c15 bm 90,10 m 106,10 m 107,16 m 102,16 m 105,34 m
100,34 m 96,16 m 91,16 m 90,10": PAINT (100,14), 15: DRAW "c15 bm
152,10 m 156,10 m 152,30 m 165,10 m 171,10 m 155,34 m 147,34 m
152,10": PAINT (153,14), 15
70 DRAW "c15 bm 110,10 m 126,10 m 126,24 m 123,24 m 127,34 m
123,34 m 118,24 m 118,20 m 122,20 m 122,16 m 115,16 m 115,34 m
111,34 m 110,10": PAINT (120,14), 15: DRAW "c15 bm 130,10 m
149,10 m 144,34 m 141,34 m 142,24 m 134,24 m 133,34 m 130,34 m
130,10 bm 134,20 m 134,16 m 143,16 m 142,20 m 134,20"
80 PAINT (140,14), 15: DRAW "c13 bm 28,40 m 30,42 m 226,42 m
228,40 m 28,40": PAINT (40,41), 13
90 DRAW "c6 bm 32,46 m 34,48 m 222,48 m 224,46 m 32,46": PAINT
(40,47), 6
100 DRAW "c10 bm 36,52 m 38,54 m 217,54 m 219,52 m 36,52": PAINT
(50,53), 10
110 GOTO 110
```

MISCELLANEOUS CURIOSITIES & TABLES

SPECTRAVIDEO NEVER INTENDED....

This section is devoted to the curiosities and bizarre things that can happen with your computer. The routines and instructions were not sort after intentionally, but were stumbled upon when searching for something else. They are included in this book purely for interest sake and should not be taken seriously unless you can find a good application for some of them.

Most of the routines and instructions will lock up the computer, so don't leave anything in RAM that you may want to save later, as switching the machine off then on again is sometimes the only way to get your computer to respond again.

The following discussions have no set format and are presented almost in the haphazard way in which they were discovered.

So here we go then, clear RAM first, either saving your program or by typing NEW.

To turn TRON on without the usual BASIC instruction, use **POKE &HF8F1,1**. Input a small program into RAM and run it to confirm that in fact TRON was switched on. To turn TRON off or activate the TROFF function use **POKE &HF8F1,0**.

The next command only works if you have a printer attached. **POKE 62785,1** causes a LPRINT statement to the printer. If you don't have a printer connected, then the command becomes void.

Try the following:- **POKE 64003,78**. This doesn't do anything on its own, but press any character key and the colour will change. If you hold down the BACKSPACE key the screen gets very excited. Try **POKE 64003,129** and repeat the above procedure. You can return the screen to normal at any time by issuing a **SCREEN 0** or **CLS** instruction.

The next location is related to the ERROR messages that control the computer. It is rather devastating at times to any programs you have in RAM, but there is no fear of damage to the computers circuitry.

```
POKE 63887,1   - causes "overflow" and machine to reset.
POKE 63887,8   - causes "Illegal function call"
POKE 63887,14  - causes "Syntax Error"
POKE 63887,16  - causes "Internal Error" (none really)
POKE 63887,43  - causes "Device I/O Error"
POKE 63887,49  - causes numerous output.
POKE 63887,51  - causes colour change before reset.
POKE 63887,198 - causes removal of characters from screen.
POKE 63887,221 - causes "Out of DATA"
```

Press the LIST function key after giving the following instruction to the computer. **POKE 62795,1**.

Automatic line numbering can be achieved with the next statement so long as there is a line number already in RAM. If you don't have a program in RAM then use say **10 PRINT "SPECTRAVIDEO"** and then **POKE 63446,1** to activate AUTO.

It may be necessary sometimes in your program to check to see if the cassette is in one state or another. The following routine checks to see if any key was depressed under program control.

```
10  IF (INP(&H98)AND 64)<>64 THEN PRINT "A KEY WAS  PRESSED"
    ELSE 10
```

The POKE command that controls the Text Screen width that was described in the chapter on the TEXT screen has one other curious attribute. It has the abilitiy to provide constant sideways scrolling although the machine locks up, it still is impressive.

Give the following command **POKE &HF543,255**. Move the cursor up to any text you may have on the screen and then press the INSERT key, then the SPACE BAR as if you were inserting a character. The screen will begin to scroll sideways and can only be stopped by switching the machine off, never mind. . . . , a nice try though.

A rather self explanatory routine when used, is as follows:-
DEFUSR = 0 : X = USR (0). This gives the computer the ability to forget everything that you may have had in it. It basically resets the machine.

The next series of locations control the colours of the High Resolution Graphic Screens. They are as follows:-

```
&HFA0A - Location of TEXT colour on the Hi-Res. Screen.
&HFA0B - Location  for  the  foreground colour  on  the  Hi-Res.
         Screen.
&HFA0C - Location  for  the background  colour  on  the  HI-Res.
         Screen.
```

If a number corresponding to a colour from 0 to 15 is POKED into these locations then the relevant screen characteristic will change accordingly. Below are some example programs.

```
10 SCREEN 1
20 FOR X = 0 TO 15
30 PRINT "SPECTRAVIDEO";
40 POKE &HFA0A,X
50 NEXT X
60 GOTO 20
```

```
10 FOR X = 0 TO 15
20 SCREEN 1
30 POKE &HA0B,X
40 NEXT X
50 GOTO 10
```

```
10 FOR X = 0 TO 15
20 SCREEN 1
30 POKE &HFA0C,X
40 NEXT X
50 GOTO 10
```

Some rather curious locations that affect the video RAM are best demonstrated with the following programs. Most are similar, except for the screen mode selected at the beginning of the programs.

```
10 SCREEN 0
20 FOR X = 0 TO 255
30 POKE 64003,X
40 PRINT X
50 NEXT X
60 GOTO 20
```

```
10 SCREEN 0
20 FOR X = 0 TO 255
30 POKE 64006,X
40 PRINT X
50 NEXT X
60 GOTO 20
```

```
10 SCREEN 1
20 FOR X = 0 TO 255
30 POKE 64010,X
40 PRINT "SPECTRAVIDEO"
50 NEXT X
60 GOTO 20
```

```
10 SCREEN 0
20 FOR X = 0 TO 255
30 POKE 64006,X
40 PRINT "SPECTRAVIDEO"
50 NEXT X
60 GOTO 20
```

```
10 SCREEN 0
20 FOR X = 0 TO 255
30 POKE 64004,X
40 PRINT X
50 NEXT X
60 GOTO 20
```

```
10 SCREEN 1
20 FOR X = 0 TO 255
30 POKE 64004,X
40 PRINT X
50 NEXT X
60 GOTO 20
```

```
10 SCREEN 0
20 FOR X = 0 TO 255
30 POKE 64003,X
40 PRINT "SPECTRAVIDEO"
50 NEXT X
60 GOTO 60
```

The next few routines will cause havoc with your television picture, but don't worry the image you get doesn't harm the set in any way. Your eyesight might be a little impaired though.

```
10 SCREEN 1
20 POKE &HFA0D,1
30 LINE (0,0)-(256,192)
```

This locks up the computer, switch it off when finished.

```
10 FOR X = 0 TO 255
20 OUT &H81,X
30 PRINT X;
40 NEXT X
50 SCREEN 0
```

As you can see after running this program, it is self correcting with the SCREEN 0 command included in the program.

The patterns that this program creates keep changing each time it is run. To see the individual screens for a longer period, insert a FOR-NEXT loop into the program as shown below.

```
10 FOR X = 0 TO 255
20 OUT &H81,X
30 PRINT X;
40 FOR T = 1 TO 200
50 NEXT T
60 NEXT X
70 SCREEN 0
```

Now every time the program is run a different set of patterns can be viewed.

The last program in this section of up-setting the screen is the following. It is the most interesting, as at the end of running, your program can still be seen and edited as if there was nothing wrong with the computer.

```
10 FOR X = 65070 TO 65085
20 POKE X,1
30 PRINT "SPECTRAVIDEO"
40 NEXT X
```

For what it may be worth as a programming application, the next routine will boot the disk drive if it is attached to your computer, without affecting the CAPS LOCK key, if it is turned on.

```
DEFUSR = &H79E0 : X = USR(0)
```

Have you ever wondered what the **SELECT** and **PRINT** keys on the SV-328 computers are used for, well, I don't know, but we can utilize them in a program to either select an option from a menu or print out the results from running a program. We could do anything with them as they can be used as an input to branch to a sub-routine in any program.

The programs below use the port address for the keyboard and the byte number representing either the SELECT or PRINT key to check the keyboard input from a user to see if they match, if they do, then the program branches to the specified line number.
These instructions can be used anywhere in a program, for whatever purpose you wish.

```
10 IF INP(&H99) = 239 THEN GOTO 20 ELSE 10
20 PRINT "THE SELECT KEY WAS PRESSED"
30 END

10 IF INP(&H99) = 223 THEN GOTO 20 ELSE 10
20 PRINT "THE PRINT KEY WAS PRESSED"
30 END
```

Wouldn't it be nice to be able to call the Spectravideo logo to the screen at the start of your program or anywhere else in your program for that matter?, well the next routine allows us to do just that by calling the machine code routine from ROM and controlling it within a BASIC program.

```
10 DEFUSR = &H4782
20 FOR X = 1 TO 15
30 STOP ON
40 ON STOP GOSUB 90
50 COLOR 15,X,X
60 U = USR(0)
70 NEXT X
80 GOTO 20
90 COLOR 15,4:SCREEN 0
```

If the DEFUSR instruction is used on its own then the ROM routine will leave the screen in the high resolution mode and the computer appears to lock up. The only way to retrieve control, is to issue a **SCREEN 0** command. This will return the computer to normal.

Inside the RAM of the computers there sits the titles of the various software packages that apparently were to be supplied with your computer, but the actual programs never quite made it to the production deadline. These titles can be viewed with the following program.

```
10 FOR X = &H8120 TO &H8200
20 PRINT CHR$(PEEK(X));
30 NEXT X
```

There is one command in the BASIC ROM that relates to disk drives that, until recently, was unable to be used successfully. The command in question is COPY. As one would expect, this command copies a disk sitting in one drive to a disk sitting in the other. The command can also copy, for whatever its worth, a disk on one drive to itself. The command must be used in the following format.

```
COPY 1 FROM 1
COPY 1 FROM 2
COPY 2 FROM 2
COPY 2 FROM 1
```

This method of copying a disk under BASIC is rather slow and the programs 1COPY or COPY on your CP/M disk would be the faster of the two.

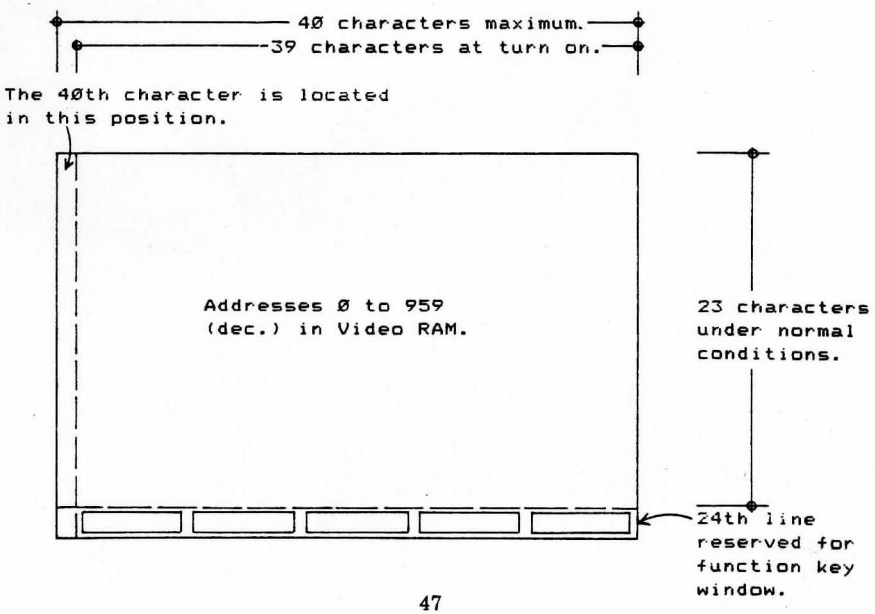
MISCELLANEOUS TABLES

The following pages include such things as screen maps and memory layouts, they can be used as reference material when creating your own programs.

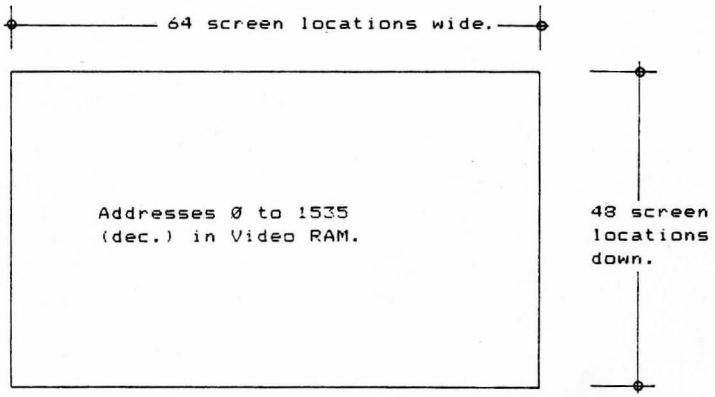
SCREEN MAPS

The following drawings are graphical representations of the various screens available on the Spectravideo and how to address each location on those screens.

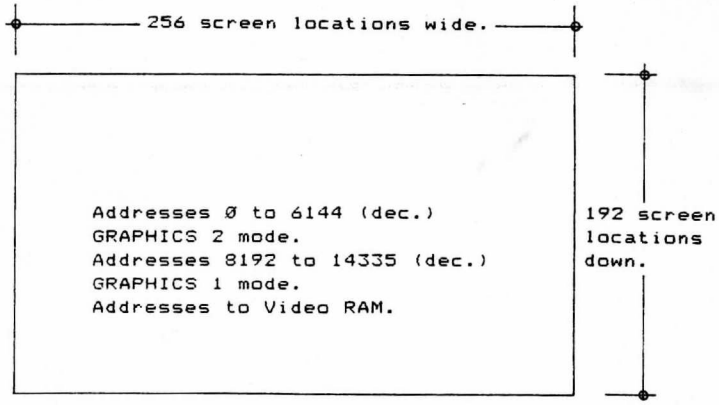
THE TEXT SCREEN



THE LOW RES. SCREEN



THE HIGH RES. SCREEN



HIGH RES. SCREEN MEMORY MAP

Color Table

The high resolution graphic screen is broken up into individual addresses as shown below, ie. the addresses start at the top left-hand corner and proceed down the screen for 8 consecutive addresses at which time they jump back up to the top of the row being addressed and begin to run consecutively down again for 8 addresses. Because the high res. screen contains a large number of addresses, the map below is not complete but the general layout can be established from it.

← 32 →

Top left-hand corner of the screen.

Top right-hand corner of the screen.

8192	8200	8208	.	.	.	.	8440
8193	8201	8209	.	.	.	.	8441
8194	8202	8210	.	.	.	.	8442
8195	8203	.	.	.	.	.	8443
8196	8204	.	.	.	.	.	8444
8197	8205	.	.	.	.	.	8445
8198	8206	.	.	.	.	.	8446
8199	8207	.	.	.	.	.	8447
8448	8456	.	.				
8449	.	.	.				
8450	..	.	.				
8451	.	.	.				
8452	..	.					
8453	.	.					
8454	.	.					
8455	.						
.	.						
.	.						
.	.						
14080	14088	.	.		.		14328
14081	14089	.	.		.		14329
14082	14090	.	.		.		14330
14083	.	.	.		.		14331
14084	.	.	.		.		14332
14085	.	.	.		.		14333
14086	.	.	.		.		14334
14087	.	.	.		.		14335

24

Bottom left-hand corner of the screen.

Bottom right-hand corner of the screen.

(0,0) → (7,0) (8,0) → (15,0)

LOW RES. SCREEN MEMORY MAP

The low resolution screen memory map shown below is not totally shown because of the large number of individual addresses. The general layout of the screen however can still be ascertained from this map.

Top left-hand corner of the screen.

Top right-hand corner of the screen.

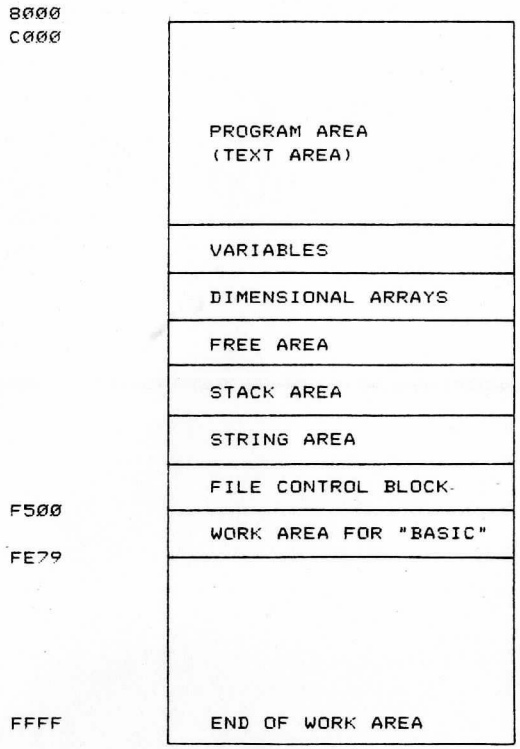
Ø	8	16	.	.	.	.	.	.	248
1	9	17	.	.	.	.	.	.	249
2	1Ø	18	.	.	.	.	.	.	25Ø
3	11	19	.	.	.	.	.	.	251
4	12	2Ø	.	.	.	.	.	.	252
5	13	21	.	.	.	.	.	.	253
6	14	22	.	.	.	.	.	.	254
7	15	23	.	.	.	.	.	.	255
256	264	.	.	.	.	.	.	.	.
257	265	.	.	.	.	.			
258	.	.	.	.	.				
259	.	.	.	.	.				
26Ø	.	.	.						
261	.	.	.						
262	.	.	.						
263	.	.							
.	.	.	.	.					
.	.	.							
.	.	.	.						
128Ø	1288	.	.	.	.	.	.	.	1528
1281	1289	.	.	.	.	.	.	.	1529
1282	129Ø	.	.	.	.	.	.	.	153Ø
1283	.	.	.	.	.	.	.	.	1531
1284	.	.	.	.	.	.	.	.	1532
1285	.	.	.	.	.	.	.	.	1533
1286	.	.	.	.	.	.	.	.	1534
1287	.	.	.	.	.	.	.	.	1535

Bottom left-hand corner of the screen.

Bottom right-hand corner of the screen.

WORKING AREA MEMORY MAP

The following chart shows the layout of the RAM when programming in BASIC. The addresses to the side of the chart indicate the start address for the various attributes. They are given in hexa-decimal numbers.



PROGRAMS

1. SUB HUNT

2. HELICOPTER

3. ENGINE

PROGRAMS

The following pages are some of the programs that have been developed either using the information in this book, or by earlier methods that were available at the time.

SUB HUNT is perhaps the only full working program. It is a game of "sink the subs" and is complete in that it has a beginning, keeps scores and has an ending.

HELICOPTER is the middle part of a program I was working on and I felt it was worthy for publication in its present form, if only to indicate the programming required if you decided to write your own game program.

ENGINE is part of an educational program written by Laurie Callender, it uses the graphic abilities of the Spectravideo to its full to give a very real representation of the workings of a 4 stroke internal combustion engine. The original program is over 20k long and covers the 2 stroke engine as well. Each part of the engines is explained and drawn individually then placed together to form a full moving model of each engine.

The program listings have been filled with spaces between individual commands on each line. This is to try and aid the keying in of these programs. The only spaces that are important in these and any other programs for the Spectravideo, are the spaces inserted between the letters and words held in quotation marks, as in PRINT statements etc.

The programs have been checked thoroughly for errors, so hopefully they will work for you.

SUB HUNT

```
10 COLOR 15, 1, 1: SCREEN 1,3: CLEAR 525
20 FOR T = 1 TO 32: READ A: G$ = G$ + CHR$(A): NEXT:  SPRITE$(12)
= G$
30 FOR T = 1 TO 32: READ A: H$ = H$ + CHR$(A): NEXT:  SPRITE$(13)
= H$
40 FOR T = 1 TO 32: READ A: I$ = I$ + CHR$(A): NEXT:  SPRITE$(14)
= I$
50 FOR T = 1 TO 32: READ A: J$ = J$ + CHR$(A): NEXT:  SPRITE$(15)
= J$
60 SPRITE$(16) = H$
70 FOR T = 1 TO 32: READ A: K$ = K$ + CHR$(A): NEXT:  SPRITE$(17)
= K$
80 FOR T = 1 TO 32: READ A: L$ = L$ + CHR$(A): NEXT:  SPRITE$(18)
= L$
90 FOR X = 1 TO 50
100 PUT SPRITE 12, (70,30), CO, 12: GOSUB 180
110 PUT SPRITE 13, (105,30), CO, 13: GOSUB 180
120 PUT SPRITE 14, (142,30), CO, 14: GOSUB 180
130 PUT SPRITE 15, (55,70), CO, 15: GOSUB 180
140 PUT SPRITE 16, (90,70), CO, 16: GOSUB 180
150 PUT SPRITE 17, (127,70), CO, 17: GOSUB 180
160 PUT SPRITE 18, (163,70), CO, 18: GOSUB 180
170 NEXT X: GOTO 250
180 CO = INT (RND (-TIME) * 13) + 2: RETURN
190 DATA 7, 31, 124, 240, 240, 252, 255, 127, 31,  3,  192,  224,
240, 255, 223, 199, 140, 236, 252, 60, 28,  12,  128,  240,  252,
254, 127, 31, 31, 254, 252, 224
200 DATA 254, 254, 124, 56, 56, 56, 56, 56, 56, 56, 56,  60,  62,
31, 31, 7, 63, 63, 30, 12, 12, 12, 12, 12, 12, 12,  12,  28,  60,
248, 240, 224
210 DATA 255, 255, 120, 56, 56, 56, 63, 63, 56, 56, 56,  56,  56,
120, 255, 255, 248, 252, 62, 30, 30, 62, 252, 248,  60,  30,  15,
15, 31, 62, 252, 248
220 DATA 254, 254, 124, 56, 56, 56, 63, 63, 56, 56, 56,  56,  56,
124, 254, 254, 127, 127, 62, 28, 28, 28, 252, 252,  28,  28,  28,
28, 28, 62, 127, 127
230 DATA 252, 252, 126, 62, 63, 63, 59, 59, 57, 57, 56,  56,  56,
124, 254, 254, 63, 63, 30, 12, 12, 12, 140, 140, 204,  204,  236,
236, 124, 124, 60, 60
240 DATA 255, 255, 243, 227, 195, 195, 3, 3, 3, 3, 3,  3,  7,
15, 15, 254, 254, 158, 142, 134, 134, 128, 128,  128,  128,  128,
128, 128, 192, 224, 224
250 COLOR 4, 15, 4: CLS: LOCATE 80,10: PRINT "* * SUB HUNT *  *":
LOCATE 20,30: PRINT "YOU ARE THE  COMMANDER  OF  FIVE  WARSHIPS":
LOCATE 20,50: PRINT "YOUR  ORDERS ARE  TO  DESTROY  AS  MANY":
LOCATE 20,70: PRINT "ENEMY  SUBMARINES AS  POSSIBLE  BEFORE"
260 LOCATE 20,90: PRINT "ALL  YOUR  DEPTH  CHARGES  HAVE  GONE."
270 L1 = 0: L2 =  256:  L3  =  192:  FOR  T  =  1  TO  100:  LINE
(L1,L1)-(L2,L3), 4, B: L1 = L1 + 1: L2 = L2 - 1: L3  =  L3  -  1:
NEXT
280 LINE (80,80)-(180,120), 15, BF: LOCATE 105,100:  PRINT  "GOOD
LUCK": FOR T = 1 TO 500: NEXT: LINE (80,80)-(180,120), 4, BF: FOR
T = 1 TO 500: NEXT
290 COLOR 15, 1, 1: SCREEN 1,2: DEFINT A-Z
300 FOR T = 1 TO 24: READ A: A$ = A$ + CHR$(A): NEXT:  SPRITE$(1)
```

```
= A$: FOR T = 1 TO 24: READ A: B$ = B$ + CHR$(A): NEXT:
SPRITE$(2) = B$
310 FOR T = 1 TO 24: READ A: C$ = C$ + CHR$(A): NEXT: SPRITE$(3)
= C$: FOR T = 1 TO 24: READ A: D$ = D$ + CHR$(A): NEXT:
SPRITE$(4) = D$
320 FOR T = 1 TO 8: READ A: E$ = E$ + CHR$(A): NEXT: SPRITE$(9) =
E$: FOR T = 1 TO 32: READ A: F$ = F$ + CHR$(A): NEXT: SPRITE$(10)
= F$
330 DATA 0, 1, 1, 3, 251, 127, 63, 31, 0, 0, 0, 0, 0, 0, 0, 0, 0,
128, 192, 192, 224, 255, 255, 254
340 DATA 0, 0, 0, 0, 0, 0, 255, 127, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 192, 240, 240, 255, 255
350 DATA 0, 0, 0, 3, 15, 15, 255, 127, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 255, 254
360 DATA 0, 1, 3, 3, 7, 31, 255, 127, 0, 0, 0, 0, 0, 0, 0, 0, 0,
128, 128, 192, 223, 254, 252, 248
370 DATA 0, 0, 0, 248, 112, 80, 112, 248
380 DATA 21, 10, 102, 22, 73, 71, 51, 207, 59, 19, 229, 11, 50,
70, 24, 32, 16, 144, 180, 68, 217, 119, 152, 167, 184, 68, 114,
74, 168, 166, 145, 136
390 SPRITE$(5) = B$: SPRITE$(6) = C$: SPRITE$(7) = C$: SPRITE$(8)
= B$: SPRITE$(20) = A$: SPRITE$(21) = A$: SPRITE$(22) = A$:
SPRITE$(23) = A$: CLEAR
400 CLS: LF = 0: SP = 24: CH = 20: DC = 76: LINE (10,5)-(69,20),
8, BF: LINE (10,20)-(69,35), 12, BF: LINE (10,35)-(69,50), 4, BF:
LINE (186,5)-(246,18), 13, BF: LINE (186,18)-(246,50), 15, BF:
LINE (72,5)-(183,35), 6, BF: LINE (186,28)-(246,38), 3, BF
410 LINE (72,40)-(183,50), 11, BF: LINE (0,55)-(256,80), 7, BF:
LINE (0,80)-(256,192), 4, BF
420 DRAW "C1 BM 0,135 R2 D2 R4 D2 R2 D2 R4 D6 R4 D2 R4 D2 R2 D6
R4 D6 P2 D2 R2 D2 R6 D10 R2 D4 R2 D2 R4 D2 R10 U2 R8 D6 R6 U2 R6
U4 R2 U2 R2 U2 R6 U4 R6 U2 R8 D4 R6 D2 R6 D2 R10 U4 R10 U2 R10 U4
R10 U2 R2 U2 R2 U2 R6 U2 R4 U6 R6 D2 R4 U6 R6 U4 R6 U6 R4 U2 R8
U2 R6 D2 R6 D8 R1 D6 R4 D2 R4 D6 R6 D2 R4 D2 R4 D2 R6 D6 R6 U2 R2
R2 R8"
430 PAINT (127,191), 1
440 DRAW "C15 BM 79,17 D2 F2 R4 F1 G1 L6 D2 R8 E2 U2 H2 L4 H1 E1
R6 U2 L8 G2 BM 93,15 D8 F2 R6 E2 U8 L2 D6 G2 L2 H2 U6 L2 BM
107,15 R8 F2 G2 F2 D2 G2 L8 U10 BM 109,17 R4 F1 G1 L4 U2 BM
109,21 R4 F1 G1 L4 U2 BM 127,15 D10 R2 U4 R6 D4 R2 U10 L2 D4 L6
U4 L2 BM 141,15 D8 F2 R6 E2 U8 L2 D6 G2 L2 H2 U6 L2"
450 DRAW "C15 BM 155,17 D8 R2 U6 E2 R2 F2 D6 R2 U8 H2 L6 G2 BM
169,15 D2 R4 D8 R2 U8 R4 U2 L10": PAINT (84,20), 15: PAINT
(97,24), 15: PAINT (111,20), 15: PAINT (132,20), 15: PAINT
(146,24), 15: PAINT (159,16), 15: PAINT (174,24), 15
460 LOCATE 18,8: COLOR 15: PRINT "CHARGES": LOCATE 16,24: PRINT
"LEFT": LOCATE 16,41: PRINT "USED": LOCATE 45,24: PRINT CH:
LOCATE 40,41: PRINT LF
470 IF SC > HS THEN HS = SC
480 SC = 0: LOCATE 200,8: PRINT "SCORE": LOCATE 195,20: COLOR 1:
PRINT SC: LOCATE 190,30: PRINT "H. SCORE": LOCATE 195,42: PRINT
HS
490 FOR X = 246 TO 127 STEP -.2: PUTSPRITE 1, (X,71), 1, 1: NEXT
500 PUT SPRITE 20, (90,40), 1, 20: FOR T = 1 TO 500: NEXT: PUT
SPRITE 21, (110,40), 1, 21: FOR T = 1 TO 500: NEXT: PUT SPRITE
22, (130,40), 1, 22: FOR T = 1 TO 500: NEXT: PUT SPRITE 23,
(150,40), 1, 23: FOR T = 1 TO 500: NEXT
510 G = STICK (0) OR STICK (1): H = STRIG (0) OR STRIG (1)
520 DC = DC + 0: PUT SPRITE 9, (X,DC), CL, 9: IF DC > 170 THEN
GOSUB 820
```

```
530 F = F - 8: PUT SPRITE 2, (F,165), 15, 2: IF F < 0 THEN F =
257
540 B = B - 5: PUT SPRITE 5, (B,145), 6, 5: IF B < 0 THEN B = 257
550 DC = DC + O: PUT SPRITE 9, (X,DC), CL, 9: IF DC > 170 THEN
GOSUB 820
560 Z = Z + 6: PUT SPRITE 6, (Z,125), 1, 6: IF Z > 257 THEN Z = 0
570 Q = Q + 2: PUT SPRITE 7, (Q,105), 2, 7: IF Q > 257 THEN Q = 0
580 E = E - 2: PUT SPRITE 8, (E,85), 10, 8: IF E < 0 THEN E = 257
590 DC = DC + O: PUT SPRITE 9, (X,DC), CL, 9: IF DC > 170 THEN
GOSUB 820
600 ON G GOSUB 630, 630, 640, 630, 630, 630, 660, 630
610 IF H THEN GOSUB 690
620 GOTO 510
630 RETURN
640 PUT SPRITE 1, (0,0), 0, 0: X = X + 3: IF X > 225 THEN X = X -
3
650 PUT SPRITE 4, (X,71), 1, 4: RETURN
660 PUT SPRITE 4, (0,0), 0, 0: X = X - 3: IF X < 20 THEN X = X +
3
670 PUT SPRITE 1, (X,71), 1, 1: RETURN
680 GOTO 510
690 SPRITE ON: O = 4: CL = 15: ON SPRITE GOSUB 700: RETURN
700 SPRITE OFF: IF DC < 100 AND DC > 80 THEN GOTO 750
710 IF DC < 120 AND DC > 100 THEN GOTO 760
720 IF DC < 140 AND DC > 120 THEN GOTO 770
730 IF DC < 160 AND DC > 140 THEN GOTO 780
740 IF DC < 170 AND DC > 160 THEN GOTO 790
750 SC = SC + 1: E = 257: PUT SPRITE 8, (E,85), 10, 8: GOTO 800
760 SC = SC + 2: Q = 0: PUT SPRITE 7, (Q,105), 2, 7: GOTO 800
770 SC = SC + 5: Z = 0: PUT SPRITE 6, (Z,125), 1, 6: GOTO 800
780 SC = SC + 10: B = 257: PUT SPRITE 5, (B,145), 6, 5: GOTO 800
790 SC = SC + 20: F = 257: PUT SPRITE 2, (F,165), 15, 2: GOTO 800
800 SOUND 0,0: SOUND 6,30: SOUND 7,0: SOUND 8,16: SOUND 9,16:
SOUND 10,16: SOUND 11,0: SOUND 12,5: SOUND 13,0: SOUND 12,26
810 PUT SPRITE 9, (0,0), 0, 0: PUT SPRITE 10, (X,DC), 9, 10: FOR
T = 1 TO 50: NEXT: PUT SPRITE 10, (0,0), 0, 0: LINE
(200,18)-(246,27), 15, BF: LOCATE 195,20: PRINT SC
820 O = 4: CL = 4: LINE (45,24)-(65,34), 12, BF: LINE
(40,41)-(62,49), 4, BF: CH = CH - 1: LF = LF + 1: IF CH = 0 THEN
GOTO 830 ELSE LOCATE 45,24: COLOR 15: PRINT CH: LOCATE 40,41:
PRINT LF: COLOR 1: PUT SPRITE 9, (X,DC+3), CL, 9: O = 0: DC = 76:
RETURN
830 CH = 20: SP = SP - 1: IF SP =19 THEN GOTO 840 ELSE PUT SPRITE
1, (0,0), 0, 0: PUT SPRITE 4, (0,0), 0, 0: PUT SPRITE 9, (0,0),
0, 0: FOR TI = 1 TO 500: NEXT: PUT SPRITE SP, (0,0), 0, 0: FOR TI
= 1 TO 500: NEXT: FOR T = 246 TO X STEP -.2: PUT SPRITE 1,
(T,71), 1, 1: NEXT: GOTO 510
840 COLOR 15: CLS: LOCATE 40,80: PRINT "G": PUT SPRITE 10,
(35,75), 6, 10: GOSUB 880: LOCATE 60,80: PRINT "A": PUT SPRITE
10, (55,75), 6, 10: GOSUB 880: LOCATE 80,80: PRINT "M": PUT
SPRITE 10, (75,75), 6, 10: GOSUB 880
850 LOCATE 100,80: PRINT "E": PUT SPRITE 10, (95,75), 6, 10:
GOSUB 880: LOCATE 140,80: PRINT "O": PUT SPRITE 10, (135,75), 6,
10: GOSUB 880: LOCATE 160,80: PRINT "V": PUT SPRITE 10, (155,75),
6, 10:GOSUB 880
860 LOCATE 180,80: PRINT "E": PUT SPRITE 10, (175,75), 6, 10:
GOSUB 880: LOCATE 200,80: PRINT "R": PUT SPRITE 10, (195,75), 6,
10: GOSUB 880
870 FOR X = 1 TO 2000: NEXT: GOTO 400
```

```
880 SOUND 0,0: SOUND 6,30: SOUND 7,0: SOUND 9,16: SOUND 10,16:
SOUND 11,0: SOUND 12,5: SOUND 13,0: SOUND 12,26
890 FOR T = 1 TO 100: NEXT: PUTSPRITE 10, (0,0), 0, 0: FOR T = 1
TO 200: NEXT: RETURN
900 SOUND 0,0: SOUND 6,30: SOUND 7,0: SOUND 8,16: SOUND 9,16:
SOUND 10,16: SOUND 11,0: SOUND 12,5: SOUND 13,0: SOUND 12,26
```

HELICOPTER 2

```
10 COLOR 12, 5, 1: SCREEN 1,2: DEFINT A-Z: GOSUB 730
20 A = 0: TI = 0: X = 15: Y = 99: LINE (0,110)-(256,192), 1, BF
30 LINE (0,110)-(256,110), 12
40 DRAW "c12 bm 0,50 r2 d2 r4 d2 r2 d2 r4 d6 r4 d2 r4 d2 r2 d6 r4
d6 r2 d2 r2 d2 r6 d10 r2 d4 r2 r2 r10 u2 r8 d6 r6 u2 r6 u4 r2 u2
r2 u2 r6 u4 r6 u2 r8 d4 r6 d2 r10 u4 r10 u2 r10 u4 r10 u2 r2 u2
r2 u2 r6 u2 r6 u2 r4 u6 r6 d2 r4 u6 r6 u4 r6 u6 r4 u2 r8 u2 r6 d2
r6 d8 r1 d6 r4 d2 r4 d6 r6 d2 r4 d2 r6 d6 r6 u2 r12 u2 r4 u2 r5
d2 r4 u6 r4"
50 PAINT (127,109), 12
60 PUT SPRITE 1, (X,Y), 15, 1
70 TI = TI + 1: IF TI = 5000 THEN PUT SPRITE 1, (0,0), 0, 0: PUT
SPRITE 2, (0,0), 0, 0: PUT SPRITE 3, (0,0), 0: PUT SPRITE 4,
(0,0), 0, 0: GOTO 20
80 S = STICK (0) OR STICK (1): F = STRIG (0) OR STRIG (1)
90 D = D + O: PUT SPRITE 4, (X+3,D), CL, 4: IF POINT (X+3,D+3) =
1 THEN GOSUB 620
100 IF D > 180 THEN D = Y: O = 0: CL = 0
110 ON S GOSUB 140, 200, 260, 320, 380, 440, 500, 560
120 IF F THEN O = 8: CL = 15
130 GOTO 70
140 Y = Y - 4: PUT SPRITE 2, (0,0), 0, 0: PUT SPRITE 3, (0,0), 0,
0: PUT SPRITE 1, (X,Y), 15, 1
150 IF POINT (X+2,Y+1) = 1 THEN GOSUB 650
160 IF POINT (X+13,Y+1) = 1 THEN GOSUB 650
170 IF POINT (X+8,Y+10) = 1 THEN GOSUB 650
180 IF Y < 5 THEN Y = Y + 4
190 RETURN
200 X = X + 4: Y = Y - 4: PUT SPRITE 1, (0,0), 0, 0: PUT SPRITE
3, (0,0), 0, 0: PUT SPRITE 2, (X,Y), 15, 2
210 IF POINT (X+2,Y+1) = 1 THEN GOSUB 650
220 IF POINT (X+13,Y+1) = 1 THEN GOSUB 650
230 IF POINT (X+8,Y+10) = 1 THEN GOSUB 650
240 IF X > 230 OR Y < 5 THEN X = X - 4: Y = Y + 4
250 RETURN
260 X = X + 4: PUT SPRITE 1, (0,0), 0, 0: PUT SPRITE 3, (0,0), 0,
0: PUT SPRITE 2, (X,Y), 15, 2
270 IF POINT (X,Y+7) = 1 THEN GOSUB 650
280 IF POINT (X+15,Y+1) = 1 THEN GOSUB 650
290 IF POINT (X+12,Y+10) = 1 THEN GOSUB 650
300 IF X > 230 THEN X = X - 4
310 RETURN
320 X = X + 4: Y = Y + 4: PUT SPRITE 1, (0,0), 0, 0: PUT SPRITE
3, (0,0), 0, 0: PUT SPRITE 2, (X,Y), 15, 2
330 IF POINT (X,Y+7) = 1 THEN GOSUB 650
```

```
340 IF POINT (X+15,Y+1) = 1 THEN GOSUB 650
350 IF POINT (X+12,Y+10) = 1 THEN GOSUB 650
360 IF X > 230 OR Y > 190 THEN X = X - 4: Y = Y - 4
370 RETURN
380 Y = Y + 4: PUT SPRITE 2, (0,0), 0, 0: PUT SPRITE 3, (0,0), 0,
0: PUT SPRITE 1, (X,Y), 15, 1
390 IF POINT (X+2,Y+1) = 1 THEN GOSUB 650
400 IF POINT (X+13,Y+1) = 1 THEN GOSUB 650
410 IF POINT (X+8,Y+10) = 1 THEN GOSUB 650
420 IF Y > 190 THEN Y = Y - 4
430 RETURN
440 X = X - 4: Y = Y + 4: PUT SPRITE 2, (0,0), 0, 0:  PUT  SPRITE
1, (0,0), 0, 0: PUT SPRITE 3, (X,Y), 15, 3
450 IF POINT (X+2,Y+1) = 1 THEN GOSUB 650
460 IF POINT (X+13,Y+1) = 1 THEN GOSUB 650
470 IF POINT (X+8,Y+10) = 1 THEN GOSUB 650
480 IF X < 10 OR Y > 190 THEN X = X + 4: Y = Y - 4
490 RETURN
500 X = X - 4: PUT SPRITE 1, (0,0), 0, 0: PUT SPRITE 2, (0,0), 0,
0: PUT SPRITE 3, (X,Y), 15, 3
510 IF POINT (X+1,Y+1) = 1 THEN GOSUB 650
520 IF POINT (X+16,Y+7) = 1 THEN GOSUB 650
530 IF POINT (X+5,Y+10) = 1 THEN GOSUB 650
540 IF X < 10 THEN X = X + 4
550 RETURN
560 X = X - 4: Y = Y - 4: PUT SPRITE 1, (0,0), 0, 0:  PUT  SPRITE
2, (0,0), 0, 0: PUT SPRITE 3, (X,Y), 15, 3
570 IF POINT (X+2,Y+1) = 1 THEN GOSUB 650
580 IF POINT (X+13,Y+1) = 1 THEN GOSUB 650
590 IF POINT (X+8,Y+10) = 1 THEN GOSUB 650
600 IF X < 10 OR Y < 5 THEN X = X + 4: Y = Y + 4
610 RETURN
620 COLOR 12: LOCATE X,D: PRINT "■": O = 0: CL = 0
630 GOSUB 710
640 PUT SPRITE 5, (X+3,D+5), 6, 5: FOR T = 1 TO  100:  NEXT:  PUT
SPRITE 5, (0,0), 0, 0: D = Y: RETURN
650 PUT SPRITE 1, (X,Y), 15, 1: GOSUB 680: FOR  T  =  1  TO  100:
NEXT: PUT SPRITE 1, (0,0), 0, 0
660 PUT SPRITE 2, (X,Y), 15, 2: GOSUB 680: FOR  T  =  1  TO  100:
NEXT: PUT SPRITE 2, (0,0), 0, 0
670 PUT SPRITE 3, (X,Y), 15, 3: GOSUB 680: FOR  T  =  1  TO  100:
NEXT: PUT SPRITE 3, (0,0), 0, 0: GOTO 650
680 A = A + 1: IF A <> 15 THEN RETURN
690 A = 0: FOR T = 1 TO 4: PUT SPRITE (T), (0,0), 0, 0: NEXTT
700 FOR T = 1 TO 5: PUT SPRITE 5, (X,Y), 6, 5: FOR X = 1 TO  200:
NEXT: PUT SPRITE 5, (0,0), 0, 0: GOSUB 710: GOTO 20
710 SOUND 0,0: SOUND 6,30: SOUND 7,0:  SOUND  8,16:  SOUND  9,16:
SOUND 10,16: SOUND 11,0: SOUND 12,5: SOUND 13,0: SOUND 12,26
720 RETURN
730 FOR T = 1 TO 32: READ A: A$ = A$ + CHR$(A): NEXT: SPRITE$ (1)
= A$
740 DATA 0, 63, 1, 3, 4, 8, 8, 8, 15, 7, 0, 0, 0,  0,  0,  0,  0,
248, 0, 128, 64, 32, 32, 32, 224, 192, 0, 0, 0, 0, 0, 0
750 FOR T = 1 TO 32: READ A: B$ = B$ + CHR$(A): NEXT: SPRITE$ (2)
= B$
760 DATA 0, 7, 0, 0, 0, 0, 67, 95, 255, 64, 64, 0, 0, 0, 0, 0, 0,
255, 32, 112, 200, 196, 226, 226, 242, 124, 0, 0, 0, 0, 0, 0
770 FOR T = 1 TO 32: READ A: C$ = C$ + CHR$(A): NEXT: SPRITE$ (3)
= C$
```

```
780 DATA 0, 255, 4, 14, 19, 35, 71, 71, 79, 62, 0, 0, 0, 0, 0, 0,
0, 224, 0, 0, 0, 0, 194, 250, 255, 2, 2, 0, 0, 0, 0, 0
790 FOR T = 1 TO 8: READ A: D$ = D$ + CHR$(A): NEXT: SPRITE$ (4)
= D$
800 DATA 0, 0, 0, 24, 24, 0, 0, 0
810 FOR T = 1 TO 32: READ A: E$ = E$ + CHR$(A): NEXT: SPRITE$ (5)
= E$
820 DATA 21, 10, 102, 22, 73, 71, 51, 207, 59, 19, 229, 11, 50,
70, 24, 32, 16, 144, 180, 68, 217, 119, 152, 167, 184, 68, 114,
74, 168, 166, 145, 136
830 RETURN
```

ENGINE

```
10 REM  BAISC ENGINE DESIGN
20 SCREEN 1,3 : CLS
30 COLOR 15,4,4 : DIM B(31)
40 FOR A = 0 TO 13
50 FOR C = 0 TO 31 : READ W$ : S$ = S$ + CHR$ (VAL(W$)) : NEXT C
: SPRITE$ (A) = S$ : S$ = ""
60 NEXT A
70 GOTO 150
80 PUT SPRITE 1,(192,24),1,12 : RETURN
90 PUT SPRITE 2,(174,22),1,13 : RETURN
100 PUT SPRITE 1,(196,20),1,12 : RETURN
110 PUT SPRITE 2,(170,18),1,13 : RETURN
120 REM spark plug fire routine
130 SOUND 6,15 : SOUND 7,7 : SOUND 8,16 : SOUND 9,16 : SOUND
10,16 : SOUND 12,16 : SOUND13,0
140 FOR Z = 14 TO 0 STEP - 2 : CIRCLE (200,40),4,Z : CIRCLE
(200,40),6,Z,3.14,6.28 : NEXT : RETURN
150 LOCATE 16,0 : PRINT CHR$ (34) ; "OTTO" ; CHR$ (34) ; " 4
CYCLE ENGINE" : LOCATE 16,8 : PRINT "~~~~~~~~~~~~~~~~~~~~~~"
160 REM CRANKCASE
170 A$ = "BM182,104C9G8D40F8R35E8U40H8" : B$ =
"BM182,104C9U60BL8D60G8D40F16R35E16U40H8U60BL8D60"
180 C$ = "BM174,44C9U10E10R31F10D10"
190 DRAW A$ : DRAW B$ : DRAW C$
200 CIRCLE (200,46),17,9,6.28,3.14
210 PAINT (200,167),9
220 LINE (175,128) - (156,136),9,BF : LINE (175,132) -
(156,132),1 : LINE (224,128) - (244,136),9,BF : LINE (224,132) -
(244,132),1 : LINE (158,124) - (161,127),14,BF : LINE (158,137) -
(161,140),14,BF
230 LINE (242,124) - (239,127),14,BF : LINE (242,137) -
(239,140),14,BF
240 REM INLET MANIFOLD
250 CIRCLE (180,35),10,5,6.28,2 : CIRCLE (176,41),8,5,6.28,2 :
LINE (190,34) - (184,39),5 : LINE (177,24) - (160,24),5 : LINE
(173,32) - (160,32),5 : LINE (160,24) - (160,32),5 : PAINT
(190,33),5 : LOCATE 155,8 : COLOR 5 :PRINT "INLET" : COLOR 15
260 REM EXHAUST MANIFOLD
```

```
270 LINE (216,40) - (224,32),8 : LINE (208,32) - (220,24),8 :
LINE (216,40) - (208,32),8 : LINE (224,32) - (240,32),8 : LINE
(220,24) - (240,24),8 : LINE (240,32) - (240,24),8 : PAINT
(230,30),8 : LOCATE 216,8 : COLOR 8 : PRINT "EXHAUST" : COLOR 15
280 REM SPARK PLUG
290 LINE (190,19) - (210,23),10,BF : LINE (198,23) -
(202,32),10,BF : LINE (200,30) - (200,32),15 : LINE (202,30) -
(202,34),10 : LINE (202,34) - (199,34),10
300 LINE (198,8) - (202,12),15,BF : LINE (196,12) -
(204,15),15,BF : LINE (194,15) - (206,19),15,BF : LINE (199,2) -
(201,8),10,BF : LINE (194,19) - (194,23),5 : LINE (206,19) -
(206,23),5
310 PUT SPRITE 1,(196,20),1,12 : PUT SPRITE 2,(170,18),1,13
320 CIRCLE (199,130),23,15,,,1.3 : PAINT (199,130),15 : CIRCLE
(199,130),4,14 : PAINT (199,130),14
330 FOR Q = 1 TO 31 : READ B(Q) : NEXT
340 Q = 1 : B = 0
350 FOR A = 1.6 TO 14.1 STEP .1
360 R = SIN (A) : B = B + 1
370 X1 = R * COS (A) : Y1 = R * SIN (A)
380 X = INT (184 - 20 * X1) : Y = INT (128 - 20 * Y1 * 1.4)
390 IF B = 1 THEN GOSUB 90 ELSE IF B = 45 THEN GOSUB 80 ELSE IF B
= 20 THEN GOSUB 110 ELSE IF B = 66 THEN GOSUB 100
400 IF B = 62 THEN GOSUB 90 ELSE IF B = 109 THEN GOSUB 80 ELSE IF
B = 129 THEN GOSUB 100 ELSE IF B = 82 THEN GOSUB 110
410 IF B = 126 THEN GOSUB 90 ELSE IF B = 34 THEN GOSUB 130 ELSE
IF B = 96 THEN GOSUB 130
420 VPOKE 14376,1 : VPOKE 14377,1 : VPOKE 14392,128 : VPOKE
14393,128
430 PUT SPRITE 4,(X,Y),1,1
440 PUT SPRITE 5,(184,Y-57),10,0
450 PUT SPRITE 3,(184,Y-59),1,11
460 PUT SPRITE 6,(185,76),1,B(Q) : Q = Q + 1 : IF Q > 31 THEN Q =
1
470 VPOKE 14375,1 : VPOKE 14378,1 : VPOKE 14391,128 : VPOKE
14394,128 : VPOKE 14376,3 : VPOKE 14377,3 : VPOKE 14392,192 :
VPOKE 14393,192
480 NEXT A
490 GOSUB 100 : GOTO 340
500 REM sprite 0 piston
510 DATA 63, 127, 255, 255, 255, 255, 255, 252, 248, 248, 248,
252, 255, 255, 255, 255, 252, 254, 255, 255, 255, 255, 255, 63,
31, 31, 31, 63, 255, 255, 255, 255
520 REM sprite 1 big end hole
530 DATA 0, 0, 0, 0, 0, 0, 1, 3, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 128, 192, 192, 128, 0, 0, 0, 0, 0, 0
540 REM sprite 2 con rod straight
550 DATA 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 128,
128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128, 128,
128, 0
560 REM sprite 3 con rod right 1
570 DATA 3, 3, 3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 0, 128,
128, 128, 128, 128, 128, 128, 192, 192, 192, 192, 192, 192, 192,
192, 0
580 REM sprite 4 con rod right 3
590 DATA 3, 3, 3, 3, 3, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 128,
128, 128, 128, 192, 192, 192, 192, 224, 224, 224, 224, 112, 112,
112, 16
600 REM sprite 5 con rod left 3
```

```
610 DATA 3, 3, 3, 3, 7, 7, 7, 7, 14, 14, 14, 14, 28, 28, 28, 16,
128, 128, 128, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
620 REM sprite 6 con rod left 2
630 DATA 3, 3, 3, 3, 3, 3, 7, 7, 7, 7, 7, 14, 14, 14, 14, 0, 128,
128, 128, 128, 128, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
640 REM sprite 7 con rod left 1
650 DATA 3, 3, 3, 3, 3, 3, 3, 7, 7, 7, 7, 7, 7, 7, 7, 0, 128,
128, 128, 128, 128, 128, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0
660 REM sprite 8 con rod right 2
670 DATA 3, 3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 128,
128, 128, 128, 128, 128, 128, 192, 192, 192, 192, 192, 224, 224, 224,
224, 0
680 REM sprite 9 con rod right 4
690 DATA 3, 3, 3, 3, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 128,
128, 128, 128, 192, 192, 192, 224, 224, 224, 112, 112, 112, 56,
56, 56, 24
700 REM sprite 10 con rod left 4
710 DATA 3, 3, 3, 3, 7, 7, 7, 14, 14, 14, 28, 28, 28, 56, 56, 48,
128, 128, 128, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
720 REM sprite 11 piston rings
730 DATA 0, 0, 0, 255, 0, 255, 0, 3, 7, 7, 7, 3, 0, 0, 0, 0, 0,
0, 0, 255, 0, 255, 0, 192, 224, 224, 224, 192, 0, 0, 0, 0
740 REM sprite 12 right valve
750 DATA 0, 0, 0, 0, 0, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 4,
8, 16, 32, 192, 192, 96, 32, 0, 0, 0, 0, 0, 0
760 REM sprite 13 left valve
770 DATA 128, 64, 32, 16, 8, 4, 2, 1, 1, 3, 2, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 96, 192, 128, 0, 0, 0, 0, 0, 0
780 REM con rod sprite No. data
790 DATA 2, 2, 3, 3, 8, 8, 4, 4, 9, 9, 4, 4, 8, 8, 3, 3, 2, 2, 7,
6, 6, 5, 5, 10, 10, 5, 5, 6, 6, 7, 7
```

APPENDIX A:

THE ASCII CODE TABLE

AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE

DECIMAL	OCTAL	HEX.	ASCII CHARACTER OR CONTROL COMMAND	
1	1	1	CONTROL A	
2	2	2	CONTROL B	Move cursor to start of word.
3	3	3	CONTROL C	Break.
4	4	4	CONTROL D	
5	5	5	CONTROL E	Clear line from cursor position.
6	6	6	CONTROL F	Move cursor next word.
7	7	7	CONTROL G	Beep.
8	10	8	CONTROL H	Backspace and delete character.
9	11	9	CONTROL I	Tab 8 spaces.
10	12	A	CONTROL J	
11	13	B	CONTROL K	Home cursor.
12	14	C	CONTROL L	Clear screen.
13	15	D	CONTROL M	Carriage return.
14	16	E	CONTROL N	Move cursor to end of line.
15	17	F	CONTROL O	
16	20	10	CONTROL P	
17	21	11	CONTROL Q	
18	22	12	CONTROL R	Toggles insert cursor.
19	23	13	CONTROL S	
20	24	14	CONTROL T	
21	25	15	CONTROL U	Clear line.
22	26	16	CONTROL V	
23	27	17	CONTROL W	
24	30	18	CONTROL X	
25	31	19	CONTROL Y	
26	32	1A	CONTROL Z	
27	33	1B	ESCAPE	
28	34	1C	CONTROL \	Move cursor to the right.
29	35	1D	CONTROL]	Move cursor to the left.
30	36	1E	CONTROL SHIFT 6	Move the cursor up.
31	37	1F	CONTROL _	Move the cursor down.
32	40	20	SPACE	
33	41	21	!	
34	42	22	"	
35	43	23	#	
36	44	24	$	
37	45	25	%	
38	46	26	&	
39	47	27	'	
40	50	28	(	
41	51	29	)	
42	52	2A	*	
43	53	2B	+	

DECIMAL	OCTAL	HEX.	ASCII
44	54	2C	,
45	55	2D	-
46	56	2E	.
47	57	2F	/
48	60	30	0
49	61	31	1
50	62	32	2
51	63	33	3
52	64	34	4
53	65	35	5
54	66	36	6
55	67	37	7
56	70	38	8
57	71	39	9
58	72	3A	:
59	73	3B	;
60	74	3C	<
61	75	3D	=
62	76	3E	>
63	77	3F	?
64	100	40	@
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4A	J
75	113	4B	K
76	114	4C	L
77	115	4D	M
78	116	4E	N
79	117	4F	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y
90	132	5A	Z
91	133	5B	[
92	134	5C	\
93	135	5D	]
94	136	5E	^
95	137	5F	_
96	140	60	\
97	141	61	a
98	142	62	b
99	143	63	c
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6A	j
107	153	6B	k
108	154	6C	l
109	155	6D	m
110	156	6E	n
111	157	6F	o
112	160	70	p
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7A	z
123	173	7B	{
124	174	7C	:
125	175	7D	}
126	176	7E	~
127	177	7F	DELETE

INDEX