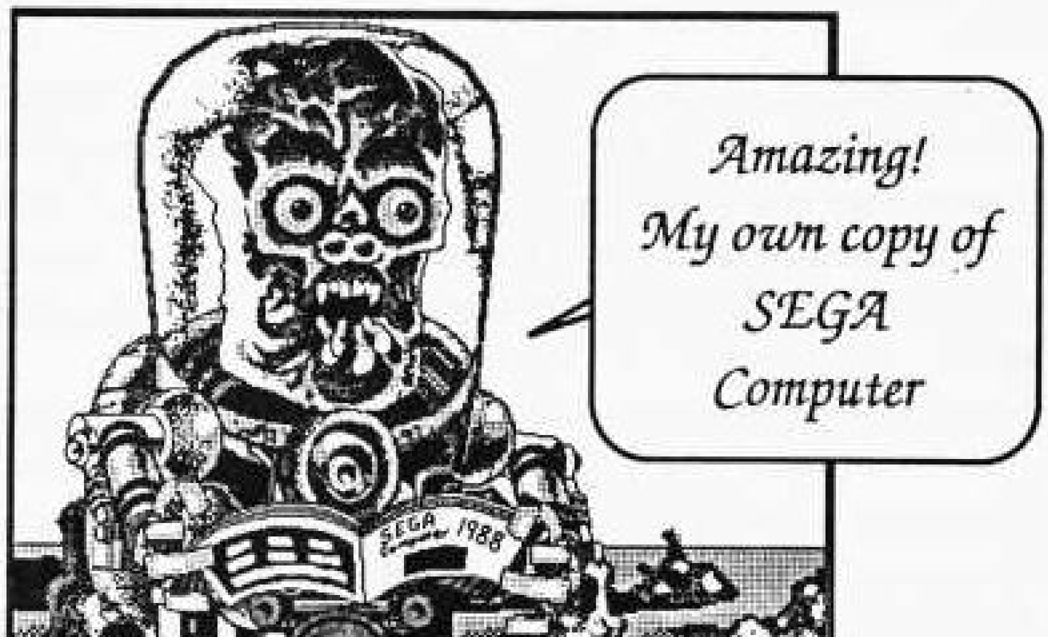
SEGA® Computer

March/April Issue 1987

The official magazine of the SEGA User Club of New Zealand



Published by MJH Software in association with Poseidon Software

Published bi-monthly by **MJH Software**
36 Colum Place Bucklands Beach
Auckland New Zealand
Telephone (09) 534-3379

in association with **Poseidon Software**
FREEPOST 243 P.O. Box 277
Tokoroa New Zealand
Telephone (0814) 67-105

- All contributions are welcome, but please include your name, address and telephone number.
- A question and answer page in the form of **Letters To The Editor** is provided and we will do our best to answer any questions about software or programming.
- It is preferable that programs be submitted on tape or disk in a listable form. (No copyright protection please). A listing is useful but don't worry if you aren't lucky enough to own a printer. Where required please include instructions on how to type in the program.
- Please check your programs thoroughly for errors and spelling mistakes before sending it to us. Please send updates if any errors are discovered, so we can publish corrections.
- All software programs received by the magazine becomes the property of **MJH Software** unless by prior arrangement. They are accepted on the basis that they are the original work of the author or required modification to run on a SEGA.
- All contributions are subject to approval by the editor and may be edited to suit the magazine style. Submitted programs will be returned on request
- Each issue two prizes of NZ$40 and NZ$20 for the two feature programs are awarded in the following categories:
  Category 1 - Games. Judged on playability and use of graphics and sound.
  Category 2 - (i) Utilities. Judged on usefulness.
  (ii) Reviews. Judged on overall presentation.

SEGA USER CLUB

MEMBERSHIP YEAR

Oct 1987 - Oct 1988

You automatically become a member of the club when you subscribe to the magazine and this qualifies you for special club benefits.

New Zealand Subscription:	NZ$25 incl GST
Australia Subscription:	A$26 Airmail

See inside cover at the back

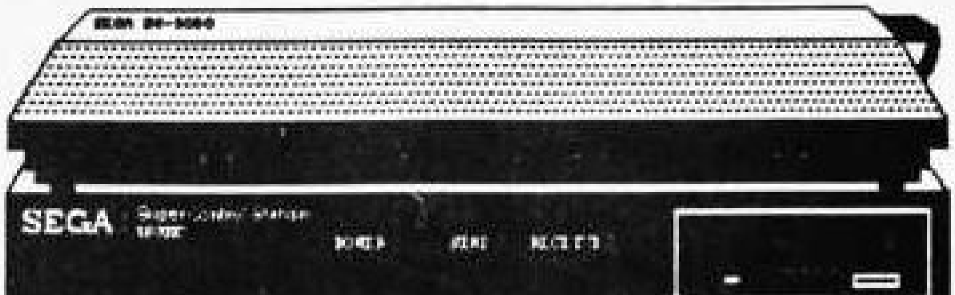
Welcome to the new look SEGA® Computer

The official magazine of the SEGA User Club of New Zealand

Issue 3

Contents

Cover illustration : Something I made up in a hurry, because I ran out of my-card covers.



Letters to the Editor

• This question and answer page is provided to help you. So send me some questions. Remember that you can ask about software or programming.

NOTES from the Editor

Please send tapes rather than listings of your programs as some of them are very very long and I don't have enough time to type the whole lot in just to see if I will put it in the magazine!

The high score table is no longer in existence, so sorry to all those people who sent in photographs of the screen. However Andrew Kerr reached level 6 of "Droll" and sent in the messages for completing each level ...

Level 1 - "Congratulations! Being a hero isn't easy. But your troubles are only beginning."
Level 2 - "Incredible! You've won the battle. But the war isn't over yet."
Level 3 - "Totally wow! Your skills are amazing. But your days may be numbered."
Level 4 - "Super duper! Few have travelled this far. Is this the end of the line for you?"
Level 5 - "Victorious! And they all lived happily ever after or did they?"

Servicing of SEGA Computers

If you know of a company or private person who is servicing SEGA Computer's could you send in a contact name and address so that we can inform all our members.

At present Andrew at Poseidon Software repairs computers at $15 per hour. So you can send your computer to Andrew care of Poseidon Software, if it can't be service anywhere else.

Lately I have also been servicing computers in Auckland so give me a call if your computer packs up and I'll have a look at it ... no guarantees that I can fix it though.

Continued on next page

Circuit Diagrams

I have copies of the Main Computer and Disk Drive circuit diagrams which are readable, so give me a call if you want a copy. If you have a good quality copy then please send a copy to me!

Dear Editor,
(a) What has happened regarding the game "Gun Fury", it was to take up the side of one disk. Is it still being worked on or has it been scrapped?
(b) Is there a tape to disk transfer program available for transferring games on tape to disk?
Rodney Jerram, Tauranga
Editors reply,
(a) Geoff at Poseidon Software has a half finished version of the game, but the writer of "Gun Fury" has now sold is SEGA and therefore will not finish it.
(b) Sorry to say there is nothing that while transfer tape games to disk. I have converted quite a few games to disk, by including a copy of Cartridge Basic on the disk (so you don't have to change any ROM calls) and modifying the loader to load off disk. I can't really explain how to do this in the magazine (it's just like explaining how to pirate software), but I might be able to arrange a way in which your cassette programs can be upgraded to disk versions.

Dear Editor,
(a) Can you save screens on disk
(b) Can you do a section explaining the third and fourth screens of the SEGA
Jason Fanning, Invercargill
Editors reply,
(a) There is only one easy way to save screens on disk. Saving a screen involves moving the screen data from VRAM to RAM and then using a SAVEM command to save the data to disk. To load a screen, you would use a LOADM command and then move the data from RAM to VRAM. This takes up heaps of memory at 12K per screen, so you need to set aside a 12K area where the screen can be stored such as #C000.

From VRAM to RAM

```
START     210004     LD HL,#C000
          AF         XOR A
          D3BF       OUT (#BF),A
          D3BF       OUT (#BF),A
          01BE00     LD BC,#BE
          1618       LD D,24
LOOP      EDA2       INI
          20FC       JR NZ,LOOP
          15         DEC D
          20F9       JR NZ,LOOP
          RET
```

From RAM to VRAM

```
START     210004     LD HL,#C000
          AF         XOR A
          D3BF       OUT (#BF),A
          3E40       LD A,64
          D3BF       OUT (#BF),A
          01BE00     LD BC,#BE
          1618       LD D,24
LOOP      EDA3       OUTI
          20FC       JR NZ,LOOP
          15         DEC D
          20F9       JR NZ,LOOP
          RET
```

(b) In the next issue I will explainhow to use the other two screens in the machine code hints and tips section.

Δ

Sorry couldn't fit in all the letters that arrived.

EDITORIAL

Sorry that this issue has arrived so late, but I have been very busy with a number of other commitments (including university) and there hasn't been much time for writing programs and articles for the magazine.

I was unable to complete "Compressed pictures" for this issue of the magazine and it will most likely appear in the next issue. In place of this I have devoted the majority of this magazine to machine code programming. It includes a bit for learners and some more information for advanced programmers - some of that stuff I've been promising.

Hopefully I can get the Auckland SEGA User Club up and running again this month, so if your interested see the next page. I will need a lot of help and support to run the club and the magazine, so any volunteers please contact me soon.

EDITOR: Michael Hadrup (no I don't look like Garfield)

If you're a Garfield fan, then see page 21 for the SEGA version of this picture. (Whoops! it was supposed to be a mystery program ... we'll see who bothers to read the Editorial page!)

News and Reviews

SEGA User Clubs about New Zealand

Do you belong to a User Club that is still up and running? If you do then send in a contact name and address, so we can inform newer members of your club. Remember there are still people buying new and second hand SEGA's who have little knowledge about their computers and are looking for information via the magazine and local user clubs

Auckland SEGA User Club

At last, I have been able to advertise the start of the Auckland SEGA User Club once again. There has been a lot of discussion concerning starting the club but as the advertisement was omitted from Poseidon Software's catalogue and from the previous magazine the club is not yet up and running.

The club will meet once a month at Auckland Technical Institute in the city (as this is central to everyone) at 7- 7:30pm and will continue no later than 10pm. The day of the week will be determined when I have contacted all those interested in attending the club.

The first club meeting will be an organisation discussion to determine what format future meetings will follow. The general format of meetings will always include "workshops" at different levels catering for people learning to use their computer, programming in BASIC and programming in machine code.

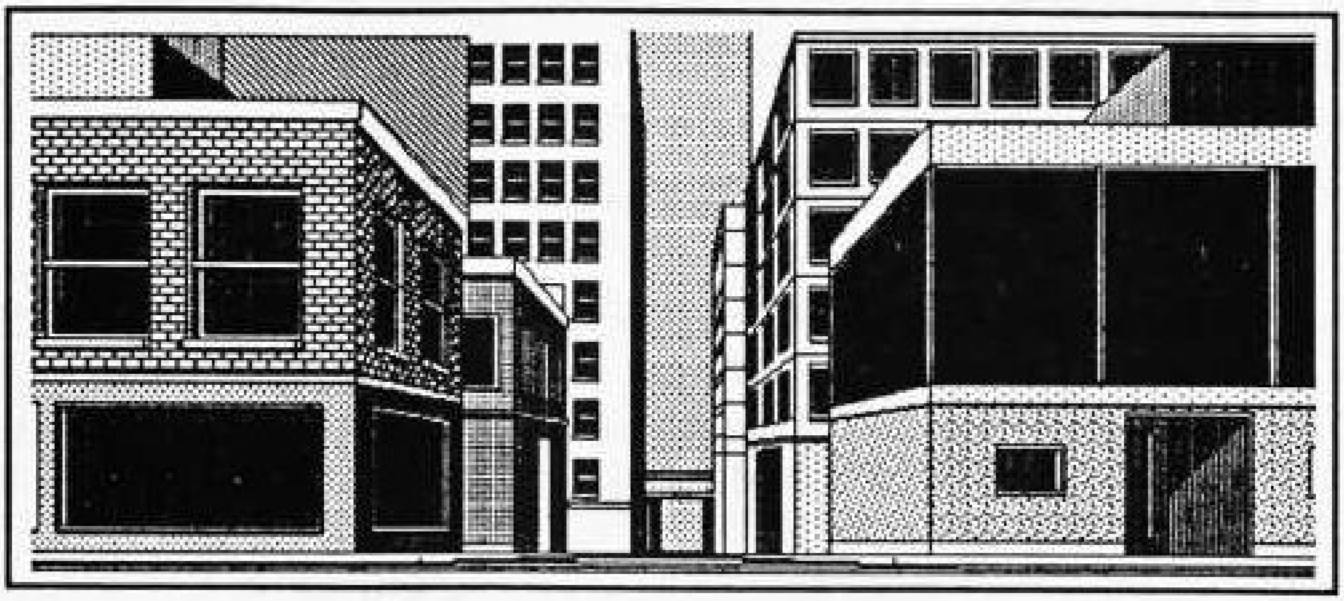
Details of meetings will be provided in a separate Newsletter (one - two pages only). Members attending will be asked to donate 80¢ (depending on the number of members) to cover costs of running the club and printing the newsletter.

The aim of the club will be to provide as much first hand knowledge as possible concernig all aspects of SEGA Computers. If your interested in attending the club then contact myself at the telephone number listied inside the front cover. It may be possible to arrange "pooling" of transport for anyone with problems getting to meetings.

Δ

3D City

By Steven Boland



```
10 REM 3-D  CITY
20 REM By Steven Boland
30 REM of Auckland
40 REM Friday 15/1/1988
50 PATTERN C#48,"708898A8C88870":PATTERN C#79,"70888888888870"
60 GOTO 1130
70 GOSUB 2140
80 MAG1:HS=0
90 SC=0:MI=0
100 GOSUB 1530
110 TX=104:TY=130:MV=0
120 SPRITE5,(TX,TY),0,9
130 IF XJ=1 THEN GOSUB 510
140 SOUND0
150 IF MI=5 THEN 940
160 IF MV=0 THEN BLINE (50,35)-(200,45),,BF:CURSOR57,38:COLOR1:PRINT "Get
 Ready - ":GOTO 760
170 REM
180 S1=STICK(1):S$=INKEY$
190 IF S$=CHR$(28)ORS1=3 THEN TX=TX+8
200 IF S$=CHR$(29)ORS1=7 THEN TX=TX-8
210 IF S$=CHR$(30)ORS1=1 THEN TY=TY-8
220 IF S$=CHR$(31)ORS1=5 THEN TY=TY+8
230 IF TX<10 THEN TX=10
240 IF TX>240 THEN TX=240
250 IF TY<100 THEN TY=100
260 IF TY>178 THEN TY=178
270 SPRITE5,(TX,TY),0,9
280 IF STRIG(1)>0ORS$=CHR$(32) THEN 380
290 REM
300 DV=INT(RND(1)*LE):V=INT(RND(1)*LE):FV=V-DV
310 IF SX+FV<10ORSX+FV>240 THEN FV=-FV
320 SX=SX+FV:SY=SY+LE
330 IF SY>133 THEN SP=48
```

$40 Prize

Program of the month

```
340 SPRITE6,(SX,SY),SP,3
350 IF SY>178 THEN MV=0:MI=MI+1:BLINE (218,20)-(230,31),,BF:CURSOR215,23:
COLOR1:PRINT MI:GOTO 140
360 IF XJ=1 THEN SOUND3,SO,15:SO=SO-25
370 GOTO 180
380 SOUND0:SN=10000:SOUND4,3,15:FOR SH=4 TO 24 STEP 4
390 SPRITE4,(TX,TY),SH,15
400 SOUND3,SN:SN=SN-1000:NEXT SH
410 IF TX<SX+8 AND TY<SY+8 AND TX+8>SX AND TY+8>SY THEN 440
420 SPRITE4,(TX,TY),28,15
430 SOUND0:GOTO 290
440 OUT &H7F,&HE4:FOR BB=&HF0 TO &HFF:OUT&H7F,BB:FOR BC=1 TO 10:NEXT BC,B
B:SOUND0
450 SPRITE4,(TX,TY),28,15
460 IF SP=44 THEN SB=50
470 IF SP=48 THEN SB=25
480 BLINE (90,20)-(125,31),,BF
490 SC=SC+SB:CURSOR90,23:COLOR1:PRINT SC
500 MV=0:GOTO 160
510 REM TUNE # 1
520 RESTORE 590
530 READ K1
540 READ J1
550 IF J1=-1 THEN 530
560 IF J1=-2 THEN SOUND0:RETURN
570 SOUND1,J1,15:FOR DE=1 TO K1:NEXT
580 GOTO 540
590 DATA 25,370,466,370,554,370,740
600 DATA -1,13,698,622,554,622
610 DATA 554,494,466,494,466,415
620 DATA -1,25,370,-2
630 REM TUNE # 2
640 RESTORE 710
650 READ K2
660 READ J2
670 IF J2=-1 THEN 650
680 IF J2=-2 THEN SOUND0:RETURN
690 SOUND1,J2,15:SOUND2,J2+5,15:FOR DE=1 TO K2:NEXT
700 GOTO 660
710 DATA 40,392,-1,10,349,330
720 DATA 349,330,-1,20,294,330
730 DATA -1,40,349,30000
740 DATA 392,-1,10,466,30000,494
750 DATA -1,15,523,-2
760 SX=INT(RND(1)*100)+75:SY=90
770 PSET (SX+8,SY+8),3:FOR DE=1 TO 100:NEXT :PRESET (SX+8,SY+8):SP=36
780 SO=2250:FOR SY=90 TO 46 STEP -LE
790 LG=LE/4
800 DV=INT(RND(1)*LG):V=INT(RND(1)*LG)
810 FV=V-DV:SX=SX+FV
820 SPRITE6,(SX,SY),SP,3
830 IF XJ=1 THEN SOUND1,SO,15:SO=SO+100
840 NEXT SY :SP=40
```

```
850 FOR SY=46 TO 88 STEP LE
860 LG=LE/2
870 DV=INT(RND(1)*LG):V=INT(RND(1)*LG)
880 FV=V-DV:SX=SX+FV
890 SPRITE6,(SX,SY),SP,3
900 IF XJ=1 THEN SOUND1,SO,15:SO=SO-100
910 NEXT SY :SP=44
920 CURSOR130,38:COLOR1:PRINT CHR$(17);"GO!!!!";CHR$(16)
930 MV=1:SOUND0:GOTO 170
940 IF XJ=1 THEN GOSUB 630
950 SCREEN 1,1:CLS
960 PRINT "**************************************************  GAME  OVER
  **************************************************"
970 CURSOR5,5:PRINT "YOUR SCORE IS "
980 IF SC>=HS THEN HS=SC:CURSOR1,7:PRINT "WELL DONE  YOU BET THE HI-SCORE
  !":GOTO 1000
990 CURSOR1,7:PRINT "YOU DIDN'T BEAT THE HI-SCORE"
1000 CURSOR5,10:PRINT "PLAY AGAIN ? (Y/N)"
1010 CURSOR20,5:PRINT SC
1020 FOR TI=1 TO 10
1030 CH$=INKEY$
1040 IF CH$="Y" THEN BEEP:GOTO 50
1050 IF CH$="N" THEN BEEP:GOTO 1080
1060 NEXT :CURSOR20,5:PRINT "      "
1070 FOR TI=1 TO 50:NEXT :GOTO 1010
1080 CURSOR15,20:PRINT "BYE !":FOR DE=1 TO 250:NEXT DE:END
1090 CH$=INKEY$
1100 IF CH$="Y" THEN CH=1:RETURN
1110 IF CH$="N" THEN CH=2:RETURN
1120 GOTO 1090
1130 SCREEN 1,1:COLOR15,1:CLS
1140 PATTERN C#113,"FFFFFFFFFFFFFFFF"
1150 PRINT" qqqq         qqq    qqqq qqq qqq q q"
1160 PRINT"    q         q  q   q     q   q  q q"
1170 PRINT"  qqq   qqq   q  q   q     q   q  qqq"
1180 PRINT"    q         q  q   q     q   q   q "
1190 PRINT" qqqq         qqq    qqqq qqq  q   q "
1200 CURSOR8,6:PRINT "INSTRUCTIONS ? (Y/N)"
1210 GOSUB 1090
1220 ONCHGOTO 1380,1230
1230 CURSOR6,8:PRINT "1 - EASY        9 - HARD":LE=1
1240 CURSOR1,12:PRINT "PUSH FIRE BUTTON OR SPACE BAR WHEN"
1250 CURSOR4,13:PRINT "THE LEVEL YOU WANT IS SHOWN"
1260 CURSOR12,10:PRINT "LEVEL : ";LE
1270 FOR DE=1 TO 25
1280 IF INKEY$=CHR$(32)ORSTRIG(1)>0 THEN BEEP:BEEP:BEEP:GOTO 1330
1290 NEXT DE
1300 CURSOR21,10:PRINT " "
1310 LE=LE+1:IF LE>9 THEN LE=1
1320 GOTO 1260
1330 REM
1340 CURSOR10,18:PRINT "SOUND ON ? (Y/N)"
1350 IF INKEY$="Y" THEN BEEP:BEEP:XJ=1:GOTO 70
```

```
1360 IF INKEY$="N" THEN BEEP:BEEP:XJ=0:GOTO 70
1370 GOTO 1350
1380 CLS:CURSOR12,0:PRINT "INSTRUCTIONS":PRINT
1390 PRINT "YOU HAVE BEEN ASSIGNED TO FIGHT":PRINT "AGAINST AN INVASION O
F GREEN ALIEN":PRINT "SPACESHIPS !!!!":PRINT
1400 PRINT "THEIR TASK IS TO DESTROY THE CITY.":PRINT "YOU HAVE TO SHOOT
DOWN AS MANY SHIPS":PRINT "AS YOU CAN TO WIPE OUT THEIR FLEET.":PRINT
1410 PRINT "SET IN THE FUTURE, THE ALIENS FLY":PRINT "TOWARD YOU IN 5 STA
GES, THUS CREATING":PRINT "A 3 DIMENSIONAL EFFECT. YOU CANNOT"
1420 PRINT "MOVE YOUR TARGET UNTIL THE ALIENS":PRINT "HAVE TAKEN OFF.":PR
INT
1430 PRINT "SCORING :- WHEN A SPACESHIP IS":PRINT "ABOUT HALF WAY DOWN TH
E PLAYING":PRINT " AREA, IT IS WORTH 50 Pts. ANY":PRINT "FURTHER DOWN IT
IS WORTH 25 Pts."
1440 PRINT:PRINT TAB(10);"PRESS ANY KEY"
1450 IF INKEY$="" THEN 1450
1460 BEEP:CLS
1470 PRINT :PRINT "IF AN ALIEN REACHES THE BOTTOM OF THE":PRINT "SCREEN T
HEN IT IS COUNTED AS A MISS.":PRINT "5 MISSES AND YOUR GAME IS OVER !"
1480 PRINT:PRINT TAB(10);"PRESS ANY KEY"
1490 IF INKEY$="" THEN 1490
1500 BEEP:GOTO 1230
1510 SX=INT(RND(1)*100)+75:SY=90
1520 END
1530 SCREEN 2,2:COLOR1,1,,1:CLS
1540 X=0:Y=10:XX=255:YY=191:E=1
1550 COLOR15,5,(0,0)-(255,49)
1560 FOR N=70 TO 0 STEP -10
1570 LINE (X,Y-E)-(XX,Y-E),15
1580 E=INT(SIN(RAD(N))*10)
1590 Y=Y+E
1600 NEXT N
1610 COLOR1,1,(0,50)-(255,99),1
1620 COLOR1,1,(0,101)-(255,191),1
1630 X=0:Y=100:XX=255:YY=191:E=1
1640 COLOR15,1
1650 FOR N=0 TO 70 STEP 10
1660 E=INT(SIN(RAD(N))*20)
1670 LINE (X,Y+E)-(XX,Y+E),11
1680 Y=Y+E
1690 NEXT N
1700 REM
1710 REM
1720 REM
1730 X=80:Y=100:XX=0:YY=191
1740 FOR U=0 TO 8
1750 LINE (X,Y)-(XX,YY),11
1760 X=X+10:XX=XX+30
1770 NEXT U
1780 X=70:Y=100:XX=0:YY=160:E=5
1790 FOR U=0 TO 2
1800 LINE (X,Y)-(XX,YY),11
1810 X=X-10:YY=YY-20
```

```
1820 NEXT U
1830 LINE (40,100)-(0,110),11
1840 X=170:Y=100:XX=255:YY=175:E=5
1850 FOR U=0 TO 2
1860 LINE (X,Y)-(XX,YY),11
1870 X=X+10:YY=YY-20
1880 NEXT U
1890 LINE (200,100)-(255,120),11
1900 LINE (220,100)-(255,110),11
1910 LINE (50,80)-(53,71),1:LINE -(57,71):LINE -(58,75):LINE -(58,63):LIN
E -(62,63):LINE -(62,75):LINE -(65,75)
1920 LINE (65,75)-(68,73):LINE -(76,73):LINE -(76,71):LINE -(79,71):LINE
-(79,61):LINE -(82,55):LINE -(86,66):LINE -(86,77)
1930 LINE (86,77)-(87,75):LINE -(89,79):LINE -(89,81):LINE -(92,81):LINE
-(92,77):LINE -(93,77):LINE -(93,74):LINE -(95,74)
1940 LINE (95,74)-(95,59):LINE -(100,59):LINE -(103,61):LINE -(103,68):LI
NE -(107,68):LINE -(108,69):LINE -(108,73)
1950 LINE (108,73)-(113,73):LINE -(113,68):LINE -(118,68):LINE -(118,52):
LINE -(125,52):LINE -(125,78):LINE -(129,76):LINE -(129,76)
1960 LINE (129,76)-(129,53):LINE -(135,53):LINE -(135,81)
1970 LINE (0,61)-(0,61):LINE -(3,69):LINE -(8,69):LINE -(8,63):LINE -(13,
62):LINE -(27,65):LINE -(27,80):LINE -(33,80)
1980 LINE (33,80)-(33,78):LINE -(43,79):LINE -(49,78):LINE -(49,80):LINE
-(50,80)
1990 LINE (135,80)-(140,75):LINE -(145,65):LINE -(150,65):LINE -(150,75):
LINE -(155,75):LINE -(155,70):LINE -(160,70)
2000 LINE (160,70)-(160,60):LINE -(165,60):LINE -(165,65):LINE -(170,65):
LINE -(170,70):LINE -(179,69):LINE -(185,70)
2010 LINE (185,70)-(185,72):LINE -(186,72):LINE -(189,73):LINE -(193,63):
LINE -(193,67):LINE -(195,67):LINE -(195,66)
2020 LINE (195,66)-(198,66):LINE -(198,67):LINE -(204,67):LINE -(204,65):
LINE -(212,65):LINE -(212,63):LINE -(216,63)
2030 LINE (216,63)-(216,68):LINE -(222,68):LINE -(222,69):LINE -(226,69):
LINE -(226,72):LINE -(228,72):LINE -(228,52)
2040 LINE (228,52)-(233,52):LINE -(237,54):LINE -(237,61):LINE -(241,61):
LINE -(241,67):LINE -(246,67):LINE -(246,61):LINE -(255,61):PAINT(0,55),8
2050 FOR I=0 TO 10
2060 X=INT(RND(1)*255)
2070 Y=INT(RND(1)*100)
2080 IF Y<61 THEN 2070
2090 PSET (X,Y),8:NEXT
2100 BLINE (10,10)-(245,33),,BF
2110 CURSOR8,12:COLOR1:PRINT CHR$(17);"..... 3-D CITY ....."
2120 CURSOR10,23:PRINT " SCORE:";CHR$(16);SC;CHR$(17);"  MISSES:";CHR$(16
);MI
2130 RETURN
2140 RESTORE 2150:FOR P=0 TO 51:READ P$:PATTERN S#P,P$:NEXT P:RETURN
2150 DATA 000039202100012A,2A0100212039,00009C0484008054,54800084049C
2160 DATA 00003F2020202020,20202020203F,0000FC0404040404,0404040404FC
2170 DATA 0000001F10101010,101010101F,000000F808080808,08080808F8
2180 DATA 000000000F080808,0808080F,00000000F0101010,101010F0
2190 DATA 0000000000070404,040407,0000000000E02020,2020E0
2200 DATA 0000000000000302,0203,000000000000C040,40C0
```

```
2210 DATA 0000000000000001,01,0000000000000080,80
2220 DATA 00,00,00,00
2230 REM EXPLOSION
2240 DATA FF,FF,FF,FF
2250 DATA 0000000000040607,0604,000000000002060E0,6020
2260 DATA 000000001018151F,151810,000000000818A8F8,A81808
2270 DATA 000040605048497F,4B49506040,000002060A1292FE,D2920A0602
2280 DATA 8080C0C0A0918AFF,8B91A0C0C08080,01010303058951FF,D1890503030101
```

Δ

Machine Code Programming

By Michael Hadrup

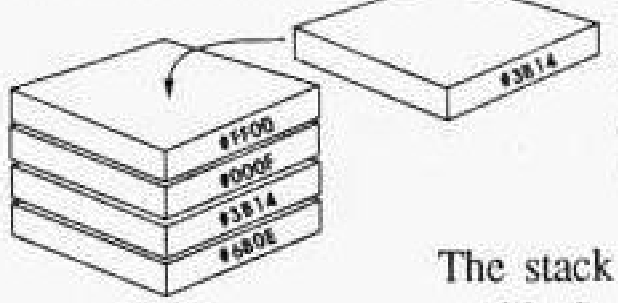
• The third part of this course covers the more advanced topics of stacks, jumping / calling and comparing.

The Stack

There is an area set aside in RAM for storing various pieces of information. Depending on the situation this can be data you have stored there, or it may be information to help the machine know what it's doing. You will see later that it is important that the stack is balanced (ie. in the right sequence) or the machine may become totally confused and crash.

The word "stack" means exactly what it sounds like! Imagine a stack of cardboard boxes. Each box can be considered a memory location, so each has an address - but you can only look into the top box easily. Therefore a box can only be added or removed to the top of the stack.

#3B14 "Pushed" on top



This idea is more commonly described as LIFO - Last in, first out and the memory locations of the stack are just like that. When you add something to the stack you "PUSH" it on the top. When you remove something from the stack (the last thing you "PUSHED" on) you "POP" it from the top of the stack.

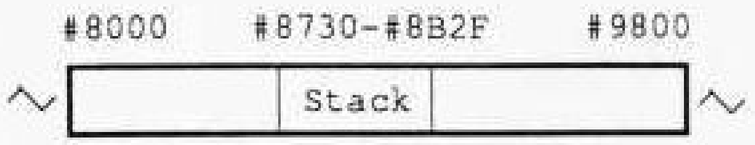
The stack is actually a very peculiar beast, because it's stored upside down in the SEGA (or any Z80 machine).

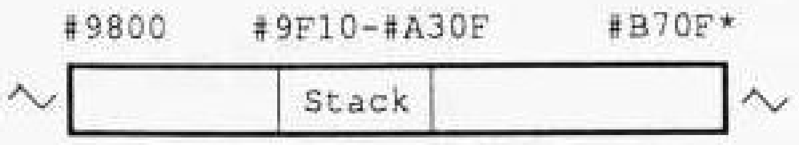
It was not designed this way to cause utter confusion, but in other computers the stack always grows downwards and your program grows upwards allowing all the free memory to sit in the middle with no restrictions on either stack or program size. BUT in the SEGA the stack is allocated it's own fixed area of RAM within the system variables.

This means that the stack can only grow within this set aside area. (When using the PAINT command which is a "stack based" program ie. it uses the stack, on complicated pictures you can get a "? Stack overflow error" message. This is because the PAINT command has used up all the stack area and has been stopped before it starts destroying other parts of the system variables.

Stack area within the system variables

16K and 32K Cartridge **SF-7000 (Disk Basic)**





Figures 3.1 and 3.2 * Varies according to maximum files. See page 210 Disk Basic manual

As mentioned on the previous page the stack is actually stored upside down, so that the start of the stack area is #8B2F or #A30F and it builds or grows downwards.

The stack is so important to the computer that a special register is set aside just to store the position of the *top* of the stack (which is actually the lowest address). That register is called SP or "Stack Pointer" and it is a register pair (16 bits - so it can point anywhere in the memory) like BC,DE and HL, but we can not separate the two halves into S or P.

Here is the list of instructions for storing data on the stack. Note that it is only possible to use register pairs with the stack. It is also worth noting the "PUSH" ing a register pair does not alter its value.

C5	PUSH BC	C1	POP BC
D5	PUSH DE	D1	POP DE
E5	PUSH HL	E1	POP HL
F5	PUSH AF	F1	POP AF

Verify that
PUSH HL
POP DE
is the same thing as
LD D,H
LD E,L

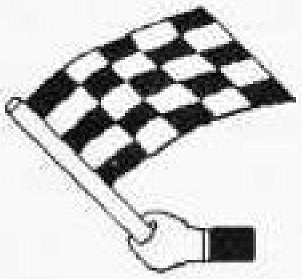
```
 PUSH HL          POP HL
 in Basic         in Basic

SP=SP-2          L=PEEK (SP)
POKE SP+1,H      H=PEEK (SP+1)
POKE SP,L        SP=SP+2
```

Don't worry about the register AF, I'll get round to explaining that next when I explain "Flags" in detail. It is safe to say now that the stack is only time that the Flag register can be accessed - when it is combined with A (accumulator) to form a register pair. To examine the value of the flag register directly you could use PUSH AF:POP BC. The contents of the flag register would be placed in C.

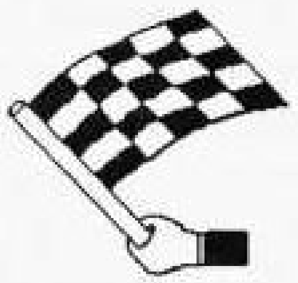
One of the major uses of PUSH AF and POP AF is simply to preserve the value of A. The fact that F is stacked with A is conveniently forgotten about. (But it is important to remember). If you need to perform a calculation that can only be done with A, such as B=B+5 , but where the value of A is needed later on in the program.

We can use SP in much the same way as BC and DE. We can add and subtract with it and load it.

```
31????      LD SP,NN        ED72        SBC HL,SP
33          INC SP          ED73????    LD (NN),SP
39          ADD HL,SP       ED7A        ADC HL,SP
3B          DEC SP          ED7B????    LD SP,(NN)
F9          LD SP,HL
```



Flag Register



A description of a "Flag" can be found on page 11 of Issue 2. The F or Flag register sometimes called the *status* register often cohabitates with A in the hope no one will notice it - PUSH AF, POP AF and EX AF,AF'. The bits of F each have a specific purpose ...

Bit 7: The *sign* flag, or S.
Bit 6: The *zero* flag, or Z.
Bit 5: Not used
Bit 4: The *half carry* flag, or H.
Bit 3: Not used.
Bit 2: The *parity / overflow* flag, or PV.
Bit 1: The *subtract* flag, or N.
Bit 0: The *carry* flag, or C. (Not to be confused with register C).

The *half carry* flag is set by arithmetic instructions if there is a carry from bit 3 into bit 4 and in the case of register pairs from bit 11 to bit 12. This is used with BCD or Binary Coded Decimal - I might discuss this in future issues.

The *subtract* flag is set by any instruction which invloves a subtraction or reset by any instruction which involves addition.

The *carry* flag is set by any instruction which causes a carry. It is reset by all AND, OR, XOR, and BIT instructions.

The *zero* flag is set by any instruction that returns a result of 0 and reset by any instruction that returns a result which isn't 0.

The *parity / overflow* flag is set by any instruction that results in an overflow and is also used to determine the parity of an arithmetic operation. Parity is defined as the number of 1's. PE (Parity Even) means the number of 1's was an even number. PO (Parity Odd) means the number of 1's was an odd number.

An overflow applies when we are using *negative numbers*. Remember the definition of *negative numbers* , if you were to add #45 to #41 (both positive) it gives #86 which is negative.

PO (parity odd) = NV (no overflow) and PE (parity even) = V (overflow)

Jumping

There is another 16 bit register call PC or Program Counter. Like the SP it is a register pair, but it cannot be broken into P and C. Its job is to remember whereabouts we are in the program. Everytime the Z80 executes a machine code instruction it takes a look at what the PC holds. If it contained #F000 then it would execute the instruction at location #F000 and it will then increment to the next instruction.

What the PC holds then depends on whether the instruction executed was one,two,three or four bytes long. If the instruction at #F000 was LD B,2 or #0600 then after the instruction is executed the PC will point to #F002 as this load instruction is *two bytes* long.

If you alter the value of PC then the effect is like a Basic GOTO. The only difference is that machine code does not use line numbers, so you have to GOTO the right *address*. You can alter the PC with the JP instruction - short for Jump. `JP #F000` means let `PC = #F000` and start executing instructions from there. Be careful, you can make an infinite loop with JP such as `#F000 JP #F000`. Luckily the SEGA has a reset key which can get you out.

When you write programs you should initally write them on paper (I often don't but I should). To make programs easier to understand we mark important lines with labels. (By the way the program doesn't do anything really).

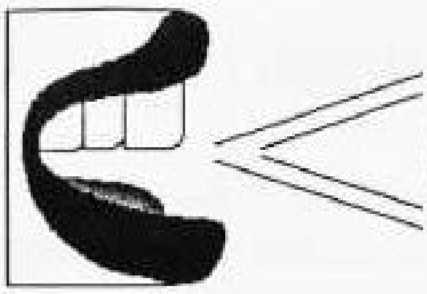
```
START  7E       LD A,(HL)     START  7E      LD A,(HL)
       23       INC HL               23      INC HL
       C3????   JP START             18FC    JR START
```

There is another instruction similar to JP called JR or Jump Relative. It means a jump forward or backwards of a given number of bytes from the end of the instruction. In many ways it is better than JP because it is only two bytes long instead of three, and because a whole routine may be relocated without changing all the JP addresses (which are "absolute").

This is where my definition of *negative numbers* is useful. With a JR instruction we can therefore jump relative in the range -128 to 127. JR 0 has no effect. JR -2 or JR #FE will jump back to the JR instruction (an infinite loop) as JR is a two byte instruction.

JR and JP are more or less useless without conditions attached such as IF ... THEN ... GOTO in Basic. Machine code allow four conditions for JR and eight conditions on JP.

JR NZ,dis	(non zero - zero flag reset)	JR Z,dis	(zero - zero flag set)
JR NC,dis	(no carry - carry flag reset)	JR C,dis	(carry - carry flag set)
JP NZ,nn	(non zero - zero flag reset)	JP Z,nn	(zero - zero flag set)
JP NC,nn	(no carry - carry flag reset)	JP C,nn	(carry - carry flag set)
JP PO,nn	(parity odd or no overflow)	JP PE,nn	(parity even or overflow)
JP P,nn	(last calculation positive)	JP M,nn	(last calculation negative)



Calling - Machine Codes Gosub

Even in machine code we can have *subroutines* . The machine code equivalent of GOSUB is CALL. Like JP we write CALL nn where nn is the address of the subroutine. The equivalent of RETURN is RET and it can be considered as "return to Basic" if there is no subroutine to return from. This is because Basic is just one big machine code program and it calls your machine program as a subroutine when you use the CALL statement in Basic. Of course there are conditional forms of the CALL and RET instructions ...

CALL NZ,nn	(non zero - zero flag reset)	CALL Z,nn	(zero - zero flag set)
CALL NC,nn	(no carry - carry flag reset)	CALL C,nn	(carry - carry flag set)
CALL PO,nn	(parity odd or no overflow)	CALL PE,nn	(parity even or overflow)
CALL P,nn	(last calculation positive)	CALL M,nn	(last calculation negative)
RET NZ	(non zero - zero flag reset)	RET Z	(zero - zero flag set)
RET NC	(no carry - carry flag reset)	RET C	(carry - carry flag set)
RET PO	(parity odd or no overflow)	RET PE	(parity even or overflow)
RET P	(last calculation positive)	RET M	(last calculation negative)

You can also use conditional RET instructions to return to Basic. In Basic there is no stack to worry about, but in machine code we must worry about it as the stack is also used to store *return address* for subroutines.

```
CALL nn is equivalent to PUSH "PC": JP nn
    and RET is equivalent to POP "PC"
```

Because the stack holds the *return address* then during a subroutine it must not alter SP or the stack and all "PUSH" instructions must be balanced by "POP" instructions, otherwise the machine may crash!

A clever trick once you know what you're doing is to alter the return address from within a subroutine by destroying the previous *return address* and replacing it with a new address on the stack. This subroutine would always return to #F000 no matter where is was called from ...

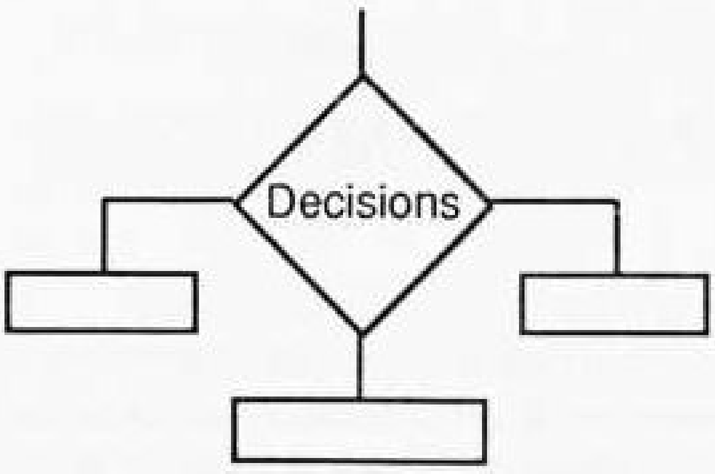
```
POP HL:LD HL,#F000:PUSH HL:RET
```

You can also accomplish a CALL "DE" instruction by using

```
PUSH DE:RET
```

Another useful technique is to store the value of SP somewhere (in this case #FF00) with a `LD (#FF00),SP` instruction. A subroutine could exit with more "PUSH" then "POP" instructions, by simply discarding them as follows ...

```
LD SP,(#FF00):RET
```



CP - Comparing

Earlier I described how the JUMP, CALL, and RET instructions can be used conditionally like IF ... THEN ... GOTO in Basic. But we need some way to set the flags. The compare instruction allows you to compare A with any other single register or number.

When you execute a CP B instruction (compare A with B) a subtraction is performed (A-B) and the result is discarded. The values of neither A nor B are altered, but the flags will change and they will tell us about the result. If A contained 3 and B contained 7 then after a CP B instruction the value of the flags will be as follows ...
C = 1 Yes there was a carry as 3 < 7
Z = 0 The result was not zero
V = 0 There was no overflow as 3 - 7 = #FC which is negative and is supposed to be

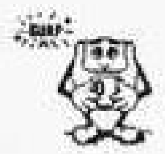
Therefore the following JP instructions would cause a jump ...

```
JP C,nn
JP NZ,nn                               IF A=B THEN GOTO    CP B:JR Z,dis
JP NV,nn or JP PO,nn                   IF A<B THEN GOTO    CP B:JR C,dis
JP M,nn (as the result is negative)    IF A>=B THEN GOTO   CP B:JR NC,dis
```

```
B8  CP B            BC  CP H
B9  CP C            BD  CP L
BA  CP D            BE  CP (HL)
BB  CP E            BF  CP A
```

```
18?? JR dis         C3???? JP nn        E2???? JP PO,nn
30?? JR NZ,dis      C2???? JP NZ,nn     EA???? JP PE,nn
28?? JR Z,dis       CA???? JP Z,nn      F2???? JP P,nn
30?? JR NC,dis      D2???? JP NC,nn     FA???? JP M,dis
38?? JR C,dis       DA???? JP C,nn
```

+1 / -1



I just realised that I forgot to mention INC (increment) and DEC (decrement) instructions. INC B adds on one to B and is equivalent to B=B+1. However INC and DEC do not alter the value of the carry flag. If A = #FF then INC A will make A zero, but the carry flag will remain unchanged.

```
03  INC BC          04  INC B       24  INC H
13  INC DE          0C  INC C       2C  INC L
23  INC HL          14  INC D       34  INC (HL)
33  INC SP          1C  INC E       3C  INC A
```

Additional Information

Although I haven't mentioned it much the microprocessor inside the SEGA is a Z80A (the A means it can work faster) which is clocked at about 3.58 MHz (how fast it goes). A lot of books have been written about the Z80. The ultimate authority is "Programming the Z80" by Rodnay Zaks, but this is quite a complicated book to read. Another good book is Z80 Programming - I can't remember who wrote it, but it's a red book (if that helps at all).

Common abbreviations:

R	= 8 bit registers	B, C, D, E, H, L, (HL), A (HL) is not a register, but it is the value at the location addressed by HL or PEEK (HL). It is included because it returns an 8 bit value like a register.
R'	= alternate 8 bit registers	B', C', D', E', H', L', (HL'), A'
RR	= 16 bit combined registers	BC, DE, HL
RR'	= alternate 16 bit registers	BC', DE', HL'
	and 16 bit dedicated registers	PC, SP, IX, IY (I will explain IX and IY in the next issue.)

In certain cases ("PUSH" and "POP") SP is replaced with AF

N	= 8 bit number
NN	= 16 bit number
8 bits =	range 0 to 255 or #FF
16 bits=	range 0 to 65535 or #FFFF

Updated MC Editor

```
10 REM MC Editor
20 REM
30 RESTORE1000
40 X=&HC000
50 READA$
60 IFA$="*"THENEND
70 IFLEFT$(A$,1)=":"THEN180
80 PRINTHEX$(X);":";
90 IFLEFT$(A$,1)="."THEN210
100 IFLEN(A$)MOD2=0THEN130
110 PRINT"Invalid data"
120 STOP
130 FORN=1TOLEN(A$)STEP2
140 POKEX,VAL("&H"+MID$(A$,N,2))
150 X=X+1
160 NEXT
170 PRINTA$:GOTO50
180 IFLEN(A$)<>5THEN110
190 X=VAL("&H"+MID$(A$,2))
200 GOTO50
210 FORN=2TOLEN(A$)
220 POKEX,ASC(MID$(A$,N,1))
230 X=X+1
240 NEXT
250 PRINT"""";MID$(A$,2)
260 GOTO50
9999 DATA *
```

This version allows you to enclose a string of ASCII characters within your machine code. The editor will automatically convert the strings to hex. To enclose a string add a line such as

```
1000 DATA .Hello
```

Δ

Machine Code Hints and Tips

Programs that save themselves

There is an area called String Register 1 in the system variables. Whenever a Basic command is executed which uses a string, such as a filename in SAVE or a PRINT statement, a copy of the string is stored in String Register 1. The SAVE routine looks at this area to decide what filename it will use when a program is saved.

For example to make a program save itself on disk as `"Hello    .BAS"` use

```
                A$="Hello   .BAS":CALL &H21CD
                              or
   PRINT "Saving """;:PRINT "Hello   .BAS""":CALL &H21CD
```

Note that you must pad (add characters or spaces) to the filename so that it fills up at least 8 characters (Disk Drive filenames) or at least 16 characters (Tape filenames).

Here are the calls to all the SAVE routines ...

Cartridge Basic	Tape save = &H7A27
Disk Basic	Tape save = &H1440
	Disk save = &H21CD

Complex Screens

A lot of people have been asking how games can store up to 64 screens in memory. The technique I am going to explain is similar to that used in Star Jacker, Sorcerer's Apprentice, and Delta Fighter and is based on building up a picture from a number of smaller characters.

A number of programs in Basic use the *graphics* characters on the keyboard to design shapes on the screen. An example of this is the boxes in some programs.

The basis behind the method is that it is possible to use the graphics screen the same way as you use the text screen. The graphics screen is divided into 32 x 24 squares of 8 x 8 pixels - this gives a 256 x 192 screen. The graphics screen is then divided into three vertical sections of 256 bytes (called the *Name Table*) and therefore each section can have 256 unique character patterns (called the *Pattern Table*) each with its only colour information (called the *Colour Table*). Within each square of individual "thirds" you can display one of the 256 patterns.

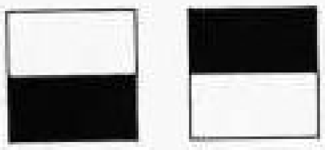
Sorcerer's Apprentice (SA from now on) uses only the two bottom thirds for displaying the game screen.. You may have noticed that certain objects such as "bushes" are repeated in many of the screens. One particular "bush" is 16 pixels high by 8 pixels wide - two patterns high. It is stored in two patterns (in the *Pattern Table*) in each of the "thirds", say pattern 0 and pattern 1. By poking a 0 and poking a 1 into the *Name Table* the pattern will be displayed, just as if you were to VPOKE &H3C00,65 to display the letter "A".

By the way the *Name Table* for the graphics screen is stored at &H3800 in VRAM, the *Pattern Table* is stored at &H0000 in VRAM and the *Colour Table* is tored at &H2000. in VRAM.I must admit that this explanation may seem very complicated. If it is don't worry have a look at the example program below ...

```
10 SCREEN2,2:CLS
20 FORN=&H3800TO&H38FFSTEP2
30 VPOKEN,0
40 VPOKEN+1,1
50 NEXT
60 FORN=0TO15
70 READA$
80 VPOKEN,VAL("&H"+A$)
90 NEXT
100 A=1
110 FORN=0TO15
120 VPOKE&H2000+N,A*16+15
130 A=(A+1)MOD15
140 NEXT
150 GOTO 110
160 DATA 0,0,0,0,FF,FF,FF,FF
170 DATA FF,FF,FF,FF,0,0,0,0
```

This sets the top "third" of the display (first 256 bytes) to alternate between pattern 0 and pattern 1.

These lines define two patterns (0 and 1).



This group of lines alters the colour information for the two patterns and the effect is scrolling of colour up each pattern.

The important points to notice are that only two patterns were defined, yet we were able to fill the top "third" of the screen easily. In machine code a normal screen is 12K long - (256 x 192 pixels divided by 8 = 6144 plus 6144 for colour). A complete screen can now be made up of 32 x 24 patterns = 768 bytes or 0.75K - a saving of 1600%. Because it is so much smaller you can store more screens and they will seem to appear instantaneously.

A game such as Sorcerer's Apprentice which uses only the two bottom "thirds" would only require 0.5K per screen and by using other techniques based on the fact that a game screen may be mostly empty space (in which the character can move) and objects are made of repeated groups of patterns, even less memory is required to store screens eg 0.25K. Hence in 20K it would be possible to store at least 80 screens or more.

The *Name Table* normally would point each square to its own pattern
square 0 = pattern 0, square 1 = pattern 1 and so on. Here is another example program ...

```
10 SCREEN2,2:CLS
20 FORN=0TO31
30 READA$
40 VPOKEN,VAL("&h"+A$)
50 NEXT
60 FORN=&H3800TO&H381F
70 IFN<>&H3800THENVPOKEN-1,4
80 IFN=&H3800THENVPOKEN+31,4
90 FORM=0TO3
100 VPOKEN,M
110 NEXT
120 NEXT
130 GOTO60
140 DATA 80,80,80,80,80,80,80,80
150 DATA 20,20,20,20,20,20,20,20
160 DATA 8,8,8,8,8,8,8,8
170 DATA 2,2,2,2,2,2,2,2
```

This defines four patterns 0-3 for the four positions of a line in each square.

Each of the 32 squares accross the screen displays the four positions of the line then moves to the next square, clearing the previous square when it steps.

This makes it look as if the line is scrolling and in machine code this is very smooth

In the next issue I might be able to combine this section with the MC course and use this technique to write an Invaders game.

??? Mystery Program ???

```
1 REM Picture expander
2 REM
3 REM By Michael Hadrup
4 REM
5 REM Original size    = 6144
6 REM Compressed size = 2030
7 REM
8 REM Saving = 67%
10 X=16:X1=192:SY=1
20 WIDTH=X1-X
30 Y=X+SY*256
40 SCREEN2,2:COLOR1,11,,11:CLS
50 RESTORE400
60 FORN=0TO3
70 READB(N)
80 NEXT
90 RESTORE410
100 FORN=0TO5
110 READC(N)
120 NEXT
130 RESTORE1000:M=0
140 GOSUB270:IFB=0THENSTOP
150 IFB>128THEN200
160 FORN=1TOB:GOSUB270
170 VPOKEY,B
180 Y=Y+1:IFYMOD256=X1THENY=Y+256-WIDTH
190 NEXT:GOTO140
200 C=B-128:GOSUB270
210 IFB=0THEN250
220 FORN=1TOC:VPOKEY,B
230 Y=Y+1:IFYMOD256=X1THENY=Y+256-WIDTH
240 NEXT:GOTO140
250 IF(C+YMOD256)>=X1THENY=Y+256-WIDTH
260 Y=Y+C:GOTO140
270 IFM<>0THEN360
280 READA$
290 IFLEN(A$)<>6THENPRINT"Error in line";1000+INT(N/64)*10
300 A=0
310 FORF=0TO5
320 B=ASC(MID$(A$,F+1))-65
330 IFB>25THENB=B-6
340 A=A+C(F)*B
350 NEXTF
360 IFA=0THENB=0:GOTO380
370 B=INT(A/B(M)):IFB>0THENA=A-B*B(M)
380 M=M+1:IFM=4THENM=0
390 RETURN
400 DATA 16777216,65536,256,1
410 DATA 380204032,7311616,140608,2704,52,1
1000 DATA IIaDrX,AQKMmf,AwTZZY,LPUOfz,DqFDAQ,FkecXe,FwDgyX,FpmAje,HbfrFE,
LNDcpo,IVbZtM,AKuqsA,AdjjvH,AhHTKO,AAFDeQ,AJKKwh
1010 DATA AAJqyM,FrCfCy,BsnOYV,GcvcAA,HtFVZA,HNpBjm,HbfoaN,D1ZIHe,HbfIdu,
HbfQOM,FmuDaY,DbwMNw,AAAMHU,BV1IBc,FxoDyp,FquGZb
1020 DATA EyHWBv,AACPiw,ACIMeo,CrGwNc,AAAAPw,CsstcC,ACPoXX,AKUpUr,DYvymw,
EQtaDM,BbMPeA,BaCAuZ,BrRszt,Bctzsh,ANzOdE,BaClIA
1030 DATA CCoevc,BbRIaK,AFnSNg,CQafAi,AiVpIR,AEfdxC,KICMww,CsAEIv,CHtCWo,
AEeVqo,AIBmHU,IaseZN,AEgmgg,DvDmTE,ACQNXN,Appyjc
1040 DATA BalAEV,DMlyoI,HIxNRk,BItmrs,BbMPeA,BbOIuw,BtmRgI,CLxVCQ,BbMPeA,
BbMPeA,BbMPeA,BbOIlG,BdcEaD,BcSBnG,AAAMXU,FhwUJd
1050 DATA AddKxy,KilYew,FrCnSs,AqaKTH,FiOlzi,AAGDJe,BWAIJv,FsFbEJ,ACtBqs,
AABtQz,AAAMXU,AMCizm,KMglBL,BOfpLm,GmTvhb,GJSade
```

```
1060 DATA GATaHz,BTVede,GAWBMA,AAANOI,IYdaAh,ALZnfj,JvHTRc,BDHUIg,BVdjZA,
AKVHJO,AUkKua,AXbQMg,DcLUSA,FsjRZf,ADZQNd,AlwEXi
1070 DATA AlAHGY,BVjNnF,CrHSyA,ANogdc,AttvfE,AidPkg,ELHvJB,AhOMiY,HDObox,
FvYoHb,KxEDfM,AlATLU,FtJvQT,AQBTMo,AEmFvu,AAAAFB
1080 DATA AJKLLo,DlZIGy,Hbfrnk,FofyeJ,AUwqVk,ASzRAA,DNqQbE,AYFDFY,ABHsvD,
AAASLo,AAAwWM,AABVeJ,IZoJdS,FiOlwO,ACQBao,AGthhA
1090 DATA ACRKLB,KTzrsg,BIxJSK,AJLIEA,ENWPFW,AXFedF,AEekOS,HbUdIk,BfLKLb,
AAAkQQ,AAAYTr,AALNgU,GXijEW,AkouXE,CrHddN,AJKKsA
1100 DATA ATAMLI,IiDeGo,FofaRs,BEfCca,JWOeaC,JbWqmS,DodahW,DonltM,CseBcA,
CuRNlA,IYkubw,DNuPtQ,ALctqs,ATeKgA,BWAILE,HVZfkB
1110 DATA ANrtsQ,BHxOBs,AAAAOo,AAOjXE,BTHGZO,BgBERM,FirdiI,ACPQcT,AADfsA,
AkpJvo,AIEVZs,AnkJBE,BjVoyZ,FnJIUh,AEavTU,FtJjHk
1120 DATA GSVxgb,CRsCLo,AASxet,HcRkMT,AADBcA,BDDuLk,LNGXaL,BONrKA,IfkcZu,
EEYVJc,AAAWlk,IlChtV,BiMdDU,ACeYjF,AHQwer,GCSNjH
1130 DATA LGLYSQ,KglLKM,LNLHLj,AAAVis,KzLXGP,GADUiX,FxoENU,AizUJs,FofOVb,
IZHaBM,AmFSIB,ADaMcR,LOROrl,JvKsmS,AJLHpF,DMqfgl
1140 DATA GfTjhG,AFNNEn,AAAqYg,AAAkjn,FxqDeb,DjmThb,AJIqBk,FpmYPD,BTkTNr,
FoeqBs,JwHovs,GkoakS,AEgauQ,AlAEHM,AAAZzc,AABUzk
1150 DATA AIDZBE,CZjfOA,KDRIBQ,FoeqBs,CrHTpA,GkobBd,FismOL,BcegVf,AjdMTj,
FtHepE,lMsHsg,AAAbFs,FiuSzG,KerqRs,ANKKTH,AQGmVc
1160 DATA AJHtiH,FvYndc,LOROrl,FjMcLJ,ANriKh,AOzPma,AEfqRs,AkmEQE,CKqIGq,
AnAWFc,CrHfFU,BfufUC,ACQBVs,AJKKrj,LFHCZQ,KzRBJc
1170 DATA FtJWzI,ASaMwg,AFIeWw,AABVTz,CpmUQK,CorvcX,LGMUvw,KzLMRL,AFluLs,
LNOOgf,KCxXtz,AIvnds,LIblfi,LIaFIQ,ANZCvU,ACSSaY
1180 DATA AIBsHV,FtJXGU,Fpmdvp,AAEWTK,CYbmsA,BWAONI,AAFGzI,DJsjMw,ASUXUS,
DMjWEY,HNgbUR,DeYiOS,FtJXBc,FiOvnk,ALZDYz,AQGmKn
1190 DATA FtFxUc,AGuCRc,FrULYs,FyRvHE,GADUdj,FxoEMR,AikYhL,AJXnLH,LMBZvv,
LMyZqH,HCOFNT,ACMarn,LKtTJs,ACLrYw,AEiUhP,BWfUAF
1200 DATA AAABWI,BWfUAD,BXeFFx,AABuYI,AJLTqK,GCSlqp,ACQNNQ,Aidhvq,AAAvAY,
AAHYhx,BenvOB,AVOHls,HEWQSg,DJdsxM,FtQWlj,AGuVlo
1210 DATA FqqUyI,JtzeXy,FeNhKn,AOCshU,AJKKzM,AABURP,FxoEIY,AROIEx,LHiapV,
CquRsQ,AgKuqo,Foepkh,LMAFPo,ANucCa,Fquqmo,AtusNg
1220 DATA ASozHz,GSWAIA,AAHAnW,CIalXk,Alugky,FismWo,AbshTm,DOIQOp,AMzKPY,
AvDepN,DhbAZv,LONHVh,AAAYxP,AADlzW,AJNVZb,CsAQnU
1230 DATA AGtBXL,AABUbF,FxoEIc,AIEtmI,KxGUUw,BVlIBU,AUgATk,Akqkth,FqyEUw,
AGQMhP,AAEihz,BUBNzV,ACQTIw,FxoEJK,ACcflM,AIuSrk
1240 DATA GADVuZ,ACRiMx,ALibvW,DOJNmC,DOJWYq,HgyLGZ,ACQNrE,BWAONv,AAFfVP,
AEfeBw,ASWOjE,EMdQhM,AJLJgQ,FjFciZ,ADaMcU,Dgvqgl
1250 DATA FquGSi,DOyqmr,DADgiA,AADOCA,AlAHGI,AjhJDV,JvcdmC,ACQNrE,AlAKKA,
BsnLYw,AFIfxq,AAMGMy,AAAAFa,ENZiET,AAAXsj,ABHiTE
1260 DATA AAAAMS,EMZriK,FpRBXY,HOjros,FqFFqX,DJtxtd,FoeqBs,CrOqCA,HVZffg,
AIDAor,ACQNXE,IZDcdd,AAAAHY,AACOIe,KFaJHE,AFmaHD
1270 DATA AAJQrG,AJNBow,CtHOMZ,BhEIRV,ASWmGo,BXHxls,AGvsHj,FotQTj,CpyOVc,
LPHMEw,LKrXaI,FidecQ,ABTQvk,ASUYlM,ABnGOk,BaAxzV
1280 DATA BMNItd,FquGjF,DAReBu,ADYXcI,ABKwky,AJKXbY,AJKLLW,ADZQEI,AbdLsg,
EMcitt,ABHsvA,AGuJzT,AEgavM,JJVvZs,FoeqOJ,AEfeBw
1290 DATA AAWlet,ABJhLI,BTVUWV,LNGrBN,JCOfaL,JtzYgg,ACAVEg,GADUnX,BDFnmY,
AEfFiO,AADpui,GYZEgI,CHATBc,AEfdpi,AABUun,LOQSTH
1300 DATA FjzwIl,AABpzU,FiurLV,FggXFU,JJVsTw,ANrJPk,Fogiwi,AEfdfj,FvYoQA,
LNKFcY,BDfsaZ,IZmosZ,AJJNzU,AdxmBr,IZmQgQ,CHUmyQ
1310 DATA KLCyGv,AAASJM,FhlPyn,FtJWzT,LPOYEw,AWxoLC,IYdNkn,BTDUgD,AABOkQ,
KHnULD,IvaXlM,AAAQhD
```

Δ

Poseidon Software Hire Club

New Titles available include

Zaxxon	Chack 'n' pop
Choplifiter	Nija Princess
GP World	Wonderboy

TANK BATTLE

By Michael Hadrup

I finally got all the bugs out of this program, during the Easter break, hence the late arrival of the magazine. I would like to make a lot more additions to the program, but the program was becoming too long for the magazine - sorry for the small lettering in the listing but I was running out of space.

Typing in the program - Disk Basic: Type in the data listing, save it using

`SAVE "TANK.Dta"`

Then run it and save the code using

`SAVEM"TANK.Cde",&HE500,&HEF2F:SAVEM"TANK.SPR",&HA316,&HA615`

Now type in the main listing below and add the extra line(see page 23). Save this using

`SAVE "TANK"`

Typing in the program - LSV Users: Type in the main listing below and add the extra lines (see pages 21 and 23). Save this using `*SAVE "TANK",RUN1`0`

Now type in the data listing, save it on another tape using

`*SAVE "TANK.Dta"`

Then run it and save the code after the main program using

`*SAVEC"TANK.Cde",&HE500,&HEF2F:*SAVEC"TANK.SPR",&H8B36,&H8E35`

Typing in the program - Cartridge Users without LSV: Type in the data listing and add the extra lines (see page 23), save it using `SAVE "TANK.Dta"`

Now type in the main listing below and add the extra lines (see pages 23 and 24).

Save this after TANK using `SAVE "TANK1"`

Playing the game: Cartridge Basic Users without LSV must load and run the data program first, which stores the machine code and sprites in memory. Press enter to the LOAD prompt and then load and run the main program. LSV and Disk Basic users only have to run the main program, as it will load the sprites and machine code.

Use the arrow keys to select the menu and press space to change options. The keys already defined are for the Joysticks. You'll have to experiment as there is no room in the magazine for full instructions ...

Main Program

```
100 A0=&HECA1:A1=&HECB4:REM P1,P2
110 A2=&HECC7:A3=&HECDA:REM F,F1
120 A4=&HECA3:A5=&HECB6:REM P Range
130 A6=&HE528:A7=&HEBBB:REM Delay,Blk
140 A8=&HE57C          :REM What
150 B0=&HEBB7:B1=&HEC6D:REM Block,Res
160 B2=&HEC1E:B3=&HE500:REM Set,Play
170 B4=&HEB43:B5=&HE5D3:REM Sprts,Win
180 B6=&HE57E          :REM KeyScan
190 B7=&HEC79:B8=&HEC85:REM Key data
```

```
200 GOSUB1470
210 CFLAG=0:MAG1:POSITION(0,0)
220 P1=PEEK(A0):P2=PEEK(A1):X=0:Y=6
230 MAXL=10:SCREEN1,1:COLOR15,3:CLS
240 CURSOR12,0:PRINTCHR$(19);"TANK BATTLE"
250 CURSOR9,2:PRINT"By Michael Hadrup
260 CURSOR11,19:PRINT"S START GAME"
270 CURSOR9,20:PRINT"C CONTINUE  GAME
280 CURSOR8,21:PRINT"1 OR 2 DEFINE KEYS
290 CURSOR14,22:PRINT"D DEMO";:GOSUB1230
```

```
300 CURSORX,Y:PRINT">"
310 FORN=1TO3000:A$=INKEY$:IFA$<>""THEN370
320 NEXT
330 POKEA0,22:POKEA1,22:P1=22:P2=22
340 POKEA7,160:POKEA4,100:POKEA5,100
350 POKEA6,0:POKEA6+1,10
360 MAXL=2:GOSUB1230:GOTO920
370 CURSORX,Y:PRINT" "
380 IFA$=CHR$(29)ANDY<15THENX=0
390 IFA$=CHR$(28)ANDY<15THENX=20
400 IFA$=CHR$(31)ANDY<17THENY=Y+1:IFY=14
THENX1=X:X=9:Y=15
410 IFA$=CHR$(30)ANDY>6THENY=Y-1:IFY=14
THENY=13:X=X1
420 IFA$="S"THEN920
430 IFA$="C"THEN1190
440 IFA$="D"THEN330
450 IFA$="1"THEN760
```

```
460 IFA$="2"THEN750
470 IFA$<>" "THEN300
480 IFY=6THENA=2
490 IFY=7THENA=1
500 IFY=8THENA=8
510 IFY=9THENA=4
520 IFY=10THENA=16
530 IFY=11THEN700
540 IFY=12THENA=64
550 IFY=13THENA=32
560 ONY-14GOTO660,640,730
570 IFX=0THENP1=P1XORA:A=P1:Z=A4
580 IFX=20THENP2=P2XORA:A=P2:Z=A5
590 BEEP:POKEA0,P1:POKEA1,P2
600 POKEA2,P1:POKEA3,P2
610 C=Y-5:GOSUB1260
620 IFC<10THENCURSORX+2,Y:PRINTA$
630 GOTO300
```

```
640 CURSOR0,22:PRINTCHR$(21);:INPUT"New blocks = ";A:IFA<1ORA>255THEN640
650 POKEA7,A:GOTO690
660 CURSOR0,22:PRINTCHR$(21);:INPUT"New delay = ";A:IFA<1ORA>20000THEN660
670 CURSOR0,22:PRINTCHR$(21):CURSOR14,22:PRINT"D DEMO
680 POKEA6,AMOD256:POKEA6+1,INT(A/256)
690 CURSOR0,22:PRINTCHR$(21):CURSOR14,22:PRINT"D DEMO":GOTO610
700 CURSOR0,22:PRINTCHR$(21);:INPUT"New range = ";A:IFA<1ORA>255THEN700
710 Z=A4:IFX=20THENZ=A5
720 POKEZ,A:GOTO690
730 CURSOR0,22:PRINTCHR$(21);:INPUT"Max lives = ";MAXL:IFMAXL<1THEN730
740 GOTO690
750 Z=B8:Z1=B7:C=P1:GOTO770
760 Z=B7:Z1=B8:C=P2
770 RESTORE910:FORN=0TO5:READA$
780 CURSOR16,22:PRINTCHR$(21);A$
790 CALLB6:IFPEEK(A8)=255THEN790
800 IFPEEK(A8)=254THEN890
810 A=PEEK(A8):B=PEEK(A8+1):IFN=0THEN840
820 FORM=0TON-1:IF(PEEK(Z+M*2)=B)AND(PEEK(Z+M*2+1)=A)THEN780
830 NEXT
840 IFCAND2THEN870
850 FORM=0TO5:IF(PEEK(Z1+M*2)=B)AND(PEEK(Z1+M*2+1)=A)THEN780
860 NEXT
870 BEEP:POKEZ+N*2,B:POKEZ+N*2+1,A:NEXTN
880 CURSOR0,22:PRINTCHR$(21):CURSOR14,22:PRINT"D DEMO":GOTO300
890 CALLB6:IFPEEK(A8)<>255THEN890
900 GOTO780
910 DATA Visible,Turn left,Turn right,Forward,Backward,Fire
920 L1=MAXL:L2=L1:SCREEN2,2:COLOR15,3,,1
930 IF(P1AND32)OR(P2AND32)=32THENCOLOR14,1,,1
940 CLS:CALLB0:REM Blocks
950 CALLB1:REM Reset
960 CALLB2:REM Set screen
970 CURSOR16,0:PRINTCHR$(17);"P1";L1
980 CURSOR88,0:PRINT"P2";L2
990 CURSOR160,0:PRINTCHR$(16);"\ exits to menu
1000 CALLB3:REM Play
1010 FORN=1TO100:NEXT
1020 WHAT=PEEK(A8):IFWHAT<>0THEN1040
1030 CFLAG=1:SCREEN1,1:GOTO300
1040 IFWHAT=1THENL1=L1-1
1050 IFWHAT=2THENL2=L2-1
1060 IFWHAT=3THENL1=L1-1:L2=L2-1
1070 IF(L1<>0)AND(L2<>0)THEN950
1080 IFL1=0THENA$=" P2 Wins":IFP2AND2THENA$="  I win"
1090 IFL2=0THENA$=" P1 Wins":IFP1AND2THENA$="  I win
```

Cartridge Basic Users add

```
1480 FORN=0TO63:A=PEEK(&H4020+N)
1520 C=&H1800+PEEK(&H3FA0+N)*8
```

Disk Basic Users add

```
95 IF(PEEK(&HE500)XORPEEK(&HE501))<>158
THENLOADM"TANK.Cde":LOADM"TANK.Spr"
```

LSV Users add

```
95 IF(PEEK(&HE500)XORPEEK(&HE501))<>158
THEN*LOADC"TANK.Cde":*LOADC"TANK.Spr"
```

```
1100 IFL1=0ANDP1AND2THENA$=" You win
1110 IFL2=0ANDP2AND2THENA$=" You win
1120 IF(L1=0)AND(L2=0)THENA$="Both lose
1130 CALLB1:CALLB4:REM Reset,Spr
1140 BLINE(48,56)-(224,136),,BF
1150 LINE(56,64)-(216,128),,B:LINE(60,68)-(212,124),,B
1160 CURSOR80,95:PRINTCHR$(17);A$
1170 CALLB5:CFLAG=0:FORN=1TO400:NEXT
1180 CFLAG=0:SCREEN1,1:GOTO300
1190 IFCFLAG=0THENBEEP2:GOTO300
1200 SCREEN2,2:COLOR15,3,(8,0)-(255,191)
1210 IF(P1AND32)OR(P2AND32)=32THENCOLOR14,1,(8,0)-(255,191)
1220 GOTO960
1230 CURSOR2,4:PRINT"Player 1":Z=A4:A=P1:B=2:GOSUB1250
1240 CURSOR22,4:PRINT"Player 2":Z=A5:A=P2:B=22
1250 PRINT:FORC=1TO8:GOSUB1260:PRINTTAB(B);A$:NEXT:GOSUB1430:GOSUB1440:GOT
O1450
1260 ONCGOTO1270,1290,1310,1330,1350,1370,1390,1410,1260,1430,1440,1450
1270 A$="COMPUTER":IF(AAND2)THENA$="COMPUTER
1280 RETURN
1290 A$="FIRE BEND":IFAAND1THENA$="FIRE BEND
1300 RETURN
1310 A$="- PARTIAL":IFAAND8THENA$="- PARTIAL
1320 RETURN
1330 A$="FIRE BOUNCE":IFAAND4THENA$="FIRE BOUNCE
1340 RETURN
1350 A$="RANDOM RANGE":IFAAND16THENA$="RANDOM RANGE
1360 RETURN
1370 A$="- RANGE ="+STR$(PEEK(Z))+"  "
1380 RETURN
1390 A$="SLOW FIRE":IFAAND64THENA$="SLOW FIRE
1400 RETURN
1410 A$="NIGHT BATTLE":IFAAND32THENA$="NIGHT BATTLE
1420 RETURN
1430 CURSOR11,15:PRINT"Delay = ";PEEK(A6)+256*PEEK(A6+1);"     ":RETURN
1440 CURSOR11,16:PRINT"Blocks = ";PEEK(A7);"  ":RETURN
1450 CURSOR11,17:PRINT"Maxlives =";MAXL;"  ":RETURN
1460 REM Inverse char CORRECT VERSION
1470 IFVPEEK(&H1D00)=223THENRETURN
1480 FORN=0TO63:A=PEEK(&H5D32+N)
1490 IFA=0THEN1550
1500 IFA=32THEN1550
1510 B=&H1D00+(A-160)*8
1520 C=&H1800+PEEK(&H5DB2+N)*8
1530 FORM=0TO7:VPOKEB+M,VPEEK(C+M)XOR255
1540 NEXTM
1550 NEXTN:RETURN
```

Type the bold text with the ENG DIER's key to get inverse characters.

See alterations on page 23 for Cartridge Basic.

Data Listing

Δ

```
10 CLS:PRINT:PRINT:PRINT"Storing machine code and sprites":PRINT:PRINT"PLE
ASE WAIT"
20 RESTORE1870
30 FORN=0TO95:READA$:PATTERNS#N,A$:NEXT
40 RESTORE1000:X=&HE500:L=1000
50 GOSUB500
60 X=&HEE30:L=L+10
70 GOSUB500
80 STOP
500 CURSOR0,0:PRINTHEX$(X),L
510 READA$:IFA$="*"THENRETURN
520 C=0:FORN=1TO64STEP2
530 POKEX,VAL("&h"+MID$(A$,N,2))
540 C=C+PEEK(X):X=X+1:NEXT
550 READA$:IFC<>VAL("&h"+A$)THENPRINT"Error in line";L:STOP
560 L=L+10:GOTO500
```

Cartridge Basic Users add

```
75 CLS:PRINT:PRINT"LOAD":PRINTCHR$(11);
```

```
1000 DATA ED7370E5F3CD54ECCD3EE6CD43EBCDCEE7CDACE9CDC3E9DBBFF5CB6FC42EE7F1
,1791
1010 DATA CB7F28F3CDEAE521000ADBBFCB7FC4EAE52B7CB520F4CDCEE7CD73E9CD89E7CD
,14C8
1020 DATA A6E9CDBDE9CD6CE83AD1EC473ADDEC4FED5F80CB111F274710FE274710FE4710
,10C9
1030 DATA FECD5AE83E05D3DEDBDD1FDA0BE5AF3106A3327CE5C33EE6327CE5C9FFFF2191
,12B1
1040 DATA EC11FFFF0608783DD3DEDBDCCD9FE5DBDDCD9FE53002CBDA10ECED537CE5C9B6
,1473
1050 DATA 232F4FA7C87A3C11FEFFC05015C506081E00791730011C10FAC11D5937C811FE
,0C16
1060 DATA FFC9AF32EBE532F0E53E0A321CE63EE6D37FC916B41EC03EE6ED793EF0ED79FB
,12FC
1070 DATA 76F3CDF1E57AA720F6C916FF7A3CC81EC00E7F7BA728247BED797AED797AC6F0
,1269
1080 DATA ED791C7BFED038023EC032F0E55F147AFE1020023EFF32EBE557C93E073D2002
,0E2A
1090 DATA 3E0A321CE6C0ED5FE607B20E7FC6F0ED79147AFE1020023EFF32EBE557C90E7F
,0F7A
1100 DATA 3E9FED793EBFED793EDFED793EFFED793EFF32EBE5C91E00FDCB00BEDD7E03FD
,12D3
1110 DATA 960330041E01ED4447DD7E04FD96043004CBCBED444FA73E10281878A7281478
,0BAC
1120 DATA 913805913E0C300B7990380590 3E0430023E0857CB4B28043E209257CB432805
,088F
1130 DATA 3E4092E63F57FD3612017AFD9605281C3002E63FFE203EFC30023E04FD8605E6
,0CB9
1140 DATA 3FCDB9E7FD7705D0FD36120421D0E6C9CD2FE9380B21D0E6FD3512C02156E6C9
,1107
1150 DATA FDCB007EC256E6FD36120821EFE6C9CD51E9380721EFE6FD3512C0ED5FE6013E
,1101
1160 DATA 0820033EF847CD22E7300678ED44CD22E7ED5FE603C602FD7712FDCB00FE21D0
,0F6D
1170 DATA E6C9FD8605E63FCDB9E7FD7705C93ACBEC5F3ACAEC573ADEEC93300EED44CB3F
,12B2
1180 DATA ED446FED44FE07D01806CB3F6FFE07D03ADDEC92300EED44CB3FED4467ED44FE
,10E7
1190 DATA 07D01806CB3F67FE07D07C8232D2EC7D8332D3EC3EBF32CAEC32DDECFD21C7EC
,11CB
1200 DATA DD21DAEC3E03C3FDE7FD21C7ECCD94E7FD21DAECFDCB00462812FDCB005E2806
,1245
1210 DATA ED5FE6072006FD7E05FD7708FDCB006EC8FD360601C9FD7E05D95FFD4603FD4E
,0FA5
1220 DATA 04D57BCDAFEAD17BD9D0FD7E05C9FD21C7ECDD21DAECCDB6E73E01380EFD21DA
,1374
1230 DATA ECDD21C7ECCDB6E7D03E0208FD7E04FD770CFD7E03FD770BFD3603BF0808DD36
,1033
1240 DATA 0BBFCD7FEBCD70EBFD360D40CDC2E5ED5FE60FFD770ECD43EB0602DBBFCB7F28
,11EF
1250 DATA FACDEAE53C281210F2FD340DFD7E0DFE6020DCFD360D5018D608C36FE506067E
,0F55
1260 DATA E607D3DE0EDC7E23FE0838020EDDED78A6200137CB132310E6C92185ECFD21DA
,0F01
1270 DATA ECDD21C7ECCD7EE8D0C370EB2179ECFD21C7ECDD21DAECCD7EE8D0C37FEBCD3D
,1613
1280 DATA E8FDCB004E282ACB6B2806FD7E11FD7706FD6E0FFD6610CDB0E8FD750FFD7410
,100E
1290 DATA FDCB0056CAE9E8FD7E05E60FC8C3E9E8E9FD7E05CB632807D604E63FCDB9E7CB
,1387
1300 DATA 5B2807C604E63FCDB9E7FD7705CB532805CD2FE91805CB4BC451E9CB6B2806FD
,0F21
1310 DATA 7E11FD7706A7CB43C8FD7E07A7C0FD360701AF32EBE53EC032F0E53EE7D37FFD
,11D4
1320 DATA 7E05FD7708210DEDCD9CE9FD7E0481FD770AFD7E0380FD7709AFFD7701FDCB00
,1051
```

```
1330 DATA 6637C8ED5FE61F070707FD770237C9FD7E05D9CD99E978FD86034779FD86044F
,0F7D
1340 DATA FD7E05CDAFEA3807FD7104FD7003A7D9C9FD7E05D9CD99E9FD7E039047FD7E04
,11CC
1350 DATA 914FFD7E05CDAFEA3807FD7104FD7003A7D9C9FD21C7ECCD7EE9FD21DAECFD7E
,1394
1360 DATA 0BFEBFC8ED5FE60FFD770EFD340DFD7E0DFE60C0FD360BBFC921EDECCB3F4F06
,1156
1370 DATA 000946234EC93AC7ECCB77C0CD7FEB3ACEECA7FD21C7ECC4D4E9C37FEB3ADAEC
,13C9
1380 DATA CB77C0CD70EB3AE1ECA7FD21DAECC4D4E9C370EBFD3401FD7E02FDBE012841FD
,142C
1390 DATA 7E08CD99E978FD86094779FD860A4FCD8DEA2007FD710AFD7009C9FDCB005628
,0FDD
1400 DATA 25FD4E0AFD4609FD7E08CD3DEAFD7208AF32EBE53EC032F0E53EE7D37FC3D4E9
,1261
1410 DATA FD4E0AFD460979D608FD770C78D608FD770BFD360D40FD360700C3C2E5F50D0D
,0E80
1420 DATA CD8DEAF50C0C0C0CCD8DEAF50D0D0505CD8DEAF504040404CD8DEA075FF1B307
,0E64
1430 DATA 5FF1B3075FF1B3075FF1577BA72804FE0F2006ED5FE63C57C97BE60C28043E40
,0DE1
1440 DATA 92577BE603C83E2092E63F57C9CD30EB79E60732A3EA7BD3BF7AD3BF00000000
,0F70
1450 DATA DBBE18060F0F0F0F0F0F0F0FE680C908CD30EBC5E579C60FE6070732DFEAFE0E
,0D41
1460 DATA 20043EF8835FD52100EEE5086F26002929291130EE19061056235E23EBAF1802
,0A29
1470 DATA 298F298F298F298F298F298F298FEBE3772C722C732CE310DFF1D12100EE01BE
,0E18
1480 DATA 10C506037BD3BF7AD3BF7BC6085F00ED78A620172C10ED7BD6175FE60720057B
,0D5E
1490 DATA D6085F14C110DAE1C1A7C9F1E1C137C979E6F85F783C0F0F0FE61F57783CE607
,1035
1500 DATA 835FC9AFD3BF3E7BD3BF01BE0421CAECEDA320FC060421DDECEDA320FC060421
,1048
1510 DATA D2ECEDA320FC060421E5ECEDA320FCC93AE1ECA7C83AE3EC473AE4EC4F180D3A
,1259
1520 DATA CEECA7C83AD0EC473AD1EC4FCD30EB7BD3BF7AD3BF79E60732ABEBDBBE4F0000
,12C3
1530 DATA 7BD3BF7AF640D3BF3E011806070707070707A9D3BEC92100011EA0ED5F4FED
,0C48
1540 DATA 5F86A8A9CB37854F23AE0F5779E6F8D3BF7AA9E61FF640FE583802D618D3BF01
,109B
1550 DATA BE08E52165ECEDA320FCE11D20D11600213849CDF9EB21B84BCD05EC24CD05EC
,0FE5
1560 DATA 24CD05EC24CD0DEC0618CD15EC7DD3BF7CD3BF06087AD3BE00000010F9C92100
,0DE1
1570 DATA 4055CD0DEC06F8CD16EC1506F8CD15EC21F8410E16CD0DEC0610CD15EC240D20
,0D82
1580 DATA F406F0CD15EC2100600E1855CD0DEC240D20F9C93E02D3BF3E80D3BF3EE2D3BF
,0F61
1590 DATA 3E81D3BFC9003C7E7E7E3C000021A1EC11C7EC012600EDB0C907200704070807
,0B53
1600 DATA 01070207100F080F010F02074007800F0800F090F190F610FE00FE00FE00FE00
,093E
1610 DATA FE1400644F4020060000000BF00400456E606011400645FC0000400000000BF
,06CB
1620 DATA 00400056E604011400644F4018060000000BF00400456E606011487F853BA24
,07B0
1630 DATA 0401201E7EBFB0600A56E6040000020102020202010200002FF02FE01FE00FEFF
,08E5
1640 DATA FEFEFEFEFFFE00FE01FE02FF0208100B101010100B1007100310FF0BFF08FF03
,0CB0
1650 DATA FFFFFFFF03FF07FF0BFF10031016CAB909210000CD3E84FE0D2009110A00010A
,0BDD
```

```
1660 DATA *
1670 REM
1680 DATA 0000000000000007FE07FE01FC03FF03FFF3FF01FC07FE07FE000000000000
,0AD6
1690 DATA 000000018001E000F801FE03FF83FF8FFE0FFE07FF81FFE078601E000600000
,0C52
1700 DATA 00000200070003800 3C00FE02FF07FF83FFC1FE80FE007F0039801CC00840000
,0AE8
1710 DATA 0000004000E003E037F03FF01FF81FF80FFC0FFC07F607F6033003300018001 8
,0B2D
1720 DATA 000000018301BB01FF01FF01FF01FF01FF01FF01FF01BB003800100010001 00
,09CD
1730 DATA 00000200070007C00FEC0FFC1FF81FF83FF03FF06FE06FE00CC00CC018001800
,0BC8
1740 DATA 0000008001C0038007E00FE41FFE3FFC7FF82FF00FE019C03380610000000000
,0B68
1750 DATA 00000018007801F007F81FFC1FFC07FE07FF1FFE7FF861E00780060000000000
,0B23
1760 DATA 0000000000000007FE07FE03F80FFCFFFC0FFC03F807FE07FE000000000000
,0A1B
1770 DATA 0000000006000780 61E07FF81FFE07FF07FE1FFC1FFC07F801F0007800180000
,0B23
1780 DATA 0000000061003380 19C00FE02FF07FF83FFC1FFE0FE407E0038001C000800000
,0B68
1790 DATA 0C000C00066006603 7F037F01FF81FF80FFC0FFC07FE07F603E003800100000 0
,0ADF
1800 DATA 0100010001000380 1BB01FF01FF01FF01FF01FF01FF01FF01BB0183000000000
,09CD
1810 DATA 0030003006600 6600FEC0FEC1FF81FF83FF03FF07FE06FE007C001C000800000
,0C64
1820 DATA 0000008401CC0398 07F00FE01FE83FFC7FF82FF00FE003C0038007000200000 0
,0AE8
1830 DATA 0000006001E00786 1FFE7FF8FFE0FFE03FF83FF81FE00F801E001800000000 0
,0C52
1840 DATA *
1850 REM
1860 REM
1870 DATA 000000007F751F38,37381F757F000000,00000000E060C070,BF70C060E0000
000
1880 DATA 0000181E0B1E3936,57F96E1B06010000,0000000080E0B878,A0A078FE86E06
000
1890 DATA 00020702030E2D7B,2B1D0A0702010000,00000080C0A070A8,BCA860F098CC8
400
1900 DATA 00000003373E151D,0A0E050703030000,0040E060D070A8B8,D4DC36F630301
818
1910 DATA 0000181B1F161D15,1D151E1B03010101,000030B0F0D07050,7050F0B080000
000
1920 DATA 000207060B0E151D,2B3B6C6F0C0C1818,000000C0EC7CA8B8,5070A0E0C0C00
000
1930 DATA 00000102070A1D2B,7B2C0F1933610000,0080C080E064BED4,B850E04080000
000
1940 DATA 00000001071D1E05,051E7F6107060000,001878D0789C6CEA,9F76D86080000
000
1950 DATA 000000000706030E,FD0E030607000000,00000000FEAEF81C,EC1CF8AEFE000
000
1960 DATA 00000607617F1E05,051E1D0701000000,0000008060D8769F,EA6C9C78D0781
800
1970 DATA 00006133190F2C7B,2B1D0A0702010000,0000008040E050B8,D4BE64E080C08
000
1980 DATA 0C0C060637361D15,0E0A070503030100,00006060F050B6A8,DCD43EF660800
000
1990 DATA 010101031B1E151D,151D161F1B180000,00000080B0F05070,5070D0F0B0300
000
2000 DATA 000006060F0A1D15,3B2B7C6F06010000,30306060EC6CB8A8,7050E0A0C0C08
000
```

```
2010 DATA 00000102070A1D2B,7B2D0E0302070200,0084CC98F060A8BC,A870A0C080000
000
2020 DATA 000001061B6EF957,36391E0B1E180000,0060E086FE78A0A0,78B8E08000000
000
2030 DATA 0000000001010303,0F03000000000000,000000000000C0F0,C0C0808000000
000
2040 DATA 0000000804070706,06070F0800000000,008080C0C8D07060,60E0F0C880808
000
2050 DATA 0000000001010303,0F03000000000000,000000000000C0F0,C0C0808000000
000
2060 DATA 00000131190F0E0C,04070F0F08100000,0004083870F06020,6060E0E0E0701
800
2070 DATA 03071F2C2B4FDCFB,F7F97F2F331F0703,C0A0F8F4FC8EF7FF,FF9FEEFCEC90E
080
2080 DATA 03071F2D3B7BBBFF,FF5D5D2B1F1F0703,C0E098C4FC6EB7B7,FFDDDEDCB4F8E
0C0
2090 DATA 03051F2F3F71EFFF,FF5D5D2B1F1F0703,C0E0F8CCF4FE9FEF,DF3BF2D434F8E
0C0
2100 DATA 000000003070B1E,0F0F050301000000,0000080C0A0D0F0,78F0E0C000000
000
```

Δ

~~BOO BOO'S~~

Any Cartridge Basic Users typing in the Graphics Screen Flip program would have found lines 90-130 created a few problems. They were not required, however LSV users may want to save the code using `*SAVE "Flip.Code",&HED00,&HED9F`

A *boo boo* in the Boo Boo's, the bit about the MC Editor should have read ...
Line 180 of the MC Editor (Figure 1.2) should have read `180 PRINT:GOTO50`

and of course a few spelling mistakes.

PS I have given up trying to correct the Inverse Characters program, but you can find it being used at the end of Tank Battle

Δ

In the next issue

No idea!
Hopefully what ever I can finish writing
between now and the next magazine
and what ever else you send in!

More on machine code programming and hints

Due out about May

Poseidon Software

NZ SEGA DISTRIBUTORS

FREEPOST 243 P.O. BOX 277 TOKOROA NEW ZEALAND

Specials

Vermin Invaders	*16K and 32K*	$15.00
Burgular Bill	*16K and 32K*	$19.00
Carverns of Karanor	*16K and 32K*	$21.00
Vortex Blaster	*32K only*	$15.00
Aerobat	*32K only*	$21.00
Orb of Fear	*32K only*	$12.00
Castle of Fear	*32K only*	$12.00
Castaway	*32K only*	$12.00
Burgular Bill	*Disk version*	$33.00
Caverns of Karanor	*Disk version*	$33.00
Michael Howard's - More than 50 Programs		$5.00
Book and Tape - Teach Yourself BASIC Programming		$6.00
LSV, Print 64 and Pattern Paint	*Disk / Tape versions*	$15.00
Magazine programs on cassette		$20.00
Machine Code Summary Sheets		$1.00

	Australia	New Zealand	
Magazine subscription (6 issues)	A$26	NZ$25	GST incl
One issue	A$7	NZ$6	

Name ______________________ **Phone** __________

Address ______________________________

____

Item	No. of items	$ Total

Add Postage **$2.50**

Payment by (circle one) **Total Enclosed $**________

Cheque Cash Bankcard Visa

Orders over $20.00 only

VISA

BANKCARD Welcome here

Credit Card No. ______________________

Signed ________________ Expires / /