# SEGA® Computer

October Issue 1987

**The official magazine of the SEGA User Club**

- *Details about the new 'my card' games*
- *64 Column printing on the screen*
- *Astro Attack by John Dowman*
- *Machine code programming course*


*Published by MJH Software in association with Poseidon Software*

Published bi-monthly by **MJH Software**
Auckland New Zealand
Telephone (09) 534-3379
All correspondence to **Poseidon Software**
P.O. Box 277
Tokoroa New Zealand

- 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.
- 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.

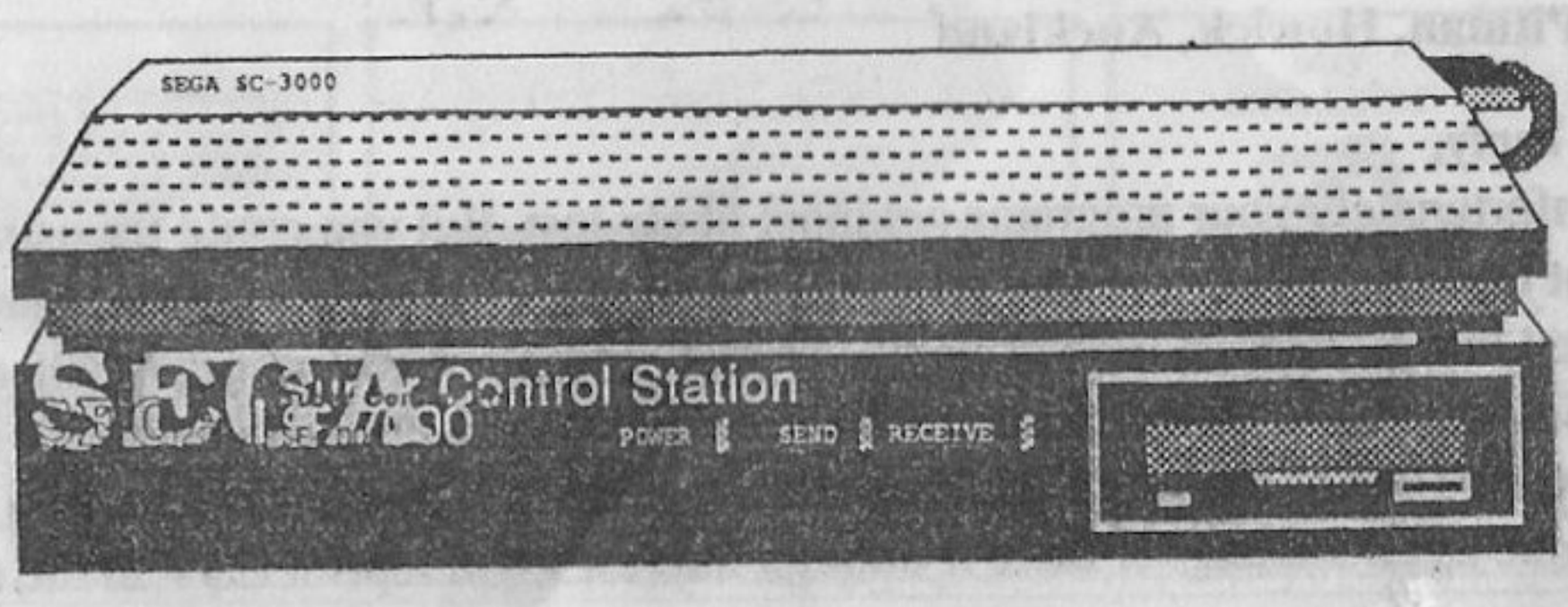
Welcome to the new look

# SEGA®
## Computer

**The official magazine of the SEGA User Club**

## Contents





# 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.**

**Dear Editor,**

(a) In one of your previous issues, the news and reveiws section stated that the new 'my card' games would be available some time in the future. What has happened to them since that article was published about July/August in 1986!

(b) Will there be a course on machine code programming in coming magazines. I would really like to learn some more about my SEGA. Could you please make it as simple as possible though, as previous explainations were sometimes a little difficult to understand.

**Simon Sharp, Bucklands Beach, Auckland**

*Editors reply,*

(a) Yes, the new 'my card' games are available for hire from Poseidon Software. See the News Section on page 5.

(b) There seems to be a lot of interest in machine code programming, so starting in this issue is a course on machine code programming that starts at the very basics. See page 6.

**Dear Editor,**

Lately I have had problems loading all my software.The programs which used to load from my National Pansonic tape deck won't load now. Adjusting the volume and tone makes no difference. They all seem to load until the very end, when a tape read error occurs. Is there some way that I can use these programs again. By the way I have a 32K cartridge.

**David Pitman, Howick, Auckland**

*Editors reply,*

Unfortunately your programs are most likely lost. But you could try using a different tape deck. There is another solution which should solve future problems. Published in the October 1985 issue of SEGA Computer was a program called LSV (Load-Save-Verify). This program makes loading and saving much easier by using extended commands such as *Load and *Save. It is much more reliable - you can change the volume as its loading! If there is enough interest I will reprint LSV in the next issue.

# EDITORIAL

Here it is! After a lot of hard work the first issue of the 'new look' SEGA Computer is published. The future of the magazine lay in limbo for a long period of time, but all that work was not in vain, as you can see.

The inital response of subscriptions to the old magazine was very disappointing with less than half of the previous readers re-subscribing. I feel this was due to the high price of the previous magazine (this was necessary as a colour gloss magazine is not cheap to produce) and the quality of service that was lacking. Be assured there will be no 'double issues' from MJH Software to cheat you out of your six magazines.

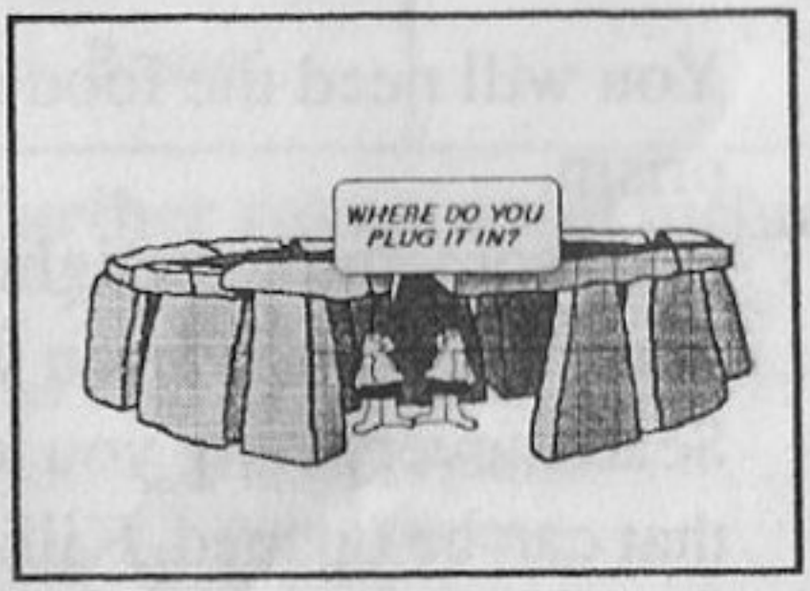
We delayed as long as possible, in an attempt to keep the original magazine going, but eventually it became obvious to most of us that this was not to be. It was time to come up with a new solution and here it is. The colour has gone and the pages are smaller, but the information on the pages has been greatly improved!

By the way, for those of you that don't know me, I'm a 7th Former at Macleans College in Auckland and I have never produced a magazine before - this will be obvious to some, it certainly is time consuming. So, if you have any ideas or *constructive* criticism, then put it on paper and send it in.

**EDITOR: Michael Hadrup**

MJHadrup

# MAN LOGIC *Original Idea by Neil Bradley*


# Adventure Section

*By an annonymous reader*

**• Hello fellow adventurers out there this is Gimbal the Wizard. With the aid of my trustee band of adventurers we will help you in your travels in adventureland with reviews of new adventures, tips to get you started and a help line to get you out**

I have seen ATFGROS Software's debut game **The Tomb Of Nozar** and on first impressions it looks a good game. Great graphics, a great parser and average puzzles. The parser is the part of the program that accepts your input and turns it into something the computer can understand.

One thing I like about the game is the ability to fight and battle the other characters. (You need to kill a few characters to get anywhere in the game.) The puzzles in the game are fairly easy but some of the responses to certain inputs are quiet humorous.

- Title: The Tomb Of Nozar
- System: 32K
- Price: $24.95  Available from Poseidon Software

Just recently I played Time Capsule and solved it within three hours of seeing it for the first time. I have also completed Death Satellite and you will find some clues below for the previously mentioned three games.

***CLUES:***

**TIME CAPSULE :** You will need the food and the hose to get the prism.

**DEATH SATELLITE :** Push something straight away and look. Wear what you find and you will be protected

**TOMB OF NOZAR :** Search everything you kill and open anything that can be opened. Killing the Elf is not the only way to get the rope. Δ

# NEWS

SEGA has long been one of the market leaders in the production of arcade games. Therefore it has been with disappointment that a number of people have noted that SEGA in Japan has reprogrammed few, if any of these games for their own computer system. The latest offering from Poseidon Software shows that SEGA in Japan is still supporting the SC-3000.

Available for hire from Poseidon Software will be over 25 games, on cartridge, including a number of amazing arcade conversions. At first only five titles will be released, followed by five new titles each month - see below.

## *Only $20 to Join*

For $20, you can become a member of the SEGA Hire Club and this entitles you to hire any game of your choice. The rate of hire per game is $7.50 for 8 days and this includes the return postage and packing. However, for each day the *Post Back Date* is exceeded, there is an extra charge of $1.50.

For the time being the 'my card' games cannot be purchased directly, but Geoff at Poseidon Software tells me there is a good chance that they will be available in the future.

*In the next issue I will review some of the games available.*

**The first five releases are :**

Bank Panic
Droll
Penguin Land
Lode Runner
Championship Lode Runner

**The next five releases will be :**

GP World
SEGA Galaga
Choplifter
Dragon Wing
Zoom 909

**Further releases will include :**

| | | |
|---|---|---|
| Hyper sports | Hustle Chummie | Zaxxon |
| Zippy Race | BombJack | C-SO! |
| Chack 'n' Pop | Elavator Action | Hangon 2 |
| Ice Hockey | Nija Princess | Pitfall 2 |
| Thinking Rabbit | Wonderboy | Rock 'n' Bolt |

# Machine Code Programming

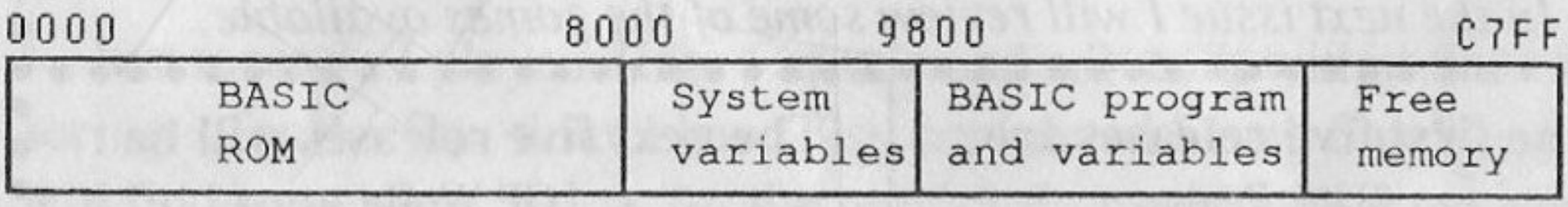
*By Michael Hadrup*

**• This is a new course for those people who a finding the gap between BASIC and machine code difficult to master. It is designed for people that have a reasonable understanding of**

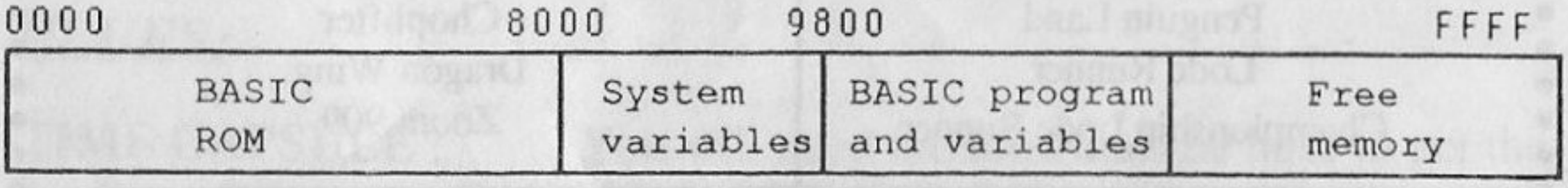
The first important fact you should know about your SEGA Computer is that someone forgot to include a button labled *"Go into MACHINE CODE MODE"*. Therefore we need to write a few programs, so that it can accept machine code instead of BASIC. For the moment these will be BASIC programs, but in the future we will develop machine code programs which will help us to write more machine code programs.

Before we can do anything, we need to know where to put our machine code programs. Depending on the memory available and whether you own a disk drive there a certain limitations to where we can put out programs. Study these simple diagrams:

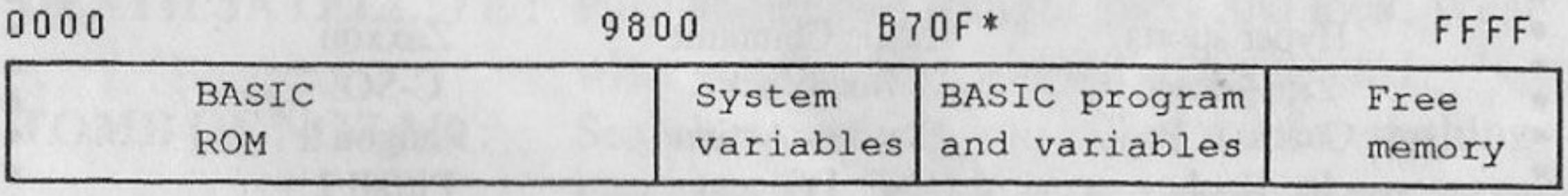
**16K Cartridge**




**32K Cartridge**




**SF-7000 Super Control Station**




* This varies according to maximum number of files. See page 210 Disk Basic manual

**Figure 1.1**

OK - now were ready to start. Type in the following BASIC program.

```
10 REM MC Editor
20 REM
30 RESTORE1000
40 X=&HC000
50 READA$
60 IFA$="*"THENEND
70 IFLEFT$(A$,1)=":"THEN190
80 IFLEN(A$)MOD2=0THEN110
90 PRINT"Invalid data"
100 STOP
110 PRINTHEX$(X);":";
120 FORN=1TOLEN(A$)STEP2
130 POKEX,VAL("&H"+MID$(A$,N,2))
140 IFPEEK(X)<16THENPRINT"0";
150 PRINTHEX$(PEEK(X));
160 X=X+1
170 NEXT
180 GOTO50
190 IFLEN(A$)<>5THEN90
200 X=VAL("&H"+MID$(A$,2))
210 GOTO50
9999 DATA *
```

**Figure 1.2**

Can you see how the program works? Or at least what it does? In brief - it will accept a machine code program (in the form of hex data stored at line 1000 onwards) and will store it at &HCOOO.

Try it - Add one of the following lines.

**16K or 32K users**

```
1000 DATA 3E02C3A056
```

**Disk Basic users**

```
1000 DATA 3E02C31E53
```

Now run the program
- the SEGA responds with

```
C000:3E02C3????
```

Where the ???? depends on what SEGA you have.

Now type `CALL &HC000`
If you hear a beep then your first machine code routine worked.

```
C000 LD A,2
C002 JP Beep routine
```

You can change where the machine code is stored by adding a line such as `1000 DATA :D000`
Now the machine code will be stored at &HD000

If you didn't understand the example above, don't worry for now. Read the following section then return to the example later.

## *What exactly is Machine Code?*

Now for the big question you've been dying to ask. Well machine code, or machine language to others, is simply another computer langauge - much like BASIC - only at a much lower level. This means that instructions even slightly complex, (for instance FOR / NEXT loops) are not available in the form you would expect. However, this makes it an easy language to learn - but a difficult language to master.

Like BASIC, machine code consists of a set of instructions, which tell the computer what to do. One such instruction is RET, which is more or less the equivalent to BASIC's RETURN.

Unlike BASIC the computer cannot understand machine code as it is written in 'English'. (However there are programs called 'Assemblers' which will do this). Each machine language instruction has a numerical code associated with it, which the computer can understand directly. (For example the code for RET is 201). Every code lies between 0 and 255. It more convenient to write these codes in a system called *hexadecimal.*

## Counting in Hexadecimal

Briefly hexadecimal, or hex for short, is a means of counting using sixteen symbols instead of ten. The six new symbols which represent the numbers ten to fifteen are **A B C D E F**

The fun really starts when we want to represent numbers bigger than fifteen, for believe it or not, sixteen is written as 10! Worse still, seventeen is written as 11 and eighteen as 12. This continues to twenty five - written as 19 - before we have to start using the new symbols again - twenty-six is written 1A and so on.

If there is any confusion about whether a number is written in hex or decimal, then you should make it clear by writting a small 'h' after the number. Some people prefer to write a '#' before the number (I do anyway) or in the case of the SEGA '&H'.

## Counting in Binary

The concept of binary is something a lot of people seem to have trouble with. So study these tables below about 8 bit binary or 'bytes'

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Power to base 2 | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| Value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

To work out the value of the binary number 11111111, add the values of bits 0-7 (all the bits that are 1) and you get a total of 255 decimal.

128+64+32+16+8+4+2+1 = 255

It so happens that this is the maximum value an 8 bit binary number can hold. You can work out any binary number this way, such as 1011011. There trick is to remember that you need 8 binary digits (there is only 7 in the number above) so you have to add leading zeros to give 01011011. Now you can work it out. 0+64+0+16+8+0+2+1 = 91

Converting binary to hex is just as easy. Every hex digit can be represented by 4 binary bits called 'nibbles'

| Hex Digit | Binary 'nibble' | Hex Digit | Binary 'nibble |
|---|---|---|---|
| 0 | 0000 | 8 | 1000 |
| 1 | 0001 | 9 | 1001 |
| 2 | 0010 | A | 1010 |
| 3 | 0011 | B | 1011 |
| 4 | 0100 | C | 1100 |
| 5 | 0101 | D | 1101 |
| 6 | 0110 | E | 1110 |
| 7 | 0111 | F | 1111 |

To convert the binary number 1011011 to hex, (remember to add a leading zero) break the binary number into two parts - 0101 and 1011 giving #5B.

You can turn hex into binary by using the same simple rule. For example the hex number #6A can be converted to binary by joining the two nibbles 0110 and 1010 to give 01101010.

Binary numbers are often shown by a small 'b' after the number. Some people prefer to write a '&' before the number (I do anyway) or in the case of the SEGA, it won't except binary!

## *Crashing the Computer*

There are some fundamental differences between machince code and BASIC. One of these fundamental differences is that of *line numbers*. As you know, every BASIC statement in a program must be preceded by a line number so that the computer knows in which order to execute them. In machine code, there are no line numbers. The instructions will be executed in the order that they are stored. For example, if the computer has just executed the instruction held at #C000, it would then go on to the instruction held in #C001. It will continue like this until is it instucted to do otherwise.

Unlike BASIC it will not stop when it reaches the end of the program. It will continue trying to execute instuctions and this often results in a crash. Even a minor mistake in in your machine code will confuse the poor machine and it may go into its "*LET'S PRODUCE SOME MODERN ART MODE* "

Luckily the SEGA has a RESET key, which in most cases will restore some order inside your SEGA. There will be times however when the only way to get back to normal, will be to switch the computer on and off. So the golden rule is SAVE IT before you CALL IT and ensure that your programs are bug-free.

**Next Issue :** Hopefully you will get to learn some actual machine code in the next lesson and there will be some example programs for you to type in. The next lesson covers registers, adding, subtracting, peeking and poking. Δ

# Program of the month
# ASTRO ATTACK

*By John Dowman*

**• For 16K, 32K or Disk Basic users**

**A good BASIC game which deserves to win the program of the month. John will receive his $40 in the mail soon.**

```
10 REM ********************************
20 REM ********************************
30 REM ***          BY              ***
40 REM ***      JOHN DOWMAN         ***
50 REM ***          OF              ***
60 REM ***       OAMARU             ***
70 REM ***        1986              ***
80 REM ********************************
90 REM ********************************
100 PATTERNC#48,"708898A8C88870"
110 PATTERNC#79,"70888888888870"
120 PATTERNC#144,"000000000000FF"
130 SCREEN 2,2:COLOR11,1,,1:CLS
140 Z$="ASTRO ATTACK!":SC=0:HS=0:ME=5:XX=150:YY=174:X1=0:Y1=0:C=0
150 COLOR,12,(20,110)-(215,113)
160 COLOR,12,(20,87)-(215,90)
170 V=23:P=0:SOUND0
180 FOR Z=1 TO LEN(Z$)
190 COLOR11,1
200 S$=MID$(Z$,Z,1)
210 FOR X=240 TO V STEP -20
220 SOUND1,X+90,15
230 CURSORX,96:PRINT CHR$(17);S$
240 CURSORX,96:PRINT CHR$(8)
250 NEXT X
260 CURSORV,96:PRINT CHR$(17);S$
270 V=V+15
280 NEXT Z
290 SOUND0
300 FOR Z=23 TO 203 STEP 15
310 LINE (127,191)-(Z+7,101),8
320 BLINE (127,191)-(Z+7,101)
330 CURSORZ,96:PRINT CHR$(8)
340 SOUND1,Z+90,15
```

```
350 NEXT Z
360 SOUND0
370 GOSUB 1020:GOSUB 2310
380 SCREEN 2,2:CLS
390 SPRITE0,(50,5),4,10
400 SPRITE1,(50,30),8,9
410 SPRITE2,(50,60),12,5
420 SPRITE3,(50,90),16,15
430 SPRITE4,(50,120),20,13
440 SPRITE5,(50,140),24,8
450 SPRITE6,(50,160),0,14
460 COLOR10:CURSOR70,10:PRINT CHR$(16);"= ?? Pts.":BEEP
470 COLOR9:CURSOR70,35:PRINT "= 50 Pts.":BEEP
480 COLOR5:CURSOR70,65:PRINT "= 40 Pts.":BEEP
490 COLOR15:CURSOR70,95:PRINT "= 30 Pts.":BEEP
500 COLOR13:CURSOR70,125:PRINT "= 75 Pts.":BEEP
510 COLOR8:CURSOR70,145:PRINT "= 100 Pts.":BEEP
520 COLOR15:CURSOR70,165:PRINT "= Your Ship":BEEP
530 RESTORE 610:Z=0
540 FOR U=1 TO 15
550 READ Z
560 SOUND1,Z,15:FOR I=1 TO 50:NEXT I
570 SOUND2,Z,15:FOR I=1 TO 50:NEXT I
580 SOUND3,Z,15:FOR I=1 TO 50:NEXT I
590 NEXT U
600 SOUND0
610 DATA 262,330,392,330,262,440,494,523,494,440,392,330,392,330,2
62
620 FOR I=1 TO 3
630 T$=TIME$
640 IF TIME$=T$ THEN 640
650 SOUND1,440,15
660 FOR U=1 TO 10
670 NEXT U
680 SOUND0
690 NEXT I
700 T$=TIME$
710 IF TIME$=T$ THEN 710
720 FOR I=15 TO 0 STEP -.2
730 SOUND1,880,I
740 NEXT I:SOUND3,110,0
750 FOR I=1 TO 500:NEXT I
760 SCREEN1,1:CLS:CURSOR7,15:COLOR13,1:PRINT "PLEASE WAIT 30 SECON
DS":SCREEN 2,1:COLOR,1,,1:CLS:COLOR,7,(0,0)-(40,191),7
770 FORI=1 TO 150
780 A=INT(RND(1)*210+45):B=INT(RND(1)*191):C=INT(RND(1)*13+2)
790 PSET (A,B),C
800 NEXT I:C=0
810 COLOR1:CURSOR10,30:PRINT "HI~SC"
820 CURSOR10,70:PRINT "SCORE"
830 CURSOR10,120:PRINT "SHIPS"
840 CURSOR10,45:PRINT HS
```

```
850 CURSOR10,85:PRINT SC
860 CURSOR10,135:PRINT ME:X=INT(RND(1)*150+75)
870 BLINE (45,180)-(255,191),,BF
880 CIRCLE(151,191),104,4,.1,.5,1,BF
890 SCREEN 1,1:CURSOR7,15:PRINT"PRESS ANY KEY TO BEGIN"
900 IF INKEY$="" THEN 900
910 BEEP
920 SCREEN 2,2
930 O=INT(RND(1)*6)+1
940 ONOGOSUB1150,1160,1170,1180,1190,1200
950 IF C=10 THEN 1210
960 IF C=9 THEN 1330
970 IF C=5 THEN 1450
980 IF C=15 THEN 1570
990 IF C=13 THEN 1580
1000 IF C=8 THEN 1590
1010 GOTO 930
1020 Z$="":Z=0:MAG1:RESTORE1070
1030 FOR Z=0 TO 31
1040 READ Z$
1050 PATTERNS#Z,Z$
1060 NEXT Z:RETURN
1070 DATA 010107272C3C2E7F,0F7D3F7F5B67987F,8080E0E4343C74FE,F0BEF
CFEDAE619FE
1080 DATA 000103674267311B,3F3E3F78FF7F2912,0080C0E642E68CD8,FC7CF
C1EFFFE9448
1090 DATA 0000000000000102,03060E0702040818,000000C0C0C0E0D0,F0D81
CF8D0C80406
1100 DATA 000001070F797F3F,3F77783F53890406,000080E0F09EFEFC,FCEE1
EFCCA912060
1110 DATA 001E5FC7FF33FF7E,3BFC4B2F276E24E0,0078FAE3FFCCFF7E,DC3FD
2F4E4762407
1120 DATA 00000C7FFF7F0D,01,00003C9EC890A0E0,ECFE7C
1130 DATA 00000000A45B7F5B,5B7CC6,00000E1DBF6EE46A,6ADB68
1140 DATA 000AB57B2F7E9B2A,2A1B9F7E2F7BB50A,0050ADDEFA7ED954,E4D8F
97EF4DEAD50
1150 N=4:C=10:P=INT(RND(1)*5+5):P=P*10:RETURN
1160 N=8:C=9:P=50:RETURN
1170 N=12:C=5:P=40:RETURN
1180 N=16:C=15:P=30:RETURN
1190 N=20:C=13:P=75:RETURN
1200 N=24:C=8:P=100:RETURN
1210 A=INT(RND(1)*10+10):Y1=INT(RND(1)*30+40)
1220 FOR X1=60 TO 239 STEP A
1230 SPRITE1,(X1,Y1),N,C
1240 SOUND4,2,10
1250 A$=INKEY$
1260 IF A$=CHR$(32) THEN GOSUB 1600
1270 IF A$=CHR$(29)ANDXX>60 THEN XX=XX-8
1280 IF A$=CHR$(28)ANDXX<220 THEN XX=XX+8
1290 SPRITE0,(XX,YY),0,14
1300 IF RND(1)<.01 THEN GOSUB 1660
```

```
1310 NEXT X1
1320 GOTO 1010
1330 A=INT(RND(1)*10+10):X1=INT(RND(1)*160+75)
1340 FOR Y1=60 TO 178STEP A
1350 SPRITE1,(X1,Y1),N,C
1360 SOUND4,2,10
1370 A$=INKEY$
1380 IF A$=CHR$(32) THEN GOSUB 1600
1390 IF A$=CHR$(29)ANDXX>60 THEN XX=XX-8
1400 IF A$=CHR$(28)ANDXX<220 THEN XX=XX+8
1410 SPRITE0,(XX,YY),0,14
1420 IF RND(1)<.01 THEN GOSUB 1660
1430 NEXT Y1
1440 GOTO 1010
1450 A=INT(RND(1)*10+10):Y1=INT(RND(1)*75)+30
1460 FOR X1=238 TO 60 STEP -A
1470 SPRITE1,(X1,Y1),N,C
1480 SOUND4,2,10
1490 A$=INKEY$
1500 IF A$=CHR$(32) THEN GOSUB 1600
1510 IF A$=CHR$(29)ANDXX>60 THEN XX=XX-8
1520 IF A$=CHR$(28)ANDXX<220 THEN XX=XX+8
1530 SPRITE0,(XX,YY),0,14
1540 IF RND(1)<.01 THEN GOSUB 1660
1550 NEXT X1
1560 GOTO 1010
1570 GOTO 1330
1580 GOTO 1450
1590 GOTO 1210
1600 LINE (XX+8,YY)-(XX+8,10),C
1610 SOUND5,RND(1)*4,10
1620 BLINE (XX+8,YY)-(XX+8,10)
1630 IF XX+8>X1ANDXX+8<X1+16 THEN GOSUB 1790
1640 SOUND0
1650 RETURN
1660 LINE (X1+8,Y1+8)-(XX+8,YY+2),4
1670 SOUND4,2,15
1680 BLINE (X1+8,Y1+8)-(XX+8,YY+2)
1690 FOR I=1 TO 10
1700 FOR CO=2 TO 15
1710 SPRITE0,(XX,YY),0,CO
1720 NEXT CO
1730 NEXT I
1740 ME=ME-1
1750 IF ME<=0 THEN 1950
1760 BLINE (0,132)-(40,145),,BF
1770 COLOR1:CURSOR10,135:PRINT ME
1780 SOUND0:RETURN
1790 OUT127,228:FORF=240 TO 255
1800 SPRITE1,(X1,Y1),28,INT(RND(1)*13)+2
1810 OUT127,F:FOR U=0 TO 5:NEXT U
1820 NEXT F
```

```
1830 SC=SC+P:PRINTCHR$(16)
1840 BLINE (0,80)-(45,94),,BF
1850 IFSC>2000ANDSC<2100THENSOUND2,150,15:ME=ME+2:BLINE(0,132)-(40
,145),,BF:COLOR1:CURSOR10,135:PRINT ME:SOUND0
1860 CURSOR10,85:COLOR1:PRINT SC
1870 IF C=10THENX1=239
1880 IF C=9 THENY1=174
1890 IF C=5 THEN X1=60
1900 IF C=15THENY1=174
1910 IF C=13THEN X1=60
1920 IF C=8 THENX1=239
1930 SOUND0
1940 RETURN
1950 SCREEN 2,2:COLOR10,10,,11:CLS
1960 COLOR,1,(70,70)-(185,88)
1970 CURSOR75,75:PRINT CHR$(17);"GAME OVER":SOUND0
1980 SOUND1,587,15
1990 FOR I=1 TO 100:NEXT I
2000 SOUND0
2010 SOUND1,587,15
2020 FOR I=1 TO 100:NEXT I
2030 SOUND0
2040 SOUND1,587,15
2050 FOR I=1 TO 50:NEXT I
2060 SOUND0
2070 SOUND1,587,15
2080 FOR I=1 TO 100:NEXT I
2090 SOUND1,698,15
2100 FOR I=1 TO 100:NEXT I:SOUND1,659,15
2110 FOR I=1 TO 50:NEXT I
2120 SOUND0
2130 SOUND1,659,15
2140 FOR I=1 TO 100:NEXT I
2150 SOUND1,587,15:FOR I=1 TO 50:NEXT I:SOUND0
2160 SOUND1,587,15
2170 FOR I=1 TO 50:NEXT I
2180 SOUND1,554,15
2190 FOR I=1 TO 50:NEXT I
2200 SOUND1,587,15
2210 FOR I=1 TO 150:NEXT I
2220 SOUND0
2230 FOR I=1 TO 500:NEXT
2240 PRINT CHR$(16)
2250 SCREEN 1,1:CLS
2260 PRINT"PLAY AGAIN {Y/N}?"
2270 IF INKEY$="" THEN 2270
2280 IF INKEY$="N" THEN CLS:PRINT "Bye Bye!!":FOR I=1 TO 10000:COL
OR,INT(RND(1)*15)+1:NEXT
2290 IF INKEY$="Y" THEN ME=5:IFHS<SC THEN HS=SC
2300 SC=0:GOTO 760
2310 SCREEN 1,1:COLOR13,1:CLS:PRINT "INSTRUCTIONS {Y/N}"
2320 IF INKEY$="" THEN 2320
```

```
2330 IF INKEY$="N" THEN RETURN
2340 CLS:PRINTSPC(10);"INSTRUCTIONS"
2350 PRINT:PRINT" You are on a planet firing at the   evil aliens
."
2360 PRINT:PRINT" Use the CURSOR arrows to move <- and -> and the
space-bar to fire. You get an extra 2 men if your score is in   b
etween 2000 and 2100"
2370 PRINT:PRINT" Every so often they will fire at you,and they do
n't miss!!"
2380 PRINT:PRINT"     The score table follows."
2390 PRINT:PRINT"   PRESS ANY KEY..."
2400 IF INKEY$="" THEN 2400
2410 RETURN
```

Δ

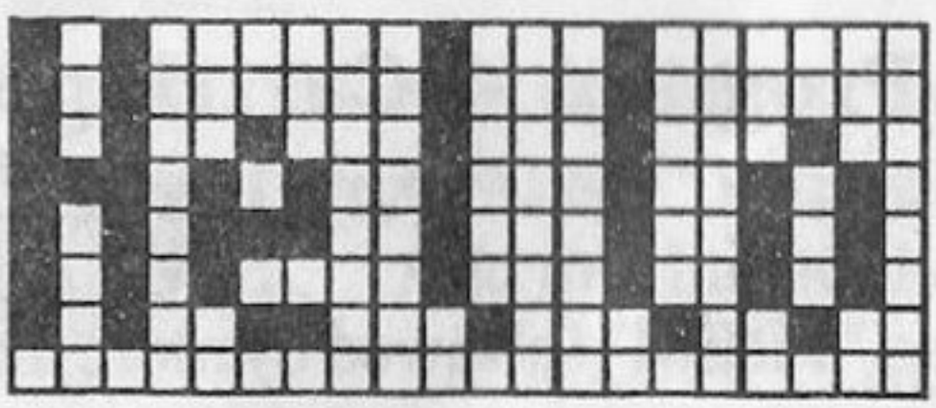
*You can help keep the magazine going by sending in some programs! If you would like to see your name in print (with chance to win some money), then start programming!*

# Printing 64 Columns

*By Michael Hadrup*

• ***For 32K and Disk Basic***

We all know that the SEGA does not provide an 80 column screen. However with this program you can overcome the restriction of 38 columns. By using the normal PRINT command, this program will displays text characters on the graphics screen. The new characters are stored in a 4x8 matrix. This allows a screen of 64x24 text characters, as the graphics screen has 256x192 pixels. See the example below



Remember that you don't have to know the technical side of how this program works, the point is, it does and it is easy to use - see over the page

## Typing in the Program - Disk Basic Users

Type in listing 1 followed by listing 2 and listing 3, carefully. Each line in listing 3 contains 32 bytes of machine code followed by a checksum for that line. The program can be saved at any time by typing

```
SAVE "Print64.Dta"
```

When you have finished type RUN and wait. The machine code is being poked to &HF000. If all goes well then the BASIC program will print the message "No errors!!". If you don't get this message then check the offending data line for typing errors, and type RUN again.

When you get (finally?), the message "No errors!!", insert a disk onto which the code can be saved and press space. The program will also automatically saye itself. (See machine code "Hints and Tips" in the next magazine if you want to know how it does this.)

## Typing in the Program - Cartridge Basic Users Using LSV

If you have typed in LSV from the October '86 issue of SEGA Computer, then the process of typing Print 64 is simplified. (If you are unsure exactly what LSV is, then look at the bottom of page 2). Type in listing 1 followed by listing 3 and carefully make the alterations for Cartridge Basic by replacing those lines as shown in listing 4. The program can be saved at any time by typing `*SAVE "Print 64.Data"`

When you have finished, save it, type RUN and wait. The machine code is being poked to &HF000. If all goes well then the BASIC program will print the message "No errors!!". If you don't get this message then check the offending data line for typing errors, and type RUN again.

When you get (finally?), the message "No errors!!", save the machine code using `*SAVEC "Print 64.Code",&HF000,&HF645`

## Typing in the Program - Cartridge Basic Users without LSV

Start by typing a line "1 REM" followed by more than 256 characters. (The easiest way to do this, is by typing "1 REM" followed by seven lines of zeros). The REM statements are where the machine code will be stored and this will allow us to load and save machine code in the form of a BASIC program.

When you list the program, the line should look like this...

```
1 REM  0000000000000000000000000000000
0000000000000000000000000000000000000
0000000000000000000000000000000000000
0000000000000000000000000000000000000
0000000000000000000000000000000000000
0000000000000000000000000000000000000
0000000000000000000000000000000
```

Type `PRINT PEEK(&H9800)` . If the answer is not 251, then add some more characters to the REM statement and recheck.

Now duplicate line 1 as line 2. Duplicate line 2 as line 3 and so on, until you have 8 lines of the same REM statement. To make the listing easier to read, type as a direct command.

```
FORN=0TO7:POKE&H9806+N*257,13:NEXT
```

Now add to this Listing 3 and carefully make the alterations for Cartridge Basic by replacing those lines as shown in listing 4. The program can be saved at any time by typing

```
SAVE "Print 64.Data"
```

When you have finished, save it, type RUN and wait. The machine code is being poked into the REM statements. If all goes well then the BASIC program will print the message "No errors!!". If you don't get this message then check the offending data line for typing errors, and type RUN again.

When you get (finally?), the message "No errors!!", delete lines 100 onwards using `DELETE 100-`

Save this program using `SAVE "Print 64"`

## Using the Program

Before you can use Print 64, the machine code must be stored at &HF000. There are three ways to do this, depending on whether you are using LSV, Disk Basic or Cartridge Basic.

**Disk Basic Users :** Add this line to the start of your program

```
LOADM "Print 64.Cde",&HF000
```

**Cartridge Basic with LSV :** Add this line to the start of your program

```
*LOADC "Print 64.Code",&HF000
```

**Cartridge Basic without LSV :** Add your program after the REM statements and this line

```
CALL &H9F34
```

The routine at &H9F34 copies the machine code from the REM statements to &HF000 - (Cartridge Basic Users without LSV only).

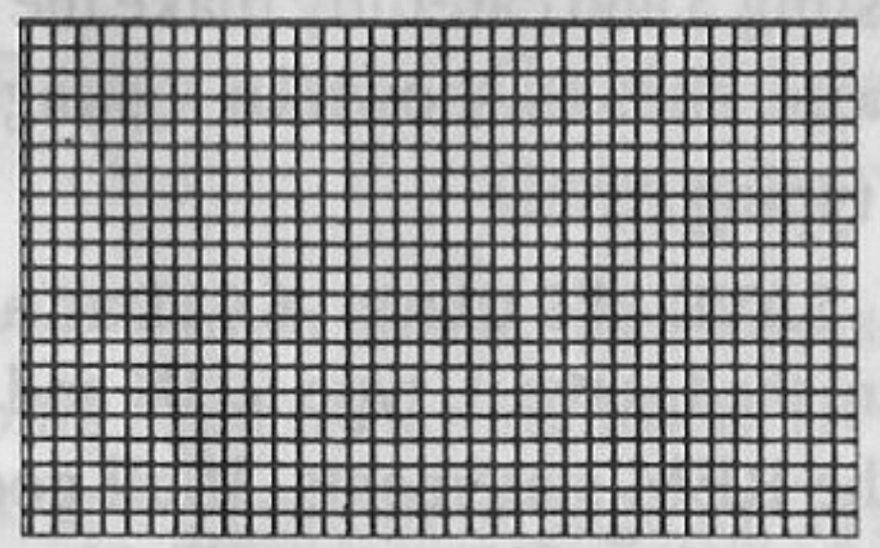
## *Print 64 uses the normal Basic PRINT command*

Once the machine code is loaded into memory you can switch Print 64 on using `CALL &HF000`. From now on anything you print with the PRINT command is diverted to Print 64, and is displayed on the graphics screen in 64 columns. If you want to use the normal PRINT command then you can switch Print 64 off using `CALL &HF007`.

## *Printing in the window*

You can restrict where Print 64 prints on the graphics screen, by specifying a window. The window is determined by the top-left and bottom-right coordinates. The window for the full screen would be

(0,0)



(63,23)

However most televisions lose the left-most 16 pixels or four Print 64 columns. So the more usual window is defined by (4,0),(63,23). There is a restriction on the window horizontally. The left side of the window must lie on an even numbered column and the right side must lie on an odd numbered column. In other words you can't have a window such as (1,0),(63,23). To change the window in which Print 64 will print, poke the coordinates into the following addresses:

| | |
|---|---|
| &HF640 | Top-left x coordinate |
| &HF641 | Top-left y coordinate |
| &HF642 | Bottom-right x coordinate |
| &HF643 | Bottom-right y coordinate |

So if you wanted to set up the following window (8,4),(55,19) then you would add these pokes to your program

```
POKE &HF640,8:POKE &HF641,4
POKE &HF642,55:POKE &HF643,19
```

Another useful point to know, is that if you continue printing text into a window , Print 64 will scroll the text only within the window.

The cursor command has no effect on Print 64, so to change the cursor position you have to use pokes. For example to set the cursor as `CURSOR 8,4` you would add these pokes to your program:

```
POKE &HF644,8:POKE &HF645,4
```

A number of control codes can be used by a `PRINT CHR$(n);` statement to turn on inverse printing and other controls:

| | |
|---|---|
| CHR$(0) | None |
| CHR$(1) | Inverse printing off |
| CHR$(2) | Inverse printing on |
| CHR$(3) | Carriage return |
| CHR$(4) | None |
| CHR$(5) | Same as CONTROL E in Basic |
| CHR$(6) | None |
| CHR$(7) | BEEP |
| CHR$(8) | Backspace |
| CHR$(9) | Tab every 8 columns |
| CHR$(10) | Line feed |
| CHR$(11) | Home |
| CHR$(12) | Cls window with colours set by COLOR command |
| CHR$(13) | Carriage return |
| CHR$(14) | None |
| CHR$(15) | Swith text <> graphics screen |
| CHR$(16) | None |
| CHR$(17) | None |
| CHR$(18) | Insert space at cursor position |
| CHR$(19) | Set uppercase |
| CHR$(20) | Set lowercase |
| CHR$(21) | Clear line, same as CONTROL U in Basic |
| CHR$(22) | Reset screen |
| CHR$(23) | None |
| CHR$(24) | Toggle sound on/off, same as CONTROL X in Basic |
| CHR$(25) | Delete characters toward cursor |
| CHR$(26) | Toggle Centronics/Parallel port (Disk Basic only) |
| CHR$(27) | None |
| CHR$(28) | Cursor right |
| CHR$(29) | Cursor left |
| CHR$(30) | Cursor up |
| CHR$(31) | Cursor down |

Now you can try the demo program in listing 5. Before you run this program, make sure the machine code is stored at #F000 using one of the methods described on page 17. The demo program scrolls random messages in randomly sized windows with random colours!

Hopefully I have now given you most of the information required to use Print 64 in your own programs. Try experimenting! In the next issue there will be a more detailed technical description of Print 64.

# Listing 1 - Disk Basic and LSV Users

```
10 CLS
20 X=&HF000:RESTORE1000
30 FORN=1000TO2000STEP20:C=0
40 CURSOR0,0:PRINTHEX$(X)
50 FORM=0TO31:READA$:POKEX,VAL("&H"+A$):C=C+PEEK(X):X=X+1:NEXTM
60 READA$:IFC<>VAL("&H"+A$)THENBEEP2:PRINT"Error in line ";N:STOP
70 NEXT
80 BEEP:BEEP:PRINT"No errors!!"
```

# Listing 2 - Disk Basic Users only

```
90 PRINT:PRINT"Insert DISK and press space"
100 A$=INKEY$:IFA$<>" "THEN100
110 SAVEM"Print 64.Cde",&HF000,&HF645
120 PRINT:PRINT"Saving """;
130 PRINT"Print 64.Dta""":CALL&H21D4
140 END
```

# Listing 3 - Data Listing

```
1000 DATA 21,E,F0,22,4E,AC,C9,21,DF,4C,22,4E,AC,C9,F5,C5,D5,E5,DD,
E5,DD,21,40,F6,F3,CD,6B,F0,FB,DD,E1,E1,1454
1010 REM
1020 DATA D1,C1,F1,C9,7D,D3,BF,7C,D3,BF,C9,7D,D3,BF,7C,F6,40,D3,BF
,C9,79,CB,3F,7,7,7,6F,60,79,1F,E,F0,124B
1030 REM
1040 DATA D0,E,F,C9,DD,66,1,DD,7E,0,4F,CB,3F,7,7,7,6F,DB,BF,DD,7E,
2,3C,91,4F,DD,7E,3,3C,94,47,C9,D83
1050 REM
1060 DATA 26,F7,79,CB,3F,7,7,CB,27,5F,C9,FE,80,D2,88,F0,FE,20,D2,A
6,F0,21,0,F3,87,5F,16,0,19,5E,23,56,F16
1070 REM
1080 DATA D5,DD,4E,4,DD,46,5,C9,21,40,F3,E6,7F,CA,99,F0,47,CB,7E,2
3,CA,91,F0,10,F8,7E,E6,7F,CD,E,F0,CB,1280
1090 REM
```

```
1100 DATA 7E,23,CA,99,F0,C9,CD,81,F0,C5,F5,DB,BF,CD,34,F0,F1,CD,B8
,F0,C1,C3,91,F1,E5,6F,26,0,29,29,29,11,13B2
1110 REM
1120 DATA 40,F2,19,EB,E1,6,8,C5,CD,24,F0,0,0,DB,BE,47,79,2F,A0,47,
CD,2B,F0,1A,CB,41,CA,E1,F0,7,7,7,EF8
1130 REM
1140 DATA 7,0,B0,D3,BE,2C,13,C1,10,DD,C9,AF,32,E1,F0,C9,3E,A9,32,E
1,F0,C9,AF,DD,77,0,DD,77,1,DD,36,2,1099
1150 REM
1160 DATA 3F,DD,36,3,17,DD,77,4,DD,77,4,D3,BF,3E,78,D3,BF,1,0,0,79
,D3,BE,3,78,FE,3,C2,14,F1,AF,D3,EC6
1170 REM
1180 DATA BF,47,3E,7B,D3,BF,3E,C0,D3,BE,0,0,10,FA,CD,44,F0,CD,62,F
0,C5,CD,E6,F1,C1,24,10,F8,CD,48,F1,DD,1343
1190 REM
1200 DATA 4E,0,DD,46,1,C3,A4,F1,CD,44,F0,CD,62,F0,CB,EC,C5,43,CD,2
B,F0,3A,1A,AB,D3,BE,0,0,10,FA,24,C1,1110
1210 REM
1220 DATA 10,EE,C9,DD,66,5,CD,47,F0,CD,E6,F1,44,DD,4E,0,C3,A4,F1,C
D,6D,F1,0,C3,8A,21,79,D,DD,BE,0,C2,11FA
1230 REM
1240 DATA A4,F1,DD,4E,2,78,5,DD,BE,1,C2,A4,F1,4,C3,A4,F1,79,C,DD,B
E,2,C2,A4,F1,DD,4E,0,78,4,DD,BE,1149
1250 REM
1260 DATA 3,CA,AB,F1,DD,71,4,DD,70,5,C9,5,CD,A4,F1,CD,44,F0,5,CA,E
6,F1,CD,62,F0,E,BE,7B,54,5D,C5,24,11E4
1270 REM
1280 DATA E5,F5,CD,24,F0,F1,47,26,F7,ED,A2,20,FC,E1,EB,E5,F5,CD,2B
,F0,F1,47,26,F7,ED,A3,20,FC,E1,EB,14,C1,15EB
1290 REM
1300 DATA 10,DC,EB,CD,47,F0,CD,2B,F0,CD,62,F0,43,AF,D3,BE,0,0,10,F
A,C9,CD,7A,F1,C5,3E,20,CD,A9,F0,C1,C3,137D
1310 REM
1320 DATA A4,F1,DD,7E,4,E6,F8,C6,8,DD,BE,2,DA,12,F2,DD,7E,0,DD,77,
4,C9,CD,6E,F2,CA,1F,F2,2C,ED,B0,3E,12A6
1330 REM
1340 DATA 20,12,C3,43,F2,CD,6E,F2,CA,32,F2,DD,6E,2,5D,2D,ED,B8,DD,
5E,4,C3,1F,F2,CD,6E,F2,36,20,CA,43,F2,1156
1350 REM
1360 DATA 1C,ED,B0,CD,44,F0,DD,66,5,16,F8,DD,5E,0,41,E,F0,C5,D5,1A
,CD,B8,F0,D1,C1,1C,79,2F,4F,1F,D2,65,10AE
1370 REM
1380 DATA F2,7D,D6,8,6F,10,EA,C9,DD,66,5,CD,47,F0,CD,82,F2,DD,7E,4
,3C,CD,4A,F0,6,0,DD,6E,4,5D,54,79,102D
1390 REM
1400 DATA A7,C9,CD,44,F0,DD,66,5,C5,CD,24,F0,CD,60,F0,55,43,E,BE,E
D,A2,20,FC,C1,6A,41,E,F0,26,F7,D9,DD,12C8
1410 REM
1420 DATA 6E,0,26,F8,E,20,D9,C5,11,41,F8,6,8,7E,A1,CB,41,CA,B8,F2,
7,7,7,7,12,1C,2C,10,F0,E5,21,40,C0B
1430 REM
```

```
1440 DATA F3,11,41,F8,6,8,1A,BE,C2,EB,F2,23,1C,10,F7,D9,71,2C,E,20
,D9,E1,C1,79,2F,4F,1F,D2,E2,F2,7D,D6,1036
1450 REM
1460 DATA 8,6F,10,C3,D9,36,0,DD,6E,4,C9,D9,C,79,FE,80,CA,FB,F2,D9,
16,0,58,19,C3,C1,F2,E,20,C3,D0,F2,108D
1470 REM
1480 DATA EA,F0,EB,F0,F0,F0,6D,F1,EA,F0,38,F2,EA,F0,B,52,F5,F1,2,F
2,9C,F1,3F,F1,2E,F1,73,F1,EA,F0,6C,52,16E0
1490 REM
1500 DATA EA,F0,EA,F0,25,F2,63,5A,6B,5A,63,F1,F6,F0,48,F1,9E,52,16
,F2,A7,52,99,F1,91,F1,7A,F1,85,F1,9C,F1,152B
1510 REM
1520 DATA 0,0,0,0,0,0,0,0,40,40,40,40,40,0,40,0,A0,A0,A0,0,0,0,0,0
,60,60,F0,60,F0,60,60,0,720
1530 REM
1540 DATA 40,E0,C0,60,60,E0,40,0,A0,A0,20,40,80,A0,A0,0,40,A0,40,C
0,A0,90,60,0,40,40,80,0,0,0,0,0,B90
1550 REM
1560 DATA 20,40,80,80,80,40,20,0,80,40,20,20,20,40,80,0,0,0,A0,40,
A0,0,0,0,0,40,40,E0,40,40,0,0,780
1570 REM
1580 DATA 0,0,0,0,60,20,40,0,0,0,0,E0,0,0,0,0,0,0,0,0,0,60,60,0,0,
20,20,40,80,80,0,0,3E0
1590 REM
1600 DATA 40,A0,A0,E0,A0,A0,40,0,40,C0,40,40,40,40,E0,0,40,A0,20,4
0,80,80,E0,0,40,A0,20,40,20,A0,40,0,C20
1610 REM
1620 DATA 20,60,A0,A0,E0,20,20,0,E0,80,80,C0,20,A0,40,0,40,A0,80,C
0,A0,A0,40,0,E0,20,20,40,40,40,40,0,C40
1630 REM
1640 DATA 40,A0,A0,40,A0,A0,40,0,40,A0,A0,60,20,A0,40,0,0,40,40,0,
40,40,0,0,0,40,40,0,40,40,80,0,8A0
1650 REM
1660 DATA 0,20,40,80,40,20,0,0,0,0,E0,0,E0,0,0,0,0,80,40,20,40,80,
0,0,40,A0,20,40,40,0,40,0,660
1670 REM
1680 DATA 60,90,10,50,D0,D0,60,0,40,A0,A0,A0,E0,A0,A0,0,C0,A0,A0,C
0,A0,A0,C0,0,60,80,80,80,80,80,60,0,F90
1690 REM
1700 DATA C0,A0,A0,A0,A0,A0,C0,0,E0,80,80,E0,80,80,E0,0,E0,80,80,E
0,80,80,80,0,40,A0,80,80,A0,A0,60,0,1100
1710 REM
1720 DATA A0,A0,A0,E0,A0,A0,A0,0,E0,40,40,40,40,40,E0,0,20,20,20,2
0,20,A0,40,0,A0,A0,C0,80,C0,A0,A0,0,DA0
1730 REM
1740 DATA 80,80,80,80,80,80,E0,0,A0,E0,E0,E0,A0,A0,A0,0,C0,A0,A0,A
0,A0,A0,A0,0,40,A0,A0,A0,A0,A0,40,0,1120
1750 REM
1760 DATA C0,A0,A0,A0,C0,80,80,0,40,A0,A0,A0,A0,A0,40,20,C0,A0,A0,
A0,C0,A0,A0,0,40,A0,80,40,20,A0,40,0,F60
1770 REM
```

```
1780 DATA E0,40,40,40,40,40,40,0,A0,A0,A0,A0,A0,A0,60,0,A0,A0,A0,A
0,A0,40,40,0,A0,A0,A0,E0,E0,E0,A0,0,F40
1790 REM
1800 DATA A0,A0,A0,40,A0,A0,A0,0,A0,A0,A0,60,20,40,80,0,E0,20,20,4
0,80,80,E0,0,E0,80,80,80,80,80,E0,0,EA0
1810 REM
1820 DATA 0,80,80,40,20,20,0,0,E0,20,20,20,20,20,E0,0,40,A0,0,0,0,
0,0,0,0,0,0,0,0,0,E0,0,5A0
1830 REM
1840 DATA 40,40,20,0,0,0,0,0,0,0,0,40,20,60,A0,60,0,80,80,C0,A0,A0,A
0,C0,0,0,0,60,80,80,80,60,0,900
1850 REM
1860 DATA 20,20,60,A0,A0,A0,60,0,0,0,40,A0,E0,80,60,0,20,40,40,E0,
40,40,40,0,0,0,60,A0,A0,60,20,C0,AA0
1870 REM
1880 DATA 80,80,C0,A0,A0,A0,A0,0,40,0,40,40,40,40,40,0,20,0,20,20,
20,20,20,C0,80,80,A0,A0,C0,A0,A0,0,B80
1890 REM
1900 DATA 40,40,40,40,40,40,20,0,0,0,A0,E0,E0,A0,A0,0,0,0,C0,A0,A0
,A0,A0,0,0,0,40,A0,A0,A0,40,0,AE0
1910 REM
1920 DATA 0,0,C0,A0,A0,A0,C0,80,0,0,60,A0,A0,A0,60,20,0,0,A0,C0,80
,80,80,0,0,0,60,80,40,20,C0,0,B80
1930 REM
1940 DATA 40,40,E0,40,40,40,20,0,0,0,A0,A0,A0,A0,60,0,0,0,A0,A0,A0
,40,40,0,0,0,A0,A0,E0,E0,A0,0,B20
1950 REM
1960 DATA 0,0,A0,A0,40,A0,A0,0,0,0,A0,A0,A0,60,20,C0,0,0,E0,20,40,
80,E0,0,20,40,40,80,40,40,20,0,A40
1970 REM
1980 DATA 40,40,40,0,40,40,40,0,80,40,40,20,40,40,80,0,0,0,C0,40,6
0,0,0,0,F0,F0,F0,F0,F0,F0,F0,F0,C80
1990 REM
2000 DATA 0,0,3F,17,0,0,21,7,98,11,0,F0,6,8,C5,1,E0,0,ED,B0,1,21,0
,9,C1,10,F3,C9,0,0,0,0,820
```

# Listing 4 - Alterations to Listing 3

## Cartridge Basic Users only

```
100 CLS
110 X=&H9807:RESTORE1000
120 FORN=1000TO1970STEP140
130 FORF=0TO6:C=0
140 CURSOR0,0:PRINTHEX$(X)
150 FORM=0TO31:READA$:POKEX,VAL("&H"+A$):C=C+PEEK(X):X=X+1:NEXTM
160 READA$:IFC<>VAL("&H"+A$)THENBEEP2:PRINT"Error in line ";N+F*20
:STOP
170 NEXTF:X=X+33
180 NEXTN
190 FORF=0TO2:C=0
```

```
200 CURSOR0,0:PRINTHEX$(X)
210 FORM=0TO31:READA$:POKEX,VAL("&H"+A$):C=C+PEEK(X):X=X+1:NEXTM
220 READA$:IFC<>VAL("&H"+A$)THENBEEP2:PRINT"Error in line ";1980+F
*20:STOP
230 NEXTF
240 BEEP:BEEP:PRINT"No errors!!"
250 END
1000 DATA 21,E,F0,22,A2,94,C9,21,99,50,22,A2,94,C9,F5,C5,D5,E5,DD,
E5,DD,21,40,F6,F3,CD,6B,F0,FB,DD,E1,E1,148A
1200 DATA 4E,0,DD,46,1,C3,A4,F1,CD,44,F0,CD,62,F0,CB,EC,C5,43,CD,2
B,F0,3A,3A,93,D3,BE,0,0,10,FA,24,C1,1118
1220 DATA 10,EE,C9,DD,66,5,CD,47,F0,CD,E6,F1,44,DD,4E,0,C3,A4,F1,C
D,6D,F1,F5,C3,F5,4B,79,D,DD,BE,0,C2,1384
1480 DATA EA,F0,EB,F0,F0,F0,6D,F1,EA,F0,38,F2,EA,F0,B,51,F5,F1,2,F
2,9C,F1,3F,F1,2E,F1,73,F1,EA,F0,4F,51,16C1
1500 DATA EA,F0,EA,F0,25,F2,66,47,6E,47,63,F1,F6,F0,48,F1,80,51,16
,F2,E8,F0,99,F1,91,F1,7A,F1,85,F1,9C,F1,15CB
```

# Listing 5 - Demo Program

```
100 DIMA$(5)
110 A$(1)="Print 64 was written by Michael Hadrup
          "
120 A$(2)="Fit more information on the screen
          "
130 A$(3)="Use it easily in your own programs
          "
140 A$(4)="The quick brown fox jumped over the lazy dog
          "
150 A$(5)="It can print inverse characters as well
          "
160 POKE&HF016,64
170 POKE&HF017,246
180 SCREEN2,2:COLOR15,4,,4:CLS
190 CALL&HF000
200 POKE&HF640,2*INT(RND(1)*16)
210 POKE&HF641,INT(RND(1)*12)
220 POKE&HF642,2*INT(RND(1)*16)+33
230 POKE&HF643,INT(RND(1)*12)+12
240 COLORRND(1)*14+2,1:PRINTCHR$(12);
250 WIDTH=PEEK(&HF642)-PEEK(&HF640)+1
260 FORN=1TO32
270 PRINTCHR$(1+INT(RND(1)*2));LEFT$(A$(RND(1)*5+1),WIDTH);
280 FORM=1TO32
290 NEXTM,N
300 COLOR15,4:PRINTCHR$(12);
310 GOTO200
```

Remember to add the correct line(s) for you system, which will store the machine code at &HF000

Δ

# BASIC Programming

*By Allan Clarke*

I would like to start this column by wishing Michael success with his new magazine and I hope that all readers support his efforts to keep Sega users informed. Even if you cannot write articles or programs for the magazine, a letter to Michael will let him know that his work is appreciated. There is nothing worse than getting no response from readers at all. I am sure that he would not even mind the odd critical letter (constructive of course!)

Michael asked me to contribute some articles about BASIC programming. After a long discussion, we decided to pitch these at a level a bit above beginners BASIC. There are a number of areas in which people experience difficulties that some of us have more or less forgotten about but caused us some bother at first. You will have to tell us what kind of articles you want to see.

The subject for this issue is arrays. One night I was talking to a club about programming and in passing mentioned the use of arrays for various purposes. I noted a couple of blank looks and later brought this up at supper. I was stunned to find that 80% of those present did not know what an array was and had never used them in their own programs.

## *Arrays*

An array is way of storing data in memory in an ordered and concise fashion. Instead of using a new variable name for every new bit of data, we use a common variable name and refer to a particular bit of data by number reference (or subscript). An array is named in two parts; a variable name followed immediately by a numerical reference in round brackets. We can have number arrays or string arrays.

**String Arrays -** For the first example let us consider a game where you have to input the names of four players. You could use variables NA$, NB$, NC$ and ND$ for the names or we could use array variables N$(1),N$(2),N$(3) and N$(4).

```
eg.  Name      Variable   Array
     Fred      NA$        N$(1)
     Mavis     NB$        N$(2)
     Joe       NC$        N$(3)
     Jill      ND$        N$(4)
```

'So what?' you say. But what if your program requires you to input 100 names? Get the picture now? Let us look at a simple program to store 100 names in memory as a simple data base.

```
100 DIM N$(100)
110 FOR K=1 TO 100
120 PRINT"Name?";
130 INPUT N$(K)
140 NEXT K
150 REM Now print the names
160 FOR K=1 TO 100
170 PRINT K;N$(K)
180 NEXT K
```

You may wish to decrease the number of names from 100 when you type the program into your machine. Just change all the 100's to whatever you want.

There are several points raised in this little program:

- We need to specify a 'dimension' (DIM) for an array. This is to tell the Sega to reserve a space in memory for the array (for up to 100 strings in this case).
- The programming is very compact as we only need to name one variable, N$, for the set of names.
- We can refer to the place number in the array with another variable; K in this program.
- It is easy to number or order the names in the program.
- You can use the same numbering variable to refer to other arrays, eg. addresses could be included in the above program by adding the following lines:

```
100 DIM N$(100),A$(100)   and
135 PRINT"Address?";:INPUT A$(K)
```

**Number Arrays -** These can be used to store numerical data in memory or to carry out a set of calculations. The general layout of a program is the same as before. A short program to draw a diagonal on the text screen is show on the next page:

```
100 DIM X(23),Y(23)
110 FOR N=1 TO 23
120 X(N)=23-N: Y(N)=N
130 NEXT N
140 REM Now draw the line
150 FOR N=1 TO 23
160 CURSOR(X(N),Y(N))
170 PRINT"*"
180 NEXT N
```

This is a very simple example but it shows that the calculation loop only needs to be carried out once. If the diagonal is needed several times in a program, its display is speeded up as time is no longer lost in calculation. You would not notice any difference in this program but if the calculations were long and complex, it would be very noticeable.

**Multi-Dimension Arrays** - The examples before have illustrated the use of an array where each member of the array is referred to by a single number. These are called 'single dimension arrays'. Your Sega is capable of referring to arrays with up to three numbers at a time.

A two-dimensional array is like having a map and each bit of data is found on the grid using row and column reference numbers. Remember graphs at school? We mostly use two dimensional arrays when we are writing game programs because we need to tell the Sega where to put something on the screen. In the next magazine, I will publish my Battleships program which makes extensive use of two dimensional arrays.

You can actually get away without using a DIM statement if the number of items in the array is 10 or less but it is good progamming practice to use the DIM statement at all times.

Your Sega uses arrays in its BASIC and you have probably used them without noticing. The CHR$ function is a single dimension array; CHR$(65) is the letter A. Positions on the screen are located with the CURSOR statement which is a two dimensional array. Thus the last program actually used two arrays within an array to locate the cursor! TAB is another function that uses a single dimension array.

There is a sorting program in the Sega Disk manual (p156). This uses a single dimension array to sort a set of names into alphabetical order. Michael might reprint this for non-disk users.

*(Sorry there wasn't enough space in this issue. In the next issue there will be a program disection of a sorting program. - Editor)*

Δ

# Inverse Characters

By Michael Hadrup

- This program creates an inverse character set using the ENG DIER'S characters which matches the alphabet keys on the keyboard. Type in the listing below for a demonstration and make the necessary alterations for your system. If you wish to add the inverse characters to your own program then delete lines 10-240 and lines 350-390 and add the remaining lines to your program.

```
10 COLOR 15,1
20 CLS
30 PRINT "INVERSE CHARACTER S
ET",,,
40 PRINT "REPLACES ENG DIER'S
CHARACTERS",,,
50 PRINT "BY MICHAEL HADRUP",
,,,,,
60 FORN=160 TO 223:PRINT CHR
$ (N):NEXT
70 FOR N=0TO63
80 A=PEEK (&H5D32+N)
90 IF A=0 THEN 340
100 IF A=32 THEN 340
110 B=&H1D00+(A-160)*8
120 C=&H1800+PEEK (&H5DB2+N)*8
130 FOR M=0 TO 7
140 VPOKE B+M,VPEEK (C+M)XOR
255
150 NEXT M
160 NEXT N
170 PRINT ,,,"**FINISHED **"
180 BEEP
```

## • Alterations for Cartridge Basic

```
80 A=PEEK (&H4020+N)
120 C=&H1800+PEEK (&H3FA0+N)*8
```

*Type italic text with the ENG DIER's key* Δ

# *In the next issue*

**PATTERN PAINT** Program
*for 32K and Disk Basic Users*

Fast Disk Copy and File Copy Routines
*for Disk Basic Users*

More on machine code
and BASIC programming
And maybe 40 pages of useful information,
but that is up to you!

***Due out just before Christmas***

# Poseidon Software

## NZ SEGA DISTRIBUTORS

FREEPOST 243 P.O. BOX 277 TOKOROA NEW ZEALAND

## *Specials*

| | | |
|---|---|---|
| Aerobat | *32K only* | $21.00 |
| Burgular Bill | *16K and 32K* | $19.00 |
| Carverns of Karanor | *16K and 32K* | $21.00 |
| Vortex Blaster | *32K only* | $15.00 |
| Vermin Invaders | *16K and 32K* | $15.00 |
| Castle of Fear | *32K only* | $12.00 |
| Castaway | *32K only* | $12.00 |
| Orb of Fear | *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 |

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

**Name** Glen Mackie **Phone** 8325052

**Address** 23 Metcalf Rd.

| **Item** | **No. of items** | **$ Total** |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |

**Add Postage** **$2.50**

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

Cheque Cash Bankcard Visa

Orders over $20.00 only

Credit Card No. _____

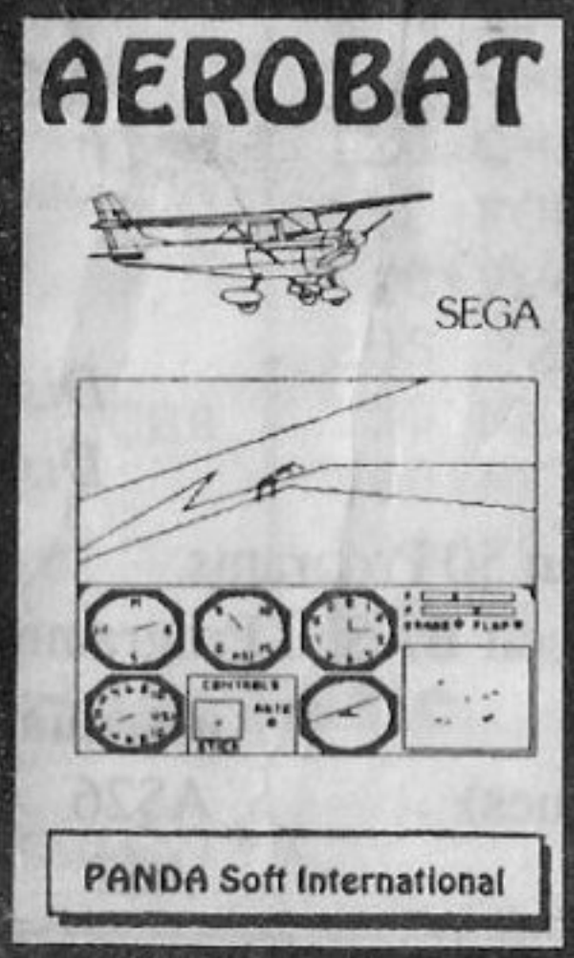
Signed _____ Expires / /

VISA

Welcome here

# GET THE MOST OUT OF YOUR

# SEGA






## Poseidon Software

NZ SEGA DISTRIBUTORS

P.O. BOX 277 TOKOROA NEW ZEALAND