SOUTH AUSTRALIAN

SEGA

MAGAZINE

THE WORLD

Vol. 1 No. 2

WRITTEN AND PUBLISHED BY THE ADELAIDE SEGA USER CLUB

Editorial

Welcome to the second issue of South Australian Sega Magazine (SASM!). We are sorry that this issue is late, but we had an accute attack of Hofstadter's Law!! Hofstadter's Law - Everything takes longer to complete than you first planned, even when you take Hofstadter's Law into account!!. I experienced a problem with my printer, and it is presently being repaired. I would like to thank Andonis (Tony) Symeonidis for the loan of his Brother M-1109. Whilst thanking people, I would like to thank John Maynard for supplying us with the picture on the front cover. If you have a picture that could go on the cover then send it in (on tape or disk if a Sega program)!

This week we have the second parts of all of the columns started last issue, plus the first parts of "Dexter's Workshop" and "Machine Code Tutorial"!! We also have some reviews, and many great programs in the back.

Well, it has been a busy couple of months. As you may or may not know, JSE no longer exists!!! Their position has been taken over by Sega Australia. This new company has a direct relationship with Sega Japan, so we can look forward to a more competative and re-vitalised market (Sega Japan have told Sega Australia to go "all out" in their new advertising campaign, and have given them the money to do it!)

What does this mean for current Sega owners like you and me? Firstly it means more software! All of the games cartridges sold in Japan are going to appear on the Australian market (Pitfall II, Zippy Racer, GP World, Zoom 909, Zaxxon etc). To start off with, SA is going to ship over 30 new titles, and then steadily bring new titles out. Eventually, everything that Sega produces over in Japan will come onto the market. This includes hand-held video games, robots, toys, games machines, and new computers!

Secondly it means a company with a higher profile. You will be able to say proudly "I own a Sega!", and not be met with alot of blank

expressions. A higher profile means that we will get back into the department stores like Myer, John Martins and David Jones.

Thirdly, it means alot of new hardware, like graphics tablets, a new disk drive and "credit card" games (more later). With this new line up, we now have a good chance of taking the top position from Amstrad and Commodore. Not only will more things be out, everything will probably be alot cheaper!

I have heard that there is a new Sega computer in Australia. This computer is only for evaluation only, and may never hit the Australian market. Apparently, it has a built in drive (possibly 3.5"!!!) 128Kb RAM (expandable to 256Kb), an Artificially Intelligent (AI) user interface (possibly with a new programming language!). There is also a built in graphics tablet. Overlays will be supplied with the computer, so that you just press the words on the tablet instead of pressing [FUNC] + [:] say to get PRINT. This is all very exciting, but we must remember that it may never be seen on the market. If it does go onto the market, it will probably sell for under $700.

I hope that they do bring this new computer out, as Sega badly need a new machine out on the market that can compete with the likes of the C128, Amstrad and beyond. I hope that Sega are aiming to produce a "MAC like" machine that can compete with the Amega, Atari 520 ST and the soon to come 1040ST. If they want to do this, then they had better do it fast, otherwise they will loose out.

As I hinted before, I heard a rumour that they are planning to release a "cheapy" disk drive. It would probably come with the bare essentials, this means no RS-232 or Centronics interfaces. Price $300 - $350 maybe.

Games are now going to come on "credit-cards". These are small cards which fit into an adaptor ("Card Catcher", which goes into the cartridge port). This may sound a bit gimmicky, but listen to this, the cards are re-programmable!. This means that you pay about $30 for the card, and when you get sick of the game, you just take it back to the store and have the game changed for a small price of about $10. All the store does it put the card into

a machine, and press a button. The machine erases the card, and burns a new game of your choice in. It is alot better than being stuck with a large library of useless cartridges that you don't play anymore!

Oh well, time to close now, so segalater!

Jamie Anderson

Problems and Letters to the Editor

I would like to thank the many people who took the time to write to us. We had many complimentary letters, and a few verbal compliments and complaints (about the set-up of some programs published). Your response has been phenomenal, and it has been appreciated greatly. Here are a couple of the letters sent in.

Dear Editor,

Congratulations! Your first magazine is a very fine production. It's pleasing to see a group of dedicated and talented people put personal profits to one side to objectively direct their energy to help those less informed and less capable. Quite refreshing.

To have such expertise contained within the one production must benefit everyone Australia wide. With such aims and objectives expressed in your first editorial then reiterated through the pages, I will certainly recommend your magazine to evry Sega user.

Yours faithfully,

Keith D Zuch,
COLLINGWOOD PARK QLD.

Dear Editor,

I know that I speak for most of the Users in the Victorian Group when I commend you and those that have contributed to this excellent magazine.

It is about time that the Users in Australia got more for their money and weren't wasting it on magazines that have too many advertisements and very little information on programming and use of the Sega computer.

Your magazine was very informative with excellent articles and I look forward to your next edition.

Yours Faithfully,

Les Becall,
President Vic User's Group,
CROYDON VIC.

All I can say is thank you to Keith and Les and everyone else who sent letters, or made verbal comments. I hope that we will be able to maintain this standard in the future editions.

I also received a letter from someone in Salisbury, Adelaide who wanted to know how to go over the wall in "Thermonuclear Wargames". I am deeply sorry that I lost the letter, but I will be able to help you. My good friend David tells me that you just have to type "JUMP FENCE" after joining the rods.

If you want to grizzle about or praise SASM, express your views on a subject brought up in the magazine, or even just want to share a thought with us all, then write to

Letters to the Editor,
C/-14 Derribong Road,
Modbury North,
ADELAIDE SA 5092.

Jamie Anderson

Last issue, I talked about the basics of the modem, now I will teach you how to use the software that drives the modem (SegaCOM), and how to get the most out of it.

To check if an error has occured in the transmission process, a "PARITY BIT" is usually sent. This bit is derived from the character code itself. The parity bit helps the computer check for errors, beacuse it describes the character (ie, code can be divided evenly by two etc). The options are odd, even, mark or space depending on the computer you are communicating with. Some do not use a parity bit at all, and therefore have very limited error checking. So that you can understand parity better, let's go through an example. Say that we are "talking" to a database that uses EVEN parity, and we send the letter "B" down the line. The code for "B" is 66 (01000010 in binary), so SegaCOM would send "01000010"+"1" down, the extra one is sent, because 66 can be EVENly divided by 2. If we send the letter "A" (01000001), "01000001"+"0" would be sent, because the character code for "A" (65) is not EVENly divisible by two. If the computer on the other end received "01000001"+"1", or the letter "A" plus an even parity condition, the comuter would know that an error has occured, and it would tell SegaCOM to repeat the letter.

The [CR] key on your Sega performs a carriage return with line feed, ie it brings the cursor to the beginning of the line and moves it down to the next line. Most databases send a carriage return without a line feed, so the software must take care of this. Automatic line feed with carriage return can be turned on or off to cater for the requirements of specific databases.

Information may be sent with the intention for it to be displayed neatly on a 40, 80 or some other column width display. 40 column width is coming into prevalent use, as it can easily be read from a domestic television screen.

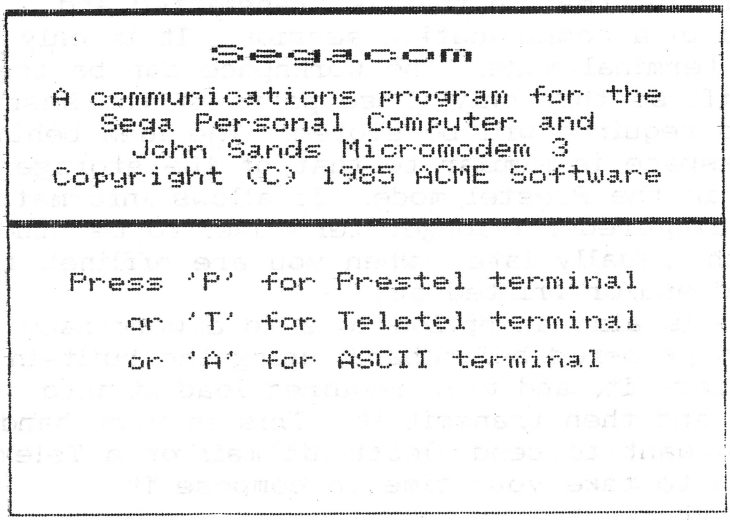
All these selections particularly parity, and

[CR] selection will have to be set up, if you want to communicate with databases other than those using the PRESTEL format.

Now that we know how to set up the computer to access different databases, let's look at what we can do once we are in the database.

The process of acquiring programs from a database or host computer is called "DOWNLOADING". Similarly, the transferring of programs from your system to another via the modem is called "UPLOADING". Prestel mode allows for the downloading of software which may be saved to tape or disk. The programs are stored on Prestel systems as "Screens". When downloading, SEGACOM will tell you how many screens the program occupies, what screen you are up to in the download, how much has been saved in bytes and also a listing of what is actually comming into the computer. In terminal mode programs can be down or uploaded.

MAIN MENU



SEGACOM allows you to store screens away in memory. This will let you keep the screens to study closely after you "log off" (ie told the system that you don't want to communicate with it any longer). The screens can then be scrolled through at your leisure without wasting money on "On-line" charges

that mount up whilst you are in the system (ie as in Viatel which is about 5c a minute). If you want to keep them for longer term reference you may save them to tape or disk or send them to the printer. If you fill the 26 screen storage area, you can save the pages while still "on line" (still communicating with the database or host computer) and then continue storing more screens.

To get a hard copy of the information you have gathered, two print out routines have been incorporated into SEGACOM, ASCII and GRAPHICS. ASCII output takes all the textual part of what you want printed and sends it out the usual Sega serial printer port. This mode allows for a fast printout and works with most printers. All graphics are ignored so printers that do not have graphics capability (eg Daisywheel printers, SP-400) may be used. GRAPHICS output prints out text and graphics. This mode makes the most of printers with bit-image graphics capabilities.

The workspace is an area set aside in the memory to record information that is transmitted and received in a communication session. It is only used in terminal mode. The workspace can be toggled on or off, so that only those parts of the session that are required will be stored. The idea behind the workspace is similar to that of the storage screens in the Prestel mode. It allows information that is required for longer term use, to be studied at length (usually later, when you are offline), to be saved and/or printed out.

There is also an option to send a workspace that has been prepared beforehand, using the built-in editor, save it, and when required load it into SEGACOM and then transmit it. This is very handy when you want to send electronic mail or a Telex (and wish to take your time to compose it beforehand).

Upon selecting Prestel mode from the MAIN MENU, you will obtain a HELP/STATUS screen by pressing the [FUNC] key and holding it down for more than one second. The HELP/STATUS screen displays Help information in the top section of the screen, showing the options which are available. The required option is obtained by continuing to hold

down the [FUNC] key and pressing the key noted next to the option required. The lower section of the screen displays the current printer options, the current page buffer pointer and the page bufers which are currently in use. If a page buffer letter if underlined, then that means that that page has been used. An empty page buffer is signified by an un-underlined letter. There are 26 page buffers to choose from, listed in alphabetical order A-Z.

HELP/STATUS SCREEN

```
↓ Store page                    I Index
↑ Recall page                   R Repeat page
→ Recall next                   U Update page
← Recall previous               V Previous page
                                Q Quit database
S Save page(s)
L Load page(s)                  C Carousel
                                A Reveal
P Print page                    E Erase page
O Printer options               X Cancel entry
                                K Enter editor
                                J Exit editor
                                T Transmit page


Printer          :Graphics,CR/LF
Current page     :A
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

To store a page in memory press [FUNC] + [↓]. This will cause the message "STORE PAGE" to be displayed. Entering the letter "A" will store the page in the memory devoted to the page buffer named "A". By pressing the [BREAK] key you will abort the function.

To recall a specific page from memory press [FUNC] + [^] followed by the letter corresponding to the page required. The page can then be read, printed, edited or transmitted. The pages that have been stored in memory can be stepped through by pressing the [FUNC] key and using the left and right cursor keys to go backwards and forwards (respectively) through the stored pages.

To save pages which are stored in memory to tape or disk press [FUNC] + [S]. One page or a range of

pages may be saved. Once this option is selected, the letter corresponding to the start and end page required, are entered to save the page or pages.

Pages which have been saved to either cassette or disk can be reloaded for display, printing or transmission by pressing [FUNC] + [L]. The pages stored on cassette or disk will be loaded back into the same buffer it was saved from. This means that if you store page number "E" on tape or disk, when you load it up again it will be stored in page "E" in memory, replacing anything that was in "E" before, so beware!!

Pressing [FUNC] + [P] will cause the printer to print the displayed page with reference to the output mode, selected by pressing [FUNC] + [O]. As mentioned a while back, [FUNC] + [O] allows you to select between graphics or ASCII output or line feed mode.

By pressing [FUNC] + [K] you will enter the edit mode and can create your own pages or fill out forms (off line) which can be stored in memory or tape/disk for later transmission to the database.

I will explain more about the edit mode (EDITOR) a little later on. Until next issue, happy communicating!

John Maynard

In this issue, we continue examining the parts of CP/M, and start to look at one of the important programs that is supplied with CP/M, or in Sega's case SegaDOS 2. For simplicity, we will collectively refer to both CP/M and SegaDOS 2 as CP/M.

Although the CCP must listen to your commands that you type on the keyboard, read and write files to the disk and control the screen, it doesn't carry out these actual input/output operations itself, it passes all the work onto FDOS

FDOS consists of two programs, BDOS, and BIOS. Firstly we will look at BDOS. BDOS is short for "Basic Disk Operating System". BDOS sits directly between the CCP and BIOS. BDOS dosen't actually preform the I/O functions either, it formatts the informaton that it is given into something that BIOS understands. It then instructs BIOS to complete the command.

Finally there is BIOS, short for "Basic Input/Output System". BIOS, knows your computer imtimately, and is capable of talking it into doing anything you want. BIOS is important, as it contains all the routines that actually "talk" to the disk drive, printer, screen etc. As BIOS knows how to talk to the system, it is the slave of BDOS. This is why CP/M is found on so many computers, because all you have to do is change the BIOS to get it to work with your particular system. BIOS sits right at the top of the memory (&HFFFF [65535 decimal] down).

Now that we know about CP/M itself,let's take a look at the software that is supplied on the system disk.

The TEXT EDITOR - "ED.COM"

A text editor program allows you to create and edit text files like letters, novels, poems, business forms or anything comprised of characters.

"ED", the text editor provided with CP/M, is only a minimal editor and is not as easy to use as most

other text editors. If you plan to do a significant amount of word processing, you should obtain a more powerful editor. There are many text editors and word processing packages available for CP/M, the most popular being WORDSTAR. When buying CP/M software, you should first ask the computer salesman if his shop could transfer the program to 3" disk for you, or at least help you do it. Most CP/M programs come in 5 1/4" format, which won't despite my best efforts (short of bending the disk in half!!) fit into my 3" drive!!!!

The editor program "ED.COM" is usually found on the System Disk. You run it by typing "ED ", followed by the name of the file that you want to work on. If the file that you want to work with isn't on the disk, then a "NEW FILE" message will be displayed. For example, if you wanted to edit SAMPLE.TXT, your screen might look like this -

```
A>ED SAMPLE.TXT

NEW FILE

*_
```

In this case, the file SAMPLE.TXT wasn't on the disk. Do not type just "ED". This is a common mistake, the program will just throw you back into the CCP, after displaying some sort of error message. You must supply a file name!

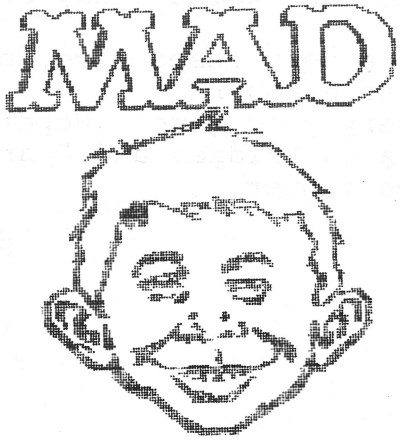
Below is a table of the ED control codes. I will continue next issue.

ED control characters

Keys		Meaning
[CTR]L+[C]	(^C)	System restart (warm boot)
[CTR]L+[E]	(^E)	Moves cursor to next line
[CTRL]+[H]	(^H)	Erase last character type
[CTRL]+[I]	(^I)	Moves cursor to the next "TAB stop"
[CTRL]+[J]	(^J)	Same as CR
[CTRL]+[M]	(^M)	Same as CR
[CTRL]+[L]	(^L)	Replacement for CR in strings
[CTRL]+[R]	(^R)	Retype current line

[CTRL]+[U]	(^U)	Delete current line
[CTRL]+[X]	(^X)	Backspace to beginning of current line and erase line
[CTRL]+[Z]	(^Z)	Terminate the (I) command
[CTRL]+[P]	(^P)	Toggle printer on and off (ie send all screen output to printer as well as screen)
[CTRL]+[S]	(^S)	Temporarily halt display
BREAK		Discontinue execution of currently-executing ED command

Jan Jacobsen



CHOPLIFTER ~ reviewed

Choplifter is the first of the "credit-card" games to make it to Australia. In fact the card that I had for review was the only one of its kind in South Australia.

SEGA's version of choplifter is one of the best versions I have seen, and I have seen both of the Atari and Commodore versions. It is a game where you have a chance to become a total hero "RAMBO" style.

You are one of the elite band of chopper pilots who are given the following mission: To fly into hostile territory and pick up 64 hostages.

"That's not to hard!" I hear you say, well if it sounds all too easy your chopper can only carry 16 passengers at one time. Once in enemy territory you must rescue the hostages, trapped in buildings that must have a hole blown in them by you. Also you have to avoid being shot down by attacking aircraft. When on the ground picking up men, you watch out for the deadly and accurate tanks.

The game starts of at a fairly easy pace, but is by no means a push over. Once you have rescued the first 64 men the game speed quickens at a frightening rate. There are two different screens which alternate after every lot of 64 hostages have been rescued. The first is on the land while the second is over the sea, where enemy boats take the place of tanks.

The controls are very responsive, so the chopper responds is easy to manouvre

All in all a very good game! Hopefully all of SEGA's forthcoming cartridges and "card" games will follow in the same manner.

Paul Schwarz

Welcome back. In this second part of this series, the use of machine code routines to achieve rapid and smooth movement of sprites will be demonstrated.

The Sega computer uses a Z80A microprocessor (CPU for short). Therefore we must use Z80 machine code instructions for the Z80 to understand us. Other computers such as the Macintosh use a different CPU (68000 in the Macintosh's case), and therefore require a different set of machine code instructions. This is like in BASIC. A BASIC program on the Commodore Amiga for example, will not RUN on the Sega, because their BASICs while still the same is essence, are different in the commands that they use.

Machine code is the code in which the Sega does all its work. BASIC programs are translated bit by bit into machine code as they run, this is why they work slowly. Machine code on the other hand needs no translating, so machine code programs run very fast. Whats more, machine code programs take up less memory as well. When you first start to program in machine code, it may seem hard, as all the instructions are just groups of two numbers and/or letters (eg CD stands for CALL, and 3C means add one to the A register etc), but if you keep at it, you will win eventually. Do you remember learning BASIC? It was hard at first, but with a little practice it became very easy to write BASIC programs.

As the purpose of this series of articles is to help you perfect your machine code skills, not teach it to you, I suggest that you read the machine code tutorials written by Paul Schwartz if you know nothing about machine code.

The key to understanding how to move sprites is to remember that the sprite co-ordinates are stored in VRAM from &H3B00 onwards. This means that for sprite number 0, the Y co-ordinate is stored in &H3B00 in VRAM, and the X co-ordinate is in &H3B01 in the VRAM. By changing the values kept in these memory locations, we can move the sprites. If we do

it in machine code, the sprite will move quickly and smoothly.

There are several different ways of writing machine code programs for the Sega. I find the "HEX loader" the most convenient for small routines. The Hexadecimal (from now on abbreviated to HEX!) numbering system uses sixteen digits (HEX - 6 DEC - 10 HEXaDECimal 6+10 = 16!!!!), the numbers 0-9, and the letters A-F to represent numbers. Sometimes you will need to convert from decimal to HEX or vica-versa. To convert from decimal to HEX, type

```
PRINT HEX$(100) [CR]
```

In this case, we converted the number 100 to HEX (100d = &H64). To convert HEX to decimal type

```
PRINT &H64 [CR]
```

You can see that we converted &H64 back into 100 in decimal.

Earlier, I mentioned that the Sega stored the value of the X coodrinate of sprite number 0 at &H3B01. One way of understanding how a computer memory works is to think of it as lots of little boxes. Each box can hold a byte of information. Each box has a number on it which is called its address. If the computer wants some information, it will go to the box in question, and read or change the information it contains. Now let's see what this means in practice.

The following BASIC program puts a sprite on SCREEN 2.

```
10 SCREEN 2,2:CLS:PATTERN S#0, "183C7EFFC3C36624"
20 SPRITE 0,(120,60),0,1
30 GOTO 30
```

After RUNning it, type in this

```
PRINT VPEEK(&H3B01),VPEEK(&H3B00)
```

The numbers 120 and 60 should come up onto the screen if you have typed it correctly. Now we know for sure where the co-odinates for sprite number 0

are stored, we can now start on our machine code program. If you still need convincing, try changing the co-ordinates in line 20, then RUN it again, and re-type the VPEEK line. The big question now is how do we move sprites in machine code??

Well, we know that we put machine code programs into memory via a HEX loader, but what is it?? Essentially it is two or more lines of code. You usually have the part which actually POKEs the machine code into memory, and the DATA statements containing the machine code program. Here is an example.

```
10 DATA 21,01,3B
20 DATA FOR A = &HA000 TO &HA002:READ A$:POKE
A,VAL("&H"+A$):NEXT A
```

In this case, I have decided to start storing my machine code in memory starting from &HA000.

Now we need to work out the machine code instructions, so that we can create the data for the second part of the HEX loader. The program will have to

- Get the X position from &H3B01
- Add one to that value
- Put the next value back into &H3B01
- Wait (delay loop)
- Go back and do it again

I have already written the machine code program, and it is listed at the end of this article. In the program I have included some REMarks. You do not have to type them in. To save space, you could also type all the data statements onto one line eg,

```
10 DATA 3E,00
20 DATA DE,01
becomes
10 DATA 3E,00,DE,01
```

As the disk drive is different internally than the cartridge system, you will have to make some modifications for the program to work on disk drive systems.

Line 120 should read DATA CD,6F,00
Line 230 should read DATA CD,69,00
Line 390 should read AD=&HF000

After you type the program in, I suggest that you SAVE it before RUNning, as machine code is very unforgiving. If you have typed something wrong, the computer will "crash", which means that it will go beserk and probably damage or wipe your program. Disk drivers should be especially careful, and take the disk out after SAVEing. If the program crashes, there is a high probability that the drive will be activated, and precious programs and or data lost, SO BEWARE!!!!!!

You might like to try some experiments, like changing the value in line 100 from 01 to 00. You could try changing the number 3C in line 170 to 3D, or you could play with the values 00 and 03 in line 290. The later is particularly interesting, as it controls the speed at which the sprite moves. If the number is at 100 (00,01) then the sprite moves so fast that it catches up with itself before your TV set can finish printing the sprite, and so you get a very funny effect. If you try the value 00,00 the sprite will move very s-l-o-w-l-y. Anyway, I'm running out of space, so I will continue next issue. Have fun!

Ian Dunn

```
10 REM **********************
20 REM **   ARCADE GAME    **
30 REM **     PART II      **
40 REM **        BY        **
50 REM **    IAN DUNN      **
60 REM **********************
70 REM
80 REM
90 REM
100 DATA 21,01,3B:REM Loads HL register
110 REM                with VRAM address &H3
B01
120 DATA CD,CB,2B:REM CALLs routine in memo
```

```
ry
130 REM                  which gets the displa
y
140 REM                  chip ready to VPEEK
150 DATA DB,BE    :REM Places the X value of
160 REM                  sprite in A register
170 DATA 3C       :REM Increases the value i
n
180 REM                  the A register, which
190 REM                  moves the sprite to t
he
200 REM                  right by one when the
210 REM                  next couple of lines
220 REM                  are executed
230 DATA CD,C5,2B:REM CALLs routine in memo
ry
240 REM                  which gets the displa
y
250 REM                  chip ready to VPOKE
260 DATA D3,BE    :REM VPOKE's  &H3B01 with
270 REM                  with the value of the
280 REM                  A register
290 DATA01,00,03 :REM Start of delay loop
300 REM                  (puts &h300 in BC reg
310 DATA 0B       :REM Takes 1 from the BC re
g
320 DATA 3E,00    :REM Puts 00 in the A reg
              "A" ZERO
330 DATA B8       :REM Compares A with B
340 DATA 20,FA    :REM If A<>B then go back
350 REM                  6 (FA) bytes
360 DATA 18,E7    :REM If A=B then jump back
370 REM                  25 (E7) bytes to the
380 REM                  start
390 AD=&HA000
400 FOR A=AD TO AD+24:READ A$:POKEA,VAL("&H
"+A$):NEXT
410 SCREEN 2,2:CLS:PATTERNS#0,"183C7EFFC3C3
6624"
420 SPRITE0,(120,60),0,1
430 CALLAD
```

Machine code is not as difficult as a lot of BASIC programmers think. The best way to learn and understand machine code programming is to actually sit down and do it. Before we get stuck into programming, there is a need to know what is inside your computer.

The Sega uses the Z80A microprocessor. The Z80 is the brains (or more technically the CPU- Central Processing Unit) of the computer. We will first take a brief look at what is inside the Z80.

Inside the CPU are 21 user registers, a register is simply an 8 bit memory location, that the Z80 uses within a program. This is where the art of machine code programming comes in - we must tell the Z80 how to use them. Figure one shows the registers available.

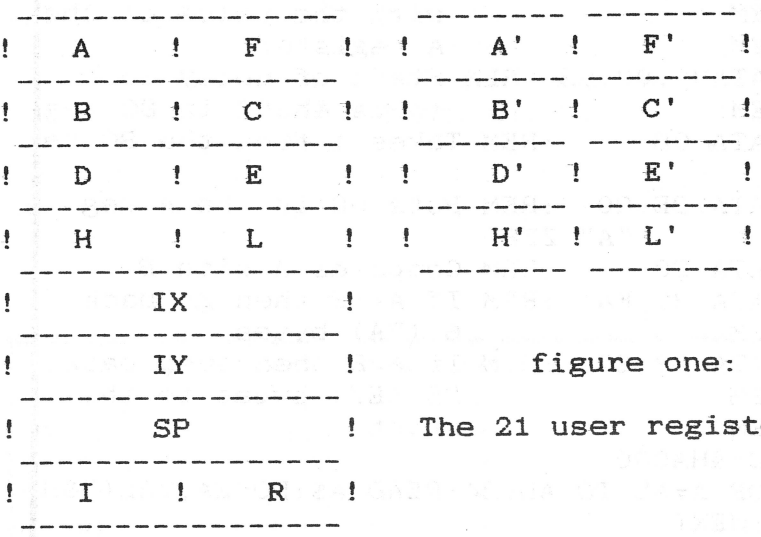


figure one:

The 21 user registers

As you can see some of the registers are twice the size of others, this is because some of the registers are 16 bit and others are 8 bit. If you are unsure of binary maths and the terms "8 bit" and "16 bit" then I suggest you find a book on the subject, "Programming the Z80" by Rodey Zaks, is a very good book and retails for about $20.

We will take a more detailed look at some the Z80's instruction set a little later. Also inside the Sega are a few specialised chips. One of these is the Texas Instruments TMS9929A Video Display Processor (VDP). The VDP is the chip that allows you to use all of the fancy graphics capabilities of the Sega. When you want to print something on the screen (in machine code) you must do it directly through the video chip. All you will need to know at the moment, is that the video chip is "port mapped" at &HBE and &HBF, all this means is the these are the two ports that you use to give the VDP data and instructions.

There is also, as you are all aware, a Programmable Sound Generator (PSG). This is the SN76487AN from TI. Like the VDP the sound generator is also port mapped but this time the port &H7F is used.

The last chip that you will need to know about at this stage is the Programmable Peripheral Interface chip (PPI). This is the chip that controls the input and output of data to and from the computer. This includes the keyboard, printer, joystick and cassette. I will take a more detailed look at the PPI later.

Now that I have very briefly explained what is inside the computer, I hope I haven't scared anyone with all the jargon about the chips. There is no need to worry, as you will soon learn what it all means and how simple it is. The next thing to do is to take a closer look at the Z80 and its instruction set.

Most of the instructions are self explanatory so I won't go into the finer details other than explain what they do.

The first and most frequently used intruction is the "LD" instruction. "LD" simply means LoaD. These instructions are used to load a register with data or a memory location with data. An example is "LD A,B", all this does is simply load register A with the value that is in register B - a basic equivalent is "LET A=B", simple isn't it ?

There are many different forms of this instruction, and I have divided them into different groups:

1 - LOADING REGISTERS WITH CONSTANTS

The length of all these instructions is 2 bytes, one byte for the instruction proper and one byte for the constant.

eg	MNEMONIC	HEX CODE
	LD A,dd	3E dd

2 - LOADING REGISTER PAIRS WITH 2-BYTE CONSTANTS

The instruction length of these will be either 3 or 4 bytes.

eg	MNEMONIC	HEX CODE
	LD HL,aabb	21 bb aa
	LD IX,+aabb	DD 21 bb aa

The first byte of the 2-byte constant goes to the low register (ie L,C,E,X,Y or P) and the second byte to the high register (ie H,B,D,I or S).

3 - REGISTER TO REGISTER COPYING

There too many of these instructions to list here, but they are all one byte long.

eg	MNEMONIC	HEX CODE
	LD A,B	78

4 - LOADING REGISTERS WITH THE CONTENTS OF MEMORY LOCATIONS

This set of instructions can be divided into 3 more sub-groups:

(a) "ABSOLUTE ADDRESSING" - when the actual address is held as data after the instruction proper.

(b) "INDIRECT ADDRESSING" - when the required address is already held in the HL,BC or DE register pairs.

(c) "INDEXED ADDRESSING" - when the address is formed by the addition of a displacement value to a "BASE" address already held in an index register

pair (IX or IY).

(A) ABSOLUTE ADDRESSING

```
eg      MNEMONIC            HEX CODE
        LD A,(aabb)         3A bb aa
        LD HL,(aabb)        2A bb aa
```

Note the brackets around the address, this means that we are loading the register with the CONTENTS of that memory location. Note also, how I have represented the address in HEX code. I have split the number in two and reversed the digits (ie LD A,(FDEC) becomes 3A EC FD)

(B) INDIRECT ADDRESSING

The instructions in this group use the HL,BC or DE register pairs to "point" to the required memory location.

```
eg      MNEMONIC            HEX CODE
        LD A,(HL)           7E
        LD A,(DE)           1A
```

Note the length of these instructions is one byte, and they all result in the loading of just one byte into a single register.

(C) INDEXED ADDRESSING

The instructions in this group allow the programmer to load the main single register with a copy of a byte of data specified as being held in a table or block of code. The base address is held in the IX or IY register pairs.

```
eg      MNEMONIC            HEX CODE
        LD A,(IX+dd)        DD 73 dd
```

where "dd" is the displacement from the base address held in the IX or Iy register pairs. Oh well, I'm running out of space, I will continue next issue.

Hi! In this column, I will show you all sorts of hints, tips, software and hardware modifications. If you have anything to send in, then send it to

Dexter's Workshop,
C/- 14 Derribong Road,
Modbury North,
Adelaide 5092.

"SASM can accept no responsibility for any dammage caused by using these tips, and readers should be advised that hardware modifications will render your warranty incalid!" - Ed.

This week I have a utility program for you. This program allows you to access any of the screens that the VDP is capable of displaying.

After you RUN this program, you will be required to supply some information. Firstly you will have to tell it which screen you will be using (0-3), then which screen you will CALL it from (Text - 4, Graphics - 5). You type this in to the first prompt (eg 24). Then you will have to tell it if you want the "X bit" set (Y/N). The X bit is to ... err ... umm ... err ... Well I forget, but try it anyway. After answering all the prompts, you will see your screen. Pressing any key takes you back to the text screen. Once back, the program asks you if you want the program bytes printed out. If you say yes (Y), then a list of the machine code bytes is printed. To use them, you just POKE then into memory, and CALL it

ie

```
List program bytes?Y

10 20 30 40 50 60 70 70 80 90 C9
Break in 100
Ready
NEW
10 DATA 10,20,30,40,50,60,70
```

```
20 DATA 70,80,90,C9,END
30 AD=&HFF00
40 READ A$:IF A$="END" THEN END
50 POKE AD,VAL("&H"+A$):AD=AD+1:GOTO 40
RUN
Ready
CALL &HFF00:FOR X=1 TO 10000:NEXT
Ready
```

The FOR/NEXT loop is to keep the computer busy enough for a while, so that you can see your screen (when the computer comes back from a program or prints the cursor on the screen, you always come back to the text screen!) Untill next issue, happy computing!!

1	2	3	Screen Type	Resolution
0	0	0	GRAPH mode 1	(32x24)
0	0	1	GRAPH mode 2	(256x192)
0	1	0	MULTICOLOR	(64x48)
1	0	0	TEXT	(40x24)

Mark Fisher (Dexter)

```
100 REM         Screengen
110 REM    By Mark Fisher 1985
120 REM
130 SCREEN 1,1:CLS:RESTORE
140 REM
150 REM    Start address
160 REM
170 S=&HF000
180 REM
190 REM    Change S to &HA000 for
200 REM       non disk systems
210 REM
220 PRINT "0. SCREEN 000"
230 PRINT "1. SCREEN 001"
240 PRINT "2. SCREEN 010"
250 PRINT "3. SCREEN 100":PRINT
260 PRINT "4. CALL FROM TEXT SCREEN"
270 PRINT "5. CALL FROM GRPH SCREEN"
280 PRINT:PRINT
290 PRINT "INPUT CHOICE eg 14 or 35"
```

```
300 INPUT "";C$
310 IF LEN(C$)<>2 THEN BEEP2:GOTO 130
320 A1=VAL(LEFT$(C$,1))
330 IF A1<0ORA1>3 THEN BEEP2:GOTO 130
340 A2=VAL(RIGHT$(C$,1))
350 IF A2<4ORA2>5 THEN BEEP2:GOTO 130
360 INPUT "SET X";Y$
370 R=S
380 DATA F3,DB,BF
390 DATA 3E,00,D3,BF
400 DATA 3E,80,D3,BF
410 DATA 3E,E0,D3,BF
420 DATA 3E,81,D3,BF
430 DATA FB,C9,*
440 READ A$:IF A$="*" THEN GOTO 460
450 A=VAL("&H"+A$):POKE R,A:R=R+1:GOTO 440
460 IF A1=0 THEN POKE&HF004,0:POKE &HF00C,&
HE0
470 IF A1=1 THEN POKE&HF004,0:POKE &HF00C,&
HE4
480 IF A1=2 THEN POKE&HF004,0:POKE &HF00C,&
HE8
490 IF A1=3 THEN POKE&HF004,0:POKE &HF00C,&
HEC
500 IF A2=4 THEN SCREEN 1,1
510 IF A2=5 THEN SCREEN 2,2
520 IF Y$="Y" THEN POKE &HF004,2
530 CALL &HF000
540 IF INKEY$="" THEN 540
550 SCREEN 1,1:CLS:INPUT "DISPLAY MEM (Y/N)
";A$:IF A$<>"Y" THEN 130
560 FOR R=S TO S+&H14:PRINT HEX$(PEEK(R));"
,";:NEXT
570 IF INKEY$=""THEN 570
580 GOTO 130
590 REM
600 REM   Put scribble on graphics
610 REM      screen for tests
620 REM
630 FOR R=1 TO 50:LINE-(RND(1)*255,RND(1)*1
91):NEXT
```

```
10 REM This program allows you to
20 REM print the graphics screen, in 4
30 REM colours on the SP-400.
40 REM It is for SF-7000 owners only.
50 REM a version for tape systems will
60 REM be published next issue
70 REM                          Alex Farkas
80 REM
90 DATA 0,0,0,ED,5B,00,F0,CD,00,FF,FE,0,C0,
ED,5B,0,F0,1C,AF,BB,20,A,14,3E,C0,BA,20,4,3
2,2,F0,C9,ED,53,0,F0,C3,3,F0
100 FOR A=&HF000TO&HF026:READA$:POKEA,VAL("
&H"+A$):NEXT
110 DATA 3A,00,F0,5F,3A,01,F0,00,00,00,00,5
7,AF,CD,14,01,32,F2,FF,CD,81,00,11,00,20,ED
,5A,CD,6F,00,DB,BE,CB,3F,CB,3F,CB,3F,CB,3F,
32,F3,FF,3A,F2,FF,C9
120 FOR A=&HFF00 TO &HFF2E:READ A$:POKEA,VA
L("&H"+A$):NEXT
130 SCREEN 2,2:LPRINT CHR$(18):POKE&HF000,0
:POKE&HF001,0:POKE&HF002,0:CALL&HF003
140 CALL&HF00D
150 LPRINT "M";PEEK(&HF000);",";191-PEEK(&H
F001)
160 GOSUB 190:P=A
170 LPRINT "J1,0"
180 GOTO 140
190 B=PEEK(&HFFF3)
200 A=INT(B/4)+1
210 IFP=ATHENRETURN
220 ONAGOTO230,240,250,260
230 LPRINT"C0":FORH=1TO400:NEXTH:RETURN
240 LPRINT"C1":FORH=1TO400:NEXTH:RETURN
250 LPRINT"C2":FORH=1TO400:NEXTH:RETURN
260 LPRINT"C3":FORH=1TO400:NEXTH:RETURN
270 RETURN
```

```
20 REM This program rotates and
30 REM shrinks a given polygon
40 REM after drawing, you will hear
41 REM a beep, press SPACE when you
42 REM want another picture.
43 REM
50 BEEP:INPUT "SIZE
          >";MP:MY=MP+20
```

```
60 IF MP>70 THEN BEEP2:GOTO 50
70 IF MP<2 THEN BEEP2:GOTO 50
80 BEEP:INPUT "NUMBER OF SIDES
          >";N
90 IF N<0 THEN BEEP2:GOTO 80
110 BEEP:SCREEN 2,2:COLOR1,1,,1:CLS
120 FOR J=0 TO 2.5*PI STEP PI/20
130 FOR K=0+J TO 2*PI+J+.1 STEP 2*PI/N
140 X=INT(MP*1.3*SIN(K)+128)
150 Y=INT(MY*COS(K)+96)
160 IF MY<1 THEN 210
170 IF K>0+J THEN LINE (X1,Y1)-(X,Y),15
180 X1=X:Y1=Y:NEXT
190 MP=MP-2
200 MY=MY-2.5:NEXT
210 BEEP
220 IF INKEY$="" THEN 220
230 BEEP:GOTO 50
```

```
10 REM In The Mood
20 REM 3 part harmony
30 REM By Jamie Anderson
40 REM
50 READ L:IF L=0 THEN SOUND0:END
60 L=L/2:READ A,B
70 IF A THEN SOUND 1,A,15:VA=15
80 IF B THEN SOUND 2,B,13:VB=7
90 FOR X=1 TO L:SOUND 1,,VA:VA=VA+1*(VA>1):
NEXT:GOTO 50
100 DATA 2,330,196,2,392,196
110 DATA 2,523,220,2,330,220
120 DATA 2,392,233,2,523,233
130 DATA 2,330,220,2,392,220
140 DATA 2,523,196,2,330,196
150 DATA 2,392,220,2,523,220
160 DATA 8,0,233
170 DATA 2,330,196,2,392,196
180 DATA 2,523,220,2,330,220
190 DATA 2,392,233,2,523,233
200 DATA 2,330,220,2,392,220
210 DATA 2,523,196,2,330,196
220 DATA 2,392,220,2,523,220
230 DATA 8,0,233
240 DATA 2,349,262,2,440,262
```

```
250 DATA 2,523,294,2,349,294
260 DATA 2,440,311,2,523,311
270 DATA 2,349,294,2,440,294
280 DATA 2,523,262,2,349,262
290 DATA 2,440,294,2,523,294
300 DATA 8,0,311
310 DATA 2,330,196,2,392,196
320 DATA 2,523,220,2,330,220
330 DATA 2,392,233,2,523,233
340 DATA 2,330,220,2,392,220
350 DATA 2,523,196,2,330,196
360 DATA 2,392,220,2,523,220
370 DATA 8,0,233
380 DATA 2,392,294,2,494,294
390 DATA 2,587,330,2,392,330
400 DATA 2,494,349,2,587,349
410 DATA 2,0,330
420 DATA 2,349,262,2,440,262
430 DATA 2,523,294,2,349,294
440 DATA 2,440,311,2,523,311
450 DATA 2,0,294
460 DATA 2,330,196,2,392,196
470 DATA 2,523,220,2,330,220
480 DATA 2,392,233,2,523,233
490 DATA 2,330,220,2,392,220
500 DATA 2,523,196,2,330,196
510 DATA 2,392,220,2,523,220
520 DATA 16,0,233
530 DATA 0
```

```
1 REM COLDITZ
2 REM
10 MAG3:PATTERNC#160,"0C3C3C7C7CFCFCFC"
20 PATTERNC#161,"C0F0F0F8F8FCFCFC"
30 PATTERNC#162,"FCFCFC7C7C3C3C0C"
40 PATTERNC#163,"FCFCFCF8F8F0F0C0"
50 PATTERNS#0,"0000000000050504"
60 PATTERNS#1,"0701010102040810"
70 PATTERNS#2,"0000000000808000"
80 PATTERNS#3,"E0202020807010O0"
90 PATTERNS#4,"0000000000010100"
100 PATTERNS#5,"0101070101010103"
110 PATTERNS#6,"0000000000808000"
120 PATTERNS#7,"8080808080000040"
```

```
130 PATTERNS#8,"070F3F7F7FFFFFFF"
140 PATTERNS#9,"FFFFFF7E7E3E0E06"
150 PATTERNS#10,"E0F0FCFEFEFFFFFF"
160 PATTERNS#11,"FFFFFF7E7E7C7060"
170 PATTERNS#12,"0000000000000000"
180 PATTERNS#13,"0D0D010700000000"
190 PATTERNS#14,"000000000000040A"
200 PATTERNS#15,"90FE020000000000"
210 PATTERNS#16,"1F3F7FFFFE3F3F3F"
220 PATTERNS#18,"F8FCFEFFFFFCFCFC"
230 PATTERNS#19,"FCFCFCFCFCFCFCFC"
240 PATTERNS#17,"3F3F3F3F3F3F3F3F"
250 PATTERNS#20,"010D2F7F7F7F1F3F"
260 PATTERNS#21,"7F7F3D3921010103"
270 PATTERNS#22,"88FCF8FCFCFEFEFC"
280 PATTERNS#23,"FCFE9E8C80C0C0E0"
290 PATTERNS#24,"000000000501030F"
300 PATTERNS#25,"060F020401000000"
310 PATTERNS#26,"000000004050E078"
320 PATTERNS#27,"C0F0C040200000000"
330 GOSUB910
340 GOSUB800
350 X1=5:Y1=10:MH=0:ME=0
360 BLINE(220,172)-(255,191),,BF
370 Y=INT(RND(-1)*100)+50
380 FORX=255TO5STEP((INT(SC+1/5))+8)*-1
390 S=Y
400 DY=INT(RND(-1)*3)+2
410 IFRND(-1)>.4THENDY=DY*-1
420 Y=Y+DY:IFY<10THENY=10
430 IFY>150THENY=150
440 IFPO=0THENPO=4:GOTO460
450 PO=0
460 SPRITE1,(X,Y),PO,1:SOUND1,110,10:SOUND0
470 PSET(225+INT(X/10),173+INT(Y/10)),1
480 PRESET(225+INT((X+(INT(SC+1/5))+8)/10),17
3+INT(S/10)),1:SOUND1,110,10:SOUND0
490 ONSTICK(1)GOSUB590,610,640,660,690,710,74
0,770
500 SPRITE8,(X1,Y1),8,3
510 IFSTRIG(1)<>0THEN1210
520 NEXTX
530 FORBV=300TO110STEP-10:SOUND1,BV,10:NEXTBV
:SOUND0
540 ME=ME+1:IFME>4THEN1310
```

```
550 COLOR10:CURSOR233,0:PRINTCHR$(8);CHR$(8)
560 CURSOR233,0:PRINTME
570 FORBV=300TO110STEP-10:SOUND1,BV,10:NEXTBV
:SOUND0
580 GOTO360
590 Y1=Y1-6:IFY1<10THENY1=10
600 RETURN
610 Y1=Y1-6:IFY1<10THENY1=10
620 X1=X1+6:IFX1>230THENX1=230
630 RETURN
640 X1=X1+6:IFX1>230THENX1=230
650 RETURN
660 X1=X1+6:IFX1>230THENX1=230
670 Y1=Y1+6:IFY1>160THENY1=160
680 RETURN
690 Y1=Y1+6:IFY1>160THENY1=160
700 RETURN
710 Y1=Y1+6:IFY1>160THENY1=160
720 X1=X1-6:IFX1<5THENX1=5
730 RETURN
740 X1=X1-6:IFX1<5THENX1=5
750 X1=X1-6:IFX1<5THENX1=5
760 RETURN
770 Y1=Y1-6:IFY1<10THENY1=10
780 X1=X1-6:IFX1<5THENX1=5
790 RETURN
800 SCREEN2,2:COLOR,1,,1:CLS
810 SPRITE2,(20,8),20,1
820 SPRITE3,(220,32),16,1
830 SPRITE4,(70,64),20,1
840 SPRITE5,(120,96),16,1
850 SPRITE6,(40,128),20,1
860 SPRITE7,(90,160),16,1
870 COLOR,10,(230,172)-(255,191)
880 COLOR10:CURSOR224,162:PRINT"RADAR"
890 CURSOR0,0:PRINT" PRISONERS SHOT:0    :PRIS
ONERS ESCAPED:0 "
900 RETURN
910 CLS:CURSOR9,5:PRINT"* C O L D I T Z *":CU
RSOR13,12:COLOR4,15:PRINT"PRESS FIRE"
920 FORU=1TO15:SOUND1,300,U:FORT=1TO10:NEXTT:
NEXTU:SOUND0
930 IFSTRIG(1)=0THEN930
940 SCREEN1,1
950 CLS:COLOR4,1
```

```
960 PRINT"Instructions
970 PRINT"
980 PRINT"You are a German prison guard
990 PRINT"on duty outside Colditz P.O.W.
1000 PRINT"Camp. You guard a nearby field
1010 PRINT"which is popular with escapees
1020 PRINT". When you hear the sound of
1030 PRINT"a running P.O.W. you must use
1040 PRINT"your radar to locate him with
1050 PRINT"your spotlight. There are trees
1060 PRINT"and huts in the field which may
1070 PRINT"obstruct your view. When you
1080 PRINT"think you've got him in the
1090 PRINT"sights of your spotlight, fire
1100 PRINT"your machine gun. A successful
1110 PRINT"shot and your spotlight will
1120 PRINT"reveal a dead P.O.W. If he
1130 PRINT"should make it across the field
1140 PRINT"you will get a demerit point.
1150 PRINT"5 demerit points and you're fired.
1160 PRINT"The escapees get quicker.
1170 PRINT"
1180 PRINT" PRESS FIRE WHEN YOU ARE READY
1190 IFSTRIG(1)=0THEN1190
1200 RETURN
1210 SPRITE0,(X1,Y1),24,8
1220 OUT127,228:FORNM=1TO6:OUT127,240:OUT127,
245:OUT127,250:NEXTNM:OUT127,255:SPRITE0,(255
,191),,0
1230 IFX>X1-6ANDX<X1+16ANDY<Y1+12ANDY>Y1-12TH
EN1250
1240 GOTO520
1250 SPRITE1,(X,Y+8),12,1
1260 FORYT=1TO5:SOUND1,110,15:SOUND0:NEXTYT
1270 FORBN=1TO30:NEXTBN
1280 MH=MH+1:CURSOR96,0:PRINTCHR$(8);CHR$(8);
CHR$(8);CHR$(8)
1290 COLOR10:CURSOR96,0:PRINTMH
1300 GOTO360
1310 SCREEN1,1:COLOR6,15:CLS:PRINT"GAME OVER
                                  ---------"
1320 PRINT:PRINT"YOU KILLED ";MH;" ESCAPEES"
1330 PRINT:PRINT"YOUR ADOLF HITLER MERIT RATI
NG IS -"
1340 IFMH<5THENRR$="GRANNY WITH A WATER PISTO
```

```
L!"
1350 IFMH>4ANDMH<11THENRR$="BEGINNER WHO NEED
S GLASSES!"
1360 IFMH>10ANDMH<21THENRR$="INEFFICIENT GERM
AN GUARD -- OBVIOUSLY NOT A TRUE GERMAN!"
1370 IFMH>20ANDMH<31THENRR$="SECOND RATE GUAR
D-GUN MADE IN TAIWAN!"
1380 IFMH>30ANDMH<41THENRR$="GOOD PERFORMANCE
 - POTENTIAL HITLER YOUTH!"
1390 IFMH>40THENRR$="CRACK SHOT-SHOULD BE S.S
., NOT GUARD!"
1400 PRINT:PRINTRR$:PRINT:PRINT:PRINT"PRESS F
IRE TO PLAY AGAIN."
1410 IFSTRIG(1)=0THEN1410
1420 GOTO340
```

```
10 PATTERNC#255,"6000000000000060"
20 POKE&H6188,0:POKE&H6189,255
30 POKE&H1839,0:POKE&H183A,255
40 DATAFE,00,C8,3E,FF,CD,F3,00,C9
50 FORX=&HFF00TO&HFF08:READA$:POKEX,VAL("&H"+A$):NEXT
60 SCREEN 2,2:CLS
70 IF RND(1)<.5THEN90
80 LINE(INT(RND(1)*255),INT(RND(1)*191))-(INT(RND(1)*255),
INT(RND(1)*191)),INT(RND(1)*13+2):GOTO 70
90 CIRCLE (INT(RND(1)*255),INT(RND(1)*191)),INT(RND(1)*60)
+1,INT(RND(1)*13)+2
100 GOTO 70
```

```
1 REM            BLOCK GAME
2 REM FROM THE SEGA DISK DRIVE MANUAL
3 REM IS NOW IN MACHINE CODE FOR DISK
4 REM                 BY
5 REM            ALEX FARKAS
6 REM
7 REM     ADELAIDE USER CLUB 1986
8 REM
9 REM
10 DATA F3,CD,3E,01,3A,22,AC,FE,01,C8,3A,20,AC,CB,6F,20,09
,3A,21,AC,CB,6F,20,10,18,1A,3A,FE,FF,FE,19,28,F7,D6,02,32,
FE,FF,18,0C,3A,FE,FF,FE,DF,28,05,C6,02,32,FE,FF
20 DATA 3E,90,32,E4,AB,3A,FE,FF,32,E5,AB,3E,00,32,E6,AB,3E
,06,32,E7,AB,3E,00,CD,A5,00,0E,13,06,00,10,FE,0D,20,F9
30 DATA 3A,F0,FF,FE,01,28,13
40 DATA 3A,F2,FF,FE,ED,28,07,C6,01,32,F2,FF,18,15,3E,01,32
,F0,FF
50 DATA 3A,F2,FF,FE,19,28,04,D6,01,18,EB,3E,00,32,F0,FF
60 DATA 3A,F1,FF,FE,01,28,21
70 DATA 3A,F3,FF,FE,8F,28,07,C6,01,32,F3,FF,18,23,3A,FE,FF
,67,C6,10,6F,3A,F2,FF,CD,E1,83,D8,3E,01,32,F1,FF
80 DATA 3A,F3,FF,FE,01,28,04,D6,01,18,DD,3E,00,32,F1,FF
90 DATA 3A,F3,FF,32,E4,AB,3A,F2,FF,32,E5,AB,3E,01,32,E6,AB
,3E,0C,32,E7,AB,3E,01,CD,A5,00
100 DATA ED,5B,F2,FF,CD,14,01,FE,01,C2,01,E0
110 DATA 1D,CD,14,01,FE,01,28,F8,1C,14,CD,14,01,FE,01,28,F
8
120 DATA 62,7A,D6,08,57,7B,C6,0C,6F,CD,0E,01,3A,F5,FF,3D,3
2,F5,FF,3A,F5,FF,FE,00,C8,3A,F1,FF,FE,00,28,04,3E,00,18,02
,3E,01,32,F1,FF,21,00,E5,CD,43,74,3A,10,A3,21,17,00,CD,E1,
83,30,03,C3,01,E0
130 DATA 3A,F0,FF,FE,01,28,04,3E,01,18,02,3E,00,32,F0,FF,C
3,01,E0,END
140 X=&HE000
150 READA$:IFA$<>"END"THENPOKEX,VAL("&H"+A$):X=X+1:GOTO 15
0
160 SCREEN 2,2:COLOR2,4,,4:CLS
170 COLOR,15,(24,0)-(239,160):LINE(24,0)-(239,160),1,B:COL
OR7:CURSOR50,172:PRINT "Block buster 3000 mk II":COLOR2
180 PATTERNC#229,"FCFCFCFCFCFCFCFC":FOR A=8TO80STEP16:CURS
ORO,A:PRINT "          ▬ ▬ ▬ ▬ ▬ ▬ ▬ ▬ ▬ ▬ ▬":NEXT
```

```
190 PATTERNS#0,"7F7F7F7F00000000"
200 PATTERNS#1,"8000000000000000":MAG2:ME=5:POKE&HFFFE,127
:POKE&HFFF5,55:RO=1:SC=0:POKE&HE04F,&H13
210 BEEP1:POKE&HFFF2,INT(RND(1)*180)+32:POKE&HFFF3,92
220 POKE&HFFF0,INT(RND(1)*2):POKE&HFFF1,0:BEEP0
230 CALL&HE000
240 IF PEEK(&HFFF5)=0THEN270
250 FORA=1TO40:BEEP1:BEEP0:NEXT:ME=ME-1:IFME=0THENSC=SC+(5
5-PEEK(&HFFF5)):GOTO 340
260 GOTO210
270 SC=SC+55:X=PEEK(&HE04F):IF X=1THEN290
280 X=X-1:POKE&HE04F,X
290 RO=RO+1:IFROMOD3=0THENME=ME+1:BEEP:BEEP
300 IF RO/2<>INT(RO/2)THENFOR A=8TO80STEP16:CURSORO,A:PRIN
T "                                                    ":NEXT:POKE&HFFF5
,55:GOTO210
310 CURSORO,8:PRINT "                                  
■":CURSORO,24:PRINT "                                  
320 CURSORO,40:PRINT "                                 
■":CURSORO,56:PRINT "                                  
"
330 CURSORO,72:PRINT "                                 
■":POKE&HFFF5,55:CURSORO,88:PRINT "                                    
■":GOTO 210
340 COLOR1:CURSOR80,100:PRINT "Game over.":CURSOR80,116:PR
INT "Score:";SC:CURSOR80,132:PRINT "Round:";RO
350 IFINKEY$<>" "THEN350
360 BLINE(25,1)-(238,159),,BF:BEEP:COLOR2:GOTO 180
```

PROGRAMS FROM N.S.W

CARDFILE : BURGLAR BILL **
SEGA MONITOR : BASWORD
ONE DAY CRICKET : BACKGAMMON
DUNGEONS/CAIRO : CAVERNS OF KARA
SATALITE SALVAGE **
BUSSINESS PAC I
PANDAMONIUM **
AERO-BAT (FLIGHT) **
DISKWASHER II **
CENTRONICS INTERFACE:RS232C INTERFACE
PROGRAMS IN MACHINE CODE **
CONTACK THE ADELAIDE SEGA USER CLUB
TEL 382-7967 : 263-5020 : 264-2747
JAN JAMIE JOHN

<<*>> I K E L A <<*>>

IKELA V1.1 is a multi-function program in which an Editor/Macro Assembler/Source File creator and a Debugger have been combined. This package brings machine code within the reach of all and by using the TRACE facility is an excellent aid to learning Assembly language.

IKELA is supplied under licence which entitles the licencee to one years Guarantee, backup's And to any updates that happen within that time plus any other help that may be needed.

IKELA supports either a Serial or Centronics Printer and comes to you for the LOW price of $ 80.00 which includes a very comprehensive manual of 90 pages.

For further information contact

Scott MacDonald PHONE : 046.668956
TAWARRI
2 Coolalie Ave,
CAMDEN 2570
New South Wales.


John Sands
SEGA®
John Sands
SM8000
Monitor
SEGA SC-3000H
SegaCom