MICRO-80

P.O. BOX 213, GOODWOOD, S.A. 5034 AUSTRALIA
TELEPHONE (08) 381 8542

ISSUE 2
JANUARY 1980 Price $2.50

CONTENTS

MICRO-80 is registered for posting as a publication - CATEGORY B.

Printed by: The Shovel and Bull Printers,
312 A Unley Road, Hyde Park, SA 5061.

All enquiries should be sent to:-
MICRO-80, P.O. BOX 213, GOODWOOD, SA 5034.
'Phone: (08) 381 8542

MICRO-80

MICRO-80

P.O. BOX 213, GOODWOOD, S.A. 5034 AUSTRALIA
TELEPHONE (08) 381 8542

** ABOUT MICRO-80 **

MICRO-80 is the first Australian monthly magazine devoted entirely to the TRS-80 microcomputer. It is available by subscription $24.00 for 12 months or by mail order at $2.50 per copy. A cassette containing all the programs in each month's issue is available for an additional $3.50 or a combined annual subscription to both magazine and cassette, is available for $60.00. Special bulk purchase rates are also available to computer shops etetc. Please use the form in this issue to order your copy or subscription.

The purpose of MICRO-80 is to publish software and other information to help you get the most from your TRS-80 computer and its peripherals. MICRO-80 is in no way connected with the TANDY organisation.

** WE WILL BUY YOUR PROGRAMS **

Most of the information we publish is provided by our readers, to whom we pay royalties. An application form containing full details of how you can use your TRS-80 to earn some extra income is included in every issue.

** CONTENT **

Each month we publish at least one applications program in Level 1 BASIC, one in Level 2 BASIC and one in DISK BASIC (or Disk compatible Level 2). We aslo publish Utility programs in Level 2 BASIC and Machine language. At least every second issue has an article on hardware modifications or a constructional article for useful peripherals. In addition, we run articles on programming techniques both in Assembly language and BASIC, we print news from TRS-80 Users Clubs, letters to the Editor and new product reviews.

** ADVERTISING **

We accept camera ready copy for display advertising at the following rates:

- FULL PAGE (19cm wide x 24cm high) $120
- 1/2 PAGE (19cm wide x 14cm high) $ 60
- 1/4 PAGE (19cm wide x 7 cm high) $ 30

Classified ads are $8.00 for up to 50 words. Ads must be submitted by the 15th of each month in order to appear in the following month's issue. A Company Order or payment must be included with the advert.

** TRS-80 USERS CLUB NEWS **

We are prepared to print news of the activities of TRS-80 Users Clubs up to a maximum of 200 words per Club per month, space permitting. Copy must be TYPED with DOUBLE LINE SPACING and reach us NO LATER than the 15th of each month in order to appear in the following month's issue.

** COPYRIGHT **

P.O. BOX 213, GOODWOOD, S.A. 5034 AUSTRALIA
TELEPHONE (08) 381 8542

** EDITORIAL **

Well, MICRO-80 got off to a great start with issue 1. A lot of people who bought single copies "on approval" have sent in for subscriptions or ordered cassettes and many have written congratulating us. We've already had a number of programs submitted for possible publication. We are serious about paying to publish your programs. For instance, we have paid out over $100 to readers for the programs published this month. As our circulation grows, we will be able to afford to pay even more. If you like MICRO-80, why not tell your friends about us and encourage them to take out subscriptions. That way the magazine will grow and you'll get even more progrms.

One of our problems is finding interesting Level 1 programs. Most experienced programmers of our acquaintance work with Level 2. So, if you have a Level 1 machine, why not send in those programs you have been working on, they're probably just what we need . We've been asked if programs have to be written efficiently to be published. The answer is no they don't, as long as they perform adequately. We consider one of our objectives is to help TRS-80 users become better programmers. What better way to do this than to help each other. If we publish a program and you know a better way to acieve the same result then why not write in to explain it. We'll publish your letter and then we will all learn something.

A number of our readers have one or more Disk drives. There is now a wide variety of Disk Operating systems available including TRS DOS 2.1, 2.2, 2.3, NEWDOS, VTOS and DOS 3.0. We would be interested in publishing a review of these various systems so, if you have experience of using one or more of them, perhaps you'd like to write in with your opinions. Better still, maybe there is someone out there who has wide experience of most or all of them and could write a comparative review single-handed. If so, we'd like to hear from you.

Future plans include a series of articles on advanced BASIC programming . The objective is to help you improve your programming technique and make the most of Level 2 and Disk BASIC. We envisage that this series will require a number of contributors and would like to hear from any of our readers who feel they could write one or more articles. Naturally, we will pay for all material published but, before sending in an article, write and tell us the topics you want to cover so that we can co-ordinate your efforts.

We are also working on some hardware projects. The light pen project was announced last month. It is proving to be a little erratic in operation so we still have a few bugs to sort out. Look for the article on it in about 3 to 4 months time. Not because it is going to take that long to sort out but because it has now taken a back seat to an even more exciting project. We are developing a memory expansion board wich will take a set of 4K chips and up to two sets of 16K chips. This board will connect to the back of the CPU via a cable and have its own power supply. It will cost very much les than the expansion interface and will enable you to take advantage of the cheap 16K chips which are available from a number of suppliers for around $110 per set. Using this board you could buy a set of 16K chips install them in the CPU, then plug the 4K chips from the CPU into the board thus giving you a 20K machine. Later you could buy another set of 16K chips and plug them into the board to get a 36K machine. Or you could have two 16K sets in the board,

making a 48K machine. For those who want lots of memory but don't need Disk drives, this should be just the shot. It's a big project and will take several months to complete. In the meantime, we'd be interested to hear from readers who would consider buying such a unit. It is too early to be certain about price but we should be able to sell a fully assembled unit without memory chps but including cable, case and power supply, for around $200.

As you can see, things are really happening at MICRO-80.

** ABOUT OUR OWN SYSTEM (AND LEVEL 1) **

MICRO-80 has a Level 2, 48K machine with lower-case modification, a single Disk drive and a Comprint 912 parallel interface printer. This month we have composed the magazine using the Electric Pencil word processing program and the Comprint printer which prints characters in a 12 x 9 dot matrix on electro sensitive paper.

We generally run our disk system under TRSDOS 2.2 but have recently acquired NEWDOS +. The big advantage of NEWDOS + to a magazine such as ours is that it provides a Level 1 in Level 2 program. That enables us to write, evaluate, save, load, modify etc., Level 1 programs on our Level 2 machine. Printing Level 1 programs is another matter. Last month we had to convert our Level 1 programs into Level 2 and print them out using the LLIST command. For this month, we have written a special machine language routine that loads into high memory (7000H). We use the NEWDOS SAVE* command to move the Level 1 pogram to 8000H, then load and run our special LLIST routine. It lists the program, starting from the first line number, located at 8003H.

In writing this special machine language program, we've learned some things about Level 1 BASIC which might be of interest to our readers. Level 1 BASIC programs start at 4200H. The first byte is 2A Hex or * in ASC11. The next two bytes give the program length in Hex, least significant byte first. The next two bytes then give the first line number in Hex, again least significant byte first. Then follows the first program line in ASC11 terminated with a line feed (0DH). The next two bytes are the second line number in Hex, least significant byte first, then the program line in ASC11 and so on. Unlike Level 2, Level 1 has no special end of program identification, since the length of the program is indicated at the beginning. All this will be clearer from the example below.

** Example **

Consider the simple 2 line program:

```
 1 F.A=1 TO 5
10 P.A:N.A
```

Counting 2 bytes for each line number, one space between the line number and the program text and a Carriage Return at the end of each line, this program takes 25 bytes.

A Hex dump of the memory block containing this program is shown below.

```
4200 2A (19 00) (01 00) (20  46 2E 41 3D 31 20 54 4F 20 35)
     *   length  line no.          program line

4210 (0D) (0A 00) (20 50 2E 41 3A 4E 2E 41) 0D
      CR   line no.      program line       CR
```

The equivalent ASC11 dump is:

```
4200 *.... F.A=1 TO 5
4210 ... P.A:N.A.
```

Compare the Hex dump with the ASCII dump. The first byte 2AH, corresponds with the ASCII * and is a start of program identifier. The next two bytes taken in reverse order, are 0019H, which corresponds to 16 + 9 = 25 (Dec). This is the length of the program. Since there is no ASCII character equivalent to these Hex numbers, they are shown as full-stops. The next two bytes, taken in reverse order, are 001H = 1 (Dec) which is the first line number. Again there is no equivalent ASCII character, so these are shown as full-stops too. The first program line starts with 20H, which is an ASCII "space", typed in as part of the program. Although when you list a Level 1 program on the screen, a space always appears between the line number and the first program character, this space is not stored in memory so, if you want to conserve RAM, don't type in a space after the line number. The next byte is 46H, equivalent to ASCII "F", the first instruction and so on to address 4210H which contains 0DH, the Hex equivalent of an ASCII Carriage Return (CR). Again, there is no ASCII character for a CR (it is a control code), so it is shown as a full-stop. Then follows the next program line until the 25th byte which is 0DH, the CR for the last line.

-00000-

********** NEXT MONTH'S ISSUE **********

The February issue of MICRO-80 will contain at least the following programs:

HANGMAN (L1) - This program is similar to the Level 2 version published this month but, of course achieves the result in a different manner, since Level 1 does not support string manipulation.

AMAZIN (L2) - A program which generates a series of mazes on the screen, for you to find your way through.

ANALOGUE CLOCK (L2 + m/c language) - This program draws a large clock face on the screen, it uses high speed m/c language sub-routines to redraw the clock hands as they move.

SET 2 (m/c language) - This m/c language program adds a new graphics command to Level 2 BASIC - SET<X1 Y1,X2 Y2> which rapidly draws a line between point X1 Y1 & X2 Y2. It can save a lot of tedious programming and is much faster than BASIC graphics.

MONITOR FOR BASIC (Installment 1) (m/c language) - This m/c language program, which will be presented in self-contained modules over the next three issues, greatly improves the versatility of the TRS-80 when programming in Level 2 BASIC. For example, it gives you a MERGE COMMAND, which "hides" the resident BASIC program, allows you to load in another program, automatically renumbers it so that no line numbers clash and then joins the two together. Another instruction allows you to "hide" a program in memory, load another, modify it and save it and then return to the original program. It will also enable you to renumber BASIC programs, list all variables used, make system tapes and carry out a number of other useful functions.

February's issue will also contain the next installment on Assembly Language programming.

-00000-

** ASSEMBLY LANGUAGE PROGRAMMING **

This is the first article in a series which will continue for about six months. It is assumed that, at present, the reader has very little knowledge about the inner workings of the TRS-80 or its Microprocessor, the Z80. This series will therefore start at a very basic level, defining terms, explaining memory mapping and so on. By the time the series is finished, however, we will have covered some quite sophisticated programming techniques and readers should have a good understanding of the Level 2 ROM, how to use many of its sub-routines in their own programs and what many of the addresses in reserved RAM are used for.

In order to make the best use of these articles, you will need three programs: an Editor/Assembler, a Monitor and a Disassembler. Tandy can supply the first two. The Editor/Assemler is good but has two disadvantages. First, it is expensive ($59.95 on cassette, $139.95, for the mpre sophisticated disk version). Second, all source code, comments etc., are stored in RAM in uncompressed ASCII format, this takes a lot of space, sometimes as much as 5 or 6 times the length of the final object (machine language) program produed. When you consider that the Editor/Assembler itself takes about 5K, if you have a 16K machine, you would be hard put to it to assemble an object program of more than about 2K. Many machine language programmers we know even run into problems with 32K machines. That having been said, the Tandy program is a good one and is convenient to use. However, if you havn't already purchased an Editor/Assembler, you might like to wait a few weeks. MICRO-80 is currently negotiating with some American software suppliers and we hope to be able to offer our readers an Editor/Assembler for no more than $45.

Tandy's monitor program is TBUG. It's greatest virtue is that it only takes about 1K of RAM. A number of other monitors which are available, such as MON 3 and RSM-2, are much more sophisticated than TBUG but do take 3 to 4K of RAM. The major criticism of TBUG is its price. At $27.95 it really is not very good value. Again, if you can wait we should soon have a range of monitors. A small one, equivalent to TBUG, for under $20. and more sophisticated monitors for about $28 to $30.

Finally, there is the Disassembler. Tandy doesn't offer one nor, to our knowledge, does anyone else in Australia, except as part of the MON 3 monitor. In view of this, we are looking for a suitable, stand-alone disassembler to offer in our software range. We expect that all the above programs should be available from MICRO-80 by March.

As a further help to the Assembly Language programmer, we are preparing a disassembled listing of the Level 2 ROM with comments and entry points for useful sub-routines, together with a detailed map of reserved RAM. This should be available as a separate publication in March.

For the moment, however, let's discuss some of the fundamentals of the TRS-80 microcomputer. The heart of the TRS-80 is the Zilog Z 80 microprocessor. This is a third generation, 8-bit microprocessor. The first generation was represented by the 8008 and the second by the 8080. In developing each generation, the designers followed a policy of "upward compatability" for software. This means that programs developed for computers using the 8008 will run on computers using the 8080 or the Z 80. This was achieved by preserving the basic 8008 instruction set (the codes which cause the microprocessor to carry out different operations such as moving data, adding, comparing, etc.,) through each generation. By the time the Z 80 was

designed, this instruction set and some of the basic "architecture" which went with it were rather outmoded, so the Z 80 has it's own more powerful instruction set, added to the 8080 set of 78 instructions. There are 158 different instructions you can use with the Z 80.

Wait, before you turn over and go back to your BASIC programs you should know that you don't need to be totally familiar with all 158. Because of the upward compatability mentioned earlier, many of the 8080 instructions can be ignorged because there are much more powerful ones available in the new Z 80 instructions. There are two schools of thought on learning Assembly Language programming. One says you should learn to crawl before you walk and use the Z 80 like an 8080 until you are "experienced" enough to go on to the Z80 instructions. The other, and the one to which we subscribe, is that you should start off from scratch learning to use the Z 80 instruction set to its full capacity. There is no doubt that pogramming an 8080 is more tedious than programming a Z 80. Life's too short to waste that way, so we will use the best instructions available on the Z 80 for each task, be they from the 8080 sub-set or the new Z 80 set.

A single Z 80 instruction is like a BASIC command or statement but much less comprehensive. Whereas, with one BASIC statement such as CLS you can clear the screen and return the cursor to its home position, to do the same thing in machine language would require dozens of instructions. In fact, when you use any BASIC statement the Z 80 first converts it into a series of Z 80 instructions, under the control of the BASIC INTERPRETER program stored in the ROM (read-only memory). It then executes these instructions. This can result in the Z 80 carrying out hundreds or even thousands of instructions for one BASIC statement. Therefore, if you write programs directly in machine language, (Z 80 instructions) a given result will be obtained many times faster than if you had used a high level language like BASIC.

As with any other computer, there are three fundamental components in the TRS-80:

- CPU (Central Processing Unit), the Z 80
- Memory to store programs and data
- Input-Output devices so that you can communicate with the computer and it with you.

The basic TRS 80 comes with three input-output devices. The keyboard, which is an input only device. The video monitor, which is an output only device and the cassette recorder, which is both an input and an output device. In addition, you can buy other output devices such as a printer or a voice synthesizer and input-output devices such as floppy disk drives or the RS 232C serial interface.

The memory of a computer, is to a great extent, a measure of its data processing power, so computer owners usually classify their machines by memory size eg. 4K, 16K, 32K, or 48K. K means 1024 bytes. The byte is the basic memory cell, so a 4K machine has 4 x 1024 = 4096 separate memory cells in which information can be stored. In the case of the TRS-80, a byte consists of 8 binary digits (bits). If you are familiar with physics, you could liken a byte to a molecule and a bit to one of the atoms which go to make up the molecule. Since there are 8 bits in a byte and every bit can assume the value of either 0 or1, there are 2 to the power 8 = 256 possible combinations of bits and hence a byte can assume any value between 0 and 255 (decimal)

depending on the individual values of the bits which it contains. So, if you want to store a number between 0 and 255, you can store it in a single memory cell. But, if you want to store higher numbers, you must use 2 or more memory cells for each number.

We mentioned that the byte is the basic memory cell in the TRS-80 and that you can store values in these cells. However, there is little point in storing values if you can't use them again later, so any computer must have some system for retrieving values from its memory. The TRS-80, in common with all modern digital computers, uses a random access memory system (RAM). In this system, every memory cell has a unique location or address, just as houses along a street each have their own number. The Z 80 CPU has 16 address lines, each of which can be on or off (0 or 1). The maximum number of combinations for these address lines is therefore 2 to the power 16 = 65,536. The Z 80 can therefore address 65,536 unique locations (or bytes of memory). There are 1024 bytes per K byte, 65,536/1024 = 64, hence the Z 80 can address 64K of memory. (Note that 0 is the first address so that the highest address is 65,535 not 65,536).

There are three different types of memory used in the TRS-80 under this random access system. The first is ROM or "read-only memory". This means that the values are put into the memory by Tandy and can't be changed by the user. The Z 80 can read these memory locations, however, and act on the instructions contained in them. The ROM contains the Level 1 or Level 2 BASIC interpreter program which is always there. In fact, having the interpreter in ROM was one of the best features of the TRS-80 when it was first announced. Many earlier computers required the user to load the BASIC interpreter in from a cassette each time the computer was turned on. Imagine how tedious that would be! The ROM in the TRS-80 occupies the lower memory addresses: Level 1 from 0 to 4095 (4K) and Level 2 from 0 to 12,287 (12K). On power up, the Z 80 goes straight to an address in ROM and starts executing the program contained there. The BASIC program you write is actually data used by this interpreter program.

Another kind of memory used by the TRS-80 is Read-Write memory. This is "volatile" memory in which the Z 80 can store (write) data or from which it can retrieve (read) it. Your BASIC programs are stored in this memory and, when you switch the computer off, you lose everything in it. Obviously, this is a pretty important kind of memory, so why haven't you ever heard of it? You have, but it is generally referred to as RAM (for random access memory). As you can see from the above, it is no more nor less random access than ROM but, who are we to argue with the crowd? From now on, we shall refer to read-write memory as RAM. There are several important things to know about RAM. First of all, it occupies the higher memory addresses, starting from location 16,384 and, if you can afford it, going all the way up to location 65,535. You cannot place RAM in the first 16K memory locations because these are occupied by ROM and memory-mapped devices (see below) thus the maximum RAM capacity in a TRS-80 is 48K. All RAM starts at memory address 16,384, 4K goes up to address 20,479, 16K goes to 32,767 and 32Kto 49,152. Finally, not all of RAM is available for your own programs, data etc. Some is reserved for use by the ROM. In Level 1 machines, the first 513 bytes of RAM are reserved whilst in Level 2 machines the first 812 bytes are reserved. But don't feel cheated. This reserved RAM is a godsend for the machine language programmer as it lets you escape from ROM routines back to your own programs, thus letting you make use of a lot of the programming done by Microsoft (who developed the Level 2 BASIC ROM for Tandy).

There are still 4K of memory locations unaccounted for, from 12,288 to 16,383. These addresses are used for MEMORY-MAPPED input-output devices. The upper 1K (15,360 to 16,383) is used for the video display. This display has its own set of RAM chips and you can use them like any other part of RAM, with the added advantage that you can see what's in them! Type in and run the following program:

```
10 CLS
20 FOR C=15360 TO 15871
30 POKE C,65
40 NEXT C
```

Well, you've filled half the screen with A's, so why is the prompt at the top of the screen? The reason is that you've stored the ASCII equivalent of A directly into many of the screen memory locations without the cursor control routine knowing anything about it, as it would have, had you used a PRINT statement. Now add the following lines to your program and run it again.

```
50 FOR C=15360 TO 15871
60 POKE C,32
70 NEXT C
```

This time you cleared the screen by poking ASCII spaces into its memory addresses and still the cursor control knew nothing about it. This gives you an idea of how much is involved in a simple PRINT statement which not only puts the appropriate value into the correct memory address but also advances the cursor, moves it to the next line and scrolls everything up when you finish printing the bottom line. All the instructions to accomplish this are contained in a sub-routine in the ROM. In a later article, we will show you how to use this sub-routine to save yourself a lot of programming/debugging time.

The next most important use of the memory-mapped area is the keyboard which occupies addresses from 14,366 to 14,464. Again, in later articles we will show how to make use of this information in your own programs. The rest of the memory-mapped area is used to communicate with the printer, floppy disk, cassette etc., or for interrupts or is not used at all, at least at present.

We explained earlier, that the basic unit used by the TRS-80 is a byte which can have 256 different values depending on the status of each of its 8 bits. The value of a byte can be described in a number of different ways. Until now, we have used the familiar decimal notation and that is the notation used by Tandy in Level 2 BASIC. All addresses are expressed as decimals and, when POKE statements are used, decimal values are POKED into the decimal address specified. The decimal system would be fine if we were always dealing with numbers, but the value of a byte can represent things besides numbers. It might, for instance, represent a letter of the alphabet, or status information about the line printer or an instruction code for the Z 80 microprocessor and so on. Again, we may not want simply to carry out mathematical operations on it. We may want to do some logic operations on certain bits or split it into two blocks of four bits rather than one of eight. Much of this would be difficult to do if we used only decimal numbers to represent the value of a byte. An alternative would be to consider all eight bits individually. This involves the binary system of mathematics and because it is the most basic system, is often used in computers. Its disadvantage is that binary numbers are long and tedious to use and it is very easy to make mistakes.

To overcome this the Z 80 uses a Hexadecimal system. This is a system of mathematics using the base 16 (decimal). A single Hexadecimal number (identified by the suffix H) can have any decimal value between 0 and 15. The numerals 0-9 are used to represent the first 10 values then the first six letters of the alphabet, A-F, are used to represent values from 10-15.

Eg. 9H = 9 Dec., AH = 10 Dec, FH = 15 Dec.
10H = 16+0 = 16 Dec, A2H = 10x16+2=162 Dec., etc.

The great advantage of the Hexadecimal system is that it allows any 4-bit quantity to be expressed as one digit so that two digits express the value of any Byte. It can be easily converted to binary and is much easier to work with. Believe it or not, after a little practice, you will find it easier to think of memory addresses and byte values in Hex than in decimal.

ASCII stands for American Standard Code for Information Interchange. It is a code, formulated several years ago, by which all of the characters that may be input from the keyboard or printed on the screen are represented numerically. Although ASCII is only a 7-bit code, 8-bit bytes are always used to hold the ASCII values inside the TRS-80. Appendix C of the Level 2 BASIC reference manual lists the codes. For example, 65 is the letter "A" and 32 indicates a blank space. Although the TRS 80 can only display upper case ltters, it can both input and output lower case letters. If you hold down the shift key and type any letter, you are actually typing a lower case letter although you'd never know it because it is displayed in upper case. (Note that this is the reverse of typewriter keyboards). Furthermore, if you type in a BASIC program in lower case, it will still work. The only discrepancy is with the "@", if you PRINT @ somewhere and type a lower case "@", it won't work.

The important point about upper and lower case is that the TRS-80 is fully capable of computing with lower case letters, it merely can't display them on the video monitor. In fact, the TRS-80 has most of the hardware required to display lower case letters and there is a simple modification to enable it to do so. We have converted ours and find it very useful when using the Electric Pencil. In a future issue, we will describe how to carry out this conversion on your computer for just a few dollars.

The 7-bit ASCII code has room for 128 values, but not all of these are used for displayable characters. The first 32 values (0-31) are used for control codes, which are also described in Appendix C. Since the 7-bit values are always kept in 8-bit bytes, that leaves room for 128 more values for special purposes. The numbers from 129 to 191 are used for Graphics codes whilst the remaining numbers from 192 to 255 are used for space compression codes.

Numerical values used in calculations are stored in several ways. The simplest are BASIC integers, which are held in 2-byte numbers. The maximum value of an integer is 32,767 rather than 65,535, because the first bit is used for the sign, so that you can compute with negative numbers. That explains why, when you want to POKE a value into an address above 32,767 you need to use the formula "POKE" - 1* (address required - 32767),XX". There is an unusual feature about 2-byte integer numbers in the Z 80 when they are used to represent addresses of memory locations. The two bytes are stoed "backwards" in memory, so that address 1234H would be stored in memory as :

XXX1H 34
XXX2H 12

This is a hangover from the days of the 8008 and 8080. Just remeber when interpreting addresses from a Hex listing of memory contents the least significant byte comes first, followed by the most significant byte (see the paragraph in this issue headed - About our own system (and Level 1)).

The storage and manipulation of single-precision and double-precision, floating-point numbers is a much more complicated process and will be a large part of the contents of a future article. Next month we will discuss the architecture of the Z 80 and start looking at its instruction set.

-00000-

** Game of LIFE continued **

Only when the new display is ready is it transferred to screen memory. Bluntly it is the last of the rules that makes Life rather difficult to set up on a computer with reasonable running speed. It is also this last rule which makes possible some of the very beautiful and interesting patterns which develop as each successive generation is displayed.

The original concept of Life was that the overall population would enjoy periods of growth and expansion, suffer periods of recession and contraction, and that in all cases would stabilise - with either a pattern of growth and contraction that repeats continuously, or with a stable and non-changing pattern.

This has always proved to be the case!

MICRO-80

P.O. BOX 213, GOODWOOD, S.A. 5034 AUSTRALIA
TELEPHONE (08) 381 8542

** SOFTWARE SECTION **

The listings for these programs are at the back of the magazine. This section explains what each program is about and how to use it.

** RICOCHET - L1/4K ** by C. Bartlett

4K users will have to use full abbreviations and all multiple statement lines in order to fit this program in. Unfortunately, our nice new machine language LLIST program mentioned earlier in this issue, introduced one or two new statements of its own, so we have had to convert Ricochet to Level 2 and LLIST it that way. Remember to use PRINT AT instead of PRINT @. Otherwise, everything should be straightforward.

This program commences by drawing a wall down the centre of the screen. Randomly spaced along the wall are seven gaps and lined up with these gaps are seven targets to the far right. Walking up and down the screen at the left is a man. When you think he is in the right position, press the "A" key, the man will stop, raise his arm and fire. If you stopped him corrctly the bullet will pass through the gap in the wall and hit the target. If not, the bullet will ricochet off the wall and kill him!

You have 3 men and 7 targets per game. Pressing the "ENTER" ke will cause the man to reverse direction. Sounds easy doesn't it - try it!!!

** FRUSTRATION L2/4K ** by Peter Hartley

This is a good party game for kids. When you run the program, a pair of Zeros flashes randomly and rapidly in the centre of the screen. At a random time, this "clock" starts counting. The idea is to push any key as quickly as possible after the clock starts running. If you push too early, the computer accuses you of cheating and then the clock starts all over again. After ten tries, the computer prints out your highest and lowest scores and the number of times you cheated. It also computes a % rating in accordance with the formula in line 330. Just a warning, if you don't like having the keys on your comouter bashed, go on to the next program!

** HANGMAN L2/4K ** by Bernie Simson

Last month, we advertised this program for L1 mahines. It turned out to be for L2 machines. It can be done on a L1 machine - we've got a half-developed program to do it. It's a lot more difficult of course, because you can't use string manipulation. We will publish the L1 Hangman next month. In the meantime, those with L2 machines can enjoy this very well presented version by Bernie Simson. This is a good educational game for kids from about 6 up.

If you leave out lines 2-9 inclusive, Hangman should fit into a 4K Level 2 machine (just). The game starts by asking you whether you are a novice or professional (professionals get harder words). It explains the scoring and then prints out a series of dashes near the top of the screen. Each dash represents a letter so, 5 dashes - 5 letters in the word. You then enter a letter. If it is contained in the word, the computer puts it into its correct position(s). If it is not in the word, the computer starts drawing a scaffold with a man on it. Too many errors and the man (that's you) gets hanged. One of the nice features about this program is that, if you win, next time you get a harder word, but if you lose you get an easier word. We know you'll enjoy this one.

** GAME OF LIFE L2/16K ** by Peter Hartley

This program is in two parts, a machine language sub-routine which carries out most of the computations and a BASIC program used to set up initial parameters. The BASIC program calls the machine language program via the USR command. To enter the program first answer the Memory Size question with 20479, then, using a suitable monitor, type in the machine language program from the Hex listing. At this point you should use your monitor to make a system tape. Parameters are:- Start 5000H, End 519AH, Entry 0000H. Now enter the BASIC listing and CSAVE that. To start, simply enter RUN and the BASIC program will take over control and call the machine language sub-routine as required. When re-loading LIFE from tape first protect memory size then enter the machine language sub-routine using the system command, return to BASIC then CLOAD the BASIC program and run it.

The game of life was originally discussed in The Scientific American some eighteen years ago. It isn't really a game at all, but a rather complex set of rules governing the growth of cells (or civilisations), which this programme illustrates graphically on the TRS-80's screen.

The rules are as follows:-
1) Each cell (2X by 1Y in this version) is capable of life
2) Life will be created in a cell location if it has exactly three living neighbours
3) Life will continue if a cell has only two or three living neighbours
4) Life will cease if a cell has less than two living neighbours (we all need some companionship!)
5) Life will cease if a cell has more than three living neighbours (since three cells are required to create life,it sands to reason that "four's a crowd" therefore five is unlivable!)
6) All births and deaths must occur simultaneously.

This version of life is in two parts. The Basic programme is quite self explanatory, offering the option of user or computer generated starting patterns, variable generation speeds (up to one per second), and single key termination at any time.

The machine language routine does all the real work, making massive use of subroutines. Each run through this routine will create one new generation.

First of all the screen display is duplicated into memory. This is to eliminate annoying "glitches" as the display is accessed through the routine.

Each pair of bits (0-1,2-3,and 4-5) are treated as one - this eliminates vertical stretching of the display - and each dislay line is treated separately. The top and bottom lines are "wasted" at this stage. This version does not permit any vertical "wrap-around", but it does occur horizontally. If you don't want this you will need to add a line to the basic routine, to reset X0, X1, X126, and X127 right down the screen prior to the USR call.

In turn, the neighbours of each bit-pair (or cell) are examined to see whether or not they are live (set). A tally is kept and a second display is gradually built up in memory, on the basis of the rules outlined above.

** Continued on Page 10 **

** STOCK RECORDING SYSTEM L 2/4K ** by Keith Neighbour

This program is really neat as it illustrates a method of sorting which is delightfully simple but which we have never seen used before. The idea is to make part of the stock number of your items the same as the line number where its data is stored in the program. You do this by entering the information as a DATA statement. The TRS-80 then stores this information in order of line number (and hence stock number). You can then list or print out any information you want about the stock items and they will be listed in numerival order. Simple isn't it. Why didn't we think of it first?

In fact, this program was originally written to help a Tandy Store Manager keep track of his own stock, soles etc., and it has proved very useful. As well as a 4 digit serial number, Tandy uses a 2 digit series number, as a prefix. The program, as listed, will list all items in any nominated series, and compute the total sales value by multiplying the change in quantity of each item by its selling price and totalling them. It will also list out the data concerning each stock item and of course, new stock items can be added or information about existing ones altered. In order to show you how it works, the following examples have been included:

Series #	Cat. #
120	654
120	655
14	841
40	878
21	998
12	1313
12	1346
12	1879
31	1984
31	2081
12	2432
21	9458

Lines 1 to 9999 are reserved for the data statements. Line 9999 itself is used as an end of data marker. When the program is listing items and encounters 9999 in a data line, it knows it has reached the end of the stock list. You can, of course, use any valid line number for your DATA statements, but, make sure you don't crash the program itself, which is best kept as a block of line numbers on its own.

Type in the program and run it. A menu will be printed out on the screen. Enter the appropriate number for the action you want to take. Full instructions are included in the program. They do need careful reading. Don't forget to enclose the data in " "s because it is read from the data statement by a string variable.

The logic behind this program can also be used to write a simple, data-base management system. Keith has one of those too, which we will publish next month.

** MONITOR IN BASIC L2/4K ** by Ian Vagg

This program makes use of PEEK and POKE statements and also HEX/DEC and DEC/HEX convertor sub-routines to allow you to examine and modify memory location. It will enable you to key in the machine language programs we publish even if you haven't got a m/c language monitor.

But, perhaps best of all, it allows you to dump the contents of a block of memory to tape and then load it again. Admittedly it is slow, but it works.

First of all, you specify the program title and its start, end and entry addresses. These are recorded on to cassette as data using the PRINT#-1, statement. Then 60 bytes of memory are PEEKed at and their contents are packed into a string variale, using a minus (-) sign as a separator. This string variable is then recorded as ddata and so on until the whole block is dumped. The loader program carries out the reverse operation and POKEs the data back into the addresses it came from. It is slow because each 60 bytes is separated by the 4 second tape leader. Nevertheless, it does enable a standard Level 2 machine, working in BASIC, to load and save machine language programs on tape.

After you have entered the program, type RUN and a menu will be displayed on the screen. This should be self explanatory. You can return to the menu at any time by entering H.

-00000-

** CORRECTIONS **

DIGITAL CLOCK (December Issue) Well, it happens to everyone else and now it has happened to us. There were a couple of errors in the listing of the Digital Clock program published last month. Those who ordered their magazine later in the month got copies corrected by hand (ugh!). For the rest of our readers please make the following changes:

Line 280 GOTO 400
Line 285 change the line number to 245

-*****-

** HARDWARE SECTION **

Last month we published details of some modifications which can be made to improve the operation of the earlier cassette recorder supplied by Tandy, the CTR-41. Some of these were designed to give the CTR-41 many of the features of the newer CTR-80, others allow you to hear your CLOADs and CSAVEs and would be useful on a CTR-80 also. Last months instructions should enable you to carry out this mod.

One problem peculiar to the CTR-80 is that, with some models, if you use the recorder STOP key whilst loading, a pulse is recorded in the middle of your program and the tape is unusuable. Apparently the pulse is caused by an inductive spike from the motor, finding its way to the recording head. Tandy have fixed this on models manufactured after 1979. Look inside the battery compartment of your recorder. There should be a sticker carrying the serial number. After the serial number are three or four characters such as 10A9 or 1A9. These record the manufacturing date. The first number (1 or 10 in the example) is the month of manufacture, 1 = January, 10 = October etc. The A is a separator and appears to have no significance (it probably means 1970's). The last number is the last digit of the year of manufacture. 8 = 1978, 9 = 1979 etc., If your recorder was manufactured earlier than February 1979 (2A9) then you probably have this problem. Various fixes have been tried, including diodes and large capacitors across the motor but we recommend that you take it back to Tandy, who should fix it for free.

-00000-

** LETTERS TO THE EDITOR **

From: E. J. MONAGHAN, "WARRAWEE" GREENWELL RD., PYREE NSW 2540
Tel. (044)47 1743/1728

We are using a TRS-80 Level 2,32K with two disk drives and a Line printer to control a Freisan Dairy Stud of about 500 animals and some farms. We would be interested in contacting by letter or 'phone, anyone who is using or considering using a TRS-80 for rural purposes or who has software on cassette, disk or listing which we may be able to convert to our use. We will accept reverse charges for 'phone calls.
We wonder if you are considering the supply of the programs on disk, as we would prefer to buy disks.

(If anyone can help, please contact reader MONAGHAN direct). As far as disks are concerned, we have no plans to supply them at present. There is a number of good reasons. First, they are much more subject to damage in transit than cassettes. Second, there is the question of copyright. To supply a program on disk would involve also supplying some component of TRSDOS which, of course, is a Tandy copyright. Tandy has already warned some program publishers in the States that they face legal action if they continue to sell programs on TRSDOS disks. That's a hassle we don't want at present. Finally, there is the price. The price of the disk would swamp the price we charge for the program. If you really must have programs on disk, we would be prepared to record them onto your own minimum system disk which you would have first to post to us. For this we would make a charge of $5.00 (insted of the $3.50 for a cassette). The charge, quite frankly is to discourage you from using disks for this purpose, since at present, such a service would be difficult for us to provide. However, if this reply meets with a storm of protest from disk-drive owning readers, we will reconsider our policy. So it's over to our readers - Ed.)

From Dr. JOHN CASTLE, Brighton S.A.

In your subsequent issues, perhaps you may be able to help me and others, by publishing the following:

1. A machine language routine similar to that in G2 Level 3 which gives the BREAK Key a Level 2 type RESET function. This would be very useful when the computer locks up and the expansion interface is connected.
2. How to get into and out of ROM to use existing routines in ROM in one's own machine language program.

(You'll no doubt be interested in our Assembly Language Programming series, starting in this issue. That series will certainly cover point 2 very thoroughly over the next few months. As far as the BREAK key routine, we agree it would be useful and will look into it in the near future. If anyone has already written such a program, why not send it in and we will publish it and pay accordingly - Ed.)

From R.F. MOORHOUSE Oakleigh Vic.

It would be helpful if you could publish a list of forthcoming programs in your magazine. This would help readers decide whether a 12 month subscription is practical.

(I wish we knew! Each month we publish a MINIMUM list of programs for the following month and we give as much information as we can concerning forthcoming articles of general interest, constructional articles etc. However, many of the programs which we will be publishing will be coming from our readers and we havn't received them yet. Nor do we really know what readers want - until we hear from them over the next few months - Ed.)

-00000-

** STOP PRESS **

** NEXT MONTH'S PROGRAMS **

In addition to the programs described on Page 4, February's issue will also contain a BIORYTHM program and a simple DATA-BASE MANAGEMENT SYSTEM, both for L2/4K machines.

- 00000 -

** USERS CLUB NEWS **

The ADELAIDE TRS-80 USERS CLUB meets at the Gawler Place TANDY shop at 7.30 pm on the first Thursday of the month. The next meting is on Thursday 7/2/80.

-00000-

```
1 REM AUTHOR C.BARTLETT :1979: RICOCHET
5 CLS:R=0:A(1)=0:Q=2:Z=1:S=4:T=40:U=1:V=43:GOSUB3000
10 RESTORE
12 FORX=0TO127:SET(X,0):SET(X,47):NEXTX:FORY=0TO47:SET(127,Y):NEXTY
20 READA,B,C,D,E,F,G,H,I,J,K,L,M,N,O
25 DATA7,8,9,10,11,12,13,14,32,33,34,35,36,37,38
30 SET(S,T):SET(U,V):PRINT@798;"RICOCHET";
50 SET(D,I):SET(E,I)
55 SET(A,J):SET(B,J):SET(C,J):SET(D,J):SET(E,J):SET(F,J):SET(G,J):SET(H,J)
65 FORY=KTOL:SET(A,Y):SET(C,Y):SET(D,Y):SET(E,Y):SET(F,Y):SET(H,Y):NEXTY
75 FORY=MTOO:SET(C,Y):SET(F,Y):NEXTY:SET(B,O):SET(G,O)
86 IF(POINT(S,T)=0)*(POINT(U,V)=1)GOSUB1000
87 IF(POINT(S,T)=0)*(POINT(U,V)=0)GOSUB2000:SET(U,V)
88 PRINT@832;" ";:IFZ=1GOTO96
90 RESET(F,O):RESET(G,O):SET(G,N):Z=1:IFQ=1RESET(G,O-1)
92 GOTO97
96 RESET(B,O):RESET(C,O):SET(B,N):Z=2:IFQ=1RESET(B,O-1)
97 IFQ=1GOTO105
100 I=I-1:J=J-1:K=K-1:L=L-1:M=M-1:N=N-1:O=O-1
101 IFI=1Q=1
102 GOTO120
105 I=I+1:J=J+1:K=K+1:L=L+1:M=M+1:N=N+1:O=O+1
106 IFO=37Q=2
107 GOTO132
120 RESET(A,L+1):RESET(B,J+1):RESET(B,O+1):RESET(C,O+1):RESET(D,L+1)
130 RESET(E,L+1):RESET(F,O+1):RESET(G,O+1):RESET(G,J+1):RESET(H,L+1)
131 GOTO150
132 RESET(A,J-1):RESET(B,J-1):RESET(C,J-1):RESET(D,I-1):RESET(E,I-1):RESET(F,J-1
)
150 RESET(G,J-1):RESET(H,J-1):PRINT@832;" ";:GOTO50
1000 RESET(H,K):RESET(H,L):SET(H+1,K):SET(H+2,L):RESET(H+1,K):RESET(H+2,L)
1005 SET(H+1,J):SET(H+2,K):RESET(H+2,K):SET(H+1,J):SET(H+2,J):SET(H+3,J)
1050 P=H+4:W=J
1060 SET(P,W):RESET(P,W):P=P+1:SET(P,W)
1082 IF(POINT(P+1,W)=1)*(P<60)GOTO1200
1084 IF(POINT(P+1,W)=1)*(P>60)R=R+1:GOSUB8000
1090 IFP=125RESET(P,W):GOTO1110
1100 GOTO1060
1110 RESET(H+3,J):RESET(H+2,J):SET(H+2,K):RESET(H+2,K):SET(H+1,K):SET(H+2,L)
1115 RESET(H+1,J):RESET(H+1,K):RESET(H+2,L):SET(H,K):SET(H,L):SET(S,T):SET(U,V):
RETURN
1200 RESET(P,W):P=P-1:SET(P,W):IFPOINT(P-1,W)=1GOTO1250
1240 GOTO1200
1250 RESET(H+1,J):RESET(H+2,J):RESET(H+3,J):RESET(H+4,J):RESET(A,K):RESET(A,L)
1260 SET(H,I):SET(H+1,I-1):SET(A-1,K):SET(A-2,L):RESET(H,I):RESET(H+1,I-1)
1270 RESET(A-1,K):RESET(A-2,L):SET(H,I):SET(H,I-1):SET(A-1,J):SET(A-2,J)
1280 SET(A-3,J):RESET(A-1,J):RESET(A-2,J):RESET(A-3,J):SET(A,I):SET(A,I-1)
1290 SET(A,I-2):RESET(A,I-2)
1300 RESET(A,J):RESET(A,I):RESET(A,I-1):RESET(D,I):RESET(E,I):RESET(H,I):RESET(H
,I-1)
1320 SET(G,I-1):SET(G,I):SET(H+1,I):SET(H+2,I):SET(H+4,I):SET(H+4,I-1)
1330 RESET(C,L):RESET(D,L):RESET(E,L):SET(G,L):SET(H,L):SET(H+1,L)
1340 RESET(C,K):RESET(D,K):RESET(E,K):RESET(F,K):SET(G,K):SET(H,K):SET(H+1,K)
1345 SET(H+2,K):RESET(C,J):RESET(D,J):RESET(E,J):RESET(F,J):SET(H+1,J):SET(H+2,J
)
1350 SET(H+3,J):SET(H+4,J):RESET(C,M):RESET(C,N):SET(D,N):SET(E,M):RESET(F,M)
1360 RESET(F,N):SET(H,N):SET(H+1,M):RESET(A,J):RESET(B,J)
1410 SET(B,N):SET(B,O):SET(C,O):SET(D,O):SET(E,O):SET(F,O):SET(G,O):SET(H,O)
```

```
1420 SET(G,N):SET(H,N):SET(H+1,N):SET(H,M):SET(H+1,M):SET(H+2,M)
1430 SET(H+1,L):SET(H+2,L):SET(H+3,L):SET(H+4,L):SET(H,K):SET(H+2,K)
1440 SET(H+3,K):SET(H+5,K):SET(H,J):SET(H+6,J)
1500 RESET(G,I-1):RESET(G,I):RESET(H+1,I):RESET(H+2,I):RESET(H+4,I):RESET(H+4,I-
1)
1510 RESET(D,N):RESET(E,M):RESET(F,L):RESET(G,L):RESET(H,L):RESET(G,J):RESET(H+1
,K)
1520 RESET(H+4,K):RESET(H+1,J):RESET(H+2,J):RESET(H+3,J):RESET(G,K):RESET(H+4,J)

1540 RESET(H,J):RESET(H,K):RESET(H+6,J):RESET(H+5,K):RESET(H+2,K):RESET(H+3,K)
1550 RESET(H+1,L):RESET(H+2,L):RESET(H+3,L):RESET(H+4,L)
1560 RESET(H,M):RESET(H+1,M):RESET(H+2,M):SET(H,O):SET(H+1,O):SET(H+2,O)
1570 SET(H+3,O):SET(H+4,O):SET(H+5,O):SET(H+6,O):SET(H+7,O):SET(H+2,N)
1580 SET(H+3,N):SET(H+4,N):SET(H+5,N)
1600 FORX=1TO2000:NEXTX:FORX=BTOH+7:RESET(X,N):RESET(X,O):NEXTX:A(1)=A(1)+1
1650 IFA(1)=3THEN1670
1660 PRINT@862;R;" TARGETS HIT ";:PRINT@926;"AND";A(1);" MEN ARE DEAD";
1663 SET(S,T):SET(U,V):RETURN
1670 PRINT@862;"ALL YOUR MEN ARE DEAD      ";
1675 PRINT@926;"                                                    ";
1680 FORY=1TO3000:NEXTX:GOTO5
2000 IFO=1O=2:SET(S,T):SET(U,V):RETURN
2005 IFO=2O=1:SET(S,T):SET(U,V):RETURN
3000 FORX=60TO70:FORY=2TO32:SET(X,Y):NEXTY:NEXTX
3001 A=0:X=60
3005 B=RND(30):IF(B>30)+(B<3)GOTO3005
3006 IF(POINT(X+1,B)=0)+(POINT(X+1,B+1)=0)+(POINT(X+1,B-1)=0)GOTO3005
3010 FORX=60TO70:RESET(X,B):NEXTX:SET(110,B)
3015 A=A+1:IFA=7RETURN
3060 X=60:GOTO3005
8000 RESET(P,W):RESET(P+1,W):P=125
8020 PRINT@862;R;"TARGETS HIT ";:PRINT@926;"AND";A(1);" MEN ARE DEAD      ";
8040 IFR=7CLS:PRINT@798:"RICOCHET";:PRINT@862;"YOU WIN !  ":FORX=1TO3000:NEXTX
8050 IFR=7GOTO5
8060 RETURN
```

```
10  REGISTRATION
100     COPYRIGHT PETER G. HARTLEY, 57 MAIN AVENUE, FREWVILLE,
SOUTH AUSTRALIA, 5063:- SEPTEMBER 1979
130 RANDOM:S%=9999:CLS:GOTO190
140 K%=K%+1:IFK%>15PRINT@P,K%:ELSEPRINT@P-2,K%
150 D$="":D$=INKEY$:IFD$=""GOTO140
160 IFK%>W%THENW%=K%
170 IFK%<S%THENS%=K%
180 RETURN
190 GOSUB350:P=540:A1=1:FORQ=1TO10:K%=0:GOSUB210:GOSUB140:NEXT:GOSUB300:PRINT@P-
5,"                ";:FORT=0TO1000:NEXT:PRINT:PRINT"PLAY AGAIN?";:D$=""
200 D$=INKEY$:IFD$=""THENGOTO200:ELSEIFD$="Y"THENCLS:CLEAR:GOTO370:ELSECLS:PRINT
"BYE.":END
210 FORD%=0TORND(250):GOSUB270:PRINT@P-4,"                ":GOSUB220:PRINT@P,K$:NE
XT:RETURN
220 Z%=RND(6):ONZ%GOSUB230,240,250,260,250,250:RETURN
230 K$="0 ":RETURN
240 K$=" 0":RETURN
250 K$="00":RETURN
```

```
260 K$="  ":RETURN
270 D$="":D$=INKEY$:IFD$=""THENRETURN:ELSEGOSUB290
280 RETURN
290 C%=C%+1:FORF=0TO10:PRINT@P-2,"CHEAT";:FORT=0TO60:NEXT:PRINT@P-2,"     ":FORT
=0TO60:NEXT:NEXT:RETURN
300 CLS:IFC%>1PRINT@0,"YOU HAVE BEEN FOUND GUILTY OF CHEATING";C%;"TIMES.":ELSEI
FC%=1PRINT@0,"YOU HAVE BEEN FOUND GUILTY OF CHEATING ONE!":ELSEPRINT
310 PRINT@128,"YOUR BEST SCORE WAS";S%
320 PRINT@256,"YOUR WORST SCORE WAS";W%
330 R=(100-(C%*5))*(100-S%)/100:PRINT@384,"YOUR RATING... ";R;"%"
340 RETURN
350 CLS:PRINTCHR$(23);"F R U S T R A T I O N

A GAME OF SKILL FOR PEOPLE
WITH LITTLE ELSE TO DO..."
360 FORT=0TO2000:NEXT:CLS
370 PRINT@0,"
WATCH THE ELECTRONIC CLOCK IN THE CENTRE OF THE SCREEN.
WHEN IT STARTS TO COUNT, YOU ARE REQUIRED TO STOP IT -
AS QUICKLY AS POSSIBLE - BY PRESSING ANY KEY.
SOMETIMES IT TAKES A LONG TIME TO START, THOUGH,";
380 PRINT"
AND IT ALWAYS LOOKS AS THOUGH IT'S ABOUT TO START!!!!!"
390 FORT=0TO6000:NEXT:CLS:PRINTCHR$(23):RETURN
```

```
2 ' HANGMAN GAME. AUTHOR: BERNIE SIMSON, 18 BULLER TCE.
4 '      CHELTENHAM  STH AUST.      5014. PH 47 7528
6 ' NUMBER OF WORDS AVAILABLE DEPENDS ON MEMORY SIZE. TO CHANGE THE NUMBER OF WO
RDS, CHANGE VALUE OF LM (LIMIT) IN LINE 30.
8 ' WORDS ARE STORED IN DATA STATEMENTS. LINE 800 CONTAINS WORDSFOR NOVICES, LIN
E 900 FOR PROFESSIONALS. CLUES TO THE LATTER ARESTORED IN LINE 910. CHANGE THE D
ATA STATEMENTS AS REQUIRED.
9 '
10 CLS:PRINT@80,"H A N G M A N . . . . ":PRINT@192,"YOU HAVE A CHOICE OF PLAYING
 AT NOVICE STANDARD OR PROFESSIONAL STANDARD.":PRINT@384,"WHEN CHOOSING LETTERS,
 DON'T PRESS THE 'ENTER' KEY."
12 PRINT@448,"WIN = +5 POINTS, LOSE = -5 POINTS";:PRINT@590,"* *   NOTE FOR PROF
ESSIONALS   * *":PRINT@640,"YOU GET A CLUE WHEN THE ROPE APPEARS (IF THERE ARE M
ORE THAN 2  LETTERS LEFT) BUT YOU WILL LOSE 2 POINTS FOR THE CLUE";
20 PRINT@848,"A W A Y   Y O U   G O ....(DON'T GET HANGED !!)"
25 PRINT:INPUT"HIT ENTER TO START THE GAME";X
30 CLS:CLEAR100:LM=10
40 DIMCH$(LM),RA(LM)
50 INPUT"NOVICE OR PROFESSIONAL";NP$:IFLEFT$(NP$,1)="N"GOTO70
60 FORL=1TOLM:READCH$(L):NEXT
70 FORL=1TOLM:READCH$(L):NEXT:RANDOM
80 CLS:DP$=STRING$(26," "):GC=0:CC=0:CL=0:G=G+1
85 IFG>LMPRINT"CAN'T THINK OF ANY MORE WORDS !":END
90 R=RND(LM)
100 FORL=1TOG:IFR=RA(L)GOTO90
110 NEXT:CLS
130 RA(G)=R:FORL=1TOLEN(CH$(R))*2STEP2:PRINT@L+19,".";:NEXT:PRINT@1,"HIDDEN WORD
 -->"
140 PRINT@970,"CHOOSE A LETTER:";
150 A$="":A$=INKEY$:IFA$=""GOTO150
152 IFA$=CHR$(13)THEN150
```

```
155 PRINT@987,A$;
160 FORL=1TOLEN(DP$)
170 IFMID$(DP$,L,1)=A$PRINT@998,"ALREADY CHOSEN, DUMMY !";:FORM=1TO900:NEXT:PRIN
T@987,CHR$(30);:GOTO150
180 NEXTL:PRINT@970,CHR$(30);
190 FORL=1TOLEN(CH$(R))
200 IFMID$(CH$(R),L,1)=A$THEN280ELSENEXT ' NOT FOUND
205 DP$=DP$+A$
210 GC=GC+1:ONGCGOSUB420,440,500,520,560,600,620,670
220 IFGC<>8GOTO140 ' GALLOWS NOT COMPLETE
230 PRINT@490,"YOU' RE HUNG !";:FORL=1TO900:NEXT::PRINT@74,"THE WORD WAS:  ";CH$(
R)::W=-5
235 FORL=1TO500:NEXT:PRINT@970,"ANOTHER GAME? (Y/N)";
237 SC=SC+W:IFLEFT$(NP$,1)="P"AND(SGN(W)=1ANDCL=1)SC=SC-2
238 PRINT@174,"SCORE: ";SC;
240 A$="":A$=INKEY$:IFA$=""GOTO240
242 IFA$=CHR$(13)THEN240
245 PRINT@991,A$;
250 IFA$<>"Y"THENCLS:END
275 GOTO80
280 FORL=1TOLEN(CH$(R))
290 IFMID$(CH$(R),L,1)=A$PRINT@18+L*2,A$;:CC=CC+1
300 NEXT
310 IFCC=LEN(CH$(R))PRINT@484,CHR$(30);:PRINT@490,"YOU WIN !";:W=5:GOTO235
320 DP$=DP$+A$:GOTO140
390 '
400 '   HANGMAN STRUCTURE GRAPHICS ROUTINE - 8 STAGES
420 FORX=25TO80:SET(X,38):NEXT:RETURN
440 FORY=38TO32STEP-1:SET(25,Y):NEXT
460 FORY=38TO32STEP-1:SET(50,Y):NEXT
480 FORX=25TO50:SET(X,32):NEXT:RETURN
500 FORY=38TO8STEP-1:SET(65,Y):NEXT:RETURN
520 Y=39:FORX=58TO65:Y=Y-1:SET(X,Y):NEXT
540 Y=39:FORX=72TO65STEP-1:Y=Y-1:SET(X,Y):NEXT:RETURN
560 FORX=65TO27STEP-1:SET(X,8):NEXT
580 Y=16:FORX=65TO58STEP-1:Y=Y-1:SET(X,Y):NEXT:RETURN
600 FORY=8TO14STEP2:SET(37,Y):NEXT:IFLEFT$(NP$,1)="N"ORLEN(CH$(R))-CC<3RETURN
610 FORL=1TO10:READC$
613 IFL=RPRINT@832,"CLUE:";:PRINT@839,C$;
616 NEXT:RESTORE:FORL=1TO20:READC$:NEXT:CL=1:RETURN
620 FORX=35TO39:SET(X,14):SET(X,19):NEXT:SET(34,15):SET(40,15):SET(34,18):SET(40
,18):FORY=16TO17:SET(33,Y):SET(41,Y):NEXT:SET(36,16):SET(38,16)
630 FORY=20TO26:SET(37,Y):NEXT:X=37:FORY=26TO31:X=X-1:SET(X,Y):NEXT:X=37:FORY=26
TO31:X=X+1:SET(X,Y):NEXT:SET(30,31):SET(44,31)
640 FORX=35TO39:SET(X,22):NEXT:X=35:FORY=23TO25:X=X-1:SET(X,Y):NEXT:X=39:FORY=23
TO25:X=X+1:SET(X,Y):NEXT
650 PRINT@484,"LAST CHANCE !";:RETURN
670 PRINT@484,CHR$(30);
680 FORX=25TO50:RESET(X,32):NEXT:FORY=32TO37:RESET(25,Y):RESET(50,Y):NEXT:RETURN

800 DATACOMPUTER,RULER,TABLE,CAT,MAT,GIRL,BOY,KITTEN,POODLE,MIXER
900 DATADILAPIDATION,SKULDUGGERY,SCRUMPTIOUS,POLYPHONY,PERAMBULATOR,PERIPHERY,WA
NTON,VERNACULAR,EXACERBATE,APERTURE
910 DATA"RUN DOWN","SOME SNEEK","GOOD TO EAT","WHAT HARMONY","FOR BABY","OUTER L
IMITS...","UNBRIDLED","STRAIGHT TALK","BUG","PEEK-A-BOO !!"
```

```
2 'GAME OF LIFE - BASIC LISTING
5 CLS:INPUT"OPERATING SPEED  (0) FAST! (9) SLOW";Q:Q=Q*250
10 POKE16526,62:POKE16527,81
20 CLS:PRINT"INITIAL PATTERN BY (O)PERATOR OR (C)OMPUTER?":D$=INKEY$:D$=""
30 D$=INKEY$:IFD$=""THEN30ELSEIFD$="O"THEN50ELSEIFD$="C"THEN40ELSE30
40 FORX=46TO68STEP2:FORY=15TO24:Z=RND(03):IFZ<>3THENNEXT:NEXT:GOTO200:ELSESET(X,
Y):SET(X+1,Y):SET(127-X,Y):SET(126-X,Y):SET(X,47-Y):SET(X+1,47-Y):SET(127-X,47-Y
):SET(126-X,47-Y):NEXT:NEXT:GOTO200
50 CLS:PRINT"CONTROL THE CELLS WITH THE KEYS MARKED WITH ARROWS .
PRESSING <S> WILL SET THAT POSITION, AND MOVE THE 'SPOT' IN THE
PREVIOUS DIRECTION, BUT GOING THROUGH THAT POSITION AGAIN WILL
WIPE IT OUT.
<ENTER> WILL START PROGRAMME.":D$=INKEY$:D$=""
55 D$="":D$=INKEY$:IFD$=""THEN55ELSECLS
60 D$="":D$=INKEY$:RESET(X,Y):RESET(X+1,Y):IFD$=""THEN150ELSEIFD$="S"THEN190ELSE
Z=ASC(D$)
70 IFZ=91THENY=Y-1:IFY<0THENY=0:GOTO150
80 IFZ=10THENY=Y+1:IFY>47THENY=47:GOTO150
90 IFZ=9THENX=X+2:IFX>127THENX=127:GOTO150
100 IFZ=8THENX=X-2:IFX<0THENX=0:GOTO150
110 IFZ=13THENGOTO210
150 SET(X,Y):SET(X+1,Y):GOTO60
190 SET(X,Y):SET(X+1,Y):GOTO70
200 PRINT@0,"ANY KEY WILL TERMINATE RUN.                    ":FORT=0TO2000:NEX
T:PRINT@0,"";:PRINTCHR$(28);:PRINTCHR$(30);:
210 X=USR(0):D$="":D$=INKEY$:IFD$<>""THEN10ELSEFORK=0TOQ:NEXT:GOTO210
```

GAME OF LIFE M/C LANGUAGE SUB-ROUTINE - HEX LISTING

```
5000:  21 00 3C 11 9B 55 01 00 04 ED B0 C9 21 9B 55 22
5010:  9B 51 23 22 9D 51 23 22 9F 51 21 DB 55 22 A1 51
5020:  23 22 A3 51 23 22 A5 51 21 1B 56 22 A7 51 23 22
5030:  A9 51 23 22 AB 51 CD DF 50 21 DC 59 01 7E 03 3E
5040:  00 C9 CA 46 50 3C C9 CB 46 CD 42 50 C9 CB 56 CD
5050:  42 50 C9 CB 66 CD 42 50 C9 2A 9B 51 CD 53 50 2A
5060:  9D 51 CD 53 50 2A 9F 51 CD 53 50 C9 2A A7 51 CD
5070:  47 50 2A A9 51 CD 47 50 2A AB 51 CD 47 50 C9 2A
5080:  A1 51 CD 4D 50 C9 2A A5 51 CD 4D 50 C9 2A A3 51
5090:  C9 3E 00 CD 59 50 CD 7F 50 CD 47 50 CD 8D 50 CD
50A0:  4D 50 CD 86 50 CD 47 50 C9 3E 00 CD 7F 50 CD 47
50B0:  50 CD 53 50 CD 8D 50 CD 47 50 CD 53 50 CD 86 50
50C0:  CD 47 50 CD 53 50 C9 3E 00 CD 7F 50 CD 53 50 CD
50D0:  6C 50 CD 86 50 CD 53 50 CD 8D 50 CD 4D 50 C9 21
50E0:  9B 59 01 FF 03 11 9C 59 36 80 ED B0 C9 D9 CD 91
50F0:  50 3D 3D 20 10 CD 8D 50 CB 46 28 0C D9 CB C6 CB
5100:  CE D9 C3 08 51 3D 28 F4 CD A9 50 3D 3D 20 10 CD
5110:  8D 50 CB 56 28 0C D9 CB D6 CB DE D9 C3 22 51 3D
5120:  28 F4 CD C7 50 3D 3D 20 10 CD 8D 50 CB 66 28 0C
5130:  D9 CB E6 CB EE D9 C3 3C 51 3D 28 F4 D9 C9 CD 00
5140:  50 CD 0C 50 CD ED 50 D9 2A 9B 51 23 22 9B 51 2A
5150:  9D 51 23 22 9D 51 2A 9F 51 23 22 9F 51 2A A1 51
5160:  23 22 A1 51 2A A3 51 23 22 A3 51 2A A5 51 23 22
5170:  A5 51 2A A7 51 23 22 A7 51 2A A9 51 23 22 A9 51
5180:  2A AB 51 23 22 AB 51 D9 23 0B 78 B1 C2 44 51 21
5190:  9B 59 11 00 3C 01 00 04 ED B0 C9
```

```
                    STOCK RECORDING SYSTEM
1 D1$="2/6/79"
654 DATA 120,654,4,"D",69.95,3
655 DATA 120,655,6,"D",39.95,4
841 DATA 14,841,2,"15/3/79",59.95,2
978 DATA 40,978,20,"15/12/78",12.95,11
996 DATA 21,996,8,"13/2/79",49.95,6
1313 DATA 12,1313,30,"1/11/78",2.95,19
1346 DATA 12,1346,10,"D",19.95,3
1879 DATA 12,1879,3,"20/12/78",129.95,2
1984 DATA 31,1984,7,"10/3/79",69.95,5, 40,1984,5,"12/3/79",59.95,3
2081 DATA 31,2081,2,"12/3/79",399.95,1
2432 DATA 12,2432,33,"12/6/79",5.99,25
9441 DATA 21,9441,4,"20/1/79",149.95,3
9458 DATA 21,9458,2,"21/12/78",449.95,1
9999 DATA 9999
10000 CLS:E=9999:PRINT:PRINT:PRINT TAB(22)"I N V E N T O R Y"
10010 PRINT:PRINT TAB(18)"S T O C K    C O N T R O L"
10020 REM * PROGRAM BY KEITH NEIGHBOUR *
10030 FOR N=1  TO  1000: NEXT N
10040 CLS: PRINT"ENTER NUMBER ( ) FOR THE ACTION REQUIRED"
10050 PRINT:PRINT"( 1 ) TO COMPILE OR ADD TO THE INVENTORY"
10060 PRINT"( 2 ) TO ALTER OR UP-DATE INVENTORY ITEMS"
10070 PRINT"( 3 ) TO  REVIEW  A COMPLETE SERIES"
10080 PRINT"( 4 ) TO SEE A  PARTICULAR CATALOGUE ITEM"
10090 INPUT Z:IF Z<1 OR Z>4 THEN CLS:GOTO 10040
10100 CLS: ON Z GOTO 10110,10300,10400,10500
10110 F1=0:PRINT"TO COMPILE: KEY IN CATALOGUE NUMBER AS A NEW LINE NUMBER"
10120 PRINT"THEN KEY IN THE FOLLOWING (COMMAS AS SHOWN IN EXAMPLE) AND ENTER"
10130 PRINT TAB(0)"LINE#";TAB(7)"' DATA' ";TAB(14)"SERIES#";TAB(23)"CAT.#";TAB(30)
"QUANT.";TAB(37)"(AT)DATE";TAB(47)"PRICE";TAB(54)"QUANT.NOW"
10140 GOSUB 10250:PRINT "EXAMPLE :"
10150 GOSUB 10250:PRINT TAB(0)1313;:GOSUB 10250:PRINTTAB(8)"DATA";:GOSUB 10250:P
RINT TAB(15)12;",";:GOSUB10250:PRINT TAB(22)1313;",";:GOSUB 10250
10160 PRINT TAB(31)56;",";:GOSUB 10250:PRINT TAB(37)"23/2/79";" ,";:GOSUB 10250:
PRINT TAB(46)2.95;",";:GOSUB 10250:PRINT TAB(56)39
10170 PRINT"( NOTE : DATE MUST BE ENCLOSED IN QUOTES )":GOSUB 10250: IF F1=1 THE
N END
10172 PRINT @ 640,"IF A COMMON DATE APPLIES TO ALL ENTRIES"
10174 PRINT "TYPE (LINE) 1 D1$=' D/M/Y' & ' D' FOR DATE IN DATA LINES"
10176 PRINT "HIT ANY KEY TO PROCEED";
10178 IF INKEY$="" GOTO 10178
10179 GOSUB 11080
10180 PRINT @ 640,"WHERE MORE THAN ONE ITEM HAS THE SAME CATALOGUE NUMBER"
10190 PRINT"( ..... BUT A DIFFERENT SERIES NUMBER )"
10200 PRINT"PUT ALL ' SETS ' IN THE SAME DATA LINE"
10210 PRINT:PRINT"( HIT ANY KEY TO PROCEED )"
10220 IF INKEY$="" GOTO 10220
10230 CLS: F1=1: GOTO 10130
10250 FOR N=1 TO 100:NEXT N:RETURN
```

```
10300 PRINT"TO ALTER OR UP-DATE ITEMS: KEY 'EDIT' & CATALOG NO (LINE NO.)"
10310 PRINT"EDIT AS REQUIRED, THEN ENTER. REPEAT AS NECESSARY."
10320 PRINT:PRINT"( HIT ANY KEY TO PROCEED )"
10330 IF INKEY$="" GOTO 10330
10340 END
10400 CLS:PRINT"ENTER SERIES NUMBER";:INPUT S1:CLS
10410 K=1: T=0: GOSUB 11000
10420 FOR N=1 TO E
10430 READ S:IF S=E GOTO 11200
10440 READ C,Q1,D$:IF D$="D" THEN D$=D1$
10445 READ P,Q2
10450 IF S=S1 THEN K=K+1:GOSUB 11010:T=T+Q2*P
10460 NEXT N
10500 CLS:PRINT"ENTER SERIES NUMBER (COMMA) CATALOG NUMBER";:INPUT S1,C1:CLS
10510 K=0:N1=0:GOSUB 11000
10520 FOR N=1 TO E
10530 READ S:IF S=E GOTO 11250
10540 READ C,Q1,D$:IF D$="D" THEN D$=D1$
10545 READ P,Q2
10550 IF C=C1 AND S=S1 GOSUB 11010: N1=Q1-Q2: V=N1*P: GOTO 10570
10560 NEXT N
10570 PRINT:PRINT"NO OF ITEMS SOLD SINCE ";D$;" =";N1;" (VALUE = $";V;")"
10580 GOTO 11260
11000 PRINT TAB(0)"SERIES";TAB(10)"CAT. NO";TAB(20)"QUANT @";TAB(30)"DATE";TAB(4
0)"PRICE $";TAB(50)"QUANT.NOW":RETURN
11010 IF K=15 GOSUB 11050
11020 PRINT TAB(0)S;TAB(5)"-";TAB(10)C;TAB(20)Q1;TAB(29)D$;TAB(39)P;TAB(52)Q2:RE
TURN
11050 PRINT @ 960,"( HIT ANY KEY TO TURN THE PAGE )";
11060 IF INKEY$="" GOTO 11060
11070 CLS:GOSUB 11000: K=1: RETURN
11080 PRINT @ 640,CHR$(252):PRINT CHR$(252):PRINT CHR$(252):RETURN
11200 PRINT @960,"TOTAL VALUE OF SERIES";S1;"= $";T;" (HIT ENTER TO CONTINUE)";:
INPUT Z
11210 RESTORE: GOTO 10040
11250 PRINT @ 832,"THIS ITEM NO. IS NOT LISTED"
11260 PRINT @ 896,"FOR OTHER ITEMS HIT ENTER. OTHERWISE ENTER '1' ";
11270 INPUT Z:RESTORE:IF Z=1 GOTO 10040
11280 GOTO 10500
```

```
5 'MONITOR IN BASIC, VERSION 1.2
10 CLS:' THIS PROGRAM ENABLES YOU TO EXAMINE THE HEX OR
DECIMAL VALUE IN ANY MEMORY LOCATION AND CHANGE IT.
20 CLS:PRINT TAB(27);"COMMANDS"
30 PRINT"M ##### OR";TAB(12);"- DISPLAY CONTENTS OF MEMORY"
40 PRINT"M ####H";TAB(12);"  LOCATION #####(DEC) OR ####(HEX)"
50 PRINT"[ ###[";TAB(12);"- DISPLAY CONTENTS OF MEMORY LOCATION "
60 PRINT TAB(12);"  ### LESS THAN CURRENT LOCATION"
80 PRINT"+ ###+ ";TAB(12);"- DISPLAY CONTENTS OF MEMORY LOCATION"
90 PRINT TAB(12);"  ### MORE THAN CURRENT LOCATION"
95 PRINT "D";TAB(12);"- DUMP MEMORY BLOCK TO TAPE"
100 PRINT "L";TAB(12);"- LOAD MEMORY BLOCK FROM TAPE"
110 PRINT"SPACE BAR";TAB(12);"- DISPLAY CONTENTS OF NEXT MEMORY LOCATION"
120 PRINT TAB(12);"  LEAVES CURRENT LOCATION UNCHANGED.":PRINT
130 PRINT"HELP";TAB(12);"- LIST COMMAND TABLE":PRINT
```

```
140 INPUT"ENTER COMMAND REQUIRED";A1$
150 T$=LEFT$(A1$,1):GOSUB 1000
160 ON A GOTO 20,165,700,800,20,900,8000,6000
165 A1$=MID$(A1$,2)
170 ' TEST FOR HEX VALUES
175 H1=0:IF RIGHT$(A1$,1)<>"H" THEN 190
180 H1=1:A2$=LEFT$(A1$,5):GOSUB 3000
185 A$=STR$(D):GOTO 199
190 A$=A1$
199 GOTO200
200 ' DISPLAY AND MODIFY MEMORY IN DECIMAL
205 IF H1=1 THEN 400
210 PRINT VAL(A$),:GOSUB 4000
215 PRINT M1,
220 B$=INKEY$:IFB$="" THEN 220 ELSE PRINT B$;:T$=B$:GOSUB 1000
230 ON A GOTO 250,20,700,800,340,900,8000,6000
250 C$=INKEY$:IF C$="" THEN 250 ELSE PRINT C$;:T$=C$:GOSUB 1000
260 ON A GOTO 280,20,700,800,340,900,8000,6000
280 D$=INKEY$:IF D$="" THEN 280 ELSE PRINT D$;:T$=D$:GOSUB 1000
290 ON A GOTO 310,20,700,800,340,900,8000,6000
310 E$=B$+C$+D$
315 B$="":C$="":D$=""
320  IF (VAL(E$)<0)+(VAL(E$)>255) PRINT"  VALUE MUST BE FROM 0-255":GOTO 200
330 D=VAL(E$):GOSUB 5000
340 PRINT:A$=STR$(VAL(A$)+1):GOTO 200
400 ' DISPLAY AND MODIFY MEMORY IN HEX
410 GOSUB 4000
420 A2$=A$:GOSUB 2000
430 PRINT H$;" H",:A2$=STR$(M1):GOSUB 2000
440 PRINTH$,
450 B$=INKEY$:IF B$="" THEN 450 ELSE PRINT B$;:T$=B$:GOSUB 1000
460 ON A GOTO 480,20,700,800,525,900,8000,6000
480 C$=INKEY$:IF C$="" THEN 480 ELSE PRINT C$;:T$=C$:GOSUB 1000
490 ON A GOTO 510,20,700,800,525,900,8000,6000
510 A2$=B$+C$:GOSUB 3000
520 GOSUB 5000
525 PRINT
530 A$=STR$(VAL(A$)+1):GOTO 400
700 ' RESPOND TO [ COMMAND
710 E$=""
730 B$=INKEY$:IF B$="" THEN 730 ELSE IF B$="[" THEN 745  ELSE PRINT B$;
740 E$=E$+B$:GOTO 730
745 IF H1=1 THEN 760
750 PRINT:A$=STR$(VAL(A$)-VAL(E$)):E$="":GOTO 200
760 A2$=E$:GOSUB 3000
770 PRINT:A$=STR$(VAL(A$)-D):E$="":GOTO 400
800 ' RESPOND TO + COMMAND
810 E$=""
820 B$=INKEY$:IF B$="" THEN 820 ELSE IF B$="+" THEN 840 ELSE PRINT B$;
830 E$=E$+B$:GOTO 820
840 IF H1=1 THEN 860
850 PRINT:A$=STR$(VAL(A$)+VAL(E$)):E$="":GOTO 200
860 A2$=E$:GOSUB 3000
870 PRINT:A$=STR$(VAL(A$)+D):E$="":GOTO 400
899 END
900 GOTO 20
999 END
1000 '  TEST INPUT COMMAND
```

```
1010 A=1
1020 IF LEFT$(T$,1)="M" A=2
1030 IF LEFT$(T$,1)="[" A=3
1040 IF LEFT$(T$,1)="+" A=4
1050 IF LEFT$(T$,1)=" " A=5
1060 IF LEFT$(T$,1)="H" A=6
1070 IF LEFT$(T$,1)="L" A=7
1080 IF LEFT$(T$,1)="D" A=8
1090 RETURN
2000 'DECIMAL TO HEX CONVERSION
2010 H$="":D=VAL(A2$):H1$="0123456789ABCDEF"
2020 D=D/16:Z=(D-INT(D))*16
2030 H$=MID$(H1$,Z+1,1)+H$
2050 D=INT(D):IF D=0 RETURN
2060 GOTO 2020
3000 'HEX TO DECIMAL CONVERSION
3010 D=0:F=1:FOR H=1 TO LEN(A2$):H$=MID$(RIGHT$(A2$,H),1,1)
3020 IF H$<"A" THEN I=VAL(H$):GOTO 3040
3030 I=ASC(H$)-55
3040 D=I*F+D:F=F*16:NEXT
3050 RETURN
4000 'PEEK AT MEMORY LOCATION
4010 IF VAL(A$)>32767 THEN 4030
4020 M1=PEEK(VAL(A$)):RETURN
4030 M1=PEEK(-1*(VAL(A$)-32767)):RETURN
5000 'POKE NEW VALUE INTO MEMORY LOCATION
5010 IF VAL(A$)>32767 THEN 5030
5020 POKE VAL(A$),D:RETURN
5030 POKE -1*(VAL(A$)-32767),D:RETURN
6000 'DUMP MEMORY TO TAPE
6005 CLEAR 1000:CLS
6010 INPUT "TITLE";TITLE$
6020 INPUT "START ADDRESS";START
6030 INPUT "FINISH ADDRESS";FINISH
6040 INPUT "ENTRY POINT";ENTRY
6050 PRINT "RECORDING TITLE BLOCK":PRINT#-1,TITLE$,START,FINISH,ENTRY
6060 BYTE$=""
6065 FOR C=0 TO 5
6067 PRINT
6070 FOR I=10*C TO 10*C+9
6080 BYTE=-PEEK(START+I):PRINT TAB((I-10*C)*5)-BYTE;
6085 IF BYTE=0 THEN 6095
6090 X$=STR$(BYTE):GOTO 6100
6095 X$=",-0"
6100 BYTE$=BYTE$+X$
6110 IF START+I=FINISH THEN 6150
6120 NEXT I
6125 NEXT C
6130 PRINT:PRINT "RECORDING DATA":PRINT#-1,BYTE$
6140 START=START+I:GOTO 6060
6149 END
6150 PRINT:PRINT "RECORDING DATA":PRINT#-1,BYTE$:PRINT:PRINT "SAVE COMPLETED"
6200 RETURN
8000 'LOAD MEMORY FROM DATA TAPE
8005 CLS:D=0:PRINT "SEARCHING FOR TITLE BLOCK"
8010 INPUT#-1,TITLE$,START,FINISH,ENTRY
8020 PRINT "TITLE","START","FINISH","ENTRY"
8030 PRINT TITLE$,START,FINISH,ENTRY
```

```
8040 L=(FINISH-START)/60:IF L-INT(L)>0 THEN 8060
8050 LOOPS=L:GOTO 8070
8060 LOOPS=INT(L)+1
8070 FOR I=0 TO LOOPS-1
8080 B$=""
8090 PRINT:PRINT "LOADING BLOCK";I+1;"OF";LOOPS
8100 INPUT#-1,BYTE$
8110 FOR C=2 TO LEN(BYTE$)
8120 C$=MID$(BYTE$,C,1)
8130 IF C$="-" THEN 8145
8135 B$=B$+C$
8140 IF C=LEN(BYTE$) THEN 8145
8142 GOTO 8147
8145 D=D+5:PRINT TAB(D)  VAL(B$);:POKE START+C,VAL(B$):B$="":IF D<50 THEN 8147 E
LSE PRINT:D=0
8147 NEXT C
8150 NEXT I
8160 PRINT:PRINT "LOAD COMPLETED"
8200 RETURN
```

APPLICATION FORM

To: MICRO-80, PO BOX 213, GOODWOOD, SA 5034. Phone: (08) 381 8542

Please send me:

- CURRENT ISSUE ..... $ 2.50
- CURRENT ISSUE PLUS CASSETTE ..... $ 6.00
- 12 MONTH SUBSCRIPTION, MAGAZINE ONLY $24.00
- 12 MONTH SUBSCRIPTION, MAGAZINE PLUS CASSETTE $60.00
- START SUBSCRIPTION FROM .....(Dec.79 is issue 1)

(Make cheques payable to MICRO-80)

NAME: .....

ADDRESS: .....

.....POST CODE .....

MICRO-80

P.O. BOX 213, GOODWOOD, S.A. 5034 AUSTRALIA
TELEPHONE (08) 381 8542

APPLICATION FORM

FOR PUBLICATION OF A PROGRAM

To: MICRO-80

Please consider the enclosed program for publication:

IN MICRO-80 ☐

ON CASSETTE OR DISK ONLY ☐

Tick one only

NAME .....

ADDRESS.....

.....POST CODE .....

CHECK LIST

Make sure you have PRINTED the following information **clearly on your** cassette or disk:

YOUR NAME AND ADDRESS
PROGRAM NAME
MEMORY SIZE
LEVEL 1, LEVEL 2, BASIC 2.2, SYSTEM 1 or 2, EDTASM
HOW TO START PROGRAM (Eg. Enter/32ØØØ)

Make sure you include the following information with your **cassette or disk:**

EXPLANATION OF WHAT THE PROGRAM DOES AND HOW TO USE IT

DESCRIPTION OF HOW TO CHANGE IT FOR DIFFERENT MEMORY **SIZES** AND DIFFERENT LEVELS OF BASIC

THE SIZE OF MEMORY REQUIRED AND TYPE OF SYSTEM **NEEDED TO RUN IT** (Eg. LEVEL 1 4K, 32K WITH 2 DISK DRIVES ETC.)

START, END AND ENTRY POINT FOR MACHINE LANGUAGE PROGRAMS

ANY SIMPLE CHANGES WHICH CAN BE MADE TO MAKE IT MORE **FLEXIBLE** OR USEFUL

DON'T FORGET A STAMPED SELF-ADDRESSED ENVELOPE IF YOU **WANT YOUR** CASSETTE OR DISK RETURNED.