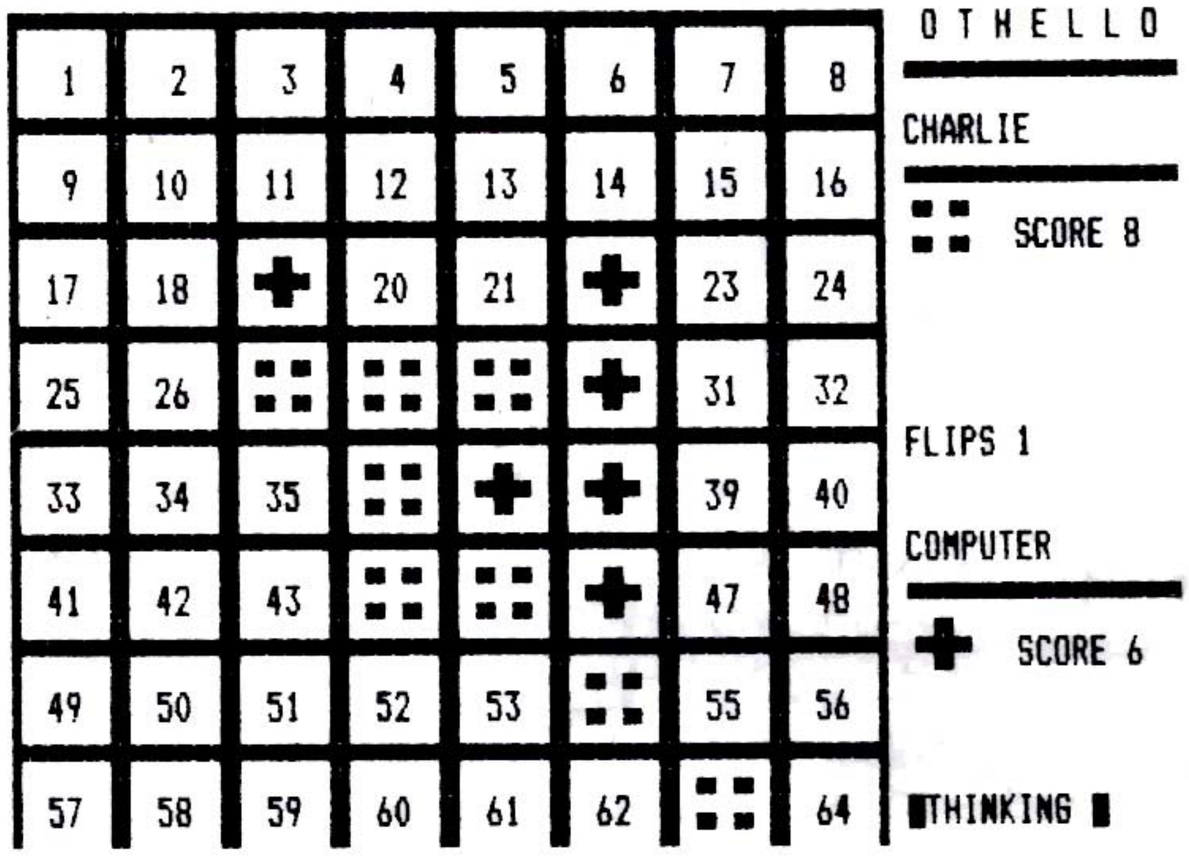
• TRS-80 • SYSTEM 80 • VIDEO GENIE
• PMC-80 • HITACHI PEACH
• TRS-80 COLOUR COMPUTER

Vol. 3, Issue 7, June 1982

OTHELLO



Also in this issue:

PROGRAMMING:

Using EDTASM+, SCRIPSIT and other machine language programs on the SYSTEM 80
Basic BASIC — Part 2
Saving and Loading long machine language programs on wafer

SOFTWARE:

- MICRO GRAND PRIX—Level II
- PASSWORD—Level II
- LOAN CALCULATION PACKAGE—Level II
- UNIT CONVERSIONS—Colour
- NORMAL DISTRIBUTION—Colour

MICRO-80

P.O. BOX 213, GOODWOOD, S.A. 5034. AUSTRALIA. TELEPHONE (08) 211 7244. PRICE: AUS. $2.50, N.Z. $4.00, U.K. £1.50
MICRO-80 is registered by Australia Post — Publication SQB 2207 Category B

***** ABOUT MICRO-80 *****

EDITOR:	IAN VAGG
ASSOCIATE EDITORS:	
SOFTWARE :	CHARLIE BARTLETT
HARDWARE :	EDWIN PAAY

MICRO-80 is an international magazine devoted entirely to the Tandy TRS-80 microcomputer and the Dick Smith System 80/Video Genie. It is available at the following prices:

	12 MONTH SUB.	SINGLE COPY
MAGAZINE ONLY	$ 26-00	$ 2-50
CASSETTE PLUS MAGAZINE	$ 65-00	$ 4-00 (cass. only)
DISK PLUS MAGAZINE	$ 125-00	$ 10-00 (disk only)

MICRO-80 is available in the United Kingdom from:
U.K. SUBSCRIPTION DEPT. 24 Woodhill Park, Pembury, Tunbridge Wells, KENT. TN2 4NW

Prices:		
MAGAZINE ONLY	£ 16-00	£ 1-50
CASSETTE PLUS MAGAZINE	£ 43-60	N|A
DISK PLUS MAGAZINE	£ 75-00	N|A

MICRO-80 is available in New Zealand from:
MICRO PROCESSOR SERVICES, 940A Columbo Street, CHRISTCHURCH 1 N.Z. Ph. 62894

Prices:		
MAGAZINE ONLY	NZ$ 43-00	NZ$ 4-00
CASSETTE PLUS MAGAZINE	NZ$ 89-00	NZ$ 5-00
DISK PLUS MAGAZINE	NZ$ 175-00	NZ$ 15-00

MICRO-80 is despatched from Australia by airmail to other countries at the following rates:

(12 MONTH SUB.)	MAGAZINE	CASS + MAG	DISK + MAG
PAPUA NEW GUINEA	Aus$ 40-00	Aus$ 83-00	Aus$ 143-00
HONG KONG/SINGAPORE	Aus$ 44-00	Aus$ 88-00	Aus$ 148-00
INDIA/JAPAN	Aus$ 49-00	Aus$ 95-00	Aus$ 155-00
USA/MIDDLE EAST/CANADA	Aus$ 55-00	Aus$ 102-00	Aus$ 162-00

Special bulk purchase rates are also available to computer shops etc. 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, System 80 or Video Genie and their peripherals. MICRO-80 is in no way connected with either the Tandy or Dick Smith organisations.

** WE WILL PAY YOU TO PUBLISH 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 or System 80 to earn some extra income is included in every issue.

** CONTENT **
Each month we publish at least one applications program in Level I BASIC, one in Level II BASIC and one in DISK BASIC (or disk compatible Level II). We also publish Utility programs in Level II BASIC and Machine Language. At least every second issue has an article on hardware modifications or a constructional article for a useful peripheral. In addition, we run articles on programming techniques both in Assembly Language and BASIC and we print letters to the Editor and new product reviews.

** LIABILITY **
The programs and other articles in MICRO-80 are published in good faith and we do our utmost to ensure that they function as described. However, no liability can be accepted for the failure of any program or other article to function satisfactorily or for any consequential damages arising from their use for any purpose whatsoever.

***** CONTENTS *****

MICRO-80 is registered by Australia Post - Publication SQB 2207 Category B

AUSTRALIAN OFFICE AND EDITOR: MICRO-80 P.O. BOX 213, GOODWOOD, SOUTH AUSTRALIA, 5034. TEL. (08) 211 7244

U.K. SUBSCRIPTION DEPT: 24 WOODHILL PARK, PEMBURY TUNBRIDGE WELLS, KENT TN2 4NW

Printed by: Shovel & Bull Printers, 379 South Road, MILE END 5031

Published in Australia by: MICRO-80, 433 Morphett Street, Adelaide.

***** EDITORIAL *****

Rumours abound that there is about to be a flood of TRS-80 Colour Computer look-alikes unleashed on the world market. The story goes that Tandy did not design the Colour Computer, rather that Motorola (the manufacturer of the 6809 micro-processor used in it) did and that Tandy's licencing agreement is about to expire. The two most common names mentioned as potential suppliers of competing machines are Motorola itself and Hitachi - manufacturer of the Peach and also a major second source manufacturer of the 6809. Naturally, we have attempted to access the truth of these rumours. It seems that Tandy has had exclusive use of the Synchronous Address Multiplexer (SAM) chip used in the Colour Computer and developed by Motorola. It also seems that this agreement is about to expire and that others will soon be able to use the SAM chip. There is no doubt that much of the versatility and power of the Colour Computer stems from this chip so that its general release should certainly lead to machines of similar performance if not using exactly the same software. We have been unable to ascertain whether Motorola itself has any intentions to release a Colour Computer whilst there is presently available in Japan a Hitachi Junior computer but we have been unable to sight a specification of this machine. It is said to be able to use Colour Computer programs but is unlikely to be seen in Australia before April 1983 if at all. Past experience suggests that the inherent egoism of computer designers will lead any potential designer of a Colour Computer look-alike to make some "improvements" thus reducing compatibility and the usefulness of the new computer. We will follow the progress of this rumour with interest.

One of the areas of most concern to our friends and readers has been the steady slippage in publication dates of MICRO-80 magazine until at present we are over four months behind our cover date. Clearly, the job of editing MICRO-80 is too large to be carried out by yours truly in conjunction with my other considerable and expanding responsibilities. Therefore, I decided some time ago that the solution was to appoint a full time Editor. The decision was much easier than its implementation since it is very difficult to find a person with the right qualifications, experience and skills to carry out this function. I am happy to announce that Richard Wiwatowski will be joining us as MICRO-80 Editor starting early in 1983. Richard has worked as a High School Science teacher for eight years, he has personally owned a System 80 computer for two years and is an experienced programmer in machine language, BASIC and several other languages. For some months now Richard has acted as Colour Computer Software Editor and is well qualified to fill his new role. I am sure you will all join with me in wishing Richard success in his new task and particularly, in catching up with the publication dates of the magazine.

Recently, I was called upon to demonstrate the Hitachi Peach to a prospective customer whose requirement was to use a computer for mathematical problem solving involving a considerable amount of "number-crunching". Inevitably, the question came up "how fast is it?" When invited to try it out the customer entered the following Short BASIC routine and timed its execution.

```
10 FOR A = 1 TO 1000
20 PRINT SQR(A);
30 NEXT A
```

The Hitachi took 60 seconds to complete the exercise. We looked at each other bemusedly, both realising that we had no basis for comparison using this improvised benchmark test. On the next table was an Olivetti M20 which uses a Z8001 16 bit micro-processor. Without more ado, I entered the same program into that machine whereupon it took 30 seconds to execute. Well, we now had something to go on; the 8 bit micro-processor in the Hitachi Peach was half as fast as the 16 bit micro-processor in the Olivetti M20. Secretly, I was disappointed at the speed of the M20 since it had such features as a 4 MHz clock speed and hardware multiplication and division on its micro-processor. My curiosity was now well and truly aroused so, that evening, I ran the same test on the TRS-80 Colour Computer; which took 75 seconds to execute the routine. I then executed the following Poke :- POKE 65495,Ø which doubles the refresh rate to 1.8 MHz and increases the computer's operating speed. Now, the exercise took 55 seconds. Casting all caution to the winds, I now executed POKE 65497 which sets all operations to high speed. This POKE causes the screen to lose synchronization so I included the speed up POKEs and then resetting POKEs (POKE 65494,Ø and POKE 65496,Ø respectively) in the BASIC program so that I could regain control of the machine. This time the Colour Computer took only 38 seconds to execute the program. (Colour Computer owners might like to experiment with these POKEs themselves but do not attempt I/O operations in the high speed mode). Next morning, now thoroughly bitten by the bug, I managed to prise Eddy Paay away from his beloved TRS-80 Model II for long enough to type in the routine and time it at 38 seconds. By now, I was very disappointed with the Z8001 but then I had an idea. What if I carried out the calculations without printing them to the screen, since the screen handling time might be significant. Generally, this sped-up the 8 bit computers by about 10 or 15 seconds. So, now the 8 bit machines were faster than the 16 bit machine! Back to the M20. This time, without screen handling, it executed the routine in 4.5 seconds. Yes, 4.5 seconds. 16 bit processing was now more than vindicated and honour was done. Eddie Paay, who by now was following my antics with some amusement looked at me in his quiet way and said well of course, high quality machines like the Model II and the M20 only update the screen during the fly-back period so streaks do not appear on it such as are seen on a standard TRS-80 Model I. The 16 bit micro-processor was spending most of its time idling waiting for the screen refresh and that is what slowed it down. I am not sure what the moral of this story is apart from the

amount of time I could have saved had I asked Eddie in the first place! But it probably should make us all aware that programming techniques are often more important than the inherent processing speed of the micro-computer itself. This is nowhere more apparent than with the 8088/8086 16 bit micro-computers which are acquiring a reputation for being very slow. Whilst these micro-processors are inherently slower than the Z8000 and the Motorola 68000 micro-processors they are certainly faster than 8 bit micro-processors but much of the software available for them has been cross-translated from 8 bit source code and does not take advantage of the much more powerful instruction set available on the 16 bit processors. Elegant programming techniques learnt on 8 bit machines will stand us all in good stead when we are fortunate enough to be able to use the 16 bit and larger machines in the future.

- 0000000000 -

***** PEEKing (UK) - by Tony Edwards *****

A few issues ago, I mentioned a new BASIC 'dialect' which was compatible with different machines. This project has now come to fruition with the "HOBBYSCOOP" Basicode. This is a BASIC language structured so that it can be saved on cassette and reloaded into many different types of computer. This is accomplished by the use of a standard signal format and a machine code translation program which re-constructs the tape signal into the correct format for the host machine. A similar arrangement is used for saving to cassette. In this way a program can be developed on say a PET, saved on cassette, loaded into a TRS-80, and RUN. This is a big step forward in universal software and is a very interesting development. Radio Netherlands International broadcasts programs in this format internationally for down loading via radio receivers. Further details will be found in the full article on the code elsewhere in this issue.

Copyright is in the news in the U.K. again. Atari has started a campaign against programs which allegedly infringe the copyright of their game 'Pac-Man'. This is a very popular game over here and there are versions of it under different names, for many machines. Atari's objection is not to pirated copies, but to the copying of the idea. This is a new line. I understand that Atari has approached a number of software companies, including Bug-Byte, A&B Software, and Micropower. They have gone as far as to issue an injunction against Commodore with respect to the game 'Jellymonsters'. If this move is successful in the courts, it will no longer be possible to copy game types, and companies will have to develop more original games and not be able to copy the ideas of successful arcade games.

- 0000000000 -

***** INPUT/OUTPUT *****

From: M. Bauk - Kalamunda, W.A.

I have been flipping through my TRS-80 Basic manual and have come upon a section titled "Important Addresses". Decimal 16416 has CURSOR POSTION N (LSB) and decimal 16417 has CURSOR POSITION N (MSB). By PEEKing at these locations I have been trying to understand how the numbers in the locations correspond to the cursor position but cannot understand the results. Could you please explain what the numbers mean and how, from these numbers, the cursor position can be known.

About 6 months ago I obtained an "Other Venture" by Jyym Pearson - this one is entitled "Escape from Traam". Since then I have become very frustrated indeed. On the back of the pack it reads, "Average Playing Time: 1 month". I haven't even got past the first location in 6 months!

Please, please can anyone tell me how to get past the first location! At least an easy hint. I suspect that the program could be faulty.

In order to interpret the cursor position found from addresses 16416 and 16417 it is necessary to understand the layout of the TRS-80 screen memory. The screen is "memory-mapped" starting at address 15360 and occupying 1024 addresses up to and including address 16383. If you use the program statement PRINT @, 0.... then the cursor will be positioned at address 15360. PRINT @ 1, gives a cursor postion of 15361 and so on. PRINT @ 63 moves the cursor to address 15423 at the extreme right of the first line on the screen. PRINT @ 64 moves the cursor to address 15424 at the extreme left of the second line on the screen.

The table below explains this:

SCREEN ADDRESSES

	LHS	RHS
Line 1	15360	15423
	15424	15487
	15488	15551
	15552	15615
	15616	15679
	15680	15743
	15744	15807
	15808	15871
	15872	15935
	15936	15999
	16000	16063
	16064	16127
	16128	16191
	16192	16255
	16256	16319
Line 16	16320	16383

To use the cursor position pointers to find the equivalent "PRINT @" value, execute the following line.

```
CURSOR = PEEK (16416) + PEEK (16417)*256-15360
```

The variable CURSOR is the PRINT @ value you require. For example, if 16416 contains the value 244 and 16417 the value 61, then the cursor would be address 15860 which is the PRINT @ 500 position on Line 8 of the screen.

I am afraid we have no experience of "Escape to Traam". Perhaps another reader can help.

From: J.D. Smith - Hawthorn, S.A.

Mr. P.R. Smith of Donvale Victoria, certainly happened to do things the hard way! I have made excellent backups of Tandy's "Microchess" in one simple way - I used the "Copier" program from the Software Library that was provided free when I renewed my subscription.

"Copier" is a great little routine for copying machine language programs and can do something very few such programs can - it can copy itself! Hence I now have "Copier" on a separate tape, plus backups, as well as backed up Microchess.

I don't say "Copier" is infallable, but it would go close!

"Copier" is different from many other utility programs which allow you to copy machine language programs in that it keeps control of the computer at all times and does not necessarily load the target program into its normal working space. It simply loads it into a buffer, byte for byte as it appears on the original tape and then punches an identical tape from the contents of its buffer. COPIER's main limitation however, is that it cannot copy programs much longer than 12K in a 16K machine. TRCOPY is another such program although rather more sophisticated. By now, most of our readers should have their Software Library containing "Copier" and would be able to use it as suggested by Reader J.D. Smith - Ed.

- 0000000000 -

***** USING EDTASM +, SCRIPSIT AND OTHER MACHINE LANGUAGE PROGRAMS ON THE SYSTEM 80 *****

We have had several letters recently from readers having difficulty in loading SYSTEM tapes created by EDTSAM + and text tapes created by SCRIPSIT, back into their SYSTEM 80/VIDEO GENIE computers. John Ross from the Adelaide Micro Users Group has been investigating this problem for some time and has discovered what appears to be a timing problem in the tape routines in

these two programs. Although the reason for the problem is not entirely clear, it could be related to the slight difference in clock frequency between the SYSTEM 80 and the TRS-80. Not all SYSTEM 80's experience the problem but if you are having trouble, the patches shown below have worked successfully for others and are worth trying. Also listed are patches which modify the print routines to drive the SYSTEM 80 printer via port FD instead of address 37E8 hex which is used on the TRS-80. Note that the appropriate cassette port on the SYSTEM 80 must be initialised and that none of the TRS-80 programs do this for you. The program will therefore SAVE to and LOAD from the last cassette port used before the program was run. If this is not the port you want to use then you must change it yourself. The most convenient way is to fit a changeover switch, otherwise, you must do so from BASIC before loading the TRS-80 program. A simple CSAVE to the port required even without a BASIC program in memory is all that is required. Incidentally, loading difficulties seem to increase as you increase the volume setting on your cassette deck. Now, on to the program patches. To make the patches, you will need a monitor program which loads into an area of memory that does not clash with the target program. For SCRIPSIT, EDTASAM + and TRCOPY, BMON or ZMONH from the Software Library would be ideal. for MON3 you should use MON3 itself or ZMONL from the Software Library. First load the target program using SYSTEM but instead of typing /(NEWLINE) in response to the second prompt, simply type (NEWLINE). Then use the SYSTEM command a second time to load your monitor. This time, answer the second prompt with /NEWLINE. After that, use the Edit Memory command to make the necessary changes at the addresses shown then punch out a SYSTEM tape using the START, END and ENTRY addresses shown for your version of this program.

TAPE VERSION EDTASM VER 1.1

START	END	ENTRY
4646	6310	4BEA

ORIGINAL TAPE WRITE ROUTINE			CORRECTED ROUTINE		
62EC	E5	PUSH HL	62EC	E5	PUSH HL
62ED	C5	PUSH BC	62ED	C5	PUSH BC
62EE	3A2844	LD A,(4428H)	62EE	212744	LD HL,4427H
62F1	C604	ADD A,04H	62F1	4E	LD C,(HL)
62F3	47	LD B,A	62F2	23	INC HL
62F4	212744	LD HL,4427H	62F3	7E	LD A,(HL)
62F7	4E	LD C,(HL)	62F4	C604	ADD A,04H
62F8	3E3C	LD A,3CH	62F6	47	LD B,A
62FA	1801	JR $+03H	62F7	3E3C	LD A,3CH
62FC	7E	LD A,(HL)	62F9	CD9846	CALL 4698H
62FD	23	INC HL	62FC	7E	LD A,(HL)
62FE	CD9846	CALL 4698H	62FD	23	INC HL
6301	10 F9	DJNZ $-05H	62FE	10F9	DJNZ $-05H
6303	79	LD A,C	6300	79	LD A,C
6304	CD9846	CALL 4698H	6301	CD9846	CALL 4698H
6307	AF	XOR A	6304	00	NOP
6308	323F45	LD (453FH),A	6305	00	NOP
630B	322844	LD (4428H),A	6306	00	NOP
630E	C1	POP BC	6307	AF	XOR A
630F	E1	POP HL	6308	323F45	LD(453FH),A
6310	C9	RET	630B	322844	LD(4428H),A
			630E	C1	POP BC
			630F	E1	POP HL
			6310	C9	RET

VER 1.08 ORIGINAL ROUTINE			CORRECTED ROUTINE		
62C5	E5	PUSH HL	62C5	E5	PUSH HL
62C6	C5	PUSH BC	62C6	C5	PUSH BC
62C7	3A0141	LD A,(4101H)	62C7	210041	LD HL,4100H
62AA	C604	ADD A,04H	62CA	4E	LD C,(HL)
62CC	47	LD B,A	62CB	23	INC HL
62CD	210041	LD HL,4100H	62CC	7E	LD A,(HL)
62D0	4E	LD C,(HL)	62CD	C604	ADD A,04H
62D1	3E3C	LD A,3C	62CF	47	LD B,A
62D3	1801	JR $+03H	62D0	3E3C	LD A,3CH
62D5	7E	LD A,(HL)	62D2	CD6402	CALL 0264H
62D6	23	INC HL	62D5	7E	LD A,(HL)
62D7	CD9943	CALL 4399H	62D6	23	INC HL
62DA	10F9	DJNZ $-05H	62D7	10F9	DJNZ $-05H
62DC	59	LD A,C	62D9	79	LD A,C
62DD	CD9943	CALL 4399H	62DA	CD6402	CALL 0264H
62E0	AF	XOR A	62DD	AF	XOR A
62E1	32DD42	LD (42DDH),A	62DE	32DD42	LD (42DDH),A
62E4	C1	POP BC	62E1	C1	POP BC
62E5	E1	POP HL	62E2	E1	POP HL
62E6	C9	RET	62E3	00	NOP
			62E4	00	NOP
			62E5	00	NOP
			62E6	C9	RET

EDTASM+ VER 1.06 ROUTINE STARTS AT 627BH
START=4380H END=7263H ENTRY=4380H

EDTASM+ VER 1.07 ROUTINE STARTS AT 6295H
START=4380H END=7280H ENTRY=4380H

EDTASM+ VER 1.08 ROUTINE STARTS AT 62C5H
START=4380H END=72BBH ENTRY=4380H

TRS M/L PROGRAMS GENERALLY USE INSTRUCTIONS 32 E8 37 - LOAD ADDRESS 37E8 WITH CONTENTS OF 'A' REGISTER - (PRINTER OUT).
3A E8 37 - LOAD 'A' REGISTER WITH CONTENTS OF 37E8 (PRINTER IN).
SYS 80 USES D3 FD - OUT PRINTER PORT FD: DB FD - IN PRINTER PORT FD.

SYSTEM-80 LINE PRINTER ROUTINES

ADDRESS	CONTENTS	CHANGE TO	DISK VERSION
6BAD	3A E8 37	00 DB FD	ELECTRIC PENCIL 1979
6BBF	3A E8 37	00 DB FD	(USE CASSETTE CHANGE OVER
6BC9	32 E8 37	00 D3 FD	S/W FOR TAPE FILES.)

SCRIPSIT VER 1.0

5254	3E A0	3E 0D	- FOR IBM SELECTRIC
5244	32 E8 37	00 D3 FD	
5F63	3A E8 37	00 DB FD	
663F	3A E8 37	00 DB FD	
6650	3A E8 37	00 DB FD	
665E	32 E8 37	00 D3 FD	
6722	32 E8 37	00 D3 FD	
7A79	32 E8 37	00 D3 FD	
7A9E	32 E8 37	00 D3 FD	

EDTASM VER 1.2

6ED8	3E 0A	3E 0D	- FOR IBM SELECTRIC
6EDA	32 E8 37	00 D3 FD	
6EE8	32 E8 37	00 D3 FD	
6EFE	3A E8 37	00 DB FD	

TR COPY

48CA	3A E8 37	00 DB FD
48D4	32 E8 37	00 D3 FD

SCRIPSIT TAPE VERSION Ver 1.0

4342	3E 0A	3E 0D			
4344	32 E8 37	00 D3 FD	START	END	ENTRY
4F18	3A E8 37	00 DB FD	4300	6A46	4300
5617	3A E8 37	00 DB FD	PRINTER ROUTINES		
5628	3A E8 37	00 DB FD			
5636	32 E8 37	00 D3 FD			
56EC	32 E8 37	00 D3 FD			
69B8	32 E8 37	00 D3 FD			
69C6	32 E8 37	00 D3 FD			
6A27	32 E8 37	00 D3 FD			
6A31	3A E8 37	00 DB FD			

MON-3

ADR.	CONTENTS	CHANGE TO
70DE	3A E8 37	DB FD 00

7826 C9 7B 21 E8 37 56 CB 7A 20 FB 77 C9 --CONTENTS
C9 DB FD CB 7F 20 FA 7B D3 FD 00 C9 --CHANGE TO

CASSETTE ROUTINES

DISK ADR.	TAPE ADR.	CONTENTS	CHANGE TO	
				SCRIPSIT Ver 1.0
638C	5341	06 42	06 08	CASSETTE ROUTINES
639C	5351	06 80	06 41	TAPE & DISK VER.
63A3	5358	06 22	06 76	

CASSETTE VERSION - SCRIPSIT VER 3.1
SYSTEM 80 LINE PRINTER ROUTINES

ADRESS	CONTENTS	CHANGE TO	SCRIPSIT Ver 3.1		
4306	3E 0A	3E 0D			
4308	32 E8 37	00 D3 FD	START	END	ENTRY
4EDA	3A E8 37	00 DB FD	42E9	6D88	4303
4EF3	32 E8 37	00 D3 FD			
4EF9	32 E8 37	00 D3 FD			

55EB	3A E8 37	00 DB FD
55FC	3A E8 37	00 DB FD
560A	32 E8 37	00 D3 FD
56C0	32 E8 37	00 D3 FD
CASSETTE ROUTINE		
5315	06 42	06 08
5325	06 80	06 41
5326	06 22	06 76

- 0000000000 -

***** BASIC BASIC - II - by Ken B. Smith *****

When the Editor wrote, "an occasional series of articles", as a header for my submission in April 82's issue, he was gifted with an amazing degree of foresight. The main reason for the delay has been that I rather thought that my submission had made it to file 13. However, when I received my April issue in September, I was delighted to find my article published intact.

It occurs to me that with the delay between articles, those of you who read my first are now several months on with your computing skills, and would be rather insulted by an offering on PRINT @ or whatever and would prefer something more challenging. But before we get into programming techniques, I would like to air a topic that cropped up the other week at our club, the MUSCAT COMPUTER GROUP (MCG).

BASIC v PASCAL : INTERPRETER v COMPILER.

Without getting involved in the endless and somewhat pointless debate about program structure, (there are horses for courses and both languages have their good and bad points), there is considerable confusion as to why BASIC is so much slower than PASCAL. The answer lies in the run time activity and is nothing to do with structure, GOSUB's, LINE NUMBERS, or anything so trivial.

A statement: BASIC ALWAYS NEEDS AN INTERPRETER. PASCAL IS ALWAYS COMPILED.

Wake up in the back row! I know this looks boring but there is a point to this - promise.

The simplest analogy for INTERPRETER & COMPILER I know goes something like this.

Imagine that you have to converse with a person who speaks only an obscure version of one of the 68 recognised languages of India. You have no knowledge of his language and he none of yours. Apart from a few obscene gestures that are universal, your 'conversation' will be rather limited. You will need a go-between who knows both of your languages. The INTERPRETER is just like it sounds. You say a few words at a time and they get INTERPRETED into another language. The process is relatively slow, but you can converse in an orderly fashion. THE COMPILER is like writing everything you wish to say in a letter and having the go-between INTERPRET the whole thing at once and give the new version to the other party as a finished package. This can be read very quickly.

The COMPILER is ideal for statements and fixed details, but the INTERPRETER is an interactive process, and much more suited to the conversations we need to have with our micros during program development. OK, where is all this leading? The point is that, although PASCAL is fast, it can be the very devil to develop a program on because every contentious step has to be compiled before you know if a particular sequence will work as planned. Someday a working PASCAL INTERPRETER will be available and interactive development will be possible in that language before the final compilation. When that occurs there will be a dramatic shift towards PASCAL for the Micro. But the average micro user needs that INTERPRETER and BASIC is the only language with a half way decent command set and a good INTERPRETER.

Back to BASIC. On your machine it is run by the INTERPRETER and unless you have a BASIC COMPILER you are stuck with the limitations and advantages of your system. Let's take a look now at some ways to help things along inside the '80.

At last...You can put the car keys back on the side, you won't need to go out now. We're back on something interesting...We had a very hot summer this year...Ruined a roll of film by leaving the camera in the car boot...Sand doesn't mix with water or disk drives...

Good, I'm happy to have your attention back. In the squadron we have a mail rack. (Don't bang your forehead like that. As an expression of desperation it is pointless and it will give you a headache, which you will have anyway if you read on). It's huge. The guy who built it was a little short on English and he thought that sixteen was sixty. Still we may get more pilots one day!! The point is that our letter rack is just like the variable storage table in the TRS-80. As variables are used or defined, the TRS-80 allocates a slot in the table for its name and value. Like our mail rack, there is no real order, no alphabetical sorting; the only logic is first come, first served. Once a program is RUN the variables are allocated to the table in used or DEFined order, and subsequent changes or references to that table are done TOP DOWN.

What all this means is: as your program progresses through its code, the variables are allocated to the table as the INTERPRETER comes to them, and when the program needs reference to a variable's value, it scans the table from the top until it finds the one it's searching for and acts accordingly. It doesn't take much thought to realise that if a variable used late in the program is used as a variable within a nested set of FOR - NEXT's or in a high speed graphics routine, then things will slow up. Every time that variable is needed the program has a little sleep while the INTERPRETER finds the reference buried somewhere in the bottom end of the table. Far from ideal, but there is an easy solution.

Write your program as normal and once it is debugged and running, think about speed. Which variables are being used inside the loops? And graphics sections? Are they on top of the table? Don't worry about it. Put them there. Let's assume that you need the variables AV,F1,H,HC & X on top of the table. The quickest and most efficient way is to DEFine them and then they will be there from the start. Now, don't get into a tizz about the arrays. They have a totally separate table, although the same logic applies. What I'm saying is that you can use the same line and statement as the array DEFining process to allocate your important variable to the top of the table. Thus:

```
100 DEF A(100,5),B(40),AV,F1,H,HC,X
```

This has cost you very little in memory but will save you a lot in speed.

I've been away into BASIC for a moment to write a little utility which will help you to understand what we have been discussing, and also to help you speed up your programs. What follows is a small BASIC add-on to type onto the end of your program once it has been written. RUN your program in the normal way and when it is finished or whenever, type BREAK : GOTO 50000. (It must be a GOTO or the implied CLEAR within a RUN will zero out the VLT (Variable List Tables).

VARIABLE LISTER - by Ken B. Smith

```
50000 Z1=PEEK(16633)+256*PEEK(16634)  : '** START OF NON ARRAY VLT IN Z1
50001 Z2=PEEK(16635)+256*PEEK(16636)  : '** START OF ARRAY VLT IN Z2
                                           THIS IS ALSO THE END OF VLT 1
50002 Z3=PEEK(Z1)+3                   : '** FIRST ITEM IS THE VARIABLE LENGTH
                                           INT=2,STRING=3,SINGLE=4,DOUBLE=8
                                           THEN ADD 3 TO JUMP THE VAR NAME
50003 PRINT CHR$(PEEK(Z1+2));         : '** Z1+2 IS FIRST CHARACTER OF VAR NAME
50004 PRINT CHR$(PEEK(Z1+1)),         : '** Z1+1 IS SECOND CHARACTER.
50005 Z1=Z1+Z3                        : '** INCREMENT Z1 POINTER TO NEXT VAR
50006 IF Z1>Z2 THEN END ELSE GOTO50002: '** ARE WE AT THE END YET?
```

A slightly quicker to type version:

```
50000 Z1=PEEK(16633)+256*PEEK(16634):Z2=PEEK(16635)+256*PEEK(16636)
50001
Z3=PEEK(Z1)+3:?CHR$(PEEK(Z1+2));CHR$(PEEK(Z1+1)),:Z1=Z1+Z3:IFZ1<Z2THEN50001
```

That will give you a listing of your variables in the order they appear in the VLT. My variables Z1,Z2 & Z3 should be at the bottom of the list!! Now I have made no attempt to add the variable declaration types to the listing - the comments show where this information is stored. If you want that little extra - add it yourself. The exercise will do you good.

A final word about this section of the INTERPRETER. As we saw in the listing above, the top of the Array VLT is the end of the non-subscripted variable table. This is a dynamic division. That is, if more variables are added to the top table (VLT1), then the whole of VLT2 is moved. Not a tragic occurrence as the '80 handles it very quickly, but it is worth considering.

Well, that was a good workout on variables and so on. Take a breather - you dun good. Now let's look at the variable type that gives the most problems and can be the most useful. Strings...

STRINGS & THING$

As we saw in the VARIABLE LISTER, the string variables take up 3 bytes in the VLT. Why is this? Scratch head, read manual, shout at parrot. No avail. The Manual says that strings can be up to 255 bytes long. No, I really have got a parrot. His name is Charlie and he pretends to be a human with wings, but actually he is a Stunted African Grey Menace with a bolt cropper for a beak. The reason he comes into this act is because he has eaten the edge with the page numbers off my Level II manual, so all references to that noble work of fantasy and omission will have to be from guesswork. (Evidently the edge with page numbers does not cause such heartburn as the rest - Charlie showed no signs of distress, but he hasn't attempted to chew off any more. Is there a moral here?)

Anyhow, somewhere in your manual it states that string$ can be up to 255 bytes long. Now the question is: You can't get 255 bytes into the 3 bytes in the VLT, so where are they? Do I hear, 'In the String Storage Area'? Of course, but what, then, lives in the VLT? The address of

the string in the string storage area. Go to the top of the RAM and sit next to the Lower Case Driver.

It seems obvious really but all the INTERPRETER needs from the VLT to find a STRING is where is it, and how many bytes. It would be tedious indeed to find this manually from the raw VLT, so DARTMOUTH COLLEGE, when defining BASIC, installed a command to give us the address of any variable within the VLT. So someone said, 'Let there be a method of finding variables , and the word was VARPTR and jolly useful it is, too.

Before we go any further, it will be prudent to explain that the TRS-80 and all other Z80 based micros store their numbers back to front. Lots of reasons are bandied around for this, but I am convinced that it was a deliberate attempt on the part of Zilog to confuse me. (The same comment applies to certain parts of Scripsit & Visicalc. I am, however, assured that this mild paranoia is curable by selling one's micro and taking up fishing....I never catch anything anyhow, so on with the important stuff).

Numbers are stored backwards, particularly the INTEGERS which are, apart from their programming uses, 2 byte numbers that hold addresses in RAM. They are stored in LSB & MSB format. That is, Least Significant Byte first and Most Significant Byte last. If you don't grasp the LSB, MSB idea, try imagining you owe me $47.63. (Hard isn't it?) Which would be the Most Significant portion of the repayment, $47.00 or .63c? Anyone who doesn't get that now, please lend me some money, say $10.99, but make the cheque out for $99.10.

When you VARPTR a numeric variable, PEEKing the next 2,4 or 8 Bytes will give you the number, sign and exponent. You will, however, find this rather a chore as the '80 can handle the storage of its numbers very well, and the quickest way to change one is to type A=241314 or whatever. It is the manipulation of string$ where the direct modification in memory is of the most value, and it is this technique that we will now examine.

Unless you have one of these dreadfully clever utilities that enables you to pack a string direct from the keyboard, you are stuck with the limitations of the '80 ROM. To this end you cannot enter any character above 127 (bit 7 set) from the board. This is simply because the codes 128 to 250 within a program are compressed tokens for BASIC keywords. (Sounds awfully grand, but it saves a lot of memory) So if you want graphic characters within a string you must resort to concatenation (that's adding them together, Brian) from a data array. This then builds your graphic string in the String Storage Area. This technique is very easy and you get the advantages of very fast graphics onto the screen. But it costs memory - once for the DATA line, again for the space in the String Storage Area, and further memory and speed for the construction. Although we are a long way from when we poor 4K Level I neophytes hung around outside TANDY shops with paper bags over our heads chanting 'OM,OM,OM', there is still a shortage of memory for the more complex applications. Let's therefore look at a simple way to cope with the dreaded STRING PACKING. (Dim lights, play Dick Tracy music!!!)

```
5  '*** EXAMPLE ONE
10 DATA 70,82,69,68
20 FOR X = 1 TO 4
30 READ A
40 A$=A$+CHR$(A)
50 NEXTX
60 '*** EXAMPLE TWO
70 B$="FRED"
```

The first example is concatenation from DATA to put the word FRED into A$. The second does the same rather quicker. However, the first could have been used for graphics characters, not the second. If you type these in as shown and PRINTA$,B$ you will see that both are the same. Now let's find each string in turn and play a few tricks and see what we can learn. Add to the program as follows - the comments for explanation only.

```
100 A=VARPTR(A$) : '** ADDRESS OF VLT ENTRY FOR A$ IN A
110 AL=PEEK(A) : PRINT AL : '** FIRST ENTRY IS LEN(A$)
120 AA=PEEK(A+1)+256*PEEK(A+2) : '** START ADDRESS OF A$ IN MEMORY.
                                    LSB/MSB FORMAT
                                    MEMORY ADDRESS=LSB+256*MSB
130 PRINTAA
140 FORX=AATOAA+AL : '** X FROM START TO END OF A$
150 ? PEEK(X),:NEXT : '** SHOW WHAT'S THERE
```

RUN that and you will get a 4, an address and the numbers 70,82,69,68 and it shouldn't take you long to realise that the numbers are the ASCII for FRED (see DATA in example one). Now make a note of the memory address. This will be the top of the String Storage Area as A$ is the first string in there. Now edit line 100 to read - 100 A=VARPTR(B$), and RUN again. The display will be very similar. But, and this is a big BUT, the memory address is lower than the previous example which we said was the top of the String Storage Area. In this short program the difference is not that great but the clue is there.....Correct, the address is actually that of the "F" in line 70. Now let's prove it. From the keyboard type : POKE AA,65.

Before we advance any further, I must issue a warning about indiscriminate POKEing and this is nothing to do with what Mother told you. Before trying to POKE into memory, make a copy of the work. If you get it wrong, funny (or not so) things happen and you could be faced with an awful lot of extra coding or another headache!! While you are working through these examples and following me exactly, fine, but on your own it could be different. I know. How do you think I came by this information. You have been warned....

Having POKEd the 65 into AA, LIST the program, paying particular attention to line 70. It should now read:

70 B$="ARED"

The 65 is the ASCII code for "A" and you have just changed a program statement without using the interpreter. Those who are still doubtful, PRINTB$. Satisfied? Now we can box clever and really upset the interpreter. Type : POKE AA,191 : LIST and study line 70 again. Confused? Try a PRINT LEN(B$) also and you will find that the interpreter is as well. Now PRINT B$ and see if you can work it out.

Of course, after seeing line 70, you went straight to your manual and opened up to the page before DERIVED FUNCTIONS titled 'E/Internal Codes for BASIC Keywords' and looked up the keyword for 191. On finding it as USING, the appearance of line 70 as : 70 B$="USINGRED" no longer held any mystery and the result of PRINT B$ as a character 191 followed by RED was obvious.

But just in case...As I mentioned earlier, the Interpreter assumes that all characters over 127 are Basic keywords if they appear inside a program. The big difference between A$ and B$ is their location. A$ was constructed and is located in the String Storage Area, B$ is part of the program and uses no string space for its storage, just the program line. This explains the difference in memory locations. (The String Storage Area is below the BASIC). On POKEing a 191 directly into the program area to replace the "F", the PRINT command accepts that and LISTs the graphic 191 as the BASIC Keyword, USING and still gives the LENgth as 4, which of course it is.

Try a few other graphic characters in place of the "F" and when you are happy with that, POKE AA+1,AA+2 & AA+3 with some other graphic codes to form a small graphic shape. Look at the listing - weird, not really now you understand. The one thing you must not do now is to let the Interpreter have control over that line again with the EDIT command. Try EDIT 70 and just exit with the ENTER key. RUN the program again, the LISTing looks the same, but try a PRINT B$ and a PRINT LEN(B$). Yes, it's all over - mine has gone USINGMERGERSETTAB with a LENgth of 17. Well, it's back to the drawing board or at least to rewriting line 70.

The main limitation of this problem with the EDIT command is that you must construct a dummy string of exactly the correct length before attempting to POKE your graphics into it. YOU CANNOT EDIT IT LATER. Remember, also, that you can use cursor control characters as well to make a shape of any size (within the 255 limit). So I'll leave you with those thoughts for now and hopefully, when you type in my AutoGraphicsPacker (AGP) from the next issue, you will understand not only the concepts behind it, but also the work that it saves...

- 0000000000 -

***** BASIC INTERCHANGE BETWEEN '80s AND OTHER MICROS - by Tony Edwards *****

A BASIC Esperanto

The exchange of BASIC programs between different brands of micro-computer is very difficult. Not only do different machines use different dialects of BASIC, but the various cassette formats make the exchange of recorded programs between different brands of computer virtually impossible. Until now, that is. A new BASIC code format has been developed called NOS-BASICODE. By means of specially written translaters it is now possible to exchange programs recorded on cassette tape directly between different brands of micros. In the case of the '80 group of machines, a minor hardware modification is necessary.

The core of the system is a standard formatted form of BASIC, a BASIC Esperanto, which can be read by the different brands of machine. This core BASIC is then translated into the machine's own BASIC format by a short machine program. The reverse is also possible as a BASIC program written for, and developed on the '80 can be saved onto cassette in NOS-BASICODE format for exchange with non-'80 machines.

Non-commercial Exploitation

Already this standard format is in use by some 1200 computer hobbyists in Holland where it was developed, and is being made available via non-commercial channels. Experimental broadcasts, receivable world wide, have been made by both NOS Radio and Radio Nederland Wereldomroep. The broadcasts have shown that it is technically feasible to broadcast and down load NOS-BASICODE programs for use on a wide range of micro-computers, including the '80 range. Later this year further test transmissions are scheduled using the standard format at 300 baud in association with Radio Netherlands' English Language communications magazine 'Media Network' on short wave international broadcasts. Transmissions at 1200 baud have been successfully down loaded weekly

using simple domestic medium wave radios. Amateur radio enthusiasts are also involved in long range link ups on the VHF band. Programs written in NOS-BASICODE are also available on cassette for testing purposes.

Range of Applications

Micro computers use normal audio-cassette recorders to store their programs, and an interface is provided which, together with a ROM routine, converts the stored program data into audio tones for recording. On play-back, the tones are translated back into program code. Unfortunately, practically all makes of micro-computers use different tones and coding systems to store data on the cassette tape. Consequently, it is not usually possible to load a cassette stored program into a brand of micro other than the one which recorded it. The Video Genie/System 80/TRS-80 group are an exception to this rule because they are almost identical machines. Most brands (including the '80) can be modified to read and write the standardised NOS-BASICODE and thus, it only remains to sort out the small differences in BASIC dialects which occur between the different machines.

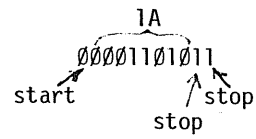
Translation programs, and where necessary, hardware modifications have been developed for the following computers:-

P2000 (Philips)
COSMICOS 1802
ACORN ATOM
TANDY TRS-80
GENIE/SYSTEM 80/PMC-80
DAI
OSI-IP with SUPERBOARD
APPLE
EXIDY SORCERER
PET/CBM
SWTPC-6800

The most conspicuous omission from this range is the ZX group of micros. Attempts have been made to include ZX-80 and ZX-81 but have not yet proved successful.

NOS-BASICODE Specifications

Two tones are used to record data onto tape, with frequencies of 1200 Hz and 2400 Hz respectively. A "Ø" is defined as one full cycle of 1200 Hz and a "1" is defined as two full cycles of 2400 Hz. Data transfer is at a baud rate of 1200. A packet of information is made up of one start bit (Ø), eight data bits (least significant bit first) and two stop bits (1). Thus HEX1A would be encoded as:-



The BASIC program is coded in the form it would appear if the resident program was LISTed. Tokens are not used for reserved words as different types of machines use different token values. All letters and ciphers are presented in ASCII form and each ASCII code in the program receives a closing bit = 1. Each BASIC instruction is followed by a space and lines are closed with "CR" (HEX 8D).

The tone format on the tape is:-

Leader: 5 seconds of stop bit (2400 Hz)
ASCII "Start Text" (HEX 82)
BASIC INFORMATION in ASCII
ASCII "End of Text" (HEX 83)
Checksum

Trailer: 5 seconds of stop bit (2400 Hz)

The check sum at the end of the tape is the result of the bit indication exclusive ORed from all previous bytes.

NOS-BASICODE Protocol

In order to ensure that a program imported from a different brand of machine, when loaded will also RUN, a protocol has been developed to avoid major incompatibilities in the BASIC program code. It is recommended that programs written for translation into NOS-BASICODE do not use the commands PEEK, POKE, DEF or USR. Variables may have up to two significant characters but the first must always be an alphabetic character. Special variables ending with $ % and ! may also be used with the same meaning as in Microsoft BASIC Level 2. It is also recommended that

variables having the same first two letters as a reserved word should not be used, but this is difficult due to the large number of reserved words in various systems. A standard is also suggested with respect to line numbers. This is shown in tables (i) and (ii). Data should be formatted to the DIF standard.

The Hardware Modification

The '80 range is one of those which requires a minor hardware modification to enable it to read and write programs in the standard format. There are two ways of doing this. The first is an extension circuit, which can be built easily, and the second is a simple internal addition.

The Software

Program 1 is the code for the translating programs for use with '80 machines and can be input using the usual techniques. This program is not the property of MICRO-80, but is reproduced here with permission of the writers for the convenience of readers. The code is not copyright, but the name "NOS-BASICODE" must always be used when referring to it, and it must not be used commercially for gain.

User's Instructions

The code is usable on systems with 4K or more and no reserved memory is required. Disk users should first enter BASIC 2 when DOS READY appears. They can first store the loaded NOS-BASICODE program using CSAVE onto a cassette and then transfer this to disk BASIC.

Load the program with the sequence SYSTEM (enter) , *?(BCODE) enter (If the internal modification is in use type in the following as direct statements:-

```
POKE 17396,255
POKE 17398,16
POKE 17404,155
POKE 17406,16
```

After loading a set of instructions will appear and about 1000 bytes of memory will have been used. The following commands are available:-

```
LOAD .... for loading a NOS-BASICODE format program
SAVE .... for writing a NOS-BASICODE format program
```

All the usual commands of BASIC are still available in addition to the two new ones. The input/output is via #1 cassette for TRS-80 users and the #2 cassette for Genie/System 80/PMC-80 users.

The Reading Routine

Unless you want the new program to be merged with an existing program, first type in NEW as the LOAD command does not have an implicit NEW as does CLOAD. Set up the cassette and start the loading procedure with LOAD. The tape will start and any faulty program lines will appear on the screen. Pressing <enter> will add the line, fault included to the memory and <backspace> will delete it. <space> has the same effect as <enter> <enter> so two lines can be added to the memory. To abort the loading press <break> . If the program fills the memory the words "GEHEVGEN RUIMTE OPGEBRUITKT" (Dutch for "Memory fully loaded") will appear, but the loaded portion is still usable. If a bad load occurs, the defective character will be highlighted by two underline (cursor) symbols. This can then be corrected later using normal EDITing features. A reading error produces the warning "CHECKSUM ERROR".

If no errors are encountered no statements appear on the screen and the cassette recorder will switch itself off at the end of the trailer.

The automatic merge arrangement can be put to good use if an attempt is being made to load a defective or poor quality tape. Repeated LOADs can be made merging good lines and avoiding bad ones with backspace .

Running Programs.

Once a program has been completely LOADed it can be run as a standard Microsoft BASIC program. In some cases a SYNTAX ERROR will appear if the program has been transported from a machine using a different dialect of BASIC. Any such errors must be corrected by use of the EDIT facilities.

Writing Routine

To store in NOS-BASICODE format no special interface is required. Once you have in memory the Microsoft BASIC program bug free and in the suggested format, simply set up the cassette recorder and type SAVE. The tape will start automatically, save the program in the standard format, and stop when the task is completed. The saving routine can only be aborted with the "re-set button".

Experimentation

It must be stressed that this system is new and still in the development stage, so some problems may occur. You are invited to experiment and report your results to me for transmission to the Production Team. If you intend to make a serious investigation of this interesting development, you are recommended to buy the instruction booklet "basicode hobbyscoop" which gives additional information in English and Dutch. It is available, together with a cassette containing the translation codes (for many different computers) and some sample NOS-BASICODE programs, from:-

NOS-BASICODE,
Administratie Algemeen Secretariaat,
NOS,
P.O. Box 10,
1200 JB Hilversum
The Netherlands.

The booklet and cassette are offered to the public at cost price (not supplied - we will publish the price as soon as it is known - Ed.) plus postage. Payment should be to "Postgiro Account No. 1419 in Hilversum" or by international money order payable to "NOS Algemeen Secretariaat". Payment should be made in Dutch guilders at the rate of ff25 in Europe and ff35 in Australasia for airmail delivery.

(The hardware circuitry and program listing supplied were not suitable for reproduction. We will publish these in a later issue - Ed.)

- 0000000000 -

***** SAVING AND LOADING LONG MACHINE LANGUAGE PROGRAMS ON WAFER - by N.J. Coleman *****

Ever wanted to save those 16K ADVENTURE games onto ESF wafer? Have you tried to load your favourite machine language program plus a monitor into just 16K and not been able to fit them both in? If you have, then read on!

This article will tell you how to save those lo-o-ng programs that just squeeze into your memory- and leave no room for your ESF monitor. It'll also describe a few pitfalls to watch out for and give you a couple of utilities to help you avoid those traps.

You may think that without a resident monitor it's pretty hard to find the start, end and entry points that you need to save a program on wafer. But it's really quite simple. Follow these simple steps.

1. Load in your monitor and relocate it to high memory. Then store a constant (zero's probably easiest) from 42ØØH up to about 6ØØØH. This provides a "background" against which the start point can be clearly seen. Now go back to BASIC. Type SYSTEM and begin to load your program from cassette. After the asterisks have flashed a few times, press the Stop key on the cassette player and then press the Reset button on the back of your keyboard. This prevents any glitches being recorded in the tape. Re-enter your monitor with SYSTEM /XXXXX and dump memory from about 42ØØH onwards. You should be able to see the start point against the background quite clearly. A common point is 43ØØH (17152 in decimal). Make a note of where it is.

OK, now you have the start point, but you still need to find the end and entry points.

2. Reload your monitor or, if you can, relocate it back down to low memory. Store a constant from about 6ØØØH to 7FFBH (so as to leave room for the four ESF floating bytes). Now go back to SYSTEM (an easy way to do this is to GOTO Ø2B5H, rather than BASIC then SYSTEM) and load your program. Let it load in all the way. When the *? appears, press BREAK.

3. At this stage you have a program in memory, but no monitor, so, load in your monitor to low memory. If the message "READING..." appears, but the motor didn't switch on, it means that the last four bytes in memory have been written over by your program. To fix this, re-initialise the ESF by typing SYSTEM /Ø Enter and then SYSTEM /12345. Now load your monitor. Examine locations 4ØDFH and 4ØEØH for the Least Significant Byte and Most Significant Byte respectively of the entry point. Alternatively, if you aren't too clued up on hex-decimal conversion, then, in the Command Mode, i.e. after pressing Break but before loading the monitor, type ?PEEK(166Ø7)+256 *PEEK(166Ø8) Enter. The entry point is displayed in decimal. This means that you only have to find the end point and you are in business!

PEEKING THE ENTRY POINT

4. Dump memory from about 7ØØØH onwards. Now BEWARE! At some stage, probably around 7FC7H, you will be seeing the stack. Don't be fooled into thinking that this is part of your program. If you see rows of ØØs, then a couple of rows of figures, then more rows of ØØs, your program ended ages before. Go back and check. If you are seeing rows of figures up past 7FC7H and maybe even up to 7FFBH (after which you should see FØ C3 97 19, the four ESF bytes) then you will need to use the utilities described in the next section. Nevertheless, write down the end point.

5. Now some is born lucky, and some ain't. If you was, then all your problems will have ended long before the stack, i.e. before approx. 7FC7H, and you can now save them on wafer using the parameters that you have just found. Don't forget to add 1 to the length, e.g. Start - End = (Length - 1).

If you ain't lucky and you're trying to save Adventure or Starfighter or something which seems to take all your memory, then you need to use the two utilities which follow but, first, a description of why you need to--for the benefit of the machine language programmers amongst you.

The whole problem is the stack. The SYSTEM command sets the Stack Pointer to low memory (4288H to be precise) and most m/l programs written for the TRS-80 assume that the SP is low. The trouble is that both the ESF and BASIC set it back to high memory, where BASIC expects it to be. So when you go to save your program, what with the ESF calling this and calling that, and returning this and that, the stack writes over your program and corrupts it! You get a VERIFY error because, naturally enough, the second time around the stack is different to what was recorded on wafer, and even if you do try to @LOAD and RUN it, the whole thing crashes because it no longer makes any logical sense.

And ditto if you somehow obtained a workable copy and tried to @LOAD and execute it. Once it started to use the stack it would write over itself in high memory. In fact, the ESF probably writes over it before auto-execution ever begins. The result, crash.

The answer to the problem is to load these two simple utilities. Their main purpose is merely to set the SP to low memory and then @SAVE or @LOAD. One utility is for @SAVE and one for @LOAD. But, I hear you ask - and this is where the story really starts - if my program is crammed into memory and I haven't even one byte spare, where am I going to put these utilities? Well, did you know that you have about eighty extra bytes available to you TOTALLY FREE! They are from 4041H to 408EH which the clock and other things in the expansion interface use, but you don't have an expansion interface, do you, because you've got an ESF you clever little munchkin. So that is where we will put the utilities. By the way, they lack a few frills (such as any sort of explanatory message at all) because they have to fit into the available space. But don't worry, as you will see, you enter the parameters in the same order as you would do for the normal ESF.

TO SAVE

Load the Saver utility. The autostart for this is 02B5H, the SYSTEM entry-point, so now just type the name of the program and press ENTER. Once the program has loaded successfully, type /16450. You should see a ? Type, in the following order and followed by ENTER on each occasion, the file #, the start, the length, and the autostart parameters, all in decimal. After you press ENTER for the fourth time, the ESF will save and verify the program on drive #0. (You can change this - see the program explanation below). If you now want to run the program, type /XXXXX (entry address), otherwise press BREAK to return to BASIC.

TO LOAD

Load the Loader utility. The autostart address is 02B5H. Type /16500 in response. You need to enter the utility via SYSTEM so that the SP is set to low memory. Now type the file number and press ENTER. The program will load on drive #0 (once again you can change this) and begin execution.

RESTRICTIONS

i) to save memory, the utilities are designed to allow file numbers 1 - 9 only. This should not be a problem because you can fit only two or three 16K programs on a 50' wafer.

ii) if you should ever crash the system, i.e. you get the Mem Size question, and you reply by ENTER, then part of Saver is overwritten and will need to be reloaded. Loader remains intact.

HOW THE UTILITIES WORK

SAVER

The Stack Pointer is set to low memory through the SYSTEM command. The sub-routine is called four times to get the parameters. Each time the sub-routine is called, the parameter is pushed onto the stack. Note the method used to set the return address up correctly. An alternative would have been to delete PUSH HL, RET and just have JP(HL). I chose this way for clarity for the new assembler programmers (i.e. me) amongst us. Next, the parameters are popped into the correct registers for the WRITE sub-routine. This sub-routine also requires that the Port Number (i.e. F0H + drive #) be placed one byte after the Top of Memory indicator. (If you want the utility to use a different drive than #0, then alter this byte to reflect the change, e.g. F1H for drive #1, etc.) Topmem is set two bytes below the resident program. Also, the SP is manipulated so that the LSB is popped into the A register, not the F register.

Next, the ERROR sub-routine is called to report any errors such as PARITY, EOT DETECTED and so on. Finally the utility jumps to SYSTEM so that you can load another tape, or execute the resident one, or whatever.

LOADER

This one is simplicity itself. The SP is set to low memory by the SYSTEM command then the file number is obtained and converted to hex by subtracting 3ØH. This was easier and uses less memory than setting up a sub-routine to cater for double-digit numbers. The utility then jumps to the load routine.

CONCLUSION

I have found these utilities invaluable. The main reason that I bought an ESF was to reduce loading times. I soon discovered that it was not possible to save and load the longer programs- the very ones that I wanted on wafer. Now that I have these utilities my problem is solved and I can put all my programs on wafer.

Here are some parameters that I have discovered already.

Program	Start	Length	Entry
ADVENT #1	17104	15664	17104
GOBBLE	17408	13312	17408 *
ASYLUM	17296	15463	17326

* N.B. GOBBLE may be saved and loaded normally if these three bytes are added to the beginning of the program: 31H 88H 42H.

```
                    00100 ;
                    00110 ;                 SAVER
                    00120 ;                 =====
                    00130 ;
                    00140 ;                         (C) 1982 N. J. COLEMAN
                    00150 ;                                  6/13 HOWITT ST
                    00160 ;                                  SOUTH YARRA VIC 3141
                    00170 ;
                    00180 ;         THIS PROGRAM GIVES THE ABILITY TO THE ESF USER TO
                    00190 ; SAVE ON WAFER THE LONG PROGRAMS THAT OCCUPY ALL OF
                    00200 ; MEMORY.  THIS ABILITY IS NORMALLY DENIED TO ESF USERS
                    00210 ; BECAUSE:-       I)  THE STACK IS POSITIONED AT HIGH MEM-
                    00220 ;                     ORY.
                    00230 ;                 II)  THE ESF USES FOUR BYTES OF HIGH MEM-
                    00240 ;                      ORY.
                    00250 ;
                    00260 ;         MAIN ROUTINE
                    00270 ;         ------------
                    00280 ;
4042                00290           ORG     4042H   ;= /16450
                    00300 ;
1E5A                00310 CONVRT    EQU     1E5AH
302A                00320 ERROR     EQU     302AH
1BB3                00330 INPUT     EQU     1BB3H
3155                00340 LOAD      EQU     3155H
02B5                00350 SYSTEM    EQU     02B5H
40B1                00360 TOPMEM    EQU     40B1H
300C                00370 WRITE     EQU     300CH
                    00380 ;
4042 0604           00390 START     LD      B,4     ;4 PARAMETERS TO INPUT
4044 CD6140         00400 LOOP      CALL    GETNUM  ;GET THEM
4047 10FB           00410           DJNZ    LOOP
4049 D1             00420           POP     DE      ;AUTOSTART
404A C1             00430           POP     BC      ;LENGTH
404B E1             00440           POP     HL      ;ADDRESS
404C 2B             00450           DEC     HL
404D 2B             00460           DEC     HL
404E 22B140         00470           LD      (TOPMEM),HL     ;SET UP POINTERS FOR ESF
4051 23             00480           INC     HL
4052 36F0           00490           LD      (HL),0F0H       ;DRIVE #0 WILL BE USED
4054 23             00500           INC     HL      ;HL BACK TO ORIGINAL
4055 3B             00510           DEC     SP      ;PERFORM TRICKY STACK MANIPULATE
4056 F1             00520           POP     AF      ;SO A GETS FILE NUMBER, NOT F
4057 33             00530           INC     SP      ;RESTORE SP
4058 CD0C30         00540           CALL    WRITE   ;WRITE FILE
405B CD2A30         00550           CALL    ERROR   ;DISPLAY ERROR MSG
405E C3B502         00560           JP      SYSTEM  ;WANT TO DO IT AGAIN?
                    00570 ;
```

```
                00580 ;         SUB ROUTINE
                00590 ;         ------------
                00600 ;
4061 D9         00610 GETNUM    EXX       ;SO THAT B COUNT FOR DJNZ REMAINS
4062 CDB31B     00620           CALL      INPUT    ;GET NUMBER
4065 D7         00630           RST       10H
4066 CD5A1E     00640           CALL      CONVRT   ;TO HEX, WITH RESULT IN DE
4069 D5         00650           PUSH      DE
406A E1         00660           POP       HL       ;NOW IN HL
406B E3         00670           EX        (SP),HL  ;GET RETURN ADDRESS
406C E5         00680           PUSH      HL       ;SET UP CORRECTLY
406D D9         00690           EXX       ;BACK AGAIN
406E C9         00700           RET
                00710 ;
                00720 ;
                00730 ;                   LOADER
                00740 ;                   ======
                00750 ;
                00760 ;                                    (C) 1982 N. J. COLEMAN
                00770 ;                                             6/13 HOWITT ST
                00780 ;                                             SOUTH YARRA VIC
                00790 ;
                00800 ;
                00810 ;         THIS PROGRAM IS DESINED TO BE USED IN CONJUNCTION
                00820 ; WITH "SAVER".  IT ENABLES A PREVIOUSLY SAVED PROGRAM
                00830 ; TO BE LOADED AND RUN.  IT IS NECESSARY TO USE THIS
                00840 ; UTILITY SINCE IT ENSURES THAT THE STACK IS IN THE
                00850 ; CORRECT LOCATION FOR LONG PROGRAMS TO BE LOADED.
                00860 ;
                00870 ;         NOTE THAT ONLY FILE #'S 1 - 9 MAY BE LOADED.
                00880 ; THIS SHOULD NOT CAUSE ANY SIGNIFICANT PROBLEMS SINCE
                00890 ; ONLY LENGTHY PROGRAMS WOULD BE LOADED USING THIS
                00900 ; UTILITY.  A FIFTY FT. WAFER HOLDS ONLY THREE OR FOUR
                00910 ; 16K PROGRAMS.
                00920 ;
                00930 ;
                00940 ;
4074            00950           ORG       4074H    ;= /16500
                00960 ;
                00970 ;
4074 CDB31B     00980 START2    CALL      INPUT    ;GET FILE #
4077 D7         00990           RST 10H
4078 7E         01000           LD        A,(HL)   ;A HAS #
4079 D630       01010           SUB       30H      ;CONVRT TO HEX
407B 4F         01020           LD        C,A      ;NECESSARY FOR ESF
407C B7         01030           OR        A        ;SET FLAGS
407D C35531     01040           JP        LOAD     ;LOAD PROGRAM
4042            01050           END       START
```

- 0000000000 -

**** SOFTWARE SECTION ****

***** UNIT CONVERSIONS PEACH and CC *****

This program first appeared in the July '81 issue and has, itself, been converted to run on the colour computers. It gives the user the choice of four different conversions:-

(1) Temperature.
    Fahrenheit to Centigrade and Centigrade to Fahrenheit.
(2) Length.
    Feet to centimetres and centimetres to feet.
    Inches to centimetres and centimetres to inches.
(3) Distance.
    Kilometres to miles and miles to kilometres.
(4) Weight
    Stones to kilograms and kilograms to stones.

One added feature allows the user to enter lengths in feet and inches, and weight in stones and pounds. The reverse conversions produce similar output.

- 0000000000 -

***** NORMAL DISTRIBUTION PEACH and CC *****

Many statistical procedures begin with the assumption that the data under study is normally distributed. From such a distribution, a series of useful statistical parameters arise and this program will compute the mean, variance, standard deviation, standard error and range for up to 100 data values. The program begins by drawing a graph of a normal distribution which shows the expected frequency of data values about the central mean, or average, value. After the data has been entered, the values are displayed for verification and may be edited if desired. Finally, the parameters described are calculated and displayed on the screen. The program is essentially the same as when first published in the May '81 issue, with the addition of the high resolution graph.

- 0000000000 -

***** MICRO GRAND PRIX L2/16K m.l. - by Ronald J. Sully *****

Micro Grand Prix is a road race game written in machine language to ensure high speed action. Your task is to steer a racing car around the curves without running off the track or crashing into an oncoming vehicle. You have control over your car's steering and speed via the following keys:-

< MOVE LEFT
> MOVE RIGHT
or ESC INCREASE SPEED
or CTRL DECREASE SPEED

The listing at the back of the magazine includes the very well commented source code and the object code. If you have an Editor Assembler such as the Radio Shack EDTASM or Microsoft Editor Assembler Plus, you should enter the source code starting with the line numbers in the third column of the listing. If you do not have an Editor Assembler, you should use the Edit Memory function of a monitor to enter the object code in the first two columns of the listing. You should use a low memory monitor such as Tandy's TBUG or ZMONL from the MICRO-80 Software Library. The 16K version of BMON or ZMONH are not suitable as they occupy the same memory area as the program being entered. When you have entered the program via the monitor, make a System tape having the following parameters:-

Start	End	Entry
7000	751C	7000

LOADING THE PROGRAM

To load the program from cassette answer MEMORY SIZE?/READY? with (ENTER/NEWLINE). Type SYSTEM (ENTER/NEWLINE). Type PRIX (or just P) (ENTER/NEWLINE). Watch the pretty asterisks then answer the next prompt with /ENTER/NEWLINE and away you go!

The procedure for loading the program from disk depends on the DOS you are using.

NEWDOS 80 ver 1 or TRSDOS

From DOS type -

PRIX and press ENTER/NEWLINE

The program will then start.

NEWDOS 80 ver 2 or the Distribution DOS.

From DOS type -

LOAD PRIX/CMD and press ENTER/NEWLINE then type BASIC press ENTER/NEWLINE then type:-

SYSTEM and press ENTER/NEWLINE then in response to the prompt *? type -

/28672 and press ENTER/NEWLINE

The program will then start.

DOSPLUS

Type -

BASIC and press ENTER/NEWLINE then from BASIC type -

CMD"PRIX/CMD" and press ENTER/NEWLINE

The program will then start.

SOURCE CODE NOTES.

The following notes are offered to those who are "into" A/l programming and wish to analyse the way this program works.

Line Nos.	Description
10 - 80	"Flag waving"
90	VIDEO = 3COOH (15360) beginning of screen
100	The ORG (ORiGin) determines where in memory the program will reside.
110-200	Initialises variables.
220	Like "GOSUB FRAM1" (see lines 2460-3480)
230-250	Is used to start game or return to DOS/BASIC
270-330	Randomly determines the position of the first character of MSGE13 and writes 64 characters of MSGE13 to the top line of the screen.
350-430	Randomly determines if the road is to bend left or right and adjust the new position accordingly.
450-540	Ensures the road stays within the limits of the screen.
600-640	Offers a 1 in 10 chance of getting an obstacle car on the road. To get more cars decrease the number in line 600. Less cars - increase the number. To have no cars at all (perhaps a speed test?) change the number in line 630 to be greater than the number in line 600.
600-690	Without this routine the program could not simulate the car moving along the road.
710-740	This automatically speeds up the program each cycle.
760 - 780	Increments the score by 1.
800-1080	This routine is used to get and act upon the function keys you press.
1120-1230	This is part of moving the car routine. It looks ahead to see if the car will be drawn on top of a space. If not (crash imminent) then the flag FLAG is set.
1260-1340	Before the car is drawn in the new position the score is written to the screen at the position determined by line 1320. This routine is located here so you don't "crash" into your score.
1360-1390	Draws the car on the screen.
1410-1430	Checks if the crash flag (FLAG) has been set: if so then GOTO CRASH.
1440-1490	Checks to make sure that the value of SPEED never gets below 20. (If SPEED becomes negative the program will slow down to a virtual stop).
1500-1510	Stops the program while the value in BC is decremented to 0. (The actual SPEED control).
1520	Completes the game cycle and continues.
1540-1570	Is a general purpose routine for getting RND numbers. Before CALLing this routine HL must contain the maximum value of the random number. On exit DE will contain the number selected. (DE = RND (HL)).
1590-1790	Is the routine to simulate a crash. The routine is inside a loop which is set at 50. That is, the cars will "flash" 50 times. To change the number of times, change the value in line 1590.
1800	Clears the screen.
1810	ROM routine to change to 32 char/line.
1830-1860	like "PRINT@VIDEO+202,MSGE14$;"
1880-1960	like "PRINT@VIDEO+228,YORSCR;"
1970-2170	Compare all the scores; sort if necessary, and set flag MSG if current score is champion score.
2180-2240	Check flag MSG and if set write appropriate message.
2250-2280	like "PRINT@VIDEO+706,MSGE15$;"
2290-2300	Reset MSG flag.
2310-2320	Scan keyboard. If key is pressed start game again. If not scan keyboard again.
2340-2420	Subroutine to randomly determine where obstacle car is to be drawn and then draws it.
2460-2930	Writes the Instructions on the screen.
3240-3480	Draws the road and the car.
3500-3810	Is the list of all the string and numeric variables used in the game. Note that line 3630 is the design of the verge of the road. If you change it, make sure you have the same number of characters. Note also (lines 3640-3660) that any text which is to be written in 32 char/line is to be formatted accordingly beforehand.
3820	The compulsory END statement.

Well, that's it! When you get sick of playing the game then perhaps you could analyse it. There may be some routines you could use in YOUR A/L game programs. MICRO-80 would welcome submissions of real-time, fast action A/L game programs.

MICRO-80 PRODUCTS - CATALOGUE

HIGH QUALITY PRODUCTS FOR YOUR COMPUTER AT UNBEATABLE PRICES.

ABOUT MICRO-80 PRODUCTS

MICRO-80 PRODUCTS was started at the request of MICRO-80 readers who wanted to obtain good quality peripherals and software for their computers at reasonable prices. In the past 2 1/2 years literally thousands of satisfied customers can attest to the fact that MICRO-80 PRODUCTS has achieved this objective. We have removed much of the mystique which surrounds the interfacing of such useful peripherals as disk drives and printers and have become the major Australian source of supply for such software products as NEWDOS and DOSPLUS which have increased the power and speed of TRS-80 micro-computers enormously. More recently, we have saved Hitachi owners considerable sums by interfacing MPI disk drives to the Hitachi Peach. We were the first in Australia (in the world?) to successfully interface the range of Olivetti electronic typewriters to be printers and have designed and produced a number of useful modifications for the TRS-80/System 80.

As the interest of micro-computer users broaden, so do our own. We now actively sell and support the TRS-80 Model 3, the Osborne 1, the Hitachi Peach, the Olivetti M20 microcomputer, the North Star Advantage and the Altos multi-user system. We would be happy to assist you in upgrading your present computer with new peripherals or even exchanging it for a more modern machine.

MAIL ORDER POLICY

Much of our business is carried out by Mail Order and our customers find it a simple and efficient way to do business. You may place your order by telephone or by mailing in the order form from any issue of MICRO-80 magazine. Generally, it takes about 1 week from receipt of order until dispatch. You should allow 2-3 days for your letter to reach us and 7-10 days for the parcel to reach you making a total turnaround time of about 3 weeks. If we are temporarily out of stock of an item, we will send you a notification of back order giving our best estimate of when it will be back in stock. Payment, which should accompany the order, may be by Cheque, Money Order, Bankcard or Access. If we are unable to supply an order immediately, we apply the following rules:

-If payment is by cheque and none of the order is in stock, the cheque is not presented until the order can be fulfilled.

-If payment is by cheque and some items are in stock, the cheque is presented and the items back ordered are shown on the invoice which accompanies the goods.

-If payment is by Bankcard or Access, only these items which can be supplied are charged. Back ordered items are not charged until available.

If you wish to speed up delivery, you may pay a special delviery fee to have the item sent by road freight or even air express. Ring for prices.

WARRANTY AND SERVICE

All hardware products carry a 90 day parts and labour warranty either from the manufacturer/distributor or from MICRO-80 PTY. LTD. In many cases, warranty servicing can be arranged in your own city, otherwise goods should be returned to MICRO-80 PTY. LTD. the cost of freight to MICRO-80 is at customer's expense. Return freight on goods which require repair or adjustment, either by road or post at MICRO-80's discretion, will be paid for by MICRO-80 PTY. LTD. Customers should obtain a return authorisation from MICRO-80 before despatching goods for warranty repair, post warranty servicing can also be carried out at very reasonable rates.

-0000000000-

TRADE-INs, EASY PAYMENT TERMS

MICRO-80 BRINGS COMMONSENSE TO COMPUTER BUYING

If you wish to buy a new car, you are able to trade-in your existing vehicle and arrange finance for your new purchase, all under the one roof. Not so with microcomputers. If you want to dispose of an existing machine, you are on your own and, in most cases, you must make your own arrangements about finance, too. Here at MICRO-80 we think this is ridiculous, so we have done something about it. We are now able to accept TRADE-INs on new COMPUTERS and PERIPHERALS and to arrange CONSUMER MORTGAGE terms to approved customers. This offer applies to our customers ALL OVER AUSTRALIA, not just in South Australia.

Here is what you do.
If you are interested in trading-in existing equipment:-

1) Write to us or phone us, describing the equipment you wish to trade-in. Make sure you tell us its age and any distinguishing features. Eg.: TRS-80 Model 1, early style keyboard with "square' monitor, L2/16K, 3 years old, good condition.

2) Tell us too, what computer you wish to purchase from our range of Hitachi, TRS-80 Model 3 Osborne, Olivetti and North Star.

3) We will write, offering you a trade-in valuation and quoting the price of the equipment requested. Our trade-in offer will be subject to inspection of the equipment at our premises. Our letter will also include instructions for sending the equipment to us in the most cost effective manner.

4) If you are satisfied with our offer and quotation, send us your equipment, together with payment for the balance (or, if you wish to purchase on terms, see 6 below) and we will send your new computer to you.

If you would like to take advantage of consumer mortgage or leasing finance, with or without a trade-in:-

5) Write or 'phone telling us the equipment you wish to purchase.

6) We will send you a written quotation, an order form and a "personal particulars" form for the appropriate finance.

7) Complete the order form and the "personal particulars" form and return them to us. We will pass on your particulars to the finance company which will contact you directly. The order is conditional upon you obtaining finance of the required amount at the quoted rate. If this is not available at the time the order is received, we will contact you for further instructions. South Australia has some of the most strigent regulations in Australia controlling consumer finance, you may rest assured that your interests will be well protected.

8) When authorised to do so by the finance company, (generally 3-7 days) we will despatch the new equipment to you.

EASY PAYMENTS TERMS ALSO AVAILABLE ON PERIPHERALS

The same consumer finance is also available on hardware peripherals selling for more than $250. For example, if you require a disk drive costing $499, you could purchase it on 10% deposit and payments of only $4.17 over a period of 36 months.

Even software can be included in the overall purchase to a limited extent. Eg.: If you purchase a new computer system then you could also finance a Disk Operating System and application programs up to about 10% of the total value of the purchase.

WE HAVE CUSTOMERS WAITING FOR USED COMPUTER SYSTEMS

In high demand are TRS-80 Model 1 systems with one or more disk drives. If you have such a system, why not trade it in on a new computer?

Finance and leasing facilities to approved clients is available through "ESANDA" Adelaide.

-0000000000-

***** BOOKS *****

THE CUSTOM TRS-80 & OTHER MYSTERIES
$32.50 + $1.20 p.&p.

The complete guide to interfacing your TRS-80 to the outside world, covering both hardware and software.

TRS-80 DISK & OTHER MYSTERIES
$27.00 + $1.20 p.&p.

A must for the serious disk user. Disk file structures revealed. DOS's compared and explained, how to recover lost files, how to rebuild crashed directories.

LEARNING LEVEL 2 NOW ONLY $7.95 +$1.20 p.&p.

Written by David Lien, the author of the TRS-80 Level 1 Handbook, this book teaches you, step-by-step, how to get the most from your Level 2 machine. Invaluable supplement to either the TRS-80 Level 2 manual or the System 80 manuals.

Level 2 ROM ASSEMBLY LANGUAGE TOOLKIT
$29.95 + $1.20 p.&p.

The definitive work on using Level 2 ROM routines in your own programs. Covers TRS-80 Model 1 and 3 and System 80. Comes complete with DBUG, a machine language debugging monitor distributed on cassette. This package is a must for machine language programmers and BASIC programmers.

BASIC BETTER AND FASTER
$32.50 + $1.20 p.&p.

Fast becoming the "bible" on the TRS-80 for BASIC programmers , this book is packed full of useful routines and techniques all fully explained, which you can use in your own programs. If you are serious about learning to program, then this is a must.

*** A FEAST OF GAMES FROM AMERICA'S TOP SOFTWARE HOUSES!!! ***

MICRO-80 NOW HAS IN STOCK, SOME OF THE BEST SPACE GAMES AND ADVENTURES WRITTEN FOR THE TRS-80. THESE PROGRAMS ARE SUPPLIED ON CASSETTE AND WILL ALL RUN IN A LEVEL 2/16K TRS-80 MODEL I & MODEL III. THEY WILL ALSO RUN ON THE SYSTEM 80 BUT SOUND MAY NOT BE AVAILABLE UNLESS A HARDWARE MODIFICATION TO REVERSE THE ROLES OF RECORDERS #1 and #2 HAS BEEN FITTED. LIMITED STOCK AVAILABLE AT THESE PRICES.

THE BEST IN SPACE GAMES FROM BIG FIVE

GALAXY INVASION - $25.50 + $1.00 p&p

A fast paced, arcade type, m/l game for 1 or 2 players; 6 different craft flying in formation are attacking Earth, after each formation they become faster and more deadly - complete with sound effects.

ROBOT ATTACK - $25.50 + $1.00 p&p

Robots have overtaken one of Earth's space stations and it is your mission to invade the station and conquer the Robots - INCLUDES *VOICE* SOUND EFFECTS.

COSMIC FIGHTER $19.95 + $1.00 p&p

Your ship comes out of hyperspace under a convoy of aliens, you destroy every one but another set appears, these seem more intelligent. You eliminate them too. Your fuel supply is diminishing. You must destroy 2 more sets before you can dock - includes sound effects.

DEFENCE COMMAND - $25.50 + $1.00 p&p

Your mission is to protect vital fuel cells from the invading aliens. However, they have captured all your fuel, beware the solar waster: - complete with sound effects.

PENETRATOR - $35.50 + $1.00 p&p

Penetrator - Superb graphics, rapid fire action, challenging situations, training options and fantastic sound combine to make penetrator the game of the year! The unique customizing feature allows you to change the landscape at will, make it easy or impossible the choice is yours - 2 cassette pack

METEOR MISSION - $19.50 + $1.00 p&p

Six stranded astronauts are shouting for help on the planet below, it is your mission to rescue them to the mother ship, but watch out for asteroids, meteor showers and alien craft - complete with sound effects.

ATTACK FORCE - $25.50 + $1.00 p&p

In this fast paced, m/l game 8 alient ramships are warping towards your ship. You must dodge them and fire your missiles before they destroy you - but watch out for the flagship and its death beam!! - complete with sound effects.

SUPER NOVA - $25.50 + $1.00 p&p

A fast paced, real-time game, for 1 or 2 players. The object is to destroy as many asteroids and aliens as possible without getting destroyed. Large asteroids shatter into smaller ones and the alien flagship fires a deadly bolt which means disaster to your mission.

STELLAR ESCORT - $25.50 + $1.00 p&p

Your mission is to intercept the supply cruisers, place them in your fighters tractor beam and escort them through the Cretonian's battle front while warding off attacks - includes sound effects.

STRIKE FORCE $25.50 + $1.00 p&p

Strike Force is one of the most difficult games for the TRS-80, making maximum use of graphics. Your mission - save 5 cities, destroy the alien craft and finally destroy their home base. Fast and Hard - with sound effects.

FROM ADVENTURE INTERNATIONAL

LUNAR LANDER - $19.50 + $1.00 p&p

Written in m/l, you will see an amazing lunar landscape scroll below your module - it is your mission to land safely before running out of fuel. A game that requires both skill and luck - complete with sound effects.

ELIMINATOR - $25.50 + $1.00 p&p

Your mission is to prevent the marauding alien hoards from recovering your energizers from the planet's surface. There are several types of alien ships - each with different weapons to destroy you!! - with sound effects.

PLANETOIDS - $25.50 + $1.00 p&p

Its your ship vs. a swarm of killer planetoids, as you try to destroy them before they destroy you - with sharp graphics and sound effects.

MISSILE ATTACK - $19.50 + $1.00 p&p

This is a real-time game with sound effects. You must protect your cities against enemy missiles, as your skill increases, so does the level of difficulty making accuracy a must.

SPACE INTRUDERS - $25.50 + $1.00 p&p

A very fast game from the deluxe version of Space Invaders, complete with "spitting" invaders and the SOS of escaping aliens - with sound effects.

ARMOURED PATROL - $31.00 + $1.00 p&p

Armoured patrol is a 3-D arcade style game. Your mission is to seek out and destroy enemy tanks and other secret weapons - incredible graphics.

ADVENTURE HINT BOOK - $9.95 + $1.00 p&p

If you can not go any further this will give you clues that may help - written by Scott Adams for Adventures 1-9.

3-D ADVENTURES

ASYLUM - $25.50 + $1.00 p&p

Asylum places you in a cell, you have to escape. Its harder than it sounds, lots of hazards will be encounted.

DEATHMAZE 5000 - $25.50 + $1.00 p&p

Deathmaze 5000 is another 3-D adventure. You move through a 5 storey building - your goal is to leave the deathmaze alive.

LABYRINTH - $25.50 + $1.00 p&p

Labyrinth - you move through a gigantic labyrinth and scattered through this nightmare are a multitude of objects and obstacles. A minotaur prowls the corridors you must kill it before it kills you, Labyrinth has over 550 locations - be patient.

SCOTT ADAMS ADVENTURE

ADVENTURELAND - $25.50 + $1.00 p&p

Wander through an enchanted world trying to recover 13 lost treasures. You'll encounter wild animals, magical beings, and many other perils and puzzles. Can you rescue the Blue Ox from the quicksand? Or find you way out of the maze of pits?

PIRATE'S ADVENTURE - $25.50 + $1.00 p&p

"Yo ho ho and a bottle of rum..." Meet the pirate and his daffy bird along with many strange sights as you attempt to get out of your London flat and get to Treasure Island. Can you recover Long John Silver's lost treasures?

MISSION IMPOSSIBLE - $25.50 + $1.00 p&p

Good morning, your mission is to...and so it begins. Will you be able to complete your mission in time? Or is the world's first automated nuclear reactor doomed? This is hard. There's no magic and no help this time, but plenty of suspense. Good luck.

VOODOO CASTLE - $25.50 + $1.00 p&p

Count Cristo has had a fiendish curse put on him by his enemies. There he lies, with you as his only hope. Will you be able to rescue him or is he forever doomed? Beware the Voodoo Man....

THE COUNT - $25.50 + $1.00 p&p

You wake up in a large brass bed in a castle, somewhere in Transylvania. Who are you, what are you doing here, and WHY did the post man deliver a bottle of blood? You'll love this adventure, in fact you might say it's Love at First Byte.

STRANGE ODYSSEY - $25.50 + $1.00 p&p

Marooned at the edge of the galaxy, you've stumbled on the ruins of an ancient alien civilization complete with fabulous treasures and unearthly technologies. Can you collect the treasures and return home or will you be marooned forever?

MYSTERY FUN HOUSE - $25.50 + $1.00 p&p

Can you even find your way in to the Strangest Fun House in existence let alone find your way completely through it or will you get kicked out when the park closes?

PYRAMID OF DOOM - $25.50 + $1.00 p&p

An Egyptian Treasure Hunt leads you into the dark recesses of a recently uncovered Pyramid. Will you recover all the treasures or more likley will you join its denizens for that long eternal sleep?

GHOST TOWN - $25.50 + $1.00 p&p

Explore a deserted western mining town in search of 13 treasures. From rattlesnakes to runaway horses, this Adventure's got em all! (Also includes new bonus scoring system).

SAVAGE ISLAND - $25.50 + $1.00 p&p

Part 1 - a small island in a remote ocean holds an awesome secret. Will you be the first to uncover it? NOTE: this is the first part of a larger adventure. it will be necessary to buy further tapes to complete the entire Adventure. WARNING: FOR EXPERIENCED ADVENTURERS ONLY!

SAVAGE ISLAND - $25.50 + $1.00 p&p

Part 2 - After struggling through Part 1, you have the consolation of knowing its half over. This concludes the two part Adventure. It requires you have completed Part 1 and received the password to start Part 2.

GOLDEN VOYAGE - $25.50 + $1.00 p&p

WARNING: For Experienced Adventurers Only! The King lies near death in the royal palace - you have only three days to bring back the elixir to cure him. Journey through the lands of magic fountains and sacred temples, stormy seas and gold, gold, GOLD!

PROGRAMS FROM MICROSOFT

Adventure on Disk		$ 41 plus $1.00 p&p
BASIC Compiler		$290 plus $1.00 p&p
Editor/Assembler +	Cassette	$ 69 plus $1.00 p&p
	Disk	$ 69 plus $1.00 p&p
Fortran 80		$177 plus $1.00 p&p
Level III Basic		$ 88 plus $1.00 p&p
Decathlon Cassette, Disk		$ 55 plus $1.00 p&p
MuMath		$145 plus $1.00 p&p
MuMath/MuSimp		$370 plus $1.00 p&p
Typing Tutor		$ 45 plus $1.00 p&p

LNW *) II * LNW 80 II * LNW 80 II * LNW 80 II * LNW 80 II * LNW 80 II * LNW 80 II * LNW 80 II

IT'S HERE AT LAST

The LNW80 II Microcomputer

Manufactured in America by LNW Research Corporation, the LNW80 II has the following outstanding features:

- o Completely software and hardware COMPATIBLE with the TRS-80 Model 1.
- o HIGH RESOLUTION COLOUR GRAPHICS - 4 MODES:
  - B/W LO-RES 128 x 48
  - B/W HI-RES 480 x 192
  - COLOUR LO-RES 128 x 192 IN 8 COLOURS
  - COLOUR HI-RES 480 x 192 IN 8 COLOURS
- o CP/M Disk Operating System
- o Single and Double Density Disk Operating System
- o Supports 5 1/4 inch or 8 inch Floppy Disk Drives
- o 48K RAM in TRS-80 mode plus 16K High Resolution graphics RAM
- o 64K RAM in CP/M mode plus 32K Banked in, usable in BASIC, plus the 16K High Resolution Graphics RAM
- o 4 MHz Z80A microprocessor - over twice the operating speed of the Model 1
- o HI-RES COLOUR (R-G-B) and B&W video outputs
- o 3 screen display modes
  - 64 characters x 16 lines
  - 80 characters x 16 lines
  - 80 characters x 24 lines
- o SOFTWARE SUPPORT

  Apart from being able to run all TRS-80 Model 1 software and all CP/M software, there is also an extended BASIC interpreter available for the LNW80 II using most of the same commands as the TRS-80 Colour Computer but with full LNW Graphics Resolution, SET, RESET, POINT, LINE and CIRCLE as well as special commands to generate sound effects and tones. TRS-80 Colour Computer BASIC programs can be transferred to the LNW with only minor changes.

The LNW80 II is the ideal computer for the serious hobbyist or businessman who is seeking a higher performance, more reliable computer to replace his TRS-80 Model 1 without sacrificing his investment in software or his programming experience. As of writing, we have one demonstration unit in Adelaide. We expect to start delivering computers in January 1983. You may reserve an LNW80 II system by paying 10% deposit now. Trade-ins will be accepted. The LNW80 II uses standard Tandy or Tandy compatible disk drives. if you already have a disk TRS-80 system you may continue to use your existing disk drives on the LNW80 II.

LNW80 II computer complete except for disk drives and monitor	$2,750 incl. S.T.
Hi-Res. Green phosphor monitor	$ 260 incl. S.T.
Super Hi-Res Hitachi RGB Colour Monitor	$1,250 incl. S.T.

SPECIAL OFFER - STOCK CLEARANCE

SAVE $600 ON A BRAND NEW OSBORNE 1
ONLY $1,995 INCL. SALES TAX

The new blue-case Osborne 1 is on the way so we are closing out our stock of current model brown-case Osborne 1's below cost. The new Osborne will differ only in the shape and colour of its case. It will be the same as the current model in every other way. Now is your chance to secure the computer bargain of a life time and we will accept trade-ins and arrange terms too! HURRY, this offer only applies while stocks last.

BUY YOUR MODEL 3 FROM MICRO-80 AND SAVE $000's

MICRO-80 fits reliable MPI disk drives to the TRS-80 Model 3 to give system capacities and capabilities far in excess of those available elsewhere. All our conversions utilise low dissipation, switching-mode power supplies to avoid screen jitter and overheating. The disk controller boards used incorporate special compensation circuitry for 80 track disk drives and may also be used to run 8 inch disk drives with an appropriate cable and DOS.

MODEL 340
2 40 TRACK SINGLE-HEAD DISK DRIVES GIVING
350K FORMATTED STORAGE, 48K RAM **$3130**

MODEL 340+
2 40 TRACK DUAL-HEAD DRIVES GIVING
700K FORMATTED STORAGE, 48K RAM **$3350**

MODEL 380+
2 80 TRACK DUAL-HEAD DRIVES GIVING
1.4 MEGABYTE FORMATTED STORAGE, 48K RAM **$3800**

★ *NEW* ★ ★ *NEW* ★ ★ *NEW* ★

***MODEL 500 — 5+* MEGABYTE MODEL 3**
1 40 TRACK DUAL-HEAD DRIVE GIVING
350K OF FLOPPY DISK STORAGE FOR TRANSFERRING
PROGRAMS AND BACKUP, 48K RAM, EXTERNAL
5 MEGABYTE WINCHESTER SUB-SYSTEM,
CP/M (ORG 4200N) DISK OPERATING SYSTEM **$5895**

The MODEL 500 offers the high speed, mass storage capacity and reliability of a Winchester drive for thousands of dollars less than you would pay for any comparable system. Model 500 is a serious business computer able to tackle the most demanding tasks.

All prices are in Australian dollars, include Sales Tax and are subject to change without notice. Prices are FOB Adelaide. Add $20 road freight anywhere in Australia. All computers and systems carry MICRO-80's 90-day Warranty covering parts and labour.

SPECIAL XMAS OFFER - $100 OFF MOST PRINTERS

Olivetti Praxis 30 Typewriter/Printer	$ 795	ITOH F10-40P character per second Daisywheel/Printer	$1850
Olivetti Praxis 35 Typewriter/Printer	$ 895		
Olivetti ET121 Typewriter/Printer	$1450	ITOH Pro-Writer 8510, 80 col. 112 cps Dot Matrix * $200 OFF!!!	$ 890
		ITOH 1550 132 col. 120 cps	$1399

PRINTERS GALORE AT UNBEATABLE PRICES

MICRO-80 has a range of printers to suit every requirement from dot-matrix to correspondence quality daisywheel. Chose from the table below:

BRAND	MODEL	TYPE	SPECIFICATIONS									
			COL	SPEED CPS	BI-DIR	LOWER CASE	PAPER FEED	GRAPHICS	INTER FACES	FREIGHT	PRICE	WEEKLY PAY-MENTS*
STAR	DP	DM	80	80	Y	ND	F/T	BLOCK	P	1	$ 575	$4.81
EPSON	MX-80	DM	80	80	Y	FULL	F	BLOCK	P	1	$ 899	$7.53
EPSON	MX-80II	DM	80	80	Y	FULL	F/T	HI RES	P	1	$ 999	$8.35
EPSON	MX-100	DM	132	100	Y	FULL	F/T	HI-RES	P	1	$1500	$12.55
MICROLINE	83A	DM	132	120	Y	FULL	F/T	BLOCK	P/S	1	$1599	$13.37
MICROLINE	84	DM	132	200	Y	FULL	F/T	HI-RES	P	1	$2220	$18.57
MICROLINE	84	DM	132	200	Y	FULL	F/T	HI-RES	S	1	$2340	$19.57
C ITOH	8510	DM	80	112	Y	FULL	F/T	HI-RES	P	1	$1099	$9.19
C ITOH	M1550	DM	132	120	Y	FULL	F/T	HI-RES	P	1	$1499	$12.54
DATA SOUTH	DS-180	DM	132	180	Y	FULL	T	OPT.	P/S	1	$2590	$21.66
OLIVETTI	PRAXIS30	DW	100	6	N	FULL	F	NO	P	1	$ 895	$7.49
OLIVETTI	PRAXIS35	DW	100	6	N	FULL	F	NO	P	1	$ 995	$8.33
OLIVETTI	ET121	DW	132	12	N	FULL	F	NO	P	2	$1500	$12.55
OLIVETTI	ET221	DW	132	16	N	FULL	F	NO	P	2	$2650	$22.17
ITOH	F10 40P	DW	132	40	Y	FULL	F	NO	P	2	$1950	$16.31
ITOH	F10 40S	DW	132	40	Y	FULL	F	NO	S	2	$2190	$18.32

NOTES: The following symbols are used

TYPE+	DM = DOT MATRIX DW = DAISYWHEEL
- BI DIRECTIONAL	Y = YES N = NO
- LOWER CASE	FULL - means Lowercase descenders go below line ND - means Lowercase descenders do not go below line
PAPER FEED	F - means Friction Feed T means Tractor Feed F/T - means both Friction and Tractor feed included in the price
- INTERFACES	P = PARALLEL (Centronics) S = SERIAL (RS232)
FREIGHT	1 - Add $10 for road freight anywhere in Australia 2 - Add $20 for road freight anywhere in Australia

Note Prices subject to change without notice. Prices quoted include Sales Tax at the 17.5% rate.

Call or write for more details.

ENHBAS

$51.95 + $1.00 p&p

ENHBAS adds over 30 new commands and functions to your BASIC interpreter including high speed SORT, labels in BASIC, RESTORE to any line number, WHILE-WEND for structured programming, SCROLL, LEFT, INVERT, DRAW and PLOT to give you ease of control over graphics, SOUND and PLAY to add realistic sound effects and many more. Makes programming a breeze! Available for Model 1 or 3, disk or cassette - specify which when ordering.

SCARFMAN

Cassette $16.95 + 60¢ p&p
Disk $22.95 + 60¢ p&p

SCARFMAN an AMAZEing game known in the arcades as Ghostmuncher or Pacman. This is by far the best implementation of this thrilling arcade game that we have seen on the TRS-80. It comes complete with realistic sounds, fast action and nine levels of play. SCARFMAN will support the use of Alpha Products Joysticks.

Specify Model 1 or Model 3.

DISK OPERATING SYSTEMS FOR TRS-80/SYSTEM 80 COMPUTERS

You can increase your programming productivity, the execution speed and 'user friendliness" of your programs by using an enhanced Disk Operating System (DOS). MICRO-80 recommends DOSPLUS and NEWDOS 80 according to your requirements and experience.

USERS REQUIREMENTS	RECOMMENDED DOS	PRICE	ORDERING INFORMATION
Single-sided Disk Drives, Economy, First-Time User (requires TRSDOS & DISK BASIC MANUAL to supplement DOSPLUS MANUAL.	DOSPLUS 3.3	$ 99.95	Specify Model 1 or Model 3. If Model 1 whether single or double density.
Single or Double-sided Disk Drives, any track count 5 inch or 8 inch. First-time or experienced user wanting Fuss-Free, Bug-Free easy to understand, but very powerful DOS which support variable length records up to 255 Bytes long with stand alone manual. High degree of compatability with TRSDOS.	DOSPLUS 3.4 Highly Recommended	$149.95	Specify Model 1 or Model 3. If Model 1 whether single or double density
Single or Double-sided single or double density disk drives, any track count. 5 inch or 8 inch. Experienced user who has already used TRSDOS and understands the manual. Requires the most powerful DOS available and is prepared to learn the somewhat complicated Syntax. Requires flexible and powerful file handing in BASIC including variable length records up to 4096 Bytes long. Definitely not for the Beginner.	NEWDOS 80 Version 2.0	$169.00	Specify Model 1 or Model 3

NEWBASIC $99.95 PLUS $1.20 P.&P.

BASIC is the programming language used on most microcomputers. One of its main limitations is its unstructured nature which not only leads to untidy and complicated code but also allows very little portability of code from one program to another. NEWBASIC overcomes this limitation by adding PROCEDURE CALLS and enabling you to define BLOCKS thus localising parts of your program yet enabling you to pass parameters to the remainder of the program. With NEWBASIC loaded on top of your BASIC interpreter, you have the familiarity and interactive nature of BASIC with many of the advantages of PASCAL. NEWBASIC adds the following facilities to your interpreter.

COMMANDS & FUNCTIONS

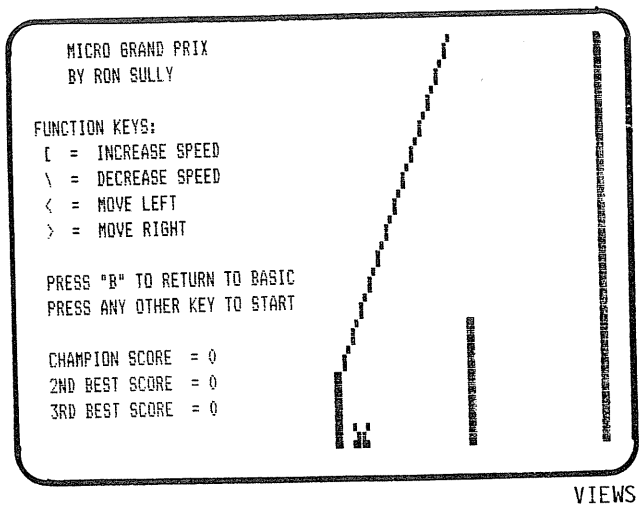
BREAK	lets you program commands for breakpoints.	0.3K
CALL	now you have procedures and sub-programs in BASIC.	*
CONT	continue after a break by just pressing enter.	0.1K
DEF BLOCK	localise parts of your programs yet pass parameters.	*
DEF END	end of a BLOCK, FUNCTION, or PROCEDURE.	*
DEF FUNCTION	start of a multi-line function.	**
DEF PROCEDURE	start of a CALLed procedure.	*
FIELD @	point strings at any part of memory.	0.1K
&FIND	find strings very quickly, anywhere in memory.	0.5K
&FN	access to multi-line functions	**
MERGE	Very speedy loading of programs	***
MOVE	copy memory anywhere, fill it with anything, fast	0.3K
PLUG	chain + pack parts of your program, keep running.	***
RESERVE	reserve and release protected memory as you run.	0.3K
STRINGS	extend and reduce string space when you want to.	0.2K
TIME	measure the time taken by any lines in your program.	1.0K

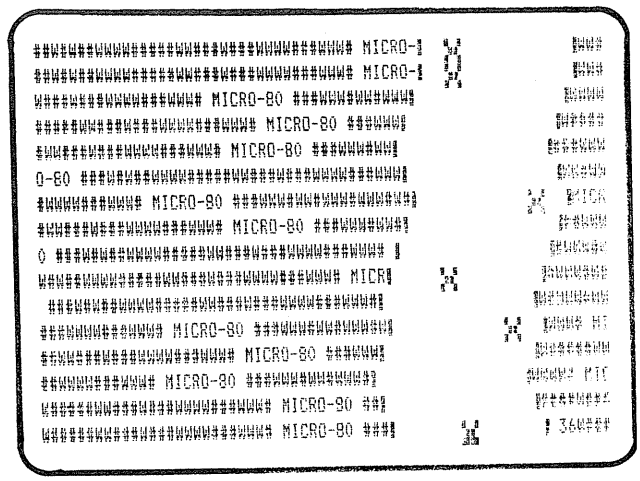
NewBasic has a 2.9K mandatory root.
* 5.0K in total for blocks.
** 0.5K for functions in addition to blocks.
*** 1.0K for segmented overlaying.

NEWBASIC requires a single disk driveTRS-80 1 or 3 with at least 32K of RAM using TRSDOS, NEWDOS, or NEWDOS 80 Versions 1 or 2.

NEWBASIC requires a single disk drive TRS-80 1 or 3 with at least 32K of RAM using TRSDOS, NEWDOS, or NEWDOS 80 Versions 1 or 2.

-0000000000-





VIEWS OF SCREEN

- 0000000000 -

***** PASSWORD LII/16K - by A. Park *****

Password is a short machine language routine for non-disk systems which allows the user to put the computer into a perpetual loop until he enters a four letter code, specified by the user, which, when entered, returns control to BASIC.

The program is loaded from cassette by typing:-

SYSTEM then press ENTER/NEWLINE

Then type:-

PASSWD and press ENTER/NEWLINE

When loaded the program will then automatically run and set memory size and display the following message:-

** PASSWORD INITIALIZED **

In order to access the routine from BASIC, type:-

LSET and press ENTER/NEWLINE

The display will then show:-

** ** TERMINAL ON STAND-BY ** **

The computer is now in a perpetual loop which cannot be broken by pressing RESET. In order to regain control of the machine the correct password must be entered - incorrect words are ignored. Consequently, a BASIC program can be stopped (using the BREAK key), type in LSET, (go and get some coffee), return and type in the password, then type CONT and your program continues!

The routine itself is very simple in structure, an initialization procedure sets Memory Size and LSET pointers and then returns to BASIC. Once LSET is typed in, the program tests characters typed in on the keyboard with the pre-defined password. The password must pass four tests (one for each character) and if they are OK then it returns control to BASIC, otherwise it waits until the correct password is entered.

For the program to run automatically, it must be typed in using an Editor Assembler such as the Radio Shack EDTASM or Microsoft Editor Assembler Plus. You should enter the source code starting with the line numbers in the third column of the listing. If you do not have an Editor Assembler, you should use the Edit Memory function of a monitor to enter the object code in the first two columns of the listing. You should use a low memory monitor such as Tandy's TBUG or ZMONL from the MICRO-80 Software Library. The 16K version of BMON or ZMONH are not suitable as they occupy the same memory areas as the program being entered. When you have entered the program via the monitor, make a System tape having the following parameters:-

START	END	ENTRY
7F2B	7FFF	7F2B

As this program is for a LII/16K machine the command file on the distribution disk, (PASSWORD/CMD), will automatically load level 2 BASIC with the password program into your machine.

A small BASIC program is also supplied which allows you to change the password while the program is in memory. Oh yes, I nearly forgot, if you have already loaded this program and typed LSET I suppose you want to know the password, well it's TEST.

If you type the program in using an Editor Assembler and you wish to change the password, change the Hex values in lines 860 and 890 to the hex value of the required characters. If you are using a monitor change the values in the following memory locations to the hex values required before you punch out the tape.

7FB8 1st character
7FB9 2nd character
7FBA 3rd character
7FBB 4th character

- 0000000000 -

***** THE GAME OF OTHELLO - by Peter R. Smith *****

The game of OTHELLO is played on an 8 by 8 grid (like a chess-board). The object of the game is to occupy more squares than your opponent.

You are each given two squares to start with, in the two diagonally opposite corners of the centre four squares of the board.

On your turn, you place one more square of your colour (or shape in the computer version), BUT this square must be placed so that at least one of your opponent's squares is directly in line between the square you just placed and another of your existing squares. The direction of the line can be horizontal, vertical or diagonal. When you do this, all the squares directly in line between two of your squares are captured and become yours. As there are eight possible directions that these lines can be in, and each line could capture or "flip" more than one square, it is possible to "flip" several squares in one turn.

If you are not too sure of the rules of placement, follow the instructions for loading the game and get the computer to play a game against itself so that you can see what goes on.

When the game starts you will be asked how many players there are to be - 0, 1 or 2. If you enter "0" the computer will play both hands itself. A "1" will allow you to play against the computer and a "2" will let two players use the computer as a playing board.

If the computer is playing one or both hands it will then ask for the strategy level that it is to adopt. There are 6 different strategies ranging from 0 (random legal moves) and 1 (maximum gains per move) through 2,3,4 (varying combinations of gain and positional play) to 5 (pure positional play). (The computer's choice of play is randomly selected from all possible moves, satisfying the best outcome for the particular strategy in use, so the chances of seeing it play the same game twice are very remote).

The only other thing that you have to do is tell the computer which square you want to use. This is done by entering the number of the square in response to the prompt "YOUR MOVE". It sometimes happens that it is not possible to make a legal move - if this does occur then enter "PASS" in response to the prompt. The computer will check your move and if it is not legal, respond with "BAD MOVE" and demand a good move.

If one player PASSes (legally) and there are no moves left for the other player, the game ends before all 64 squares have been used. The game also ends early if one player captures all of his opponent's squares.

While the computer is checking your move, or looking for the best move to make itself, you will see the word "THINKING" flashing at the bottom of the scoring area. This is to let you know the computer is THINKING and has not gone to sleep!

- 0000000000 -

***** LOAN CALCULATION PACKAGE (LII - 16K) - by K.W. Glasson *****

This program computes various figures relating to loans where interest is calculated on a daily reducing, capitalised monthly basis, e.g. building society housing loan.

The formula on which the program is based is R=L/A where:

R = repayments per month
L = loan amount
A = $(1-V^T)/I$
V = $1/(1+I)$
T = term of loan in months
I = interest rate/1200

There are five separate calculations available. They are:

1. Repayment Calculation Given the amount borrowed, interest rate and term of loan in years, it will calculate the monthly repayment and the approximate total interest which would be paid over the full term of the loan. (This section incorporates a facility to include insurance instalments with loan repayments which was applicable to my situation when I wrote the program. If not required, the amounts can be entered as zero or the relevant program lines deleted).

2. Remaining term Calculation Given the current loan balance, interest rate and amount of monthly repayment, this will calculate the length of time remaining until the loan would be paid out (e.g. If you decide to pay an extra $30.00 per month, how long would the loan then run).

3. Remaining Balance Calculation Given the current loan balance, interest rate and amount of monthly repayments this section will calculate the balance remaining after a given period of time.

4. Dissection of repayments Given the amount of the loan, interest rate, monthly repayment and term of loan, this section shows how much of each repayment is interest and how much goes toward reducing the principal. It also calculates total interest to date year by year and shows loan balance (principal) month by month.

5. Repayment Factor Calculation This section calculates a repayment 'factor' for a given interest rate. The factor is the amount of repayment per month per $1,000.00 borrowed, e.g. for interest rate of 12.75% and term of 25 years the factor is 11.09 so the monthly repayment for a loan of say $32,000 over 25 years is 32 * 11.09 or $354.88 (this section was included as a source of factors for a ready-reckoner used at my place of work.)

Being a daily reducing interest calculation, the interest charged will depend on the number of days between repayments, and to provide an acceptable average, I have used 30.4167 as the number of days per month. This is 365/12.

Sections 1, 2, 3 and 4 assign two variables to each figure entered by the user. Calculations are then done using only one of each pair of variables so that having entered your particular figures once in any of the above sections, it is only necessary to hit 'New Line' (Enter) in response to the input statements in any other section if your wish to do further calculations using the same input data.

Each section is clearly identified in the program listing by Heading and underlining, and each is fully self-contained so they can be incorporated as subroutines in another program to suit the requirements of the user.

The program when RUN asks the user to enter figures for Loan amount, Interest rate, etc. and these should be entered without dollar signs, commas, percent symbols etc.

E.g. $32,000.00 - enter as 32000
12.75% - enter as 12.75

Remaining term and remaining balance calculations take a few seconds to arrive at the answer, especially for longer terms - just be patient.

- 0000000000 -

**** UNIT CONVERSION ****
COLOUR COMPUTER

```
10 '*** DEGREE CONVERSIONS ***
20 '*       SILVIO GRECO       *
30 '*     RIVERVIEW  COURT     *
40 '*     MARIBYRNONG VIC.     *
50 '**************************
60 CLS
70  GOSUB 1110
80 REM*** COPYRIGHT(C) 1980
90 CLEAR200
100 CLS:Z=142:PRINT@Z,"MENU":PRI
NT@Z+32,"****":Z=225
110 PRINT@Z,"1. TEMPERATURE  (F,
C)";:PRINT@Z+32,"2. DISTANCE
 (MI,KM)";:PRINT@Z+64,"3. LENGTH
      (CM,IN + CM,FT)";:PRINT@Z
+96,"4. WEIGHT       (ST,KG)";:P
RINT@Z+128,"5. EXIT"
120 Q$= INKEY$:IF Q$="" THEN 120
130 Q= VAL(Q$):IF Q>5 OR Q<1 THE
N120
140 ON Q GOTO 150,350,520,930,12
10
150 CLS
160  PRINT"DEG. F -> DEG. C
  PRESS 'F'";
170 PRINTSTRING$(32,"-")
180 PRINT"DEG. C -> DEG. F
 PRESS 'C'";
190 PRINTSTRING$(32,"-")
200 A$= INKEY$:IF A$="F" THEN270
  ELSE IF A$="C" THEN 220
210 GOTO 200
220 CLS
230 INPUT"INPUT CENTIGRADE VALUE
 ";C
240 F=(9*C)/5+32
250 PRINT@224,C;"DEG. C =";F;"DE
G. F"
260 GOTO 310
270 CLS
280 INPUT"INPUT FAHRENHEIT VALUE
 ";F
290 C=(F-32)*5/9
```

```
300 PRINT@224,F;"DEG. F =";C;"DE
G. C"
310 FOR X=1TO 1000 :NEXT X
320 PRINT@419,"ANY MORE CONVERSI
ONS <Y/N>"
330 S$= INKEY$ :IF S$="Y" THEN 1
50  ELSE IF S$="N" THEN 100
340 GOTO 330
350 CLS:PRINT"MI. -> KM.
     PRESS 'M'";
360 PRINTSTRING$(32,"-")
370 PRINT"KM. -> MI.
 PRESS 'K'";
380 PRINTSTRING$(32,"-")
390 Q$= INKEY$:IF Q$="M" THEN 48
0  ELSE IF Q$ ="K" THEN 410
400 GOTO 390
410 CLS:INPUT"INPUT KILOMETRE VA
LUE ";K
420 M=K*.62137
430 PRINT@224,K;"KM. =";M;"MI."
440 FOR X=1TO1000:NEXT
450  PRINT@419,"ANY MORE CONVERS
IONS <Y/N>"
460 Q$= INKEY$: IF Q$="Y" THEN 3
50  ELSE IF Q$="N" THEN 100
470 GOTO 460
480 CLS:INPUT"INPUT MILE VALUE "
;M
490 K=M*1.609344
500 PRINT@224,M;"MI. =";K;"KM."
510 GOTO440
520 CLS:PRINT"FT./CM. CONVERSION
S    PRESS 'T'";
530 PRINTSTRING$(32,"-")
540 PRINT"IN./CM. CONVERSIONS
 PRESS 'S'";
550 PRINTSTRING$(32,"-")
560 Q$= INKEY$:IF Q$="T" THEN 75
0  ELSE IF Q$="S" THEN 580
570 GOTO 560
580 CLS:PRINT"CM.-> IN.
     PRESS 'C'";
590 PRINTSTRING$(32,"-")
600 PRINT"IN. -> CM.
 PRESS 'I'";
610 PRINTSTRING$(32,"-")
620 Q$= INKEY$:IF Q$="C" THEN 71
0  ELSE IF Q$="I" THEN 640
630 GOTO 620
640 CLS:INPUT"INPUT INCH VALUE "
;I
650 C=I*2.54
660 PRINT@224,I;"IN. =";C;"CM."
670 FOR X=1TO1000: NEXT
680  PRINT@419,"ANY MORE CONVERS
IONS <Y/N>"
690 Q$= INKEY$:IF Q$="Y" THEN 58
0  ELSE IF Q$="N" THEN 100
700 GOTO 690
710 CLS:INPUT"INPUT CENTIMETRE V
ALUE ";C
720 I=C/2.54
730 PRINT@224,C;"CM. =";I;"IN."
740 GOTO 680
750 CLS:PRINT"FT./IN. -> CM.
     PRESS 'F'";
760 PRINTSTRING$(32,"-")
770 PRINT"CM. -> FT./IN.
 PRESS 'M'";
780 PRINTSTRING$(32,"-")
790 Q$= INKEY$:IF Q$="F" THEN 89
0  ELSE IF Q$="M" THEN 810
800 GOTO 790
810 CLS
820 INPUT"INPUT CENTIMETRES VALU
E ";M
830 F=M/30.48
840 PRINT@224,M;"CM. =";INT(F);"
FT. ";INT(((F-INT(F))*120)+.5)/1
0;"IN."
850 FOR X=1TO1000:NEXT
860  PRINT@419,"ANY MORE CONVERS
IONS <Y/N>"
870 Q$= INKEY$: IF Q$="Y"  THEN
750  ELSE IF Q$="N" THEN 100
880 GOTO870
890 CLS:INPUT"INPUT 'FT.,IN.' VA
LUES ";F,M
900 F=F+M/12:M=F*30.48
910 PRINT@224,F;"FT. =";M;"CM."
920 GOTO850
930 CLS:PRINT"ST./LBS. -> KG.
     PRESS 'S'";
940 PRINTSTRING$(32,"-")
950 PRINT"KG. -> ST./LBS.
 PRESS 'K'";
960 PRINTSTRING$(32,"-")
970 Q$= INKEY$:IF Q$="S" THEN 99
0 ELSE IF Q$="K" THEN 1060
980 GOTO 970
990 CLS:INPUT"INPUT 'ST.,LBS.' V
ALUES ";S,K:S=S+K/14:IF NOT(K<14
) THEN 990
1000 K=S*6.3503
1010 PRINT@224,S;"ST. =";K;"KG."
1020 FOR X=1TO  1000:NEXT
1030 PRINT@419,"ANY MORE CONVERS
IONS <Y/N>"
1040 Q$= INKEY$:IF Q$="Y" THEN 9
30  ELSE IF Q$="N" THEN 100
1050 GOTO1040
1060 CLS:INPUT"INPUT KILOGRAMS V
ALUE ";K
1070 S=INT(K*.1575):SL=INT((K*.1
575-S)*14+.5):IF SL>13 THEN S=S+
1:SL=0
1080 PRINT@224,K;"KG. =";S;"ST.
";SL;"LBS."
1090 GOTO 1020
1100 END
1110 PRINT@10,"INSTRUCTIONS":PRI
NT@42,"************"
1120 PRINT"THIS PROGRAM WILL ENA
BLE YOU TO MAKE 4 DIFFERENT CONV
ERSIONS."
1130 PRINT"THE MENU SHOWS YOU TH
E DIFFERENTCONVERSIONS, IF YOU W
ANT A "
1140 PRINT"PARTICULAR CONVERSION
, THEN YOU PRESS THE CORRESPONDI
NG"
1150 PRINT"NUMBER.  THE LETTERS
BESIDE THE CONVERSIONS ARE TO IN
DICATE "
1160 PRINT"CONVERSION TYPE , E.G
.  (F,C)   MEANS FAHRENHEIT AND
CENTIGRADE."
1170 FOR X=1TO 1000:NEXT:PRINT@4
50,"PRESS SPACE BAR TO CONTINUE"
1180 Q$= INKEY$:IF Q$=" " THEN 1
200
1190 GOTO 1180
1200 GOTO 90
1210 CLS:END
```

****** UNIT CONVERSION ******
HITACHI PEACH

```
10 '          *** DEGREE CONVERSIONS ***
20 '          *      SILVIO GRECO      *
30 '          *    RIVERVIEW  COURT    *
40 '          *    MARIBYRNONG VIC.    *
50 '          ***************************
60 CLS
70 GOSUB 1260
80 REM*** COPYRIGHT(C) 1980
90 CLEAR200
100 CLS:LOCATE35,7:PRINT"MENU"
110 LOCATE35,8:PRINTSTRING$(4,"*");
120 LOCATE24,10:PRINT"1. TEMPERATURE  (F
,C)"
```

```
130 LOCATE24,11:PRINT"2. DISTANCE      (M
,K)"
140 LOCATE24,12:PRINT"3. LENGTH        (C
M,IN + CM,FEET)"
150 LOCATE24,13:PRINT"4. WEIGHT        (S
,KG)"
160 LOCATE24,14:PRINT"5. EXIT"
170 Q$=INKEY$:IF Q$=""THEN 170  ELSE 180
180 Q=VAL(Q$):IF Q>5 OR Q<1 THEN170
190 ON Q GOTO 200,420,610,1060,1360
200 CLS
210 LOCATE 7,8:PRINT "FOR FAHRENHEIT TO
CENTIGRADE CONVERSION PRESS 'F'"
220 LOCATE15,7:PRINTSTRING$(49,"-");
230 LOCATE7,10:PRINT"FOR CENTIGRADE TO F
AHRENHEIT CONVERSION PRESS 'C'"
240 LOCATE15,11:PRINTSTRING$(49,"-");
250 A$=INKEY$:IF A$="F" THEN330  ELSE IF
 A$="C" THEN 270
260 GOTO 250
270 CLS
280 LOCATE27,6:PRINT"INPUT CENTIGRADE VA
LUE"
290 PRINT:PRINT TAB(34);:INPUT C
300 F=(9*C)/5+32
310 LOCATE20,10:PRINT C;"CENTIGRADE =";F
;"FAHRENHEIT"
320 GOTO 380
330 CLS
340 LOCATE20,3:PRINT"INPUT FAHRENHEIT VA
LUE"
350 PRINT:PRINT TAB(37);:INPUT F
360 C=(F-32)*5/9
370 LOCATE20,10:PRINTF;"FAHRENHEIT =";C;
"CENTIGRADE"
380 FOR X=1TO 1000 :NEXT X
390 LOCATE25,16:PRINT"ANY MORE CONVERSIO
NS <Y/N>"
400 S$=INKEY$ :IF S$="Y" THEN 200  ELSE
IF S$="N" THEN 100
410 GOTO 400
420 CLS:LOCATE23,6:PRINT"FOR MILES TO KI
LOMETRES PRESS 'M'"
430 LOCATE23,7:PRINTSTRING$(32,"-");
440 LOCATE23,10:PRINT"FOR KILOMETRES TO
MILES PRESS 'K'"
450 LOCATE23,11:PRINTSTRING$(32,"-"),
460 Q$=INKEY$:IF Q$="M" THEN 560  ELSE I
F Q$ ="K" THEN 480
470 GOTO 460
480 CLS:LOCATE27,6:PRINT"INPUT KILOMETRE
 VALUE"
490 PRINT :PRINT TAB(36);:INPUT K
500 M=K*.62137
510 LOCATE25,10:PRINTK;"KILOMETRES =";M;
"MILES"
520 FOR X=1TO1000:NEXT
530 LOCATE25,16:PRINT"ANY MORE CONVERSIO
NS <Y/N>"
540 Q$=INKEY$: IF Q$="Y" THEN 420  ELSE
IF Q$="N" THEN 100
550 GOTO 540
560 CLS:LOCATE29,6:PRINT"INPUT MILE VALU
E"
570 PRINT:PRINT TAB(36);:INPUT M
580 K=M*1.60934
590 LOCATE22,10:PRINTM;"MILES =";K;"KILO
METRES"
600 GOTO520
610 CLS:LOCATE15,6:PRINT"FOR FEET AND CE
NTIMETRE CONVERSIONS PRESS 'T'"
620 LOCATE15,7:PRINTSTRING$(45,"-");
630 LOCATE15,10:PRINT"FOR INCH AND CENTI
METRE CONVERSION PRESS 'S'"
640 LOCATE15,11:PRINTSTRING$(44,"-");
650 Q$=INKEY$:IF Q$="T" THEN 860  ELSE I
F Q$="S" THEN 670
660 GOTO 650
670 CLS:LOCATE23,6:PRINT"FOR CENTIMETRE
TO INCHES PRESS 'C'"
680 LOCATE23,7:PRINTSTRING$(34,"-");
690 LOCATE23,10:PRINT"FOR INCHES TO CENT
IMETRES PRESS 'I'"
700 LOCATE23,11:PRINTSTRING$(34,"-");
710 Q$=INKEY$:IF Q$="C" THEN 810  ELSE I
F Q$="I" THEN 730
720 GOTO 710
730 CLS:LOCATE27,6:PRINT"INPUT INCH VALU
E"
740 PRINT:PRINT TAB(34);:INPUT I
750 C=I*2.54
760 LOCATE20,10:PRINTI;"INCHES =";C;"CEN
TIMETRES"
770 FOR X=1TO1000: NEXT
780 LOCATE23,16:PRINT"ANY MORE CONVERSIO
NS <Y/N>"
790 Q$=INKEY$:IF Q$="Y" THEN 670  ELSE I
F Q$="N" THEN 100
800 GOTO 790
810 CLS:LOCATE27,6:PRINT"INPUT CENTIMETR
E VALUE"
820 PRINT:PRINT TAB(34);:INPUT C
830 I=C/2.54
840 LOCATE20,10:PRINTC;"CENTIMETRES =";I
;"INCHES"
850 GOTO 780
860 CLS:LOCATE23,6:PRINT"FOR FEET TO CEN
TIMETRES PRESS 'F'"
870 LOCATE23,7:PRINTSTRING$(33,"-");
880 LOCATE23,10:PRINT"FOR CENTIMETRES TO
 FEET PRESS 'M'"
890 LOCATE23,11:PRINTSTRING$(33,"-");
900 Q$=INKEY$:IF Q$="F" THEN 1010  ELSE
IF Q$="M" THEN 920
910 GOTO 900
920 CLS
930 LOCATE27,6:PRINT"INPUT CENTIMETRES V
ALUE"
940 PRINT TAB(34);:INPUT M
950 F=M/30.48
960 LOCATE25,10:PRINTM;"CENTIMETRES = ";
INT(F);"FEET ";INT(((F-INT(F))*120)+.5)/
10;"INCHES"
970 FOR X=1TO1000:NEXT
980 LOCATE25,16:PRINT"ANY MORE CONVERSIO
NS <Y/N>"
990 Q$=INKEY$: IF Q$="Y"  THEN 860  ELSE
 IF Q$="N" THEN 100
1000 GOTO990
1010 CLS:LOCATE29,6:PRINT"INPUT FEET ','
 INCH VALUE"
1020 PRINT TAB(34);:INPUT F,M:F=F+M/12:I
F NOT(M<12) THEN 1010
1030 M=F*30.48
1040 LOCATE25,10:PRINTF;"FEET =";M;"CENT
IMETRES"
1050 GOTO970
1060 CLS:LOCATE23,6:PRINT"FOR STONES TO
KILOGRAMS PRESS 'S'"
1070 LOCATE23,7:PRINTSTRING$(33,"-");
1080 LOCATE23,9:PRINT"FOR KILOGRAMS TO S
TONES PRESS 'K'"
1090 LOCATE23,10:PRINTSTRING$(33,"-");
1100 Q$=INKEY$:IF Q$="S" THEN 1120 ELSE
IF Q$="K" THEN 1200
1110 GOTO 1100
1120 CLS:LOCATE29,6:PRINT"INPUT STONE ',
' LBS. VALUE"
1130 PRINT TAB(34);:INPUT S,K:S=S+K/14:I
F NOT(K<14) THEN 1120
1140 K=S*6.3504
1150 LOCATE24,10:PRINTS;"STONES =";K;"KI
LOGRAMS"
1160 FOR X=1TO  1000:NEXT
1170 LOCATE25,16:PRINT"ANY MORE CONVERSI
ONS <Y/N>"
1180 Q$=INKEY$:IF Q$="Y" THEN 1060  ELSE
 IF Q$="N" THEN 100
1190 GOTO1180
1200 CLS:LOCATE27,6:PRINT"INPUT KILOGRAM
S VALUE"
1210 PRINTTAB(34);:INPUT K
1220 S=INT(K*.1575):SL=INT((K*.1575-S)*1
4+.5):IF SL>13 THEN S=S+1:SL=0
```

```
1230 LOCATE23,10:PRINTK;"KILOGRAMS = ";S
;"STONE  ";SL;"LBS."
1240 GOTO 1160
1250 END
1260 LOCATE32,5:PRINT"INSTRUCTIONS":LOCA
TE32,6:PRINT"************"
1270 LOCATE9,8:PRINT"THIS PROGRAM WILL E
NABLE YOU TO MAKE 4 DIFFERENT CONVERSION
S."
1280 LOCATE9,9:PRINT"THE MENU SHOWS YOU
THE DIFFERENT CONVERSIONS, IF YOU WANT A
 "
1290 LOCATE9,10:PRINT"PARTICULAR CONVERS
ION, THEN YOU PRESS THE CORRESPONDING"
1300 LOCATE9,11:PRINT"NUMBER.  THE LETTE
RS BESIDE THE CONVERSIONS ARE TO INDICAT
E "
1310 LOCATE9,12:PRINT"WHAT TYPE OF CONVE
RSION, EG.(F,C) MEANS FAHRENHEIT AND CEN
TIGRADE."
1320 FOR X=1TO 1000:NEXT:LOCATE25,17:PRI
NT"PRESS SPACE BAR TO CONTINUE"
1330 Q$=INKEY$:IF Q$=" " THEN 1350
1340 GOTO 1330
1350 GOTO 90
1360 CLS:END
```

****** NORMAL DISTRIBUTION ******
COLOUR COMPUTER

```
10 'ROUTINE TO CALCULATE STASTIC
S IN RELATION TO THE NORMAL   DI
STRIBUTION.
20 '(C) COPYRIGHT 1980,
30 'TERRY JONES,
40 '43 HASTIE ST.,
50 'TATURA,     3616.
60 CLS:DIMX(100),F(10)
70 GOSUB260
80 PRINT"  NORMAL DISTRIBUTION A
NALYSIS":PRINT"  ";STRING$(28,45
)
90 PRINT:PRINT:PRINT TAB(11)"ENT
ER DATA":PRINT:PRINT"NUMBER OF D
ATA POINTS - MAX.=100"
100 PRINT@238,STRING$(32," "):PR
INT@238,"*";:INPUTND:IF ND<2 OR
ND>100 THEN 100
110 PRINT:FORI=1TOND:PRINT TAB(1
0)"X(";I;") =";:INPUTX(I):NEXTI
120 CLS:K=0:PRINT TAB(10)"DATA E
NTERED"
130 FOR I=1 TO ND:K=K+1:PRINT TA
B(10)"X(";I;") =";X(I)
140 IF K<10 THEN 170
150 PRINT@418,"PRESS <ENTER> TO
CONTINUE...";
160 A$=INKEY$:IF A$="" THEN 160
ELSE K=0:CLS
170 NEXTI
180 PRINT@490,"EDIT DATA (Y/N)";
190 I$= INKEY$:IFI$=""THEN190ELS
E IFI$="Y"THEN200ELSE IFI$="N"TH
EN210ELSE190
200 CLS:INPUT"DATA POINT TO BE E
DITED";I:IFI>NDTHEN140ELSE PRINT
"X(";I;") =";X(I):PRINT TAB(32)"
NEW X(";I;") =";:INPUTX(I):GOTO1
20
210 CLS:PRINT@233,"COMPUTING ...
."
220 X1=0:X2=0:FORI=1TOND:X1=X1+X
(I):X2=X2+(X(I)^2):NEXTI:X3=X1/N
D:X4=X2-(X1^2/ND):X5=X4/(ND-1):X
6= SQR(X5):X7=X5/ND
230 R1=X(1):R2=X(1):FORI=1TOND:I
FX(I)<R1THENR1=X(I):NEXTIELSE IF
X(I)>R2THENR2=X(I):NEXTIELSE NEX
TI
240 CLS:PRINT:PRINT"MEAN =";X3:P
RINT:PRINT"VARIANCE =";X5:PRINT:
PRINT"STANDARD DEVIATION =";X6:P
RINT:PRINT"STANDARD ERROR =";X7:
PRINT:PRINT"RANGE =";R1;"TO";R2:
PRINT
250  END
260 PMODE1,1:PCLS:SCREEN1,1
270 M=128:B=150:S=120
280 FOR I=0 TO 120
290 V=-(I/60)*(I/60):Y=S*EXP(V)
300 PSET(M+I,B-Y,0):PSET(M-I,B-Y
,0):PSET(M+I,B,0):PSET(M-I,B,0)
310 NEXTI
320 LINE (M,B-S)-(M,B),PSET
330 FOR I=1 TO 2000:NEXTI
340 RETURN
```

****** NORMAL DISTRIBUTION ******
HITACHI PEACH

```
10 'ROUTINE TO CALCULATE STATISTICS IN R
ELATION TO THE NORMAL DISTRIBUTION.
20 '(C) COPYRIGHT 1980,
30 'TERRY JONES,
40 '43 HASTIE ST.,
50 'TATURA,     3616.
60 CLS:DEFINTI-N:DIMX(100),F(10)
65 GOSUB310
70 PRINTTAB(26)"NORMAL DISTRIBUTION ANAL
YSIS":   PRINTTAB(26)STRING$(28,45)
80 PRINT:PRINTTAB(34)"ENTER DATA":PRINT:
PRINTTAB(21)  "NUMBER OF DATA POINTS - M
AXIMUM IS 100"
90 PRINTTAB(36)"*";:INPUTND:IF ND<2 ORND
>100 THEN PRINT CHR$(26);:GOTO90
100 PRINT:FORI=1TOND:PRINTTAB(35)"X(";I;
") =";:INPUTX(I):NEXTI
110 CLS:K=0:PRINTTAB(33)"DATA ENTERED"
112 FORI=1TOND:K=K+1
114 PRINTTAB(34)"X(";I;") =";X(I):IF K>9
 THEN LOCATE24,18:PRINT"TYPE <ENTER> TO
CONTINUE ...";:   INPUTI$:K=0:CLS
116 NEXTI
120 LOCATE30,18:PRINT"EDIT DATA (Y/N)";
130 I$=INKEY$:IFI$=""THEN130ELSEIFI$="Y"
THEN140ELSEIFI$="N"   THEN150ELSE130
140 CLS:INPUT"DATA POINT TO BE EDITED";I
:   IFI>NDTHEN140ELSEPRINT"X(";I;") =";X
(I):   PRINTTAB(40)CHR$(27);"NEW X(";I;"
) =";:INPUTX(I):   GOTO110
150 CLS:LOCATE32,10:PRINT"COMPUTING ....
"
160 X1=0:X2=0:FORI=1TOND:X1=X1+X(I):X2=X
2+(X(I)^2):NEXTI:   X3=X1/ND:X4=X2-(X1^2
/ND):X5=X4/(ND-1):X6=SQR(X5):X7=X5/ND
170 R1=X(1):R2=X(1)
172 FOR I=1 TO ND
174 IF X(I)<R1 THEN R1=X(I)
176 IF X(I)>R2 THEN R2=X(I)
178 NEXTI
180 CLS:PRINT:PRINT"MEAN =";X3:PRINT:
PRINT"VARIANCE =";X5:PRINT:   PRINT"STAN
DARD DEVIATION =";X6:PRINT:   PRINT"STAN
DARD ERROR =";X7:PRINT:   PRINT"RANGE ="
;R1;"TO";R2:PRINT
190 END
310 'NORMAL DISTRIBUTION SUBROUTINE
320 CLS
330 MD=320:IB=180:KS=120
340 FOR I=0 TO 300
345 V=-(I/150)*(I/150):IY=KS*EXP(V)
350 PSET(MD+I,IB-IY):PSET(MD-I,IB-IY):PS
ET(MD+I,IB):PSET(MD-I,IB)
360 NEXTI
370 LINE (MD,IB-KS)-(MD,IB),PSET
375 LOCATE 0,24
380 FOR I=1TO500:NEXTI:PRINT
390 RETURN
```

```
                00010 ; RONALD J. SULLY  &  MICRO-80 PRESENTS:
                00020 ;      **   MICRO GRAND PRIX    **
                00030 ;          (C)OPYRIGHT OCT 1981
                00040 ;             RONALD J. SULLY
                00050 ;             1 PACKHAM PLACE
                00060 ;           CHARNWOOD, ACT 2615
                00070 ;           PHONE (062) 582917
                00080 ;
3C00            00090 VIDEO   EQU     3C00H
7000            00100         ORG     7000H
7000 CDC901     00110 START   CALL    1C9H            ;CLS
7003 CDC304     00120         CALL    04C3H           ;CHANGE TO 64 CHAR/LINE
7006 11E33F     00130         LD      DE,VIDEO+995
7009 ED53FC74   00140         LD      (CARLOC),DE     ;START POSN OF CAR
700D 112E3C     00150         LD      DE,VIDEO+46
7010 ED530875   00160         LD      (POSN),DE       ;START POSN OF ROAD
7014 110012     00170         LD      DE,1200H
7017 ED530675   00180         LD      (SPEED),DE      ;SET INITIAL SPEED
701B 110000     00190         LD      DE,0
701E ED530475   00200         LD      (YORSCR),DE     ;SET SCORE TO 0
                00210 ;
7022 CD1F72     00220         CALL    FRAM1           ;DRAW INTRO TO GAME
7025 CD4900     00230         CALL    49H             ;SCAN KEYBOARD
7028 FE42       00240         CP      66              ;IS IT "B"?
702A CA6900     00250         JP      Z,69H           ;YES - GOTO BASIC
                00260 ;
702D 212800     00270 SCROL   LD      HL,40
7030 CD4671     00280         CALL    RND             ;GET RND(40)
7033 211D74     00290         LD      HL,MSGE13
7036 19         00300         ADD     HL,DE           ;DRAW THE VERGE
7037 11003C     00310         LD      DE,VIDEO        ;TO THE SCREEN
703A 014000     00320         LD      BC,64
703D EDB0       00330         LDIR
                00340 ;
703F 210200     00350         LD      HL,2
7042 CD4671     00360         CALL    RND             ;GET RND(2)
7045 7B         00370         LD      A,E
7046 ED5B0875   00380         LD      DE,(POSN)       ;USED TO DETERMINE IF
704A FE01       00390         CP      1               ;ROAD BENDS LEFT
704C 2803       00400         JR      Z,LEFT1         ;OR RIGHT
704E 13         00410         INC     DE
704F 1801       00420         JR      PRINT
7051 1B         00430 LEFT1   DEC     DE
                00440 ;
7052 212C3C     00450 PRINT   LD      HL,VIDEO+44
7055 CD390A     00460         CALL    0A39H
7058 FE01       00470         CP      1
705A 2801       00480         JR      Z,LESSTH
705C 1B         00490         DEC     DE
705D 21013C     00500 LESSTH  LD      HL,VIDEO+1
7060 CD390A     00510         CALL    0A39H
7063 FEFF       00520         CP      0FFH
7065 2801       00530         JR      Z,OK
7067 13         00540         INC     DE
7068 ED530875   00550 OK      LD      (POSN),DE       ;THIS ROUTINE DRAWS
706C 210A75     00560         LD      HL,ROAD         ;THE RACE TRACK
706F 011200     00570         LD      BC,18
7072 EDB0       00580         LDIR
                00590 ;
7074 210A00     00600         LD      HL,10
7077 CD4671     00610         CALL    RND             ;GET RND(10)
707A 7B         00620         LD      A,E
707B FE01       00630         CP      1               ;IF RND(10)=1
707D CC0B72     00640         CALL    Z,OBST          ;THEN GOSUB OBST
                00650 ;
7080 21BF3F     00660         LD      HL,VIDEO+959
7083 11FF3F     00670         LD      DE,VIDEO+1023   ;SCROLL THE SCREEN
7086 01C003     00680         LD      BC,960          ;FROM BOTTOM TO TOP
7089 EDB8       00690         LDDR
                00700 ;
708B ED5B0675   00710         LD      DE,(SPEED)
708F 1B         00720         DEC     DE              ;INCREASE SPEED OF GAME
7090 1B         00730         DEC     DE
7091 ED530675   00740         LD      (SPEED),DE
                00750 ;
7095 2A0475     00760         LD      HL,(YORSCR)
7098 23         00770         INC     HL              ;INCREMENT SCORE BY 1
7099 220475     00780         LD      (YORSCR),HL
                00790 ;
709C 3A2038     00800 FNCTN   LD      A,(3820H)       ;SCAN KEYBOARD
```

```
709F FE40      00810          CP      40H             ;IS KEY ">"?
70A1 200B      00820          JR      NZ,LEFT         ;NO - GOTO LEFT
70A3 ED5BFC74  00830          LD      DE,(CARLOC)     ;YES - SO GET OLD POSN
70A7 13        00840          INC     DE              ;CALC NEW POSN
70A8 ED53FC74  00850          LD      (CARLOC),DE     ;AND SAVE
70AC 1832      00860          JR      NOMOV           ;GOTO NOMOV
70AE FE10      00870 LEFT     CP      10H             ;IS KEY "<"?
70B0 200B      00880          JR      NZ,SPDCTL       ;NO - GOTO SPDCTL
70B2 ED5BFC74  00890          LD      DE,(CARLOC)     ;YES - SO GET OLD POSN
70B6 1B        00900          DEC     DE              ;CALC NEW POSN
70B7 ED53FC74  00910          LD      (CARLOC),DE     ;AND SAVE
70BB 1823      00920          JR      NOMOV           ;GOTO NOMOV
70BD 3A4038    00930 SPDCTL   LD      A,(3840H)       ;SCAN KEYBOARD
70C0 FE08      00940          CP      8               ;IS KEY "["?
70C2 200E      00950          JR      NZ,BRAKE        ;NO - GOTO BRAKE
70C4 2A0675    00960          LD      HL,(SPEED)      ;YES - GET OLD SPEED
70C7 B7        00970          OR      A
70C8 113200    00980          LD      DE,50
70CB ED52      00990          SBC     HL,DE           ;CALC NEW SPEED
70CD 220675    01000          LD      (SPEED),HL      ;AND SAVE
70D0 180E      01010          JR      NOMOV           ;GOTO NOMOV
70D2 FE10      01020 BRAKE    CP      10H             ;IS KEY DOWN ARROW?
70D4 200A      01030          JR      NZ,NOMOV        ;NO - GOTO NOMOV
70D6 2A0675    01040          LD      HL,(SPEED)      ;YES - SO GET OLD SPEED
70D9 115A00    01050          LD      DE,90
70DC 19        01060          ADD     HL,DE           ;CALC NEW SPEED
70DD 220675    01070          LD      (SPEED),HL      ;AND SAVE
               01080 ;
70E0 ED5BFC74  01090 NOMOV    LD      DE,(CARLOC)     ;GET OLD POSN
70E4 D5        01100          PUSH    DE              ;SAVE IT TEMORARILY
               01110 ;FOLLOWING ROUTINE CHECKS IF ROAD IS CLEAR
70E5 1A        01120 CHCK     LD      A,(DE)
70E6 FE20      01130          CP      32              ;IS A SPACE
70E8 2807      01140          JR      Z,CHCK2         ;YES - GOTO CHCK2
70EA 3E01      01150          LD      A,1             ;NO -
70EC 321C75    01160          LD      (FLAG),A        ;SO SET FLAG
70EF 180B      01170          JR      CARMOV          ;GOTO CARMOV
70F1 13        01180 CHCK2    INC     DE
70F2 1A        01190          LD      A,(DE)
70F3 FE20      01200          CP      32              ;IS IT A SPACE
70F5 2805      01210          JR      Z,CARMOV        ;YES - GOTO CARMOV
70F7 3E01      01220          LD      A,1             ;NO -
70F9 321C75    01230          LD      (FLAG),A        ;SO SET FLAG
70FC D1        01240 CARMOV   POP     DE              ;GET CAR POSN AGAIN
               01250 ;THIS WRITES SCORE TO BOTTOM RH OF SCREEN
70FD CD9D0A    01260          CALL    0A9DH
7100 210475    01270          LD      HL,YORSCR
7103 CDB109    01280          CALL    09B1H
7106 CDBD0F    01290          CALL    0FBDH
7109 AF        01300          XOR     A
710A 329C40    01310          LD      (409CH),A
710D 11F93F    01320          LD      DE,VIDEO+1017
7110 ED532040  01330          LD      (4020H),DE
7114 CDA728    01340          CALL    28A7H           ;WRITE IT!
               01350 ;
7117 21F674    01360          LD      HL,CAR          ;DRAW CAR TO SCREEN
711A ED5BFC74  01370          LD      DE,(CARLOC)
711E 010200    01380          LD      BC,2
7121 EDB0      01390          LDIR
               01400 ;
7123 3A1C75    01410          LD      A,(FLAG)
7126 FE01      01420          CP      1               ;DID CAR CRASH?
7128 CA5071    01430          JP      Z,CRASH         ;YES - GOTO CRASH
712B 2A0675    01440          LD      HL,(SPEED)      ;NO -
712E 111400    01450          LD      DE,20           ;SO CHECK SPEED IS
7131 CD390A    01460          CALL    0A39H           ;WITHIN LIMITS
7134 FE01      01470          CP      1               ;IF SO -
7136 2804      01480          JR      Z,DELAY         ;THEN GOTO DELAY
7138 ED530675  01490          LD      (SPEED),DE      ;IF NOT - THEN RESET SPEED
713C ED4B0675  01500 DELAY    LD      BC,(SPEED)      ;THIS IS A GEN PURPOSE
7140 CD6000    01510          CALL    60H             ;TIMING LOOP
7143 C32D70    01520          JP      SCROL           ;GOTO SCROL - GAME CONT.
               01530 ;
7146 CD9A0A    01540 RND      CALL    0A9AH           ;THIS ROUTINE GETS OUR
7149 CDC914    01550          CALL    14C9H           ;RND NUMBER.  ON ENTRY
714C CD052B    01560          CALL    2B05H           ;HL= MAX VALUE - ON EXIT
714F C9        01570          RET                     ;DE = RND(HL)
               01580 ;THIS ROUTINE SIMULATES THE CRASH
7150 0632      01590 CRASH    LD      B,50            ;DO IT 50 TIMES
7152 D9        01600 LOOP5    EXX                     ;USE PRIMED REG SET
7153 21FA74    01610          LD      HL,CAR2
```

```
7156 ED5BFC74 01620          LD      DE,(CARLOC)      ;DRAW CAR2 IN CAR POSN
715A 010200   01630          LD      BC,2
715D EDB0     01640          LDIR
715F 01D007   01650          LD      BC,2000          ;PAUSE
7162 CD6000   01660          CALL    60H
7165 21F674   01670          LD      HL,CAR
7168 ED5BFC74 01680          LD      DE,(CARLOC)      ;DRAW CAR AGAIN
716C 010200   01690          LD      BC,2
716F EDB0     01700          LDIR
7171 01D007   01710          LD      BC,2000          ;PAUSE
7174 CD6000   01720          CALL    60H
7177 D9       01730          EXX                      ;CHANGE REG BACK
7178 10D8     01740          DJNZ    LOOP5            ;IF B<>0 GOTO LOOP5
              01750 ;
717A 01FFFF   01760          LD      BC,-1            ;PAUSE A BIT LONGER
717D CD6000   01770          CALL    60H
7180 AF       01780          XOR     A
7181 321C75   01790          LD      (FLAG),A         ;RESET CRASH FLAG
7184 CDC901   01800          CALL    1C9H             ;CLS
7187 CDF604   01810          CALL    04F6H            ;CHANGE TO 32 CHAR/LINE
              01820 ;
718A 218574   01830          LD      HL,MSGE14
718D 11CA3C   01840          LD      DE,VIDEO+202     ;WRITE MSGE TO SCRN
7190 011800   01850          LD      BC,24
7193 EDB0     01860          LDIR
              01870 ;THIS ROUTINE WRITES YOUR SCORE TO SCREEN
7195 CD9D0A   01880          CALL    0A9DH
7198 210475   01890          LD      HL,YORSCR
719B CDB109   01900          CALL    09B1H
719E CDBD0F   01910          CALL    0FBDH
71A1 AF       01920          XOR     A
71A2 329C40   01930          LD      (409CH),A
71A5 11E43C   01940          LD      DE,VIDEO+228
71A8 ED532040 01950          LD      (4020H),DE
71AC CDA728   01960          CALL    28A7H
              01970 ;THIS ROUTINE COMPARES ALL THE SCORES AND SORTS THEM
71AF 2A0475   01980 SCRCMP   LD      HL,(YORSCR)
71B2 ED5BFE74 01990          LD      DE,(CHSCOR)
71B6 CD390A   02000          CALL    0A39H
71B9 FEFF     02010          CP      0FFH
71BB 2809     02020          JR      Z,NEXT3
71BD 22FE74   02030          LD      (CHSCOR),HL
71C0 EB       02040          EX      DE,HL
71C1 3E01     02050          LD      A,1
71C3 32F574   02060          LD      (MSG),A
71C6 ED5B0075 02070 NEXT3    LD      DE,(SECSCR)
71CA CD390A   02080          CALL    0A39H
71CD FEFF     02090          CP      0FFH
71CF 2804     02100          JR      Z,NEXT4
71D1 220075   02110          LD      (SECSCR),HL
71D4 EB       02120          EX      DE,HL
71D5 ED5B0275 02130 NEXT4    LD      DE,(THISCR)
71D9 CD390A   02140          CALL    0A39H
71DC FEFF     02150          CP      0FFH
71DE CAE471   02160          JP      Z,RESULT
71E1 220275   02170          LD      (THISCR),HL
71E4 3AF574   02180 RESULT   LD      A,(MSG)
71E7 FE01     02190          CP      1
71E9 200B     02200          JR      NZ,NOMSGE
71EB 21CE74   02210          LD      HL,MSGE16
71EE 11883D   02220          LD      DE,VIDEO+392
71F1 012800   02230          LD      BC,40
71F4 EDB0     02240          LDIR
71F6 219D74   02250 NOMSGE   LD      HL,MSGE15
71F9 11C23E   02260          LD      DE,VIDEO+706
71FC 013200   02270          LD      BC,50
71FF EDB0     02280          LDIR
7201 AF       02290          XOR     A
7202 32F574   02300          LD      (MSG),A
7205 CD4900   02310          CALL    049H     ;SCAN KEYBOARD FOR PRESSED KEY
7208 C30070   02320          JP      START    ;IF SO GOTO START
              02330 ;THIS ROUTINE DRAWS OTHER CARS ON THE ROAD
720B 210F00   02340 OBST     LD      HL,15
720E CD4671   02350          CALL    RND              ;AT RND POSN
7211 2A0875   02360          LD      HL,(POSN)
7214 19       02370          ADD     HL,DE
7215 EB       02380          EX      DE,HL
7216 21F874   02390          LD      HL,CAR1
7219 010200   02400          LD      BC,2
721C EDB0     02410          LDIR
721E C9       02420          RET                      ;RETURN FROM THIS ROUTINE
              02430 ;
```

```
                02440 ;THE FOLLOWING ROUTINE DRAWS THE INTRODUCTION FRAME
                02450 ;ON THE SCREEN
721F 212F73     02460 FRAM1   LD      HL,MSGE1
7222 11043C     02470         LD      DE,VIDEO+4
7225 011000     02480         LD      BC,16
7228 EDB0       02490         LDIR
722A 213F73     02500         LD      HL,MSGE2
722D 11443C     02510         LD      DE,VIDEO+68
7230 010C00     02520         LD      BC,12
7233 EDB0       02530         LDIR
7235 214B73     02540         LD      HL,MSGE3
7238 11C03C     02550         LD      DE,VIDEO+192
723B 010E00     02560         LD      BC,14
723E EDB0       02570         LDIR
7240 215973     02580         LD      HL,MSGE4
7243 11013D     02590         LD      DE,VIDEO+257
7246 011400     02600         LD      BC,20
7249 EDB0       02610         LDIR
724B 216D73     02620         LD      HL,MSGE5
724E 11413D     02630         LD      DE,VIDEO+321
7251 011400     02640         LD      BC,20
7254 EDB0       02650         LDIR
7256 218173     02660         LD      HL,MSGE6
7259 11813D     02670         LD      DE,VIDEO+385
725C 010F00     02680         LD      BC,15
725F EDB0       02690         LDIR
7261 219073     02700         LD      HL,MSGE7
7264 11C13D     02710         LD      DE,VIDEO+449
7267 011000     02720         LD      BC,16
726A EDB0       02730         LDIR
726C 21A073     02740         LD      HL,MSGE8
726F 11413E     02750         LD      DE,VIDEO+577
7272 011C00     02760         LD      BC,28
7275 EDB0       02770         LDIR
7277 21BC73     02780         LD      HL,MSGE9
727A 11813E     02790         LD      DE,VIDEO+641
727D 011C00     02800         LD      BC,28
7280 EDB0       02810         LDIR
7282 21D873     02820         LD      HL,MSGE10
7285 11013F     02830         LD      DE,VIDEO+769
7288 011600     02840         LD      BC,22
728B EDB0       02850         LDIR
728D 21EF73     02860         LD      HL,MSGE11
7290 11413F     02870         LD      DE,VIDEO+833
7293 011600     02880         LD      BC,22
7296 EDB0       02890         LDIR
7298 210674     02900         LD      HL,MSGE12
729B 11813F     02910         LD      DE,VIDEO+897
729E 011600     02920         LD      BC,22
72A1 EDB0       02930         LDIR
72A3 CD9D0A     02940         CALL    0A9DH
72A6 21FE74     02950         LD      HL,CHSCOR
72A9 CDB109     02960         CALL    09B1H
72AC CDBD0F     02970         CALL    0FBDH
72AF AF         02980         XOR     A
72B0 329C40     02990         LD      (409CH),A
72B3 11123F     03000         LD      DE,VIDEO+786
72B6 ED532040   03010         LD      (4020H),DE
72BA CDA728     03020         CALL    28A7H
72BD CD9D0A     03030         CALL    0A9DH
72C0 210075     03040         LD      HL,SECSCR
72C3 CDB109     03050         CALL    09B1H
72C6 CDBD0F     03060         CALL    0FBDH
72C9 AF         03070         XOR     A
72CA 329C40     03080         LD      (409CH),A
72CD 11523F     03090         LD      DE,VIDEO+850
72D0 ED532040   03100         LD      (4020H),DE
72D4 CDA728     03110         CALL    28A7H
72D7 CD9D0A     03120         CALL    0A9DH
72DA 210275     03130         LD      HL,THISCR
72DD CDB109     03140         CALL    09B1H
72E0 CDBD0F     03150         CALL    0FBDH
72E3 AF         03160         XOR     A
72E4 329C40     03170         LD      (409CH),A
72E7 11923F     03180         LD      DE,VIDEO+914
72EA ED532040   03190         LD      (4020H),DE
72EE CDA728     03200         CALL    28A7H
72F1 0610       03210         LD      B,16
72F3 213F3C     03220         LD      HL,VIDEO+63
```

```
72F6 114000    03230          LD      DE,64
72F9 36BF      03240 LOOP1    LD      (HL),191
72FB 19        03250          ADD     HL,DE
72FC 10FB      03260          DJNZ    LOOP1
72FE 0605      03270          LD      B,5
7300 21F03E    03280          LD      HL,VIDEO+752
7303 36BF      03290 LOOP2    LD      (HL),191
7305 19        03300          ADD     HL,DE
7306 10FB      03310          DJNZ    LOOP2
7308 060D      03320          LD      B,13
730A 212E3C    03330          LD      HL,VIDEO+46
730D 113F00    03340          LD      DE,63
7310 3696      03350 LOOP3    LD      (HL),150
7312 19        03360          ADD     HL,DE
7313 10FB      03370          DJNZ    LOOP3
7315 0603      03380          LD      B,3
7317 21613F    03390          LD      HL,VIDEO+865
731A 114000    03400          LD      DE,64
731D 36BF      03410 LOOP4    LD      (HL),191
731F 19        03420          ADD     HL,DE
7320 10FB      03430          DJNZ    LOOP4
7322 21F674    03440          LD      HL,CAR
7325 ED5BFC74  03450          LD      DE,(CARLOC)
7329 010200    03460          LD      BC,2
732C EDB0      03470          LDIR
732E C9        03480          RET
               03490 ;FOLLOWING IS THE LIST OF ALL THE VARIABLES
732F 4D        03500 MSGE1    DEFM    'MICRO GRAND PRIX'
733F 42        03510 MSGE2    DEFM    'BY RON SULLY'
734B 46        03520 MSGE3    DEFM    'FUNCTION KEYS:'
7359 5B        03530 MSGE4    DEFM    '[  =  INCREASE SPEED'
736D 5C        03540 MSGE5    DEFB    5CH
736E 20        03550          DEFM    '  =  DECREASE SPEED'
7381 3C        03560 MSGE6    DEFM    '<  =  MOVE LEFT'
7390 3E        03570 MSGE7    DEFM    '>  =  MOVE RIGHT'
73A0 50        03580 MSGE8    DEFM    'PRESS "B" TO RETURN TO BASIC'
73BC 50        03590 MSGE9    DEFM    'PRESS ANY OTHER KEY TO START'
73D8 43        03600 MSGE10   DEFM    'CHAMPION SCORE  =       '
73EF 32        03610 MSGE11   DEFM    '2ND BEST SCORE  =       '
7406 33        03620 MSGE12   DEFM    '3RD BEST SCORE  =       '
741D 20        03630 MSGE13   DEFM    ' MICRO-80 ###WW#WW#WWWW#WW###WW###WW##WWWW###
WWW# MICRO-80 ###WWW#WW#WWW#WWW#W######W#####WWWWWW# MICRO-80 '
7485 59        03640 MSGE14   DEFM    'Y O U R   S C O R E   = '
749D 50        03650 MSGE15   DEFM    'P R E S S   A N Y   K E Y   T O   C O N T I
 N U E'
74CE 59        03660 MSGE16   DEFM    'Y O U   A R E   T H E   C H A M P I O N'
74F5 00        03670 MSG      DEFB    0
74F6 B9B6      03680 CAR      DEFW    0B6B9H
74F8 99A6      03690 CAR1     DEFW    0A699H
74FA B6B9      03700 CAR2     DEFW    0B9B6H
74FC E33F      03710 CARLOC   DEFW    VIDEO+995
74FE 0000      03720 CHSCOR   DEFW    0
7500 0000      03730 SECSCR   DEFW    0
7502 0000      03740 THISCR   DEFW    0
7504 0000      03750 YORSCR   DEFW    0
7506 0000      03760 SPEED    DEFW    0
7508 2E3C      03770 POSN     DEFW    VIDEO+46
750A 85        03780 ROAD     DEFB    85H
750B 20        03790          DEFM    '
751B 8A        03800          DEFB    8AH
751C 00        03810 FLAG     DEFB    0
7000           03820          END     START
```

```
               00010 ;****************************
               00020 ;*         PASSWORD         *
               00030 ;*                          *
               00040 ;* BY  - ANTHONY PARK       *
               00050 ;*    MIDDLE COVE , N.S.W   *
               00060 ;*         MAY 1982         *
               00070 ;*                          *
               00080 ;* USE 'LSET' TO ACCESS     *
               00090 ;****************************
               00100 ;
7F2B           00110          ORG     7F2BH
06CC           00120 BASIC    EQU     06CCH
4020           00130 CURPOS   EQU     4020H
03E3           00140 KBDRV    EQU     03E3H
4016           00150 KBLOC    EQU     4016H
4198           00160 LSET     EQU     4198H
```

```
40B1            00170 TOPMEM  EQU     40B1H
1B6E            00180 SETPTR  EQU     1B6EH
40A0            00190 STACK   EQU     40A0H
                00200 ;
                00210 ;       ENTRY POINT HERE
                00220 ;
7F2B 21BC7F     00230 INIT    LD      HL,MESG1
7F2E CDAC7F     00240         CALL    PRINT
7F31 21803C     00250         LD      HL,3C80H
7F34 222040     00260         LD      (CURPOS),HL
7F37 21567F     00270         LD      HL,BEGIN
7F3A 229841     00280         LD      (LSET),HL
7F3D 2B         00290         DEC     HL
7F3E 2B         00300         DEC     HL
7F3F 22B140     00310         LD      (TOPMEM),HL     ;SET MEMORY SIZE
7F42 113200     00320         LD      DE,32H
7F45 B7         00330         OR      A
7F46 ED52       00340         SBC     HL,DE
7F48 22A040     00350         LD      (STACK),HL      ;ADJUST STACK
7F4B 3EC9       00360         LD      A,0C9H
7F4D 32E241     00370         LD      (41E2H),A       ;RESET SYSTEM VECTOR
7F50 CD6E1B     00380         CALL    SETPTR
7F53 C3CC06     00390         JP      BASIC
                00400 ;
                00410 ;       MAIN PROGRAM
                00420 ;
7F56 D9         00430 BEGIN   EXX
7F57 21DE7F     00440         LD      HL,MESG2
7F5A CDAC7F     00450         CALL    PRINT
7F5D 21577F     00460         LD      HL,BEGIN+1      ;IF RESET PUSHED
7F60 221640     00470         LD      (KBLOC),HL      ;THEN IGNORE IT
                00480 ;
                00490 ;       TEST PASSWORD
                00500 ;
7F63 CDE303     00510 LOOP    CALL    KBDRV
7F66 DD21B87F   00520         LD      IX,PWD1
7F6A DDBE00     00530         CP      (IX)
7F6D 20F4       00540         JR      NZ,LOOP
7F6F CDE303     00550 TEST1   CALL    KBDRV
7F72 FE00       00560         CP      0
7F74 28F9       00570         JR      Z,TEST1
7F76 DD21B97F   00580         LD      IX,PWD2
7F7A DDBE00     00590         CP      (IX)
7F7D 20E4       00600         JR      NZ,LOOP
7F7F CDE303     00610 TEST2   CALL    KBDRV
7F82 FE00       00620         CP      0
7F84 28F9       00630         JR      Z,TEST2
7F86 DD21BA7F   00640         LD      IX,PWD3
7F8A DDBE00     00650         CP      (IX)
7F8D 20D4       00660         JR      NZ,LOOP
7F8F CDE303     00670 TEST3   CALL    KBDRV
7F92 FE00       00680         CP      0
7F94 28F9       00690         JR      Z,TEST3
7F96 DD21BB7F   00700         LD      IX,PWD4
7F9A DDBE00     00710         CP      (IX)
7F9D 20C4       00720         JR      NZ,LOOP
7F9F CDC901     00730         CALL    1C9H    ;PASSWORD O.K.
7FA2 21E303     00740         LD      HL,KBDRV
7FA5 221640     00750         LD      (KBLOC),HL
7FA8 D9         00760         EXX
7FA9 C3CC06     00770         JP      BASIC
7FAC CDC901     00780 PRINT   CALL    1C9H
7FAF 11103C     00790         LD      DE,3C10H
7FB2 012200     00800         LD      BC,34
7FB5 EDB0       00810         LDIR
7FB7 C9         00820         RET
                00830 ;
                00840 ;DEFAULT PASSWORD SET HERE IN DEFB STATEMENTS
                00850 ;
7FB8 54         00860 PWD1    DEFB    54H     ;T=54H
7FB9 45         00870 PWD2    DEFB    45H     ;E
7FBA 53         00880 PWD3    DEFB    53H     ;S
7FBB 54         00890 PWD4    DEFB    54H     ;T      PASSWORD = "TEST"
                00900 ;
                00910 ;MESSAGES TO BE PRINTED
                00920 ;
7FBC 2A         00930 MESG1   DEFM    '** PASSWORD INITIALIZED **         '
7FDE 2A         00940 MESG2   DEFM    '** ** TERMINAL ON STAND-BY ** **   '
41E2            00950         ORG     41E2H   ;RUN PROGRAM
41E2 C32B7F     00960         JP      INIT
7F2B            00970         END     INIT
```

```
10 '** SIMPLE PROGRAM FOR PASSWORD CHANGE **
20 '**                 (C) a. PARK          **
30 INPUT"ENTER 4 LETTER PASSWORD";PW$:IFLEN(PW$)<>4THEN30
40 FORI=1TO4
50 A$=MID$(PW$,I,1)
60 POKE32695+I,ASC(A$)
70 NEXTI
80 PRINT"Password entered."
90 END
```

** OTHELLO L2/16K **

```
10 ' OTHELLO (C) 1981 PETER R SMITH - 33 HEADS RD. DONVALE 3111
20 GOTO 1250
30 Z2$="":Z5=0
40 PRINT @ Z4, STRING$(Z1, 95);
50 Z3$=INKEY$:IF Z3$="" GOTO 50
60 Z6=ASC(Z3$)
70 IF Z6=13 AND Z5>0 RETURN
80 IF Z6=8 OR Z6=24 THEN IF Z5>0 THEN Z5=Z5-1: Z2$=LEFT$(Z2$, Z5
): POKE 15360+Z4+Z5, 95: GOTO 50
90 IF Z6<48-16*Z0 THEN 50
100 IF Z6>57+65*Z0 THEN 50
110 Z2$=Z2$+Z3$
120 Z7=ASC(Z3$)
130 IF Z7>95 THEN Z7=Z7-32
140 POKE 15360+Z4+Z5, Z7
150 Z5=Z5+1
160 IF Z5<Z1 THEN 50
170 RETURN
180 IF A(C)=0 GOTO 620
190 IF C>NP THEN 380
200 PRINT@1009,"              ";:PRINT @ C*512-79,"YOUR MOVE?";
210 Z0=1:Z1=4:Z4=C*512-68:GOSUB30:PRINT@C*512-79,"
 ";
220 PRINT @ 1457-C*512,CHR$(207);
230 IF LEFT$(Z2$, 1)<>"P" GOTO 320
240 GOSUB 940
250 IF R>-100 THEN 300
260 C=3-C
270 GOSUB 940
280 IF R=-100 THEN 440
290 GOTO 180
300 PRINT @ 561,"BAD MOVE";
310 GOTO 180
320 J=VAL(Z2$)
330 IF J<1 OR J>64 THEN 300
340 I=INT((J-1)/8)+1
350 J=J-I*8+8
360 IF B(I, J)<>0 GOTO 300
370 GOTO 460
380 GOSUB 940
390 IF R>-100 THEN PRINT @ C*512-79,"MY MOVE IS";I*8+J-8;: PRINT
 @ 1457-512*C, CHR$(207);: GOTO 460
400 PRINT @ 512*C-79,"I CAN'T MOVE";
410 C=3-C
420 GOSUB 940
430 IF R>-100 THEN 180
440 PRINT @ 512*C-79,"NO MOVES LEFT";
450 GOTO 620
460 GOSUB 1240
470 F=0
480 GOSUB 800
490 IF W>0 THEN 530
500 M=I:N=J:GOSUB 1210
510 PRINT @ 128*I-69+J*6, I*8+J-8;
520 GOTO 300
530 PRINT @ 561,"FLIPS";W;
540 B(I, J)=C
550 M=I:N=J:GOSUB 1100
560 A(C)=A(C)+W+1
570 A(3-C)=A(3-C)-W
580 FOR I=1 TO 2
590 PRINT @ 512*I-196, A(I);
600 NEXT I
610 IF A(1)+A(2)<64 THEN C=3-C: GOTO 180
620 IF A(1)>A(2)THEN N$(0)=N$(1): GOTO 660
630 IF A(1)<A(2)THEN N$(0)=N$(2): GOTO 660
640 N$(0)=" A DRAWN GAME!"
650 GOTO 670
660 PRINT @ 433,"THE WINNER IS";
670 PRINT @ 561,"ANOTHER GAME? ";CHR$(95);
680 PRINT @ 497, STRING$(15, 32);
690 FOR J=1 TO 20
700 NEXT J
710 PRINT @ 497, N$(0);
720 FOR J=1 TO 50
730 NEXT J
740 Z2$=INKEY$
750 IF Z2$="" GOTO 680
760 IF Z2$="N" CLS: END
770 IF Z2$<>"Y" GOTO 680
780 CLS
790 GOTO 1900
800 W=0
810 FOR D=0 TO E
820 M=I:N=J:P=X(D):Q=Y(D)
830 FOR L=0 TO S
840 M=M+P:N=N+Q
850 PRINT@1009,CHR$(143);"THINKING ";CHR$(143);:IF B(M, N)=Z GOT
O 920
860 PRINT@1009,"              ";:IF B(M, N)<>C GOTO 910
870 IF L=1 THEN 920
880 W=W+L-0
890 IF F=0 GOSUB 2640
900 GOTO 920
910 NEXT L
```

```
920 NEXT D
930 RETURN
940 F=0
950 R=-100
960 IF A(C)=0 RETURN
970 FOR I=0 TO E
980 FOR J=0 TO E
990 IF Z(I, J)<>0 GOTO 1070
1000 GOSUB 800
1010 IF W=0 THEN 1070
1020 W=W*W2(ST(C))+P(I, J)*W1(ST(C))
1030 IF W<R THEN 1070
1040 IF W>R THEN R=W: U=I: V=J: TM=1: GOTO 1070
1050 TM=TM+1
1060 IF TM*RND(0)<1 THEN U=I: V=J
1070 NEXT J, I
1080 I=U:J=V
1090 RETURN
1100 Z(M, N)=1
1110 FOR K=0 TO E
1120 U=M+X(K):V=N+Y(K)
1130 Z(U, V)=B(U, V)
1140 NEXT K
1150 RETURN
1160 GOSUB 1100
1170 B(M, N)=C
1180 PRINT @ M*128+N*6-133, S1$(C-1);
1190 PRINT @ M*128+N*6-69, S2$(C-1);
1200 RETURN
1210 PRINT @ M*128+N*6-133, STRING$(4, 131);
1220 PRINT @ M*128+N*6-69, CHR$(196);
1230 RETURN
1240 M=I:N=J:GOTO 1180
1250 DEFINT A-Z
1260 CLEAR 1500
1270 CLS
1280 PRINT @ 207,"WELCOME TO THE GAME OF OTHELLO"
1290 DATA 8,2,8,2,2,3,2,2,7,2,2,7,2,7,8,0,2,4,2,5,2,5,2,3,2,2,2
1300 DATA 7,2,7,2,7,2,4,2,0,2,4,2,5,2,5,7,2,5,4,2,7,2,7,2,4,2,0
1310 DATA 2,4,2,5,2,5,2,3,2,2,2,7,2,7,2,7,2,4,2,0,8,5,2,5,2,3,2
1320 DATA 2,7,2,7,2,7,2,8,0
1330 OT$=""
1340 FORI=1 TO 45
1350 READ J,K
1360 OT$=OT$+STRING$(J,191)+CHR$(192+K)
1370 NEXT
1380 PRINT @ 972,"WOULD YOU LIKE INSTRUCTIONS (Y/N)?";
1390 Z0=1:Z1=1:Z4=1007:GOSUB30
1400 IF Z2$="N" GOTO 1600
1410 IF Z2$<>"Y" GOTO 1390
1420 CLS
1430 PRINT TAB(22);"THE GAME OF OTHELLO"
1440 PRINT"THE OBJECT OF THE GAME OF OTHELLO IS TO OCCUPY THE MO
ST SQUARES.";
1450 PRINT"THE PLAY PROCEEDS BY OCCUPYING SQUARES IN TURN  WHICH
 CAUSE YOUR";
1460 PRINT"OPPONENT'S SQUARES TO BE FLIPPED. OPPONENT'S SQUARES
ARE FLIPPED";
1470 PRINT"IF THEY ARE IN A DIRECT LINE (VERTICAL, HORIZONTAL, O
R DIAGONAL)";
1480 PRINT"BETWEEN ANY SQUARE YOU OCCUPY AND THE SQUARE YOU JUST
 PLACED."
1490 PRINT"EITHER  0, 1, OR 2 PLAYERS CAN PLAY OTHELLO.  WITH 0
PLAYERS THE";
1500 PRINT"COMPUTER PLAYS ITSELF.  WITH 1 PLAYER THE COMPUTER PL
AYS AGAINST";
1510 PRINT"YOU.  WITH 2 PLAYERS YOU CAN PLAY AGAINST A HUMAN OPP
ONENT."
1520 PRINT"MOVES ARE ENTERED IN REPLY TO THE 'YOUR MOVE' PROMPT.
  YOUR MOVE";
1530 PRINT"MUST CAUSE AT LEAST  1 OF YOUR OPPONENT'S SQUARES TO
BE FLIPPED.";
1540 PRINT"IF YOU CAN'T MOVE,  ENTER  'PASS'  INSTEAD OF A SQUAR
E'S NUMBER.";
1550 PRINT"IF YOUR MOVE IS INVALID THE COMPUTER WILL REPLY WITH
 'BAD MOVE'";
1560 PRINT"AND WILL ASK FOR 'YOUR MOVE' AGAIN. IF YOU HAVE A MOV
E THAT WILL";
1570 PRINT"FLIP AN OPPONENT'S SQUARE, YOU MUST MOVE AND NOT PASS
."
1580 PRINT TAB(20);"PRESS ANY KEY TO CONTINUE";
1590 IF INKEY$ ="" GOTO 1590
1600 CLS
1610 PRINT @ 256,OT$
1620 DIM P(8, 8), N$(2), B(9, 9), X(8), Y(8), A(2), Z(9, 9), S1$
(1), S2$(1), W1(5), W2(5), ST(2)
1630 DATA-1, 0,-1, 1, 0, 1, 1, 1, 1, 0, 1,-1, 0,-1,-1,-1
1640 DATA 0, 4, 1, 4, 1, 2, 1, 1, 1, 0
1650 FOR I=1 TO 8
1660 READ X(I), Y(I)
1670 NEXT I
1680 FOR I=1 TO 5
1690 READ W1(I), W2(I)
1700 NEXT I
1710 M=0:N=0:Z=0:C=0:O=1:E=8:S=7:K=0:L=0:R=0:W=0:T=2
1720 S1$(0)=CHR$(163)+CHR$(147)+CHR$(163)+CHR$(147)
1730 S2$(0)=CHR$(136)+CHR$(132)+CHR$(136)+CHR$(132)
1740 S1$(1)=CHR$(131)+CHR$(163)+CHR$(147)+CHR$(131)
1750 S2$(1)=CHR$(130)+CHR$(139)+CHR$(135)+CHR$(129)
1760 V2$=CHR$(151)+CHR$(131)+CHR$(131)+CHR$(131)+CHR$(131)+CHR$(
171)
1770 V3$=CHR$(149)+CHR$(196)+CHR$(170)
1780 FOR I=1 TO 3
1790 V2$=V2$+V2$
1800 V3$=V3$+V3$
1810 NEXT I
1820 FOR I=1 TO 4
1830 FOR J=1 TO 4
1840 READ P(I, J)
1850 P(9-I, J)=P(I, J)
1860 P(9-I, 9-J)=P(I, J)
1870 P(I, 9-J)=P(I, J)
```

```
1880 NEXT J, I
1890 DATA 9, 2, 8, 6, 2, 1, 3, 4, 8, 3, 7, 5, 6, 4, 5, 0
1900 RANDOM
1910 PRINT @ 256,OT$
1920 FOR I=0 TO 9
1930 FOR J=0 TO 9
1940 B(I,J)=0:Z(I,J)=1
1950 NEXT J, I
1960 CLS
1970 FOR I=0 TO 7
1980 PRINT @ I*128, V2$;
1990 PRINT @ I*128+64, V3$;
2000 NEXT I
2010 FOR I=0 TO 7
2020 FOR J=1 TO 8
2030 K=I*8+J
2040 PRINT @ 128*I+59+J*6," ";
2050 PRINT USING"##";K;
2060 NEXT J, I
2070 C=1
2080 M=4:N=4:GOSUB 1160
2090 M=5:N=5:GOSUB 1160
2100 A(1)=2
2110 C=2
2120 M=4:N=5:GOSUB 1160
2130 M=5:N=4:GOSUB 1160
2140 A(2)=2
2150 PRINT @ 49," O T H E L L O";
2160 PRINT @ 113, STRING$(15, 131);
2170 PRINT @ 177,"NUMBER OF";
2180 PRINT @ 241,"PLAYERS (0-2)?";
2190 Z0=0:Z1=1:Z4=305:GOSUB 30
2200 NP=VAL(Z2$)
2210 IF NP>2 THEN 2190
2220 GOSUB 2690
2230 N$(1)="LEFT HAND"
2240 IF NP=0 THEN N$(2)="RIGHT HAND" ELSE N$(2)="COMPUTER"
2250 IF NP=2 THEN 2370
2260 FOR C=NP+1 TO 2
2270 GOSUB 2690
2280 PRINT @ 177,"STRATEGY LEVEL";
2290 PRINT @ 241,"FOR ";N$(C);
2300 PRINT @ 305,"(0-5, 0=LOW)";
2310 Z0=0:Z1=1:Z4=318:GOSUB30
2320 ST(C)=VAL(Z2$)
2330 IF ST(C)>5 GOTO 2310
2340 NEXT C
2350 C=1
2360 IF NP=0 THEN 2530
2370 FOR C=1 TO NP
2380 PRINT @ 177,"NAME PLEASE    ";
2390 PRINT @ 241,"PLAYER";C;"?  ";
2400 Z0=1:Z1=15:Z4=305:GOSUB30
2410 N$(C)=Z2$
2420 NEXT C
2430 C=1
2440 IF NP<>1 THEN 2530
2450 GOSUB 2690
2460 PRINT @ 177,"WOULD YOU LIKE";
2470 PRINT @ 241,"TO MOVE FIRST?";
2480 PRINT @ 305,"(Y/N)";
2490 Z0=1:Z1=1:Z4=311:GOSUB 30
2500 IF Z2$="Y" GOTO 2530
2510 C=2
2520 IF Z2$<>"N" GOTO 2490
2530 K1=C
2540 GOSUB 2690
2550 FOR C=1 TO 2
2560 PRINT @ C*512-335, N$(C);
2570 PRINT @ 512*C-271, STRING$(15, 131);
2580 N=9:M=4*C-1.5:GOSUB1180
2590 PRINT @ 512*C-201,"SCORE";
2600 PRINT @ 512*C-196, A(C);
2610 NEXT C
2620 C=K1
2630 GOTO 180
2640 M=I:N=J
2650 FOR L1=0 TO L-0
2660 M=M+P:N=N+Q:GOSUB 1170
2670 NEXT L1
2680 RETURN
2690 FOR I=177 TO 369 STEP 64
2700 PRINT @ I, CHR$(207);
2710 NEXT I
2720 RETURN
```

** LOAN CALCULATION PACKAGE **

```
10 REM  LOAN CALCULATION PACKAGE FOR DAILY REDUCING
CAPITALISED MONTHLY BASIS.
AUTHOR     KEN GLASSON
           P.S. 2019 SUMMERVILLES RD
           KARALEE,Q. 4305     'PHONE 07 2821102  JULY'81
20 CLS:PRINT@147,"L O A N   P A C K A G E"
30 PRINT@211,"======================="
40 PRINT@344,"** M E N U **"
50 PRINT:PRINT"                  1.  REPAYMENT CALCULATION
                2.  REMAINING TERM CALCULATION
                3.  REMAINING BALANCE CALCULATION
                4.  DISSECTION OF REPAYMENTS
                5.  REPAYMENT FACTOR CALCULATION"

60 PRINT"                   6.  END PROGRAM"
70 PRINT:PRINT"                        ENTER 1,2,3,4,5,OR 6"
80 A$=INKEY$:IFA$=""THEN80
```

```
90 G=VAL(A$)
100 ONGGOTO110,300,420,560,830,940
110 CLS:PRINT"LOAN REPAYMENT CALCULATION"
120 PRINT"============================"
130 PRINT:PRINT:INPUT"ENTER LOAN AMOUNT.............";LA:L=LA
140 INPUT"ENTER INTEREST RATE...........";IR:I=IR
150 INPUT"ENTER TERM OF LOAN IN YEARS...";TY:T=TY
160 INPUT"ENTER INSURANCE PREMIUM-LIFE..";M
170 INPUT"ENTER INSURANCE PREMIUM-FIRE..";F
180 P=M+F:T=T*12:I=I/1200
190 V=1/(1+I):A=(1-V[T)/I
200 R=L/A
210 IFP=0THEN240
220 PRINT:PRINT"MONTHLY REPAYMENT IS $";:PRINTUSING"#,###.##";R;
:PRINT" + $";:PRINTUSING"##.##";P/12;:PRINT" OR  $";:PRINTUSING"#
,###.##";R+P/12:PRINT"($";:PRINTUSING"###.##";(R*12+P)/52;:PRINT"
 PER WEEK)"
230 GOTO250
240 PRINT:PRINT"MONTHLY REPAYMENT IS $";:PRINTUSING"#,###.##";R;
:PRINT " ($";:PRINTUSING"###.##";R*12/52;:PRINT" PER WEEK)"
250 PRINT"TOTAL INTEREST PAID WOULD BE APPROXIMATELY $";:PRINTUS
ING"###,###.##";R*T-L
260 FORD=1TO1000:NEXT:PRINT:PRINT"1. RETURN TO MENU.   2. END PR
OGRAM"
270 B$=INKEY$:IFB$=""THEN270
280 G=VAL(B$)
290 ONGGOTO20,940
300 CLS:PRINT"REMAINING TERM CALCULATION"
310 PRINT"============================"
320 PRINT:PRINT:INPUT"ENTER BALANCE  OF LOAN.........";LA:B=LA
330 INPUT"ENTER INTEREST RATE.............";IR:I=IR
340 INPUT"ENTER MONTHLY REPAYMENT
(EXCLUDING INSURANCE).........";MR:R=MR
350 PRINT:PRINT"CALCULATION IN PROGRESS"
360 I=I/1200:X=1/(1+I):Y=X
370 Z=1-(B*I/R)
380 T=0
390 A=X*Y:T=T+1
400 IFA=<ZTHENPRINT@576,"REMAINING TERM IS ";:PRINTUSING"##.##";
T/12;:PRINT" YEARS  (";T;") MONTHS":GOTO260
410 Y=A:GOTO390
420 CLS:PRINT"REMAINING BALANCE CALCULATION"
430 PRINT"============================":PRINT:PRINT
440 INPUT"ENTER LOAN AMOUNT.............";LA:L=LA
450 INPUT"ENTER INTEREST RATE..........";IR:I=IR
460 INPUT"ENTER MONTHLY REPAYMENT
(EXCLUDING INSURANCE).......";MR:R=MR
470 INPUT"ENTER TERM REQUIRED IN YEARS.";TY:T=TY
480 PRINT:PRINT"CALCULATION IN PROGRESS"
490 FORD=1TOT*12
500 C=(L*I/36500)*30.4167:  REM ##### 30.4167 IS AVERAGE DAYS
PER MONTH  (365/12)
510 L=L+C
520 L=L-R
530 NEXT
540 PRINT@640,"BALANCE REMAINING AFTER ";T;" YEARS IS $";:PRINTU
SING"###,###.##";L
550 GOTO260
560 CLS:PRINT"DISSECTION OF LOAN REPAYMENTS"
570 PRINT"============================":PRINT:PRINT
580 INPUT"ENTER LOAN AMOUNT.............";LA:L=LA
590 INPUT"ENTER INTEREST RATE.............";IR:I=IR
600 INPUT"ENTER MONTHLY REPAYMENT
(EXCLUDING INSURANCE).........";MR:R=MR
610 INPUT"ENTER TERM OF LOAN IN YEARS.....";TY:T=TY
620 S=0:Y=0:I=I/36500:C1=0
630 CLS:PRINT"REPAYMENT NO.","INTEREST","PRINCIPAL","  BALANCE"
640 FORD=1TOT*12
650 C=L*I*30.4167
660 L=L+C-R:C=(INT(C*100))/100:C1=C1+C
670 REM LOAN BALANCE SHOWN TO NEAREST 10C FOR SAKE OF SPEED
    OF EXECUTION OF PRINT STATEMENT. DOUBLE PRECISION MAY
    BE USED BY ADDING THE DOUBLE PRECISION INDICATOR '#'
    AFTER THE VARIABLE 'L' WHERE IT APPEARS IN THIS SECTION.
680 PRINTD;:PRINTTAB(16)USING"#,###.##";C;:PRINTTAB(32)USING"#,#
##.##";R-C;:PRINTTAB(48)USING"###,###.##";L
690 IFL<=RTHENPRINT"THE NEXT REPAYMENT WILL CLEAR THE LOAN
TOTAL INTEREST PAID WILL BE $";:PRINTUSING"###,###.##";C1+(L*I*3
0.4167):GOTO770
700 S=S+1:IFS=12THEN720
710 GOTO810
720 Y=Y+1:IFY=1THEN730ELSE740
730 PRINT"AFTER 1 YEAR ";:PRINTUSING"##.##";((R-C)/R)*100;:PRINT
"% OF THE REPAYMENT IS APPLIED TO":PRINT"THE PRINCIPAL AND TOTAL
INTEREST TO DATE IS $";:PRINTUSING"###,###.##";C1:GOTO750
740 PRINT"AFTER";Y;"YEARS ";:PRINTUSING"##.##";((R-C)/R)*100;:PR
INT"% OF THE REPAYMENT IS APPLIED TO":PRINT"THE PRINCIPAL AND TOT
AL INTEREST TO DATE IS $";:PRINTUSING"###,###.##";C1
750 S=0:INPUT"HIT >>NEW LINE<< TO CONTINUE";X
760 IFY=TTHENCLSELSEGOTO800
770 PRINT"1.  RETURN TO MENU      2.  END PROGRAM";
780 C$=INKEY$:IFC$=""THEN780
790 G=VAL(C$):GOTO290
800 CLS:PRINT"REPAYMENT NO.","INTEREST","PRINCIPAL","  BALANCE"
810 NEXT
820 CLS:GOTO260
830 CLS:PRINT"REPAYMENT FACTOR CALCULATIONS"
840 PRINT"============================"
850 PRINT:PRINT:INPUT"ENTER INTEREST RATE";I:CLS
860 L=1000:I=I/1200:T=1:P=0:PRINT"TERM","FACTOR (@";I*1200;"%)":
PRINT
870 FORF=1TO40
880 N=T*12:V=1/(1+I):A=(1-V[N)/I:R=L/A
890 PRINTT,:PRINTUSING"##.##";R
900 T=T+1:P=P+1
910 IFT=41PRINT:GOTO770
920 IFP=10THENPRINT:INPUT"HIT >>NEW LINE<< TO CONTINUE";X:P=0:CL
S:PRINT"TERM","FACTOR (@";I*1200;"%)":PRINT
930 NEXTF
940 CLS:PRINT"THANK YOU":END
```

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

Next month's issue will contain at least the following programs plus the usual features and articles. An (80) after a program title indicates that the program will be for TRS-80 Model 1/3 or System 80/Video Genie computers. (Colour) indicates that the program will be for the TRS-80 Colour Computer and the Hitachi Peach.

** HEX CONSTANTS (80) LII/4K m/l **

As a level 2 user, you may have wished you had the disk Basic ability to use hexadecimal values in your program instead of having to convert them to decimal. Now with this program you will be able to write a statement such as:

```
FOR I = &H3C40 TO &H3CBF : POKE I,&H86
:NEXT I
```

and the format is the same as that used in disk Basic.

** DR WHO ADVENTURE (80) 32/K Disk **

Travel through time and space with Dr. Who in the Tardis. You must find the Key of Time for the Time Lords in order to defeat the Black Guardian. You can go back and forth between six planets and Galaxy in your rather old and unreliable Tardis. Beware the maze on Peladon...

** VARIABLE WORKSHEET (COLOUR) **

Have you ever tried to modify a program you wrote months before only to find it undocumented or the documentation inadequate? Well, if you have a printer, this program will allow you to record information relating to variable usage in a consistent manner for future use, should the need arise.

** SERIES IMPEDANCE CALCULATIONS (80) LII/16K **

This program illustrates one of the fundamental formulae connected with electrical problems and will be of interest to electrical engineering students and amateur radio enthusiasts. A wide variety of problems are solvable and not confined to the general form of series resistance, inductance and capacitance alone.

** LOWER CASE CONVERTER (80) LII/16K m/l **

This assembly language program will convert all uppercase letters inside print statements into lower case with the exception of the first letter inside a quotation and the first letter after a period and two spaces which is assumed to be a new sentence. Just the thing for those long Adventure type programs that you typed in before you fitted your lower case conversion kit.

** MILEAGE CALCULATOR (COLOUR) **

Keeping track of your car's fuel consumption can be a nuisance. This program will remove some of the tedium by making the necessary calculations and, optionally, filing the data to maintain a continous record. This information could be of use when completing next year's tax return or in deciding if your car needs mechanical attention.

APPLICATION FOR PUBLICATION OF A PROGRAM IN MICRO-80

Date .....

To **MICRO-80** SOFTWARE DEPT. PO BOX 145 MORPHETTVALE SA 5162

Please consider the enclosed program for . . .

Tick where appropriate

(i) Publication in MICRO-80 .....

(ii) Publication on disk or cassette only .....

(iii) Both .....

Name .....

Address .....

.....

.....Postcode.....

*** * * CHECK LIST * * ***

Please ensure that the cassette or disk is clearly marked with your name and address, program name(s), Memory size, Level I, II, System 1 or 2, Edtasm, System, etc. The use of REM statements with your name and address is suggested, in case the program becomes separated from the accompanying literature.

Ensure that you supply adequate instructions, notes on what the program does and how it does it, etc.

For system tapes, the start, end, and entry points, etc.

The changes or improvements that you think may improve it.

Please package securely — padabags are suggested — and enclose stamps or postage if you want your cassette or disk returned.

***** CASSETTE/DISK EDITION INDEX *****

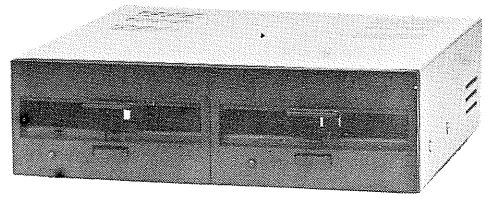
The cassette edition of MICRO-80 contains all the software listed each month, on cassette. The cassette also contains the source code for machine language programs which may not have been printed due to space restrictions. All programs are recorded twice. Level I programs can only be loaded into a Level I TRS-80 if the Level I in Level 2 program from the MICRO-80 Software Library - Vol. 1 is first loaded into your Level 2 TRS-80 or System 80/Video Genie. Note: System 80/Video Genie computers have had different tape-counters fitted at different times. The approximate start positions shown are correct for the very early System 80 without the volume control or level meter. They are probably incorrect for later machines. The rates for a cassette subscription are printed on the inside front cover of each issue of the magazine.

The disk edition contains all those programs which can be executed from disk, including Level I programs. Level I disk programs are saved in the NEWDOS format. Users require the Level I/CMD utility supplied with NEWDOS+ or NEWDOS 80 version 1.0 to run them.

SIDE 1	TYPE	I.D.	DISK FILESPEC	APPROX. START POSITION CTR-41	CTR-80	SYSTEM-80
OTHELLO	LII/4-16K	0	OTHELLO/BAS	18	10	10
"	"	"	"	89	50	53
LOAN CALC PACKAGE	LII/16K	L	LOAN/BAS	155	87	91
"	"	"	"	197	111	117
PASSWORD	LII/16K	P	PASSWORD/BAS	237	134	140
"	"	"	"	243	137	144
"	SYSTEM	PASSWD	PASSWORD/CMD	250	141	148
"	"	"	"	257	145	152
"	EDTASM	PASSWD	PASSWORD/EDT	262	148	156
"	"	"	"	282	159	167
MICRO GRAND PRIX	SYSTEM	PRIX	PRIX/CMD	302	171	180
"	"	"	"	316	179	188
SIDE 2						
MICRO GRAND PRIX	EDTASM	PRIX	PRIX/EDT	18	10	10
"	"	"	"	119	67	70

SAVE A PACKET ON MICRO-80's DISK DRIVE PACKAGES FOR TRS-80 MODEL 1 AND SYSTEM 80 MICROCOMPUTERS

SINGLE DRIVE PACKAGE *from ...* $499



DUAL DRIVE PACKAGE *from ...* $874

Bigger volume means lower cost price, which we are passing on to you. Avoid the annoying bundle of cables, wires and separate boxes. MICRO-80 is now offering our well-proven MPI disk drives in attractive, self-contained single or dual-drive cabinets complete with internal power supply. Our drive Ø and dual-drive packages also include the appropriate version of DOSPLUS and dual-drive cable.

The best news of all is the specially reduced package prices ...
SAVE $23 — $107 over our already low prices!

Choose the appropriate system from the table below:

DRIVE TYPE	No. of Tracks	No. of Heads	Capacity	Dosplus Version	Price	* Saving
DRIVE Ø						
1 x MPI B51	40	1	100K	3.3	$499	$77.95
1 x MPI B52	40	2	200K	3.4	$639	$97.95
1 x MPI B92	80	2	400K	3.4	$799	$107.95
DRIVE 1						
1 x MPI B51	40	1	100K	—	$415	$23.00
1 x MPI B52	40	2	200K	—	$525	$23.00
1 x MPI B92	80	2	400K	—	$695	$23.00

*Represents the saving compared with buying all the items included in the package separately

•Drive Ø package includes one bare disk drive, self-contained single-drive cabinet/power supply as illustrated, two drive cable and the version of DOSPLUS indicated.

•Drive 1 package includes one bare disk drive and self-contained single-drive cabinet/power supply as illustrated.

If it's a dual-drive system you need, then take advantage of our dual-drive package and
SAVE a further $40 on the price of two single-drive packages ...

DRIVE TYPE	No. of Tracks	No. of Heads	Capacity	Dosplus Version	Price
2 x MPI B51	40 ea	1 ea	2 x 100K	3.3	$874
2 x MPI B52	40 ea	2 ea	2 x 200K	3.4	$1125
2 x MPI B92	80 ea	2 ea	2 x 400K	3.4	$1454

Dual-drive package includes two bare disk drives, self-contained dual-drive cabinet/power supply as illustrated, two drive cables and the version of Dosplus indicated.

NOTE: All 40 track drives are completely compatible with 35 track operating systems such as TRSDOS. DOSPLUS allows you to realise an additional 14% capacity compared with TRSDOS. Under DOSPLUS 3.4, 80 track drives can read 35/40 track diskettes.

All disk drive components are still available separately:

BARE DRIVES — MPI drives offer the fastest track-to-track access time (5 milliseconds) available. All drives are capable of operating in double density for 80% greater storage capacity.

	Price		Freight
MPI B51 40 track, single-head, 100K	$399	New, Reduced Price	$5.00
MPI B52 40 track, dual-head, 200K	$449		$5.00
MPI B92 80 track, dual-head, 400K	$619		$5.00
Simple, wrap-around cabinet	$12		$2.00
Separate, dual-drive power supply	$85		$8.00

	Price	Freight
Self-contained, single drive cabinet/power supply	$99	$5.00
Self-contained, dual-drive cabinet/power supply	$135	$5.00
Two drive cable	$39	$2.00
Fan drive cable	$49	$2.00
DOSPLUS 3.3	$99.95	$2.00
DOSPLUS 3.4	$149.95	$2.00

Prices are FOB Adelaide. Add $5.00 freight for single drive package, $10.00 for dual-drive package. Prices are in Australian dollars. Freight is road freight anywhere in Australia.

All items carry a 90-day parts and labour warranty. Repairs to be carried out in our Adelaide workshops.

MICRO-80

LEVEL 2 ROM
ASSEMBLY LANGUAGE TOOLKIT
by Edwin Paay
FOR TRS-80 MODEL 1, MODEL 3 AND SYSTEM 80/VIDEO GENIE

This is a new package consisting of two invaluable components:

- **A ROM REFERENCE** Manual which catalogues, describes and cross-references the useful and usable ROM routines which you can incorporate into your own machine language or BASIC programs.
- **DBUG**, a machine language disassembling debugging program to speed up the development of your own machine language programs. DBUG is distributed on a cassette and may used from disk or cassette.

Part 1 of the ROM REFERENCE manual gives detailed explanations of the processes used for arithmetical calculations, logical operations, data movements etc. It also describes the various formats used for BASIC, System and Editor/Assembly tapes. There is a special section devoted to those additional routines in the TRS-80 Model 3 ROM. This is the first time this information has been made available, anywhere. Differences between the System 80/Video Genie are also described. Part 1 is organised into subject specific tables so that you can quickly locate all the routines to carry out a given function and then choose the one which meets your requirements.

Part 2 gives detailed information about each of the routines in the order in which they appear in the ROM. It describes their functions, explains how to use them in your own machine language programs and notes the effect of each on the various Z80 registers.

Part 2 also details the contents of system RAM and shows you how to intercept BASIC routines. With this knowledge, you can add your own commands to BASIC, for instance, or position BASIC programs in high memory — the only restriction is your own imagination!

The Appendices contain sample programmes which show you how you can use the ROM routines to speed up your machine language programs and reduce the amount of code you need to write.

DBUG: Eddy Paay was not satisfied with any of the commercially available debugging programs, so he developed his own. DBUG: allows you to single-step through your program; has a disassembler which disassembles the next instruction before executing it or allows you to bypass execution and pass on through the program, disassembling as you go; displays/edits memory in Hex or ASCII; allows Register editing; has the ability to read and write System tapes and all this on the bottom 3 lines of your screen, thus freeing the rest of the screen for program displays. Four versions of DBUG are included in the package to cope with different memory sizes.

The best news of all is the price. The complete Level 2 ROM ASSEMBLY LANGUAGE TOOLKIT is only:

— Aus. $29.95 + $2.00 p&p
— UK £18.00 + £1.00 p&p

SPECIAL OFFER TO OWNERS OF THE LEVEL II ROM REFERENCE MANUAL ...
UPGRADE TO THIS ASSEMBLY LANGUAGE TOOKIT FOR ONLY $19.95!
Send back your original Level II ROM Reference Manual plus a cheque, money order or Bankcard authorisation for $19.95 plus $2.00 p&p and we will send you the new ASSEMBLY LANGUAGE TOOLKIT

MICRO-80