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

1/2/83

Vol. 3, Issue 8, July 1982

Non-standard Tandy edge connector p.9



Also in this issue:

PROGRAMMING:

Automatic Graphics Packer The Theory and Techniques of Sorting Part 6

SOFTWARE:

- •HEX CONSTANTS Level II

- VARIABLE WORKSHEET Colour
- MILEAGE CALCULATOR Colour

INCLUDING A MICRO-80 CALENDAR

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

EDITOR:		RYSZARD WI	WATOWSKI
ASSOCIATE ED SOFTWARE HARDWARE	I TORS :	CHARLIE BA EDWIN PAAY	RTLETT
MICRO-80 is an international and Colour microcomputers, the It is available at the following	magazine devot Dick Smith Sy prices:	ed to the Tandy TRS ystem 80/Video Genie	5-80 Model I, Model III and the Hitachi Peach.
	Ċ.	12 MONTH SUB.	SINGLE COPY
MAGAZINE ONLY CASSETTE PLUS MAGAZINE DISK PLUS MAGAZINE		\$ 26-00 \$ 65-00 \$125-00	\$ 2-50 \$ 4-00 (cass. only) \$10-00 (disk only)
MICRO-80 is available in the Uni U.K. SUBSCRIPTION DEPT.	ted Kingdom fro 24 Woodhill Pa	m: rk, Pembury, Tunbridg	e Wells, KENT TN2 4NW
MAGAZINE ONLY CASSETTE PLUS MAGAZINE DISK PLUS MAGAZINE		£ 16-00 £ 43-60 £ 75-00	£ 1-50 £ N/A £ N/A
MICRO-80 is available in New Zea	land from:		
MICRO PROCESSOR SERVICE	S, 940A Columbo	Street, CHRISTCHURCH	1 N.Z. Ph 62894
MAGAZINE ONLY CASSETTE PLUS MAGAZINE DISK PLUS MAGAZINE		NZ\$ 43-00 NZ\$ 89-00 NZ\$175-00	NZ\$ 4-00 NZ\$ 5-00 NZ\$15-00
MICRO-80 is despatched from Aust	ralia by airmai	1 to other countries	at the following rates:
(12 MONTH	SUB) <u>MAGAZINE</u>	CASS + MAG	DISK + MAG
PAPUA NEW GUINEA HONG KONG/SINGAPORE INDIA/JAPAN USA/MIDDLE EAST/CANADA	Aus\$40-00 Aus\$44-00 Aus\$49-00 Aus\$55-00	Aus\$ 83-00 Aus\$ 88-00 Aus\$ 95-00 Aus\$102-00	Aus\$ 143-00 Aus\$ 148-00 Aus\$ 155-00 Aus\$ 162-00
Special bulk purchase rates an form in this issue to order your The purpose of MICPO-80 is to	e also availa copy or subscr	ble to computer shop iption.	s etc. Please use the
the most from your TRS-80, Sys 80 is in no way connected with a	stem 80/Video G ny of the Tandy	Genie or Peach and i , Dick Smith or Hitac	hi organisations.
** WE WILL PAY YOU TO PUBLISH Y Most of the information we pub An application form containing earn some extra income is includ	OUR PROGRAMS ** lish is provide full details ed in every iss	ed by our readers, to of how you can use ue.	whom we pay royalties. your microcomputer to
** CONTENT ** Each month we publish at least computers we support. We als We publish articles on hardwar pherals, articles on programm product reviews for both hardwar	one applicati o publish Util re modificatior ing techniques re and software	ons program in BASIC ity programs in BASI s, constructional ar both in Assembly L and we print letters	for each of the micro- C and Machine Language. ticles for useful peri- anguage and BASIC, new to the Editor.
** COPYRIGHT ** All the material published in must not copy it, except for itself or making copies of progr	this magazine your own use. ams on tape or	is under copyright. This applies to ph disk.	. This means that you otocopying the magazine
** LIABILITY ** The programs and other article utmost to ensure that they fun	es in MICRO-80 Iction as descr Im or other ar	are published in go ibed. However, no li ticle to function sa	od faith and we do our iability can be accepted tisfactorily or for any

.

***** CONTENTS *****	
	PAGE
EDITORIAL	2
PEEKING (UK) (From our U.K. Correspondent)	3
INPUT/OUTPUT - LETTERS TO THE EDITOR	3
MICROBUGS	6
TANDY'S VERSAFILE - SOFTWARE REVIEW	7
THE THEORY AND TECHNIQUES OF SORTING - PART 6	8
AUTOMATIC GRAPHICS PACKER	15
SOFTWARE SECTION	
 VARIABLE WORKSHEET. VARIABLE WORKSHEET. MILEAGE CALCULATOR. MILEAGE CALCULATOR. CC MILEAGE CALCULATOR. PEACH CALENDAR. L2/16K HEX CONSTANTS. L2/16K SERIES IMPEDANCE CIRCUIT. L2/16K DR WHO ADVENTURE. L2/16K m.1. 	18 & 24 18 & 26 18 & 25 18 & 26 18 & 28 19 & 29 20 & 29 21 & 31 21
MICRO-80 PRODUCTS CATALOGUE NEXT MONTH'S ISSUE CASSETTE/DISK EDITION INDEX ORDER FORM	CENTRE 35 36 36

MICRO-80 is registered by Australia Post - Publication SQB 2207 Category B AUSTRALIAN OFFICE AND EDITOR: U.K. SUBSCRIPTION DEPT: Printed by: Published in Australia by: MICRO-80 P.O. BOX 213, GOODWOOD, SOUTH AUSTRALIA, 5034. TEL. (08) 211 7244 24 WOODHILL PARK, PEMBURY TUNBRIDGE WELLS, KENT TN2 4NW Shovel & Bull Printers, 379 South Road, MILE END 5031 MICRO-80, 433 Morphett Street, Adelaide.

*** SPECIAL OFFER TO NEW READERS AND READERS RENEWING THEIR SUBSCRIPTION *** *** SOFTWARE LIBRARY, VALUED AT OVER \$100 - FREE!!! ***

MICRO-80 has developed a new Library of Software consisting of 7 programs and a comprehensive user manual. The Software Library, on cassette, will be sent FREE to every new subscriber and to every subscriber who renews his subscription for another 12 months. Disk subscribers will receive their Software Library on a diskette. The new Software Library contains the following Level II/Disk Programs. All programs will also operate on the Model III.

Level I in Level Π

Convert your Level II TRS-80 or System 80 to operate as a Level I machine. Opens a whole new library of software for your use.

Copier

Copies Level II System tapes, irrespective of where they load in memory. Copes with multiple ORG programs. Z80 MON

Z80 MON

A low memory, machine language monitor which enables you to set break points, edit memory, punch system tapes, etc...

Cube

An ingenious representation of the popular Rubick's cube game for Disk users.

Poker

Play poker against your computer, complete with realistic graphics.

Improved Household Accounts

Version 3.0 of this useful program. One or two bugs removed and easier data entry. This program is powerful enough to be used by a small business.

80 Composer

A music-generating program which enables you to play music via your cassette recorder and to save the music data to tape. This is an improved version of the program published in Issue 17 of Micro-80.

***** EDITORIAL *****

On behalf of all the staff here at MICRO-80, may I wish you all the best in the coming year. I feel a little nervous embarking on this, my maiden editorial, and a little worried that some of you may not think me too bright considering that my decision to accept this position at this point in time was voluntary. But, I assure you that the discrepancy between the calendar and cover month has reached a maximum and that one of my main objectives is to remove this discrepancy altogether. This will, naturally enough, take some time to accomplish and I hope that you will afford me the same degree of patience you have my predecessor, (if not more, since by comparison I am but a novice!).

Perhaps I should begin by telling you a little about myself and how I acquired my interest in computing. A long time ago, or so it seems now, when I was still in high school, I became very interested in electronics and spent the next few years trying to make the perfect hi-fi amplifier. Most of my attempts (usually designs from various electronic magazines) died horrible deaths at the moment of powering them up and I eventually bought one. Nevertheless, this fascination with electronics sustained me until I came across logic circuits in third year Physics at University. At the same time I had my first taste of computing in Applied Maths but never had the faintest suspicions that the two were closely connected (computers and logic circuits). My greatest achievement was the perfection of a program, written in FORTRAN, to produce a calendar. Elsewhere in this issue, my latest attempt appears - yet another milestone, as it is the first of my programs to ever be published! I seem to recall expending a great deal more effort the first time.

As Ian mentioned last month, I was, until recently, a high school teacher, in the areas of mathematics and science. When the Education Department took an interest in computers and purchased Apple microcomputers available for use in schools, my interest in computing was rekindled. The technological revolution in the electronics industry was well under way and the price of the personal computer had fallen to the point where it was within my reach (Yes, I had thought of making the perfect computer but my past experiences quashed that idea pretty quickly). It was some time before I became proficient in BASIC and still more time before I could cope with assembly language, but shortly thereafter, I decided to buy a computer. For financial reasons, I bought a System 80 with Level II BASIC and 16K of memory, despite the fact that the Z80 was a different processor. Transferring from the 6502 to the Z80 proved to be less difficult than I had imagined and it was not long before I had achieved the same level of proficiency with the System 80.

Why did I buy a computer? Well, to be perfectly honest, for the thrill of the arcade games - I figured that at 20¢ a time, it would eventually pay for itself! But after a while the novelty wore off and I became more interested in the computer itself, in how it worked and how it did what it did. I soon found that 16k of memory was not enough and began to hate my cassette, so I bought an expansion interface and a couple of disk drives from Dick Smith's. Needless to say, as these had only just been released, my system had teething troubles for a long time. When you add to that my disappointment in the DOS (OS-80) and the general lack of support software, I began to seriously doubt the wisdom of making such a large investment in expanding my original system. Then I had what I consider 'a lucky break' - I came across MICRO-80, right here, in my own state, no less!

I suppose there is no need to tell you the rest of the story but I found friendly help, assistance and much of the support that I was looking for elsewhere. I began to subscribe to the magazine and to learn much more about my computer, and in less time, than I would have done alone. My thirst for knowledge and understanding drove me further to the point where I undertook studies in Computing Science. At the same time, my association with Ian and this organisation grew to the stage where I am now in the position where I can offer to be of help to you through the pages of this magazine.

Lest he feel slighted in any way, I think it essential to acknowledge the tremendous effort that Ian has made. He has conceived the idea and worked hard to make it a success. The combination of his efforts and yours, the subscribers who provide most of the material and valuable input, have made this enterprise so successful that he has been very hard pressed to find the necessary time to devote to the magazine. Many thanks to you, Ian, for the work you have done - I shall endeavour to meet this challenge.

- 000000000 -

***** NOTICE TO MAIL-ORDER CUSTOMERS *****

Very occasionally, some people send cash through the post. We would strongly advise against this procedure, and make it clear that we will not accept any responsibility for any money sent as cash. For your own protection, use Postal Notes, Money Orders, etc. but never send cash.

- 000000000 -

This month I am going to talk about INPUT/OUTPUT again as there are a number of developments in this field in this country at this time. We are still promised the Mini-Disk, and the Micro-Disk but except for a few prototype units which are doing the rounds I have not seen any sign of a commercially available item. However the cost of the standard Disc drives is falling all the time, so perhaps the manufacturers are expecting the cheaper smaller units soon. Even before these units have arrived the next step is arriving - a 'hard disk', or rather a programmable memory unit which is cheap enough to allow it to be used to store programmes for practically instant recall. This is now becoming available in two forms, preprogrammed and self-programmable. The potential size of the storage available is very great indeed and as the first cost of memory devices continues to fall this sort of storage becomes attractive. I hope to test one of these devices in the next month or two and will of course report to you on the results.

Still on the subject of input/output many of our older readers will remember my reports of the 1200 baud conversion program supplied by the UK software house Kansas. It worked very well on the TRS-80, but was never successful on the GENIE/SYSTEM 80. Kansas have now produced a different program which does work on the GENIE, at least it works on mine. It is simple to use and produces tapes of both machine language and BASIC programmes which load at double speed. A good buy to load those long programmes.

- 000000000 -

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

From: Mr. C. Stobert - Wellington, N.Z.

My hobby level of computer activity has been satisfied with BASIC, though perhaps a couple of games may be better if they moved a little faster. Three BASIC programmes in "TRS80-Programs" (Cat No. 62-2064) from Tandy produce hypnotic graphics displays using strings PRINTed. Consequently, I have been curious of the display of "Graphix" since it was published in Issue 7 of Micro-80, but not sufficiently so to delve into assemblers etc. I recently compromised with the attached solution. It may not get top marks for originality, but may be of interest to other readers in a similar situation. 16K Memory is reserved from 28671. I retained operation from 7000H.

BASIC loading of memory does not take too long, but line 140 lets you see something is happening. count to 784.

Data lines were entered using two people, one reading, one typing, then checking back before entering. There are 49 lines of data entering 16 bytes per line, but it did not prove too tedious, and the result was worth the effort. We took two dumps before running, and surprise! It worked first time.

10 CLS: POKE16553,255 20 DEFINTL, R, D, P, N: DEFSTRA, B, S 30 PRINT@202,"GRAPHIX - MICRO-80 - JULY 1980" 40 PRINT@452,"PLEASE BE PATIENT WHILE I" 50 PRINT@520,"GET MY DATA INTO PLACE" 60 **P**=28672:N=1 70 READS: IFS="END"GOT0160 80 A=LEFT\$(S,1):B=RIGHT\$(S,1) 90 L=ASC(A):R=ASC(B) 100 IFL>57THENL=L-7 110 IFR>57THENR=R-7 120 L=L-48:R=R-48 130 D=16*L+R:POKEP,D 140 PRINT@896, "POKE NO. ";N 150 **P=P+1:N=N+1:GOT07**0 160 POKE16526, 0: POKE16527, 112: CLS: X=USR(0) 170 DATAF3,21,0F,72,CD,4D,71,CD,A3,71,21,4D,72,CD,4D,71 180 ° (CONTINUE DATA LINES -- NOTE NO S₱ACE AFTER "DATA" OR BETWEEN 640 ° COMMAS AND DATA BYTES) 660 DATAEND

(Thank you for this contribution, Mr. Stobert. The technique you use for POKEing machine language programs into memory is an interesting one and is certainly superior to the more common method of converting the machine code to Decimal manually and then entering it into the DATA statements. By converting HEXadecimal code in your BASIC Program, you save a considerable amount of work and, if the data statements are laid out in lines of 16 bytes per line, it is easy to check against a Hex dump. It is only recently that we have seen this technique used and you are to be congratulated in being amongst the first to develop it - Ed.)

PAGE 4

From: R. Edwards - Ivanhoe, Vic. To: M. Bauk - Kalamunda, W.A.

In response to your letter in June '82 MICRO-80, where you requested help to the "Other Venture" entitled "Excape from Tramm".

The first location is quite tricky and took me some hours to get past, so you're not entirely alone (I also believed it to be faulty). Nothing can be done on the ship - so enjoy the crash landing. Upon landing the command LOOK SHIP will reveal a "nylon rope" which requires GET NYLON ROPE to pick it up. Moving south with S you will come to a cliff. The command LOOK CLIFF will detect a bush growing on the cliff, although this last command is not necessary. Then type ROPE BUSH (another tricky part) and CLIMB ROPE to reach the top of the cliff.

From here you are on your own, hope you have fun figuring out the rest of the "Venture".

(I think Mr. Bauk will be eternally grateful for your assistance, Mr. Edwards - Ed.)

From: J. Wragg - Page, A.C.T.

Recently I had a small mishap with several tapes - all of the data just disappeared. The tapes were stored in a desk drawer in cassette boxes, but not all of the tapes were affected (i.e. some tapes remained intact while others were completely wiped). Can you offer any clues as to what may have caused this to happen?

(Recorded information on cassette tapes, or any magnetic media for that matter, will be damaged or erased by exposure to strong magnetic fields, which could have occurred by operating a television set or some appliance with an electric motor or transformer on the desk near the tapes. These types of appliances can generate strong magnetic fields. Eddy suggests that some of the cassettes may have been shielded from damage by the others, but the incident does seem mysterious. It would be wise to be careful in this respect in the future and store recorded magnetic media away from sources of heat and magnetic fields. - Ed).

From: J. Finlayson - Bluewater, Qld.

In your journal I notice fairly frequent references to difficulties with tape loading with the System 80. I have one of the early models and I have found no difficulties with loading either BASIC or System tapes provided that I clean and demagnetise the tapehead frequently. A number of my acquaintances with later models have also found this a certain cure for any difficulties but I cannot say the same thing for TRS-80 owners. A problem I have found with saving on computer grade tapes has been failure after re-using several times, this has apparently been due to incomplete erasure by the builtin deck and is easily overcome by erasing on a high quality tape deck if one is available.

(I must agree with Mr. Finlayson's comments, as I have found the same is true on my early model System 80. Although I've never demagnetised the heads, I understand that a strongly magnetised head can erase a tape while playing it! The motto - for reliable cassette operation, the mechanism needs to be clean and use good quality tapes - Ed.)

From: Mr. G. Bull - Balhannah, S.A.

Recently I bought an expansion interface and disk drives for my System 80. A copy of DOSPLUS was included with the purchase of the first drive from MICRO-80, but I have had problems copying my BASIC programs from cassette to disk. They CLOAD in Level 2 without any errors, but I can't get them to CLOAD correctly from DISK BASIC.

(DOSPLUS, NEWDOS and other DOS's more sophisticated than TRSDOS make the assumption that the user is already familiar with the Tandy version of DOS and DISK BASIC and therefore neglect to mention or emphasize the need to disable the interrupts during cassette I/O. All DOS's use the 25mS interrupt to update the software real-time clock and perform other background tasks and, as the ROM routines that read and write data to cassette have critical timing loops, it is essential that interrupts be disabled during cassette I/O. If they are not disabled, information coming in during reading will be lost and CSAVEd information will be overwritten with the interrupt pulses. To disable interrupts under DISK BASIC, type:

and to enable them, type:

CMD "T" (for TAPE?)

CMD "R" (for RESTART?)

This must be done for both reading and writing to cassette while in DISK BASIC. Note that returning to DOS or re-booting the system enables the interrupts, irrespective of whether or not CMD "R" has been entered. \sim - Ed.)

From: Mr.L. Montero - Bellambi, N.S.W.

I have bought a System 80 Blue Label and have been using it for quite a while with no problems.

I CLOADed the program "SOUND EFFECTS REVISITED", published in Issue 22, September, 1981, page 29. My problem is that the program turns on the internal cassette port instead of the internal speaker. Everything else seems to be working perfectly. I believe the problem is in the data in line 190 but I do not know what to do. Can you please help me?

(You are correct in suspecting the data in line 190. A number of people have contacted us concerning this problem - using the internal speaker on the Blue Label. One user, after much experimenting, informed us that the way to cure the problem is not to set bit 2 of port 255 (\emptyset FFH) high. Inquiries to Dick Smith's yielded the same information with an explanation.

In order to make the System 80 more compatible with the Tandy, so that it could use existing Tandy software that has sound without any patches, the Blue Label requires that bit 2 of port 255 be set low to enable the internal speaker. What this means is that there now exists an incompatibility between the Blue Label and earlier System 80's in this respect.

Line 190, as originally published is shown as:

190 DATA205,127,10,229,221,225,221,78,0,121,183,200,221,70,1,62, 5,211,255,16,254,221,70,1,62,6,211,255,16,254,13,32,235,221,35,2 2,1,35,1,255,255,33,48,0,9,56,253,24,214

To enable sound on the Blue Label change it to:

190 DATA205, 127, 10, 229, 221, 225, 221, 78, 0, 121, 183, 200, 221, 70, 1, 62, 1, 211, 255, 16, 254, 221, 70, 1, 62, 2, 211, 255, 16, 254, 13, 32, 235, 221, 35, 2, 1, 35, 1, 255, 255, 33, 48, 0, 9, 56, 253, 24, 214

Note that the only changes occur in the second line where a 5 becomes a 1 and a 6 becomes a 2. This will set bit 2 low and should fix the problem according to the above advice.

Unfortunately, Mr. Montero, your letter omitted to include your full address and I am unable to reply in writing. I hope this reply will suffice. - Ed.)

From: Mr. S. Goodhead - London, England.

I have just spent a major portion of my Christmas Holiday trying to type in a program from MICRO-80 No 6 Vol 3 (May 82) SKYDIVER.

I was not surprised when after hours of typing and editing out errors it did not work. It is a normal occurrence for me; however this time it destroyed itself - "UL ERROR 810". An attempt to LIST gave what I think you call garbage. LIST 10 = 10:+- Line 20 gives TT=26810 TA = 26869 & M = 26820.

It seems any attempt to poke a value into this area creates havoc.

I have tried under LDOS, NEWDOS, and also under BASIC 2 (without disk) Poking the USR (0) Address it still corrupted the short program - the values were now 17129, 17188, 17139.

I am wondering if there is something wrong with my computer, I would appreciate your observations. I also suspect that data statements may contain errors. Has any other reader had trouble or am I the only one? I have TRS-80 Mod 1 Lev II. Lowe electronic interface to Genie +32K expansion unit, 2 disk EPSON MX 80. It is unstable - disk unit starts up for no reason and everything locks up - I think cable connections are unreliable.

(Your last statement suggests very strongly that there is indeed something very wrong with your computer. The undefined Line error tends to make me suspect that the memory, or some parts of it, are defective. When a situation such as this arises it is virtually impossible to debug any programs in the machine - I once had a similar problem with a defective 16K block of memory in my machine. Very short, simple machine language programs would work for a moment and then fill the screen with junk, turn on the disk drives and I would have to reset the machine. It was only by accident that I suspected a hardware fault, when after altering memory contents from a monitor, I found that they had changed after examining them a few moments later, without ever running the program!

Although line 10 lists as garbage, this is, in fact, correct, but the garbage should start just after the REM, not where you have indicated. The reason for this is that the first part of the program finds where in memory the BASIC program starts and positions to the memory location immediately after the REM token. The machine language routine is then POKEd into the memory taken up by the title and explains why the warning regarding this line was necessary in the text accompanying the program. If you refer to the reply to Mr. Montero's letter you will notice that the DATA lines are the same, and there are no errors here.

What I would suggest is that you load the program into memory but do not run it. Examine line 810 after a short time (say, 10-20 seconds) and if the line lists differently then you most probably are having trouble with the memory itself. -Ed).

- 000000000 -

***** MICRO-BUGS *****

Inevitably, no matter how careful one tries to be, there will be errors, mistakes and omissions in articles and programs. Here's where we make corrections.

***** Addendum to Mr. Coleman's SAVING AND LOADING M/L PROGRAMS ON ESF WAFER *****

One very important point that was missing from my original article on SAVING AND LOADING M/L PROGRAMS ON ESF WAFER was how to save the utilities SAVER and LOADER onto wafer themselves! Hopefully these few paragraphs will clarify everything.

First, you will need to enter the machine language utilities into memory. There are a number of methods of doing this - most of them have been explained in this magazine before. Probably the quickest and easist method would be to use any monitor (such as BMON or the ESF monitor) to load the bytes directly into memory. Neither utility is lengthy so this shouldn't take too long. Start off with SAVER and load the bytes shown in the second column of the listing into the locations shown in the first column. Do the same thing with LOADER, then exit the monitor and return to BASIC.

Now get out the wafer that you want these two utilities to reside on and put it into the drive. It doesn't matter which one you save first, so now save them using these parameters:

@SAVEX,16450,31,693 and @SAVEX,16500,12,693

The reason for having an auto-start address of 693 is that this is the entry-point for the SYSTEM command.

Here is a list of some parameters that I have discovered so far (why don't you write in with some others):

PROGRAM	START	LENGTH	ENTRY
Adventure #3	17152	15616	17232
Air Traffic Controller	17152	2618	17152
Asylum *	17139	15620	17326
Back-40 (Advent. Inter.)	17408	6145	17408
BMON	29248	3263	31641
Disassembler (Instant Soft.)	28672	3012	28672
Edtasm-Plus (Microsoft)	17280	12035	17280
Eliza (Tandy)	20480	10241	20480
Penetrator	17408	15360	17408
Penetrator Editor	17408	15360	17409
S.Key	17216	1298	17216
Invaders (Tandy)	17174	3245	17717
Starfighter	17116	15652	17116
Starfighter Trainer	17116	15652	17116
Zbug stand alone	17280	6070	17280

* Note that these are the correct parameters for ASYLUM: last month's were incorrect.

LOTTO PREDICTOR - Issue 10, Sept. '80 (pp. 29-32).

This program has two errors and some potential dangers that need clarification.

(a) The copy of this program as supplied on cassette has several control characters between the OPEN statement and the comment in line 690 that list as spaces. These produce a SYNTAX ERROR during the execution of line 690 and the ON ERROR trap sends the program to line 790,

resulting in an endless loop. Re-type line 690 as it appears in the magazine listing to remove these characters.

690 OPEN "O", 1, "LOTTO/TXT": 'FOR CASSETTE USE CHANGE TO 690 INPUT "READY CASSETTE"; D\$

(b) The program should be terminated in line 780 and not allowed to proceed to line 790. Change line 780 to:

780 PRINT "RUN ENDED": CLOSE:END

With these changes the program will function as described in the accompanying text. However, during the investigation of this program, a number of interesting observations were made about the ON ERROR GOTO/RESUME statements.

- (1) The ERROR GOTO statement establishes an error trap and any subsequent errors that occur during program execution (or after!) will be processed by the program line referenced. The fact that an error at BASIC "READY" prompt causes resumed execution of the program in the error handler can be disatrous!
- (2) An error trap can be reset using the statement ON ERROR GOTO \emptyset .

***** WARNING - JOYSTICKS AND INPUT/OUTPUT PORTS FOR YOUR '80 *****

There is no actual bug in this hardware project but there is a potential source of trouble for the unwary. Mr. Tilley, from Victoria, had assembled the interface and connected it to his Model I only to find the system locked at power up. He made every effort to try and determine the cause himself, but to no avail, so he contacted us for help. I couldn't image what the trouble could be either and, in turn, contacted Allan Dent, the author. He informed me that the Tandy edge connector on the Model I has non-standard numbering in that the odd-numbered pins are on the top and not the bottom. Although there should be no problems if you follow the instructions in the article, Mr. Tilley soldered the other end of the rainbow cable to an IC plug. In this case, you must be very careful indeed, because the leftmost strand of the cable will be connected to pin 2 on the edge and not pin 1! Mr. Tilley found that after rectifying this, (and the rest of the wiring on the IC plug) the interface functioned as intended.

- 000000000 -

***** SOFTWARE REVIEW - TANDY'S VERSAFILE - by J. Dowdall *****

In my occupation as a software consultant, I have occasion to evaluate many 'off-the-shelf' programs and packages for clients. Once in a while, I find one which becomes a useful tool in my business VERSAFILE is one of these.

My client's requirement was for an easy-to-use Data Base which would enable him to store and retrieve brief details on formulae used in his Plastics manufacturing. Formulae were to be stored by product category, and data entry and retrieval had to be simple enough that relatively untrained operators could learn to handle it in a few minutes.

As it turned out, even I can be lucky! The first program I looked at was Versafile. At first it seemed too simple, and I went on to look at some more. After two or three others, though, I was back to look again.

The Command language is straight English - Versafile figures out what you mean. Data entry and retrieval are done the same way - by forming statements or questions. Close your statements with a period, and the statement is stored in a 'keyword file'. Close with a '?', and the program searches the appropriate keyword file for entries which match, and displays or prints them. The program has a total of only seven 'commands', all of which are one or two letters or symbols which fit neatly into an 'almost English' syntax.

Records can be of random lengths from one to 240 bytes and no prior formatting is needed - although the output is in the form of lines (no formatted screen). This could be a disadvantage, especially if your records are near maximum length, as words get chopped off as the line wraps around the screen. Another disadvantage is that the records aren't numbered on the display or in printouts. Considering the price of this program, however, these are minor annoyances.

The manual is supplied in a gold-leaf on brown vinyl ring-binder, with the disk in a plastic pocket at the front. Eighteen pages (about one third of them blank) with sample runs and pictures of the video are sufficient to get you up and away in less than twenty minutes.

Versafile is written in BASIC, and the manual gives a few hints on changes you can make to tailor the program to your own needs. One change not mentioned is altering the number of keyword files to be used. As supplied, eight keyword files are used. My client needed twenty, and the changes turned out to be quite simple.

Line 1000 change the 8 to 20 Line 1200 change the 45 to 57 Line 1670 change the 8 to 20 Line 2000 change the 8 to 20 Line 6025 change the 8 to 20 Line 10000 add an extra 12 keywords (DATA items)

and that's the lot! With more keyword files, Versafile's searches are faster (except global) and you can store items under relevant headings. There is only one other change I would recommend: after you have entered two or three hundred lines, print out the entire set of files with P *? and count the number of entries under each keyword. Then EDIT line 10000 again and change the order of the keywords in it to reflect the frequency of use. This gives you another boost of speed - not much, but enough to notice on long searches.

Since obtaining Versafile, I've finally found a way to store magazine article references, and retrieve them in a few minutes without having to leaf through the whole collection for the reviews I need. And a way to list all of my clients along with phone numbers, computer systems, programs used, and possible future requirements.

My rating for Versafile is: EXCELLENT ***

- 000000000 -

***** THE THEORY AND TECHNIQUES OF SORTING - PART 6 - by B. Simson *****

In the first article of this series, a simple method of sorting using exchange techniques was shown, which is the "Bubble Sort". Although it is one of the simplest algorithms around for handling more than 2 or 3 values, it is not one of the most efficient, far from it. This is easily verified by performing a test on a fixed list of numbers and comparing the time taken to execute with some of the other algorithms shown in later articles. However, don't write off techniques of sorting by exchange as being too inefficient yet, without examining at least one other exchange technique.

PARTITION-EXCHANGE SORT (QUICKSORT)

This sorting algorithm is a much more efficient method of exchange sorting than "Bubble", as its alias imples, "Quick"! Its use of specialized data structures is not as extensive as the list insertion sort or the tree sort. It just uses a stack as an auxiliary structure to the main array holding the data. The algorithm is defined in figure 1. It involves placing a particular item in its final position in the list. In so doing all items which precede this item are equal to or smaller in value, and all items that follow it are equal to or greater than this item in value. This process splits or partitions a list into two sublists, with the item between the sublists placed in the correct and final position in the list, hence the name "Partition -Exchange" - a process of partitioning a list and exchanging items. The key to understanding this algorithm is that the same process as was applied to the list before it was partitioned is applied to each sublist, and that's all. This will eventually place all items in the correct position in the list as more sublists. This algorithm can also be implemented recursively, as can be seen by its definition, but the approach in figure 1 is by iterative means. The stack, provided implicitly by recursion, is used in the iterative approach to store the bounds (positions of first and last item) of the sublists as they are created.

THE ALGORITHM IN DETAIL

The process starts by pushing the lower and upper bounds of the current sublist on the stack. Then, as the flowchart shows, this list is partitioned until the stack is empty. The partition module first gets the bounds of the sublist to be partitioned from the stack. On the first time that this module is executed, the sublist bounds popped from the stack are the bounds of the entire list, as pushed in the initialization phase, but as far as the partition module is concerned, it is just another sublist to be partitioned. The sublist bounds are used to initialize two indices into the list. Items are then compared using these two indices, which will initially point to the first and last item in the sublist. During the comparison phase, the indices are brought closer together by either incrementing the lower index, or decrementing the higher index, depending on which one was changed last. The decision on whether to increment the lower or decrement the upper depends on the status of a toggle flag. If it is off, the lower index is incremented, otherwise the upper index is decremented. The toggle flag is only toggled when item pointed to by upper index. The comparison phase is complete when the lower and upper indices point to the same item, being the item placed in the correct position in the final list. It also now separates two sublists. The larger sublist bounds are stacked, and the process repeated on the smaller sublist. Stacking the larger sublist ensures minimum stack growth. The partition module in the flowchart shows that the larger sublist is processed first, then the smaller, but since the smaller sublist bounds are stacked last, they are also popped first for processing in the partition phase when it is repeated.





ILLUSTRATING

Consider the following list: 39 24 68 45 37 57. One pass in the partition phase will be examined. An asterisk before an item indicates the current item being compared using the two indices.

						TOGGLE FLAG
*39	24	68	45	37	*57	ON
*37	24	68	45	*39	57	OFF (1 & 5 SWAPPED)
37	*24	68	45	*39	57	0FF
37	24	*39	45	*68	57	ON (3&5 SWAPPED)
37	24	*39	*45	68	57	ON
37	24	**39	45	68	57	ON

Both indices point to the third item on pass completion. This item has been placed in the correct final position in the list, and separates two sublists. The process has also guaranteed that all items in the left sublist have equal or smaller values, and all items in the right sublist have equal or greater values. Sublist 37, 24 is then processed in the same manner, while sublist 45, 68, 57 is stored. It will eventually be processed when its bounds are popped from the stack at the start of a partition phase. When a sublist of only one item is reached, its bounds are no longer stacked, and it is considered fully processed. A variation to this algorithm involves using this method to create sublists of some critical minimum size. When that size is reached, another sorting algorithm is employed to sort the sublist by passing its bounds to the auxiliary sort. This reduces the stack size in quicksort, and fewer sublists are created, but I have found that this variation is still slower, because of the less efficient sorting algorithm used to sort a particular sublist (Bubble, Selection or Insertion). The method shown here processes sublists down to their absolute minimum size (one).

DEMONSTRATION IN BASIC

Following is a BASIC program demonstrating the algorithm. You will find a great improvement in execution time when compared to Bubble, Selection or Insertion sort, particularly when the list size is large (for BASIC, say over 20). Variables used are:

I,J	INDICES INTO LIST
L,U	LOWER, UPPER SUBLIST BOUNDS
S	STACK
SP	STACK POINTER
FL	TOGGLE FLAG
L ()	LIST OF ITEMS BEING SORTED
SL,SR	SIZE OF LEFT/RIGHT SUBLIST

5000 ' PARTITION-EXCHANGE SORT (QUICKSORT) 5010 ' BY B. SIMSON. 5030 DEFINTA-Z:I=0:J=0:SP=0:T=0:FL=0 5040 CLS:PRINT@13, "PARTITION-EXCHANGE SORT (QUICKSORT)." 5050 INPUT"SIZE OF LIST TO BE SORTED";EL 5080 X=EL/4:IFX<10X=EL 5090 DIML(EL), S(X) 5100 FORM=1TOEL 5110 L(M)=RND(1000)-500 5120 NEXT 5140 GOSUB5210 5150 PRINT: INPUT "HIT ENTER TO START THE SORT"; X 5160 PRINT" SORTING...":GOSUB5260 5170 PRINT"LIST AFTER SORT: " 5180 GOSUB5210 5190 END 5200 ' PRINT L LIST 5210 FORM=1TOEL 5220 PRINTL(M); 5230 NEXT 5240 RETURN 5250 ' QUICK-SORT ROUTINE. 5260 L=1:U=EL 5270 SP=SP+1:S(SP)=L:SP=SP+1:S(SP)=U 5280 IFSP=0THEN5300 5290 GOSUB5320:GOT05280 5300 RETURN 5310 ' PARTITION LIST 5320 U=S(SP):SP=SP-1:L=S(SP):SP=SP-1 5330 I=L:J=U:FL=1 5340 IFI=JTHEN5400 5350 IFL(I)<=L(J)THEN5380 5360 T=L(I):L(I)=L(J):L(J)=T 5370 IFFL=1FL=• ELSEFL=1 5380 IFFL=1J=J-1 ELSEI=I+1 5390 GOT05340 5400 SR=U-I:SL=I-L 5410 IFSR>SLGOSUB5440:GOSUB5480 ELSEGOSUB5480:GOSUB5440 5420 RETURN 5430 ' PROCESS RIGHT SUBLIST 5440 IFSR<=1THEN5460 5450 SP=SP+1:S(SP)=I+1:SP=SP+1:S(SP)=U 5460 RETURN 5470 ' PROCESS LEFT SUBLIST 5480 IFSL<=1THEN5500 5490 SP=SP+1:S(SP)=L:SP=SP+1:S(SP)=I-1 5500 RETURN

The processing overheads of Microsoft BASIC running on a 1.7 or 4 MHz machine does not do justice to this algorithm. For my own interest, I developed an assembly language version to see just how fast it would be in machine language, and I was surprised. BASIC really does seem to take an eternity! I also developed a BASIC driver to display the numbers on the screen as they were being sorted by the machine language routine.

QUICKSORT IN Z80 ASSEMBLY

This routine will sort integers only, allowing for negative values. The following points may help to further clarify the assembly language program. The start address and length of the array to be sorted are passed to the routine to the storage area labelled "STADD" and "LENGTH". This is all that the routine needs to work on the data. The Basic stack is saved on initialization, because Quicksort will use the stack for its own operations to store sublist bounds. Then the lower and upper bound addresses are found and pushed onto the stack. Consideration is given to the fact that each item occupies 2 bytes (integers). "PARTN" involves checking if the stack is empty by comparing original stack address with current value of stack pointer. If empty, then the sort is complete. If not, the next sublist bounds are popped and stored in locations labelled "LOWER" and "UPPER". J-FLAG refers to the toggle flag, being the C register. The indices are check for equality. If not equal, then "CMPARE" is called to compare the items pointed to by HL and DE (indices). This particular version does not check which sublist is greater, but processes the right sublist. "CMPARE" involves getting the actual integers (items being compared) pointed to by HL and DE, into HL and DE for easy comparion. The the sign status is checked for both integers using logical operation "XOR". If they have the same signs, then their signs are irrelevant in their ordering. Because negative integers are stored in two's complement form. If their signs are not the same, then a further check is made in "UNEQU" to check the sign in HL. Swapping two items is done most efficiently by using the EX DE, HL instruction. The task of placing the items in the HL and DE registers back into the sort array is the reverse of the process found in "CMPARE". Since a swap was done, the flag in the C register is toggled. "TESTJ" involves testing the status of the toggle flag to determine which index to update for the next comparison process.

7F77 E5

07800

PUSH

HL

00100 ; PARTITION-EXCHANGE SORT (QUICKSORT). 00200 ; CALLED FROM BASIC AFTER PASSING START ADDRESS 00300 AND LENGTH OF ARRAY TO BE SORTED. ş 00400 AUTHOR: B. SIMSON 3 00500 ; COPYRIGHT (C) 1980 00600 3 7F00 00700 ORG 7F00H 7EFF 00800 STACK DEFL STADD-1 ; NEW STACK 7F00 0000 00900 STADD ;START ADDR DEFW Ø 7F02 0000 01000 LENGTH DEFW LENGTH OF ARRAY ø BOUNDS OF DATA 7F04 0000 01100 LOWER DEFW ø 7F06 0000 01200 UPPER DEFW ø CURRENTLY BEING PROCESSED 7F08 0000 01300 STKSAV DEFW ø ;BASIC SP 01400 ; 7F0A ED73087F 01500 START ;SAVE BASIC STACK LD (STKSAV), SP 7F0E 31FF7E 01600 LD SP, STACK ; INITIALIZE NEW 7F11 Ø8 01700 AF, AF' ;SAVE BASIC REGISTERS EΧ 7E12 D9 01800 FXX DE, (STADD) 7F13 ED5B007F 01900 LD ;START ADDR 7F17 2A027F 02000 HL, (LENGTH) ; IN INTEGERS LD 7F1A 29 02100 ADD HL,HL ; IN BYTES 7F1B 19 02200 ADD HL,DE ;HL=END+2 7F1C 2B 02300 DEC HL 7F1D 2B02400 DEC HL 7F1E EB 02500 ;HL=START, DE=END EХ DE, HL 7F1F E5 02600 **PUSH** ;LOWER HL 7F20 D5 02700 PUSH DE UPPER 7F21 21FF7E 02800 PARTN LD HL, STACK ;STACK EMPTY? 02900 7F24 B7 OR Α 7F25 ED72 03000 SBC HL,S₽ 7F27 CA417F 03100 Z, RETN J۴ ; YES 7F2A D1 03200 POP DE UPPER 7F2B E1 03300 POP HL ;LOWER 7F2C 22047F 03400 LD (LOWER), HL ; STORE 7F2F ED53067F 03500 BOUNDS (UPPER), DE LD 7F33 0E01 03600 L D ;SET J-FLAG C, 1 7F35 E5 03700 CHEKEQ **₽**USH ΗĹ ;I=J? (HL=DE?) 7F36 B7 03800 **DR** Α 7F37 ED52 03900 SBC HL,DE POP 04000 7F39 E1 HL 7F3A 280C 04100 JR Z, RIGHT ;Z IF I=J 7F3C CD7B7F 04200 CALL **CMPARE** ;COMPARE ITEMS 7F3F 18F4 04300 JR CHEKEQ 04400 RETN ;RESTORE BASIC SP 7F41 ED7B087F LD SP, (STKSAV) ;RESTORE REGISTERS 7F45 D9 04500 EXX 7F46 Ø8 04600 AF, AF' EХ 7F47 C9 04700 RET ;CHECK RIGHT SUBLIST SIZE 7F48 2A067F 04800 RIGHT LD HL, (UPPER) 7F4B B7 04900 OR Α 7F4C ED52 ;SIZE*2 IN HL 05000 SBC HL,DE 7F4E 05100 LD ; CHECK MSB 7C A,H 7F4F FE00 CP 05200 ø ;RIGHT SUBLIST BOUNDS 7F51 2005 05300 JR NZ, PUSHR 7F53 7D 05400 LD A,L ;LSB ;=2 INTEGERS CP 7F54 FE04 05500 4 05600 JR C, LEFT ;C IF HL<=4 (2 INT) 7F56 3809 7F58 D5 05700 PUSHR PUSH DE ;DE TO HL (DIVIDER) 7F59 E1 05800 POP HL INC 7F5A 23 05900 HL 7F5B 23 06000 INC HL ;NEXT INT 7F5C 06100 PUSH HL ;LOWER E5 7F5D 2A067F 06200 LD HL, (UPPER) PUSH UPPER 7F60 E5 06300 HL 06400 LEFT HL, (LOWER) ;CHECK LEFT SUBLIST SIZE LD 7F61 2A047F ; (HL)=DIVIDER, (DE)=LOWER 06500 ЕX DE,HL 7F64 EB 7F65 B7 06600 OR Α SBC ;SIZE*2 IN HL 7F66 ED52 06700 HL, DE ;SAME AS RIGHT ABOVE 7F68 7C 06800 LD A,H 7F69 FE00 06700 C₽ ø 07000 \mathbf{JR} NZ, PUSHL 7F6B 2006 7F6D 7D 07100 LD A,L 7F6E FE04 07200 CP 4 07300 J₽ C. PARTN 7F70 DA217F 7F73 D5 07400 PUSHL **₽**USH DE ;LOWER 7F74 2B 07500 DEC ΗL FREV INT DISPLACEMENT FROM LOWER 7F75 2B 07600 DEC HL 7F76 19 07700 ADD HL,DE ;HL=UPPER

7F78	C3217F	07900		JP	PARTN	
7F7B	D5	08000	CMPARE	PUSH	DE	SAVE ADDRESS
7F7C	E5	08100		PUSH	HL	; POINTERS
7F7D	7E	08200		LD	A,(HL)	GET LEFT INT
7F7E	23	08300		INC	HL	; BY LD HL,(HL)
7F7F	66	08400		LD	H,(HL)	
7F80	6F	08500		LD	L,A	
7F81	EB	08600		EX	DÉ,HL	LEFT INT IN DE
7F82	7E	08700		LD	A, (HL)	GET RIGHT INT
7F83	23	08800		INC	HĹ	
7F84	66	08700		LD	H,(HL)	
7F85	6F	07000		LD	L,A	
7F86	EB	09100		EX	DÉ, HL	;HL=LEFT,DE=RIGHT
7F87	7C	09200		LD	A,H	CHECK SIGNS
7F88	AA	09300		XOR	D	; H WITH D
7F89	FAAB7F	09400		JP	M, UNEQU	;M IF SIGNS NOT EQ
7F8C	E5	09500		PUSH	HL	SAVE LEFT
7F8D	B7	07600		OR	Α	
7F8E	ED52	09700		SBC	HL, DE	;COMPARE INTEGERS
7F90	E1	07800		POP	HL	RESTORE
7F91	381E	07700		JR	С,ОК	; NO
7F93	281C	10000		JR	z.ok	; SWAP
7F95	EB	10100	SWAP	EX	DÉ.HL	SWAP INT
7F96	7C	10200		LD	A.H	PUT LEFT BACK
7F97	45	10300		LD	B.L	
7F98	E1	10400		POP	HL	ADDR OF DEST
7F99	23	10500		INC	HL	····
7F9A	77	10600		LD	(HL) - A	:MSB
7F9B	2B	10700		DEC	HL	,
7F9C	70	10800		LD	(HL).B	;LSB
7F9D	78	10900			A.D	PUT RIGHT BACK
7F9E	43	11000			B.E	
7F9F	D1	11100		POP	DE	ADDR OF DEST
7FA0	13	11200		INC	DE	,
7FA1	12	11300		LD	(DE) - A	; MSB
7FA2	1B	11400		DEC	DE	
7FA3	78	11500			A- B	GET LSB
7FA4	12	11600			(DE) A	LSB
		11700	: TOGGL	E FLAG	·· , · ·	
7FA5	3EØ1	11800	,	LD	A. 1	
7FA7	A9	11900		XOR	C	;J-FLAG
7FA8	4F	12000		LD	C.A	
7FA9	1808	12100		JR	TESTJ	
7FAB	70	12200	UNEQU -	LD	A-H	CHECK HL SIGN
7FAC	E680	12300		AND	80H	,
7FAF	E2957E	12400		JP	P.SWAP	PIEHU (+) (HIGHER)
7FB1	F1	12500	UK	POP	H	IND SWAP
7FB2	D1	12600	0.0	POP	DE	,
7EB3	79	12700	TESTJ	I D	A. C	TEST J-FLAG
7FB4	B7	12800		OR	Δ	,
7EB5	2003	12900		JR	NZ - DECR	N7 IF SET
7FB7	23	13000		INC	H	NEXT INT
7FB9	23	13100		INC	HL	
7FR9	<u> </u>	13200		RET	•••	
7FRA	1B	13300	DECR	DEC	DF	PREV INT
7FRP	18	13400	2001	DEC	DE	
7FRC	C9	13500		RET		, TON OTEN
7500	5.	13600	,		START	
71 VH		10000			G110(1	

BASIC DRIVER

The following BASIC program will drive the sort routine. Of course, before calling the sort, the necessary parameters must be passed to it. These are the start address of the unsorted array, and the number of integers it contains for sorting. this is done in lines 1220-1250. Line 1210 pokes the entry point for the USR (0) function, which is used to invoke the machine language program. The destination of the parameters is known by the assembly program, and these are poked into their correct locations, least significant byte first. It is important at this point to mention that once the start address of the array is found by the VARPTR function, no other variables should be defined before calling the sort, since defining further scalar variables will generate an entry in the simple variable table, thereby shifting up the array, which will then have a different starting address. An option is provided for a visual presentation, with optional graphic characters to be included. If a delay is built in the machine language program, you may be able to see the workings of the algorithm as execution proceeds. To get the program running you should first protect memory at 32500 at the MEMORY SIZE question. Then load in the machine language routine using SYSTEM if a tape has been made, or by direct load with a

1000 ' PARTITION-EXCHANGE SORT DRIVER (QUICKSORT) 1010 ' B SIMSON, COPYRIGHT (C) 1980. AUTHOR: 1020 ' THE SORT PROGRAM IS WRITTEN IN Z-80 ASSEMBLY 1030 ' AND IS CALLED FROM THIS DRIVER BY A USR (.) CALL 1040 " AFTER START ADDR & LENGTH OF DATA PARAMETERS PASSED. 1050 ' SORT: ASCENDING 1060 ' DATA TYPE: SIGNED INTEGER ARRAY 1070 ' SUITABLE FOR: LEV2, 16K. 1090 DEFINTA-Y:RANDOM:Z\$="":R=0:D=0 1100 PRINTTAB(20); "QUICKSORT ROUTINE" 1110 INPUT"VISUAL DEMO (Y/N)";V\$:IFV\$="Y"THEN1360 1120 INPUT"SIZE OF DATA LIST";L 1130 DIM A(L) 1140 PRINTTAB(10); "GENERATING RANDOM INTEGERS..." 1150 FORI=1TOL A(I)=RND(1000)-200 ' GENERATE TYPICAL SIGNED DATA 1160 1170 NEXT 1180 PRINT"BEFORE SORT": GOSUB1320 1190 S=VARPTR(A(1)) 1200 GOSUB1210: GOT01260 1210 POKE16526, 10: POKE16527, 127 ' ENTRY=7FOAH 1220 POKE32512, S AND 255 START @ 7F●●H 1230 POKE32513, (S/256) AND 255 1240 POKE32514, L AND 255 ' LENGTH @ 7F02H 1250 POKE32515, (L/256) AND 255:RETURN 1260 PRINT: INPUT "HIT ENTER TO START THE SORT";S 1270 PRINT: PRINTTAB(10); "SORTING..." 1280 S=USR() ' SORT ROUTINE 1290 PRINT"AFTER SORT:" 1300 GOSUB1320 1310 END 1320 FORI=1TOL 1330 PRINTA(I); 1340 NEXT 1350 RETURN 1360 ' VISUAL DEMO DATA 1370 V\$="":INPUT"GRAPHIC & SPL CHARS INCL";V\$ 1380 IFV\$="Y"THENR=159:D=32 ELSE R=26:D=64 1390 CLS: PRINTCHR\$ (23): FORI=15360T016382STEP2 1400 POKEI, RND(R)+D:POKEI+1, 32 1410 NEXT 1420 S=15360:L=512 1430 GOSUB1210 1440 Z\$="":Z\$=INKEY\$:IFZ\$=""THEN1440 1450 S=USR() 1460 Z\$="":Z\$=INKEY\$:IFZ\$=""THEN1460 1470 END

EFFICIENCY

Quicksort performs well on large lists sizes. It does not have the same property as Bubble, etc., where sort times increase dramatically as the list size increases linearly. Its efficiency is similar to diminishing-increment and tree sort - of the order of n log n. This algorithm is one of the fastest that I have come across for average random data. There are algorithms around which will outperform this one, but only for special forms of data, e.g. the Radix sort is extremely fast on numbers which have a limited number of digits. Also, Quicksort does not perform very well on semi-ordered lists, because of uneven sublist sizes generated in the partitioning process, but for general cases of random data, it is one of the fastest. I performed some timing tests and compared them with those for the SupersnappX Sort published by Snapp Software, which compares their sort with Racet GSF. These are the results for 10,000 integers:

Racet GSF	59 seconds
SupersnappX	39 seconds
Quicksort	31 seconds

Snapp claim that SupersnappXis guaranteed to be the fastest in-memory sort on the market, which may be true since this version of Quicksort isn't on the market, but this comparison gives you an idea of its efficiency. Timing figures for some other list sizes for Quicksort are:

1,000 integers	2.2 seconds
2,000 integers	5.4 seconds
3,000 integers	8.1 seconds
4,000 integers	11.3 seconds

These were obtained on a Level 2 TRS-80 running at 1.7 MHz. In "The Art of Computer Programming, Volume 3 - Sorting and Searching", by D.E. Knuth, he compares Quicksort's efficiency with that of the diminishing increment sort, or "Shellsort" (discussed in the fourth article in this series). He shows that in the average case, Shellsort will take $15 \times n^{1.25}$ time units, and Quicksort 11.67*n*Log n - 1.74n time units, where n is the number of items being sorted. He also indicates that the maximum time for Shellsort is less than for Quicksort, a point which was referred to above. For comparison, the formalae for the average case were applied to give the following figures:

LIST SIZE	SHELLSORT	QUICKSORT
200	11281 Time Units	12018 Time Units
500	35465	35392
1000	84351	78873
2000	200621	173925
4000	477162	380206
10000	1500000	105 7 000

You can see from this table that Quicksort is more efficient in the average case as the list size increases.

TO SUMMARIZE

A method of sorting by exchange techniques known as the Partition-Exchange sort is very efficient for the average case. Items are sorted by placing them in their correct final position in the list, producing sublists on either side of it, which contain values that belong in that sublist. One sublist is stored while the other is processed in the same manner. A stack is used to store the bounds of the stored sublist.

This ends my discussion of different internal sorting algorithms, although it is not an exhaustive list of algorithms available. It should be understood that no one algorithm is the best, since different algorithms suit different applications, and some present trade-offs in memory usage that would not be acceptable in other applications.

The next article will discuss record sorting and external sorting techniques.

- 000000000 -

***** AUTOMATIC GRAPHICS PACKER - by Ken B. Smith *****

Those of you who struggled through my article on STRING\$ & THING\$ will by now have had enough practice at string packing to appreciate the time that even a simple graphic shape can consume. Even so the savings in memory and the increase in speed makes it all worthwhile. This program will enable you to draw a graphics design on the screen and automatically pack it into a string for you. This and other features make it well worth the effort to type in. What follows are simple instructions. The program itself is well REMed and those who wish will be able to decipher its logic without much effort. The REM's may be omitted for speed and ease when typing in.

Once the program is loaded and RUN the screen will request information on 'STRING NUMBER AND LENGTH TO PACK'. If the buffer contains information then answer this question with the string to pack and number of bytes to transfer but if you are starting afresh, either with a new design or a whole new run, then just press ENTER.

Once this question has been answered the screen will clear and a graphics pixel will be flashing in the centre of the screen. This single flashing pixel indicates the CURSOR Mode and all commands are available from this section. You may move this cursor around using the arrow keys, holding down the SPACE BAR will erase or leave a blank. The following commands are available from this mode using the appropriate key. (Be sure to have upper-case selected).

(P) - Pack the design on the screen to the buffer for subsequent transfer to a string. The buffer has a maximum capacity of 250 bytes so do not try too big a design to start with or problems will result from the buffer eating the program.

(C) - Clears the screen and leaves the cursor in the last position.

(R) - Repeat the contents of the buffer onto the screen. Useful for progressive animation sequences.

(A) - Enters the ASCII mode. This will show a large cursor which will respond to arrow keys to move without altering the existing graphics, or will place any selected keyboard characters onto the screen. Pressing ENTER returns control to the CURSOR mode. Auto repeat is available after about a third of a second.

(J) - If you have a printer with graphics capability this will dump the screen to paper. Initialising your printer to the correct mode is your problem. The dump is straight ASCII and emulates the JKL features of NEWDOS. As this information is taken straight from the screen, no lower case is acceptable and this is the reason the LC is not available in the ASCII mode.

.

Once the pack command has been given, the contents of the screen will be transferred to the buffer. when this is completed you will be informed of the number of bytes transferred. Edit the string array to which you wish this design to be copied to the exact number of bytes required. (You won't be able to later !!!!). The line number equals the string number +10. To aid with this chore the target strings are set to 100 bytes. Delete or add enough characters to suit your design and RUN. This time answer the question about target and length with this information. The buffer will be transferred to the target string. When this is completed, you will be back to BASIC. RERUN the program without an entry for the first question and you will be ready to make your second design.

Once you have completed your designs, merely delete all those lines not required and write your program on top of the packed strings. DO NOT ATTEMPT TO EDIT THE PACKED LINES.

This program is a fully functioning utility, but there are many other extras that could be added. Please feel free to add anything you wish, I would be pleased to see anything extra you might add. For myself I have completely automated this program, it even edits its own line numbers, but how to manage that is altogether another story.....

- 000000000 -

0 '***** Automatic Graphics Packer ******** 1 ****** Written for Micro 80 ****** ***** 2 bу ******* 3 ****** Ken B Smith ****** 4 CLEAR500 : *** 500 Bytes for String\$ 5 CLS :'** Clear Le Screen 6 DEFINTA-K, M-Z : ** Set up the integers 7 A=15360 :'** Start of Screen Memory in A 8 C=191 :'** Whole graphic in C 9 DIMA\$(10) :'** Set up the array for A\$ 10 '** The Variables A(1) to A(9) all have 100 blanks 11 A\$(1)=" 11 12 A\$(2)=" .. 13 A\$(3)=" ... 14 A\$(4)=" н 15 A\$(5)=" 11 16 A\$(6)=" 11 17 A = (7) = "11 18 A\$(8)=" ... 19 A\$(9)=" 20 '** A\$ has 250 blanks and is the Buffer 21 A\$=" 22 Z9\$=" " :'** A dummy string used in the screen print 23 INPUT"STRING NUMBER AND LENGTH TO PACK "; ZZ, Z1 24 CLS 25 X=63:Y=23 :'** Start positions for cursor 26 F\$=INKEY\$:'** Strobe Keyboard and result to F\$ 27 IFF\$="A"THEN48 :'** Was it an "A" for ASCII mode ? 28 IFF\$="C"THENCLS:GOTO26 :'** or a "C" for CLS 29 IFF\$="P"THEN67 :'** or a "P" for PACK 30 IFF\$="J"THENGOSUB108:GOTO26 :'** or"J" for Screen Print 31 IFF\$="R"THENCLS:PRINT@468,A\$:'** or an "R" for REPEAT 32 B=PEEK(14590) :'** Load B with cursor arrow PEEK 33 '** If B=O then Flash Cursor and back to get next job 34 IFB=OTHENRESET(X,Y):FORX1=1T010:NEXT:SET(X,Y):GOT026 35 '** Check for 00001000 = Up arrow and dec Y but not < 0 36 IFBAND8THENY=Y-1:IFY<OTHENY=0 37 '** Check for 00010000 = Down arrow and inc Y but not > 47 38 IFBAND16THENY=Y+1:IFY>47THENY=47 39 '** Check for 0010000 = Left arrow and dec X but not < 0 40 IFBAND32THENX=X-1:IFX<OTHENX=0 41 *** Check for 0100000 = Right arrow and inc X but not > 127 42 IFBAND64THENX=X+1:IFX>127THENX=127

PAGE 17

43 SET(X,Y) : *** SET the new value for X & Y 44 *** Check for 1000000 = Space Bar then RESET pixel and pause 45 IFBAND128THENRESET(X,Y):FORX1=1T05:NEXT 46 GOTO26 :'** Back for more inputs 47 '** The ASCII input section 48 D=PEEK(A) : ** Peek the Screen location in A, result in D 49 POKEA,C :'** Poke the Graphic Byte 191 into Screen Location 50 B\$=INKEY\$:'** INKEY\$ result to B\$ 51 '** If there is nothing from the Keyboard, Flash cursor 52 IFB\$=""THENPOKEA, 32:FORI=1T010:NEXT:POKEA, D:GOT048 53 B=ASC(B\$) :'** There was an entry. ASCII code to B 54 IFB=13THENPOKEA,D:GOTO26 :'** Was it an ENTER, then back 55 '** If entry a valid character, POKE it in and inc A 56 IFB>31ANDB<91THENPOKEA, B:A=A+1:IFA>16383THENA=16383 57 *** The following are cursor controls. POKEA,D = Skip 58 '** If entry was Left arrow then dec A and check A>15359 59 IFB=8THENPOKEA, D: A=A-1: IFA<15360THENA=15360 60 '** If entry was Right arrow then inc A and check A<16384 61 IFB=9THENPOKEA, D: A=A+1: IFA>16383THENA=16383 62 '** If entry was Down arrow then inc A+64 (1 line) and check 63 IFB=10THENPOKEA, D: A=A+64: IFA>16383THENA=A-64 64 *** If entry was Up arrow then dec A-64 and check 65 IFB=91THENPOKEA, D: A=A-64: IFA<15360THENA=A+64 66 GOT048 :'** Back again 67 IFZZ<>OTHEN100 :'** Is there a String Number from the entry 68 LO=VARPTR(A\$) :'** There wasn't so address of buffer to LO 69 LD=PEEK(LO+1)+256*PEEK(LO+2) :'** Now the address of A\$ 70 IFL0>32768THENLO=L0-65536 :'** Check for INT limits 71 L1=LO :'** Transfer to spare L1 72 FORX=15360T016383STEP64 :'** The whole screen, in lines 73 FORZ=X+63TOXSTEP-1 :'** now the lines, one byte at a time 74 IFPEEK(Z)=320RPEEK(Z)=128THENNEXT:NEXT:GOT094 :'** Blanks? 75 C1=Z :'** First non blank character. Position to C1 76 FORZ=XTOC1 :'** Now forwards along the line 77 IFPEEK(Z)=320RPEEK(Z)=128THENNEXT :'** to check for blanks 78 C=Z :'** First entry on line non-blank to C 79 FORZ=CTOC1 :'** Characters in the range C to C1 on this line 80 POKELO, PEEK(Z) : *** Poke them to the buffer string (A\$) 81 LO=LO+1 :'** Increment the buffer string pointer 82 NEXT : ** Continue till that line completed 83 XX=X+64 :'** Go down one line 84 FORZ=XXTOXX+63 :*** And check it 85 IFPEEK(Z)=320RPEEK(Z)=128THENNEXT:NEXT :*** Check for blanks 86 POKELO,26 : ** Put in a line feed 87 LO=LO+1 :'** And don't forget the buffer counter 88 C2=Z :'** Hold present position in C2 89 FORZ=C2TOC1+64 :'** End of top line to start of next 90 POKELO,24 :'** Poke in backspace characters 91 LD=LO+1 :'** And increment the pointer each time 92 NEXT :'** until enough back spaces are entered 93 IFX<16380THENNEXT :'** End of Screen - No, another line 94 X=LO-L1 :'** Number of characters into X 95 CLS :'** Must I do that again 96 PRINT"THERE ARE ";X; "STATEMENTS IN THAT DRAWING 97 PRINT"EDIT THE REQ. LINE AND ENTER STRING NUMBER AND BYTES NEXT RUN 98 END :'** End of packing. Easy heh!!!! 99 '** Transfer from buffer (A\$) to selected string A\$(ZZ) 100 LO=VARPTR(A\$(ZZ)) :'** VLT address for selected string 101 LO=PEEK(LO+1)+256*PEEK(LO+2) :'** Actual address to LO 102 FORX=1TOZ1 :'** Number of Bytes to transfer 103 POKELO,ASC(MID\$(A\$,X,1)) :'** Transfer from A\$ to A\$(ZZ) 104 LO=LO+1 :'** Increment the counter 105 NEXT : *** Until finished Z1 bytes 106 END :'** End of transfer from buffer to target string 107 '** The Screen Print Subroutine. Emulates JKL feature 108 Z9%=VARPTR(Z9\$) :*** VLT address of dummy string Z9\$ 109 POKEZ9%,64 :'** Tell the interpreter it's 64 bytes long 110 FORZ8%=15360 TO 16383 STEP 64 :'** Screen in lines to Z8% 111 Z7%=Z8%/256 : *** Extract MSB of Z8% to Z7% 112 Z6%=Z8%-Z7%*256 :'** And the LSB of Z8% to Z6% 113 POKEZ9%+1,Z6% :'** Transfer LSB of screen line to VLT 114 POKEZ9%+2,Z7% :*** Transfer MSB of Screen line to VLT 115 LPRINTZ9\$:'** Now print that screen line as Z9\$ 116 NEXT :'** And continue till end of screen 117 RETURN :'** Then go back to caller 118 END :'** This really is the end

**** SOFTWARE SECTION*

***** VARIABLE WORKSHEET Peach and CC *****

It is good programming practice to document programs as an aid to understanding how they work. At the time of writing a program, this procedure seems totally unnecessary because you are thoroughly familiar with the program, with its variable usage, and with how it functions. However. if your memory is like mine, when you come back to it after a month or so to tidy it up or improve it, you find that familiarity is gone and that you waste a lot of time trying to recover it. With this program, first published in the April '81 issue, you can record much of this precious information systematically to serve as documentation for future reference (a printer is required).

- 0000000000 -

***** MILEAGE CALCULATOR Peach and CC *****

With the aid of this program, originally published in the July '81 issue, you can use your computer to keep a record of your car's fuel consumption. The program originally used cassette tape to store the information for a particular month and these modified versions do the same. The program allows you to enter the data, make projections about fuel requirements, etc. and to produce summaries.

(1)	month number	(4)	litres
(2)	km at start	(5)	cost
(3)	km at end		

The program then works out:

kms travelled (1) (4) m.p.q. (2) miles travelled (5) litres per 100 km (3) km/litre

After this is done the user is asked if the data is to be saved to cassette. When completed, the program returns to the menu.

The second function projects the number of litres of fuel required for a given trip, given the average km/litre and the distance.

The summary mode allows you to summarise a particular month, or the full year. Given the month, the program will search the data tape (provided one has been created) and when it finds the specified month it will load the data and summarise under these headings:

TOTAL COST TOTAL KM(TRAVELLED) AVG KM/LITRE TOTAL LITRES

When asked to do a summary for the year, the data for all the months is read and summarised. Note that in this mode, the last month read must be a 12 or an error will result.

- 000000000 -

***** CALENDAR - LEVEL II *****

A program that produces a calendar seems appropriate at this time of year and this one follows the same sort of logic that a person would if the exercise were to be done by hand. In order to produce any calendar you need two pieces of information:

- the day of the week when the year begins 2)
 - if the year is a leap year or not.

Given the year for which you want the calendar, there is an algorithm called Zeller's algorithm that generates a unique day number for any day this century starting from March 1st., 1900 (hence the restriction on the year). With that it is a relatively simple task to find the first piece of information. Knowing the year allows you to quite easily get the second. Although I have not tested this program for all the 99 years this century, my confidence in Zeller and modulo arithmetic leads me to believe the code in lines 650 - 670 is correct and it does produce an accurate calendar for 1983.

The remainder of the program is concerned with formatting the layout of the calendar and is reasonably self-explanatory. The subroutine at 220 generates the calendar in a numeric array YR and that at 440 converts this array to character strings for printing one line at a time. The title is printed by the subroutine at 290. Line 200 puts my EPSON into double-strike mode and line 750 restores it to normal, followed by a hard form feed (the System 80 does not pass

MICRO-80 PRODUCTS CATALOGUE

This catalogue contains a selection from the wide range of peripherals, interfaces, computers and software carried by MICRO-80 for your computer. If you don't see the item you want, contact us, we probably have it anyway!

MICRO-80 has been supplying customers throughout Australia and the Pacific region by mail-order for $2\frac{1}{2}$ years. Our customers find this a simple and efficient way to do business. You may place your order by telephone or by mailing the order form from any issue of MICRO-80 magazine. Generally, it takes about one week from receipt of order until despatch. 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 $2^{1/2}$ -3 weeks.

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 will be repaired by our own team of technicians in our Adelaide workshops.

TRADE-INS AND TERMS

MICRO-80 can accept your existing equipment as a trade-in on new equipment. We can also arrange consumer mortgage financing or leasing on larger hardware purchases. Contact us for details.

BOOKS

LEARNING LEVEL II

by David A. Lien



Written by the author of the Level I Users Manual, Learning Level II covers all Level II BASIC beyond Level I, plus much more. It shows you how to use the Editor, explains what the many error messages are really saying, and leads you through conversions of Level I programs to Level II.

Dual cassettes, printers, the Expansion Interface with clock and other features are explained in the same easy-tolearn style that made the Level I Manual famous. Learning Level *II* is an invaluable supplement to the TRS-80 Level II and System 80 manuals and is now only \$7.95 (plus \$1.20 p&p).

BASIC FASTER AND BETTER AND OTHER MYSTERIES



by Lewis Rosenfelder Basic is not nearly as slow as most programmers think. Basic Faster and Better shows you how to super charge your BASIC with almost 300 pages of fast, functions and subroutines. You won't find any trivial poorly designed "check-book balancing" programs in this book — it's packed with useful programs. Tutorial for the beginner,

instructive for the advanced, and invaluable for the professional, this book doesn't just talk . . . it shows how! Basic Faster and Better is \$32.50 (plus \$1.20 p&p).



THE CUSTOM TRS-80 AND OTHER MYSTERIES

by Dennis Bathory Kitsz Ever wanted to do things to your TRS-80 that Radio Shack said couldn't be done? How about reverse video, high resolution graphics, and audible keystrokes

Now enough? How about turning an 8-track into a mass storage device, making music, controlling a synthesiser, individual reverse characters, and a real-time clock just to name a few?

The Custom TRS-80 and Other *Mysteries* is packed with more than 290 pages of practical information and can be yours for only \$32.50 (plus \$1.20 p&p).

TRS-80 DISK AND OTHER MYSTERIES

by H.C. Pennington TRS-80 Disk and Other *Mysteries* is the definitive fix-it book for disk users. More than 130 pages of easy to read, entertaining and immensely useful information. Find out how to recover disk files, the layout of information on disks, memory maps, problem solutions . . . the list goes on! Many readers have saved days of work by recreating disk files that were unreadable. TRS-80 Disk and Other Mysteries, which has received favorable reviews in several magazines, is yours for only \$27.00 (plus \$1.20 p&p).



TRS-80. DISK

& OTHER MYSTERIES

THE LNW80 MkII MICROCOMPUTER

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

Completely software and hardware COMPATIBLE with the TRS-80 Model 1.

RESOLUTION • HIGH 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
- CP/M Disk Operating System.

Single and Double Density Disk Operation.

Supports 5¹/₄ inch or 8 inch Floppy Disk Drives.

• 48K RAM in TRS-80 mode plus 16K High Resolution graphics RAM.

64K RAM in CP/M mode plus 32K Banked in, usable in BASIC, plus the 16K High Resolution Graphics RAM.

• 4 MHz Z80A microprocessor – over twice the operating speed of the Model 1.

video outputs.

- 3 screen display modes:
 - 64 characters x 16 lines
 - 80 characters x 16 lines
 - 80 characters x 24 lines
- 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. 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 Includes:

 CP/M Disk Operating System Dosplus 3.4 Double Density Disk Op LNW Extended Colour Basic Interpreter 	perating System \$2750 INC.S.T.
HI-RES Green Phosphor Monitor	\$265 INC.S.T. \$1250 INC.S.T.
Two Singlesided 40 Track Double Density Disk in cabinet with power supply and cable	\$825 INC.S.T.

● HI-RES COLOUR (R-G-B) and B&W





THE BEST IN ENTERTAINMENT FROM **AMERICA'S TOP SOFTWARE HOUSES**

MICRO-80 now has in stock some of the best games and adventures written for the '80s. These programs are supplied on cassette for the Level II/16K TRS-80 Model I (or III). They are also suitable for the System 80 but sound may not be available unless a hardware modification has been fitted to reverse the roles of recorders #1 and #2. Limited stock is available at these prices.

COSMIC FIGHTER

FROM BIG FIVE

\$20.95 ATTACK FORCE

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.

FROM ADVENTURE INTERNATIONAL \$26.50

ELIMINATOR

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

PLANETOIDS

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

ADVENTURELAND

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 your way out of the maze of pits?

PIRATE'S ADVENTURE

"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

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!

VOODOOCASTLE

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

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 postman deliver a bottle of blood? You'll love this adventure, in fact you might say it's Love at First Byte.

STRANGEODYSSEY

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?

MISSILE ATTACK

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.

In this fast paced, m/I game 8 alien 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.

SPACE INTRUDERS

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

MYSTERY FUN HOUSE

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

PYRAMID OF DOOM

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

GHOSTTOWN

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

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!**

SAVAGEISLAND

Part 2 — After struggling through Part 1, you have the consolation of knowing it's 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

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!

\$26.50

\$26.50

\$26.50

\$26.50

\$26.50



\$26.50

\$26.50

\$26.50

\$26.50

\$26.50

\$26.50

\$26.50

CAT 4

\$26.50

\$20.50

\$26.50

\$26.50

MICRO-80 PRODUCTS

BUY YOUR MODEL 3 FROM MICRO-80 AND SAVE \$00'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 supplies to avoid screen jitter and overheating. The disk controller boards used incorporate special compensation circuitary 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 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, DOSPLUS 4.0 DISK OPERATING SYSTEM 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. 	\$5895
WINCHESTER DISK DRIVE SUB-SYSTEM5MByte10MByte	\$2995 \$3750
This Winchester Disk Drive sub-system provides either 5 or 10 Megabyte of reliable, high speed storage. It connects to any standard Model 3 equipped with one or more floppy disk drives and does not void the Tandy warranty. Complete with DOSPLUS 4.0 Disk Operating system.	

Prices include Sales Tax and are subject to change without notice. Prices are FOB Adelaide. Add \$20 road freight anywhere in Australia. All computers and peripherals carry MICRO-80's Australia-wide. 90-day warranty covering parts and labour.

NEW

THE SMITH-CORONA TP-1 DAISY WHEEL PRINTER A low-priced letter quality printer



Ideally suited for small businesses or the home user, the TP-1 is a microprocessor controlled, correspondence quality printer that prints fully formed characters at an average print speed of 12 characters per second. This simple to operate, compact printer is compatible with most microcomputers and comes with the standard Centronics parallel interface (an optional serial data interface is available) and features:

- 128 ASCII Character Set (88 printable)
- 10 CPI or 12 CPI character spacing
- 105 characters per line (or 126 in 12 pitch)
- Handles letter and legal sized paper (up to 13" wide)
- Variable line spacing and impression control
- Prints original plus up to three copies

\Leftrightarrow NEW \Leftrightarrow NEW \Leftrightarrow NEW \Leftrightarrow NEW \Leftrightarrow

Tractor Feed Mechanism for Daisy Writer (ET121) \$380

Now your Daisywriter 2 can handle continuous stationery. Ideal for invoices and statements, etc.

Prices include Sales Tax and are subject to change without notice. Prices are FOB Adelaide. Add \$20 road freight anywhere in Australia. All equipment carries MICRO-80's Australia-wide 90-day warranty covering parts and labour.

DISK OPERATING SYSTEMS & DEVELOPMENT SOFTWARE

You can increase your programming productivity, the execution speed and 'user friendliness' of your programs by using an enhanced Disk Operating System (DOS). Together with the other utility software, you can get the most from your disk drives.

DOSPLUS 3.3

\$99.95

\$\$149.95

\$52.95

(Specify Model I single or double density or Model III) An economic DOS intended for the first-time user and requiring single-sided disk drives. (The TRSDOS & DISK BASIC MANUAL is required to supplement the DOSPLUS manual).

DOSPLUS 3.4

(Specify Model I single or double density or Model III)

With a high degree of compatibility with TRSDOS, DOSPLUS 3.4 supports single- or double-sided, single or double density, 5" or 8" disk drives with any track count (up to 96). Suitable for the first-time or experienced user wanting a fuss-free, bug-free, easy to understand but very powerful DOS which supports variable length records up to 255 bytes long. Comes with a stand alone manual.

ENHBAS

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 I or III, disk or cassette — specify which when ordering.

NEWDOS 80 VERSION 2.0

(Specify Model I or Model III)

\$169.00

\$20.95

\$26.50

\$26.50

Newdos 80 suits the experienced user who has already used TRSDOS, understands the manual and is prepared to learn the somewhat complicated syntax of one of the most powerful DOS's available. With the correct hardware, Newdos 80 supports any mix of single- or double-sided, single or double density, 5" or 8" disk drives with track counts up to 96. It provides powerful, flexible file handling in BASIC including variable length records up to 4096 bytes. Definitely not for the beginner.

MASTER DISK DIRECTORY

FIND THE PROGRAM FAST!! PAYS FOR ITSELF BY RELEASING REDUDANT DISK SPACE!! MASTER DIRECTORY records the directories of all your individual disks onto one directory disk. Then it allows you examine them, find an individual file quickly, list files alphabetically, weed out redundant files, identify disks with free space, list files by extension, etc., etc. This program is invaluable for the serious disk user and will pay for itself many times over.

THE FLOPPY DOCTOR/MEMORY DIAGNOSTICModel I Disk \$36.50Model III Disk \$43.50

THE MICRO CLINIC offers two programs designed to thoroughly check out the two most trouble-prone sections of the TRS-80 — the disk system (controller and drives) and the memory arrays. Both programs are written in Z80 machine code and are supplied together on diskette for a minimum 32K, one disk system. Specify Model I or Model III.

MORE ENTERTAINMENT SOFTWARE

ADVENTURE HINT BOOK

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

LABYRINTH

\$26.50

\$10.95

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.

ASYLUM

Asylum places you in a cell, you have to escape. It's harder than it sounds, lots of hazards will be encountered.

DEATHMAZE 5000

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

New SEIKOSHA GP-100A GRAPHICS PRINTER



PRICED AT ONLY \$475

If you have delayed buying a Printer, then now is the time to reconsider. The Seikosha has been designed for simple operation and puts full dot addressable graphics at your command. You can repeat a column of data as many times as you like with just one command. Double-width character and dot addressable positioning are software controlled. Other features:

- Automatic Printing avoiding data loss when the maximum line length is exceeded.
- Allows mixing graphics, regular and double width characters on the same line.
- Up to 50 characters per second.
- Standard Centronics type Parallel interface.
- Self-test mode.
- Optional RS-232-C Serial Interface.

PRINTERS GALORE AT UNBEATABLE PRICES

MICRO-80 has a range of printers to suit every requirement from dot-matrix to correspondence quality daisywheel. Choose 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*
EPSON	MX-80III	DM	80	80	Y	FULL	F/T	HI-RES	Р	1	\$ 999	\$ 8.35
EPSON	MX-100III	DM	132	100	Y	FULL	F/T	HI-RES	Р	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	Υ	FULL	F/T	HI-RES	Р	1	\$2220	\$18.57
MICROLINE	84	DM	132	200	Υ	FULL	F/T	HI-RES	S	1	\$2340	\$19.57
C ITOH	8510	DM	80	112	Y	FULL	F/T	HI-RES	Р	1	\$ 999	\$ 9.19
C ITOH	M1550	DM	132	120	Y	FULL	F/T	HI-RES	Р	1	\$1499	\$12.54
OLIVETTI	PRAXIS35	\mathbf{DW}	100	6	Ν	FULL	F	NO	Р	1	\$ 895	\$ 8.33
OLIVETTI	ET121	\mathbf{DW}	132	12	Ν	FULL	F	NO	Р	2	\$1500	\$12.55
OLIVETTI	ET221	\mathbf{DW}	132	16	Ν	FULL	F	NO	Р	2	\$2650	\$22.17
ITOH	F10 40P	$\mathbf{D}\mathbf{W}$	132	40	Y	FULL	F	NO	Р	2	\$1950	\$16.31
ITOH	F10 40S	DW	132	40	Y	FULL	F	NO	S	2	\$2190	\$18.32

NOTE: The following symbols are used:

TYPE+	DM = DOT MATRIX
	DW = DAISYWHEEL
BI DIRECTIONAL	Y = YES N = NO
LOWER CASE	FULL – means Lowercase descenders to
	below line
PAPER FEED	F — means Friction Feed
	T - means Tractor Feed
	F/T – means both Friction and Tractor
	Feed included in 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.

VOLUME 3 NO. 8 (JULY, 1982)

the Form Feed character (OCH) through the printer driver). Both of these lines may need to be changed depending on your computer and printer. In the double-strike mode the routine at 440 does not slow the process down, but in normal mode the printer waits for a time as each new line is constructed. The alternative would be to produce the calendar in a string array, which would use much more string space and need more memory than the 4K used by this method.

- 000000000 -

***** EXTENDING THE BASIC INTERPRETER: HEX CONSTANTS by Roger Bowler *****

As a Level II user, you may occasionally have grown tired of converting hex RAM addresses to decimal before you can use them in BASIC; or you may find programs difficult to understand because they refer to RAM addresses in decimal. Maybe you have even looked enviously at the Disk Basic manual and wished you could write statements such as:

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

Well, in the world of software nothing is impossible, and after running this short program, you will be able to write and run programs containing hex constants as in the above example. First power up the 80 with a MEMORY SIZE of 20416 (4K RAM) or 32704 (16K RAM), then CLOAD and RUN this program. All the program does is to POKE some machine code into the 60 bytes of RAM immediately above the MEMORY SIZE limit, then it executes a couple of sample statements containing hex constants, just to demonstrate that the machine code works. Now you no longer need to keep the BASIC program resident, so you can type NEW and your '80 will quite happily accept programs with hex constants in them. Incidentally, if you are typing this program in from a listing, you would be well advised to CSAVE it before running it. Of course, this program will have to be CLOADed and RUN again every time you power on.

Now how does it work? We are fortunate that the BASIC interpreter used by the '80 was designed to allow extensions such as this. There are about 2 dozen strategically located points during interpretation where the ROM code will jump out into the RAM to see whether the user wishes to do any additional processing. This is called an "exit" from the ROM, and in a normal cassette system all these exits are unused. The exit we are using is taken by the interpreter whenever it finds an expression in your program which starts with a "&"; it consists of a call out to RAM location 4162H. Now unless you have Disk Basic loaded, location 4152H contains a jump back into the ROM to a piece of code which prints the "L3 ERROR" message. What we have done is to replace this jump by a jump to our own machine code in high RAM; our code will function as an extension of the BASIC interpreter by packing the hex characters following &H into their binary equivalent, and passing the result back to the ROM.

When we come in to our code, the ROM gives us the address of the "&" character in the HL register pair. In return for this generosity, we have to update the HL register pair to point <u>past</u> the hex characters, and we store the result (i.e. their packed value) as a 16 bit integer with LSB at location 4121H and MSB at 4122H. We also have to set location 40AFH equal to 02 to tell the ROM that the value we're sending him back is an integer (as opposed to single precision or string etc.). Then we can do a simple RET to continue interpretation.

So now you can run programs with &H constants in them, and what's more, the syntax is compatible with Disk Basic!

Although the BASIC program shows the machine code being stored in high memory, this is only an example, since the machine code is completely relocatable. This means it doesn't contain any references to addresses within itself; all jumps are relative using the "JR" or "DJNZ" instructions. So you may well prefer to save the machine code somewhere else in RAM, to avoid having to protect high memory. One way of doing this is to shift the "start of basic" pointer at 40A4 to point to 4325 instead of 42E9. This is done by:

POKE 16548,37:POKE 16549,67:NEW

(The NEW forces BASIC to recompute its pointers to account for the new start address). Ignore the SN ERROR. Now you have freed up locations 17129-17188 (42E9-4324) into which you can store the machine code. Happy hexing!

I always build USR routines using dynamic string packing. This technique (described in Appendix H of the Level II Basic manual) involves packing the machine code into a string variable from DATA statements during program initialisation. It has a number of drawbacks, not the least of which being that it is rather S-L-O-W. But these I can live with. What I do draw the line at is writing machine code in DECIMAL; that is something I cannot do! Hasn't this crazy computer heard of HEX?

Well, mine has; I write my machine code like this:

200 DATA CD7F0A, 5E23, 56, EB, C39A0A, *

and I include a little basic subroutine at the end of each program to pack these DATA statements into strings.

The subroutine reads each "instruction" from the data as a string, then decodes and packs each pair of hex digits into a byte which it appends to string P\$. The subroutine continues to build up the string until it reads an asterisk. Then by means of a quick

POKE 16526,PEEK(VARPTR(P\$+1)=POKE 16527,PEEK(VARPTR(P\$)+2)

we are all ready to do a USR call.

It is quite easy to set up several independent USR routines in this way; I just make sure each set of DATA statements ends with an asterisk, and save the routine in another string variable before calling the hex pack subroutine again.

One thing to watch when using dynamic string packing - BASIC tends to move strings around in RAM (to reclaim unused string space), so set the VARPTRs immediately before making the USR call.

- 000000000 -

***** SERIES IMPEDANCE CALCULATIONS L2/16K - by W.G. Heath *****

This program illustrates one of the fundamental formulae connected with electrical problems and may be of interest to electrical engineering students and amateur radio enthusiasts.

As explained on the first two displays on the video screen, a wide variety of problems are solvable and not confined to the general form of series resistance, inductance and capacitance alone. Some of these elements need not be present but the program is still adaptable, it will also operate the OHM's law solutions of current, resistance and voltage (I=E/R)

Following an indication of the scope of the program, a display list of all the variables involved is shown. Then follows the request for input of known and unknown values related to the problem in question.

The calculations are then made and the individual variables are displayed together with their measured or calculated values.

A graphical display of the general series circuit only, follows, plus an outline of the various vector relationships and the impedance diagram.

- 000000000 -

Shown below are views of the screen:

>>> SERIES IMPEDANCE CIRCUIT <<<

A WIDE VARIETY OF ELECTRICAL CIRCUIT PROBLEMS OF THIS CATEGORY ARE SOLVEABLE DEPENDING UPON THE KNOWN AND UNKNOWN VAR-IABLES AVAILABLE.

THE PROGRAMME WILL CALCULATE ALL THE VARIABLES SHOWN IN THE FOLLOWING SCHEDULE SUBJECT TO APPROPRIATE INPUT VALUES.

IT WILL HANDLE TWO UNKNOWNS SIMULTANEOUSLY PROVIDING ONE UNKNOWN IS EITHER I, E, OR R, AND THE SECOND IS EITHER L, C OR X. MAKE NO ENTRY FOR THE UNKNOWN VALUES. I.E. PRESS $\langle \text{ENTER} \rangle$ ONLY WHEN INPUT? FOR THESE VARIABLES IS CALLED FOR.

< ENTER >?

THE PROGRAMME CAN BE USED FOR PROBLEMS WHERE NO CAPAC-ITANCE OR, ALTERNATIVELY, NO INDUCTANCE IS PRESENT.

INDUCTANCE CAN BE CALCULATED FOR AN INDUCTIVE CIRCUIT WITHOUT CAPACITANCE IF I, E, AND R, ARE KNOWN VALUES.

FINALLY THE PROSRAMME WILL ALSO OPERATE A SIMPLE OHM'S LAW SOLUTION (I = E/R) by inputting 0 (zero) for both C (capacitance) and L (inductance) simultaneously.

AFTER ALL CALCULATIONS HAVE BEEN COMPLETED A FURTHER TWO SCROLLS OF THE SCREEN CAN BE MADE WHICH WILL DISPLAY A CIR-CUIT DIAGRAM OF THE GENERAL SERIES CIRCUIT FOLLOWED BY A VECTOR DIAGRAM AND IMPEDANCE DIAGRAM < ENTER >?



***** DR. WHO ADVENTURE LII/16K or 32K/Disk

This adventure will just run in a Level 2 16K machine provided the following instructions are followed. It will run also with Disk BASIC (with at least 32K of memory).

For the disk version, lines 720 and 730 are replaced by the two lines listed separately, the program supplied on the distribution disk has already been converted for you.

This is an unusual adventure, using a data file in a manner similar to the Epyx games. The data file is the file called DRWHO/DAT on the distribution disk. This was created using the initialization program listed. If you are entering this program from the magazine you will have to run this program to create a data file either on disk or cassette.

The program uses a full 16K which is why it needs the data file.

For a cassette based system, use the following procedure:

- (1) Type in the initializer, check it, and SAVE it to tape.
- (2) Type in the adventure, check it, and SAVE it to tape (a different one...leave this positioned to just after the program).
- (3) Reload the initializer and swap back to the tape the adventure is on (it should be positioned just after the program).
- (4) Run the initializer and answer the question with T for tape.
- (5) Rewind the tape, reload the adventure, and run it.
- (6) Disk users type in both programs and SAVE them to disk. Add the lines provided to make the cassette version work on disks. Run the initializer and answer D. Then run the adventure.

NOTE: People with more than 16K and disk users may find it advantageous to merge the data in the initializer with the main program. Although no explanation of how to do this is given here, it should not be too difficult.

INSTRUCTIONS.

After Dr. Who collected the Key to Time and defeated the Black Guardian, he received many praises and went on to greater things. The Key itself was again broken into its component pieces and scattered throughout the universe.

But the dark forces threaten, and in order to save the universe, the Timelords again need the Key. You have been chosen to go forth and locate it for them. You will be given a TARDIS (rather old and unreliable, but the best available) that has the coordinates of the planets on which the six parts are located pre-programmed into it. By RESETing its controls you can travel between the six planets and Galafry. As usual, the six parts are disguised as other things, and you will have to use your intuition to figure out which is which. (There is a way to tell...)

All the planets are inhabited, and most inhabitants tend to be antisocial. Whether you TALK to them, HIDE from them, kill them, OFFER them gifts of appeasement, or simply ignore them is up to you. Most objects are obvious, but some are hidden and have to be SEARCHed for. Only one key part is on any one planet. Beware the maze on Peladon...

You can use commands of up to 64 characters. The program will ignore any words it doesn't understand. Commands can be one, two or three words long.

When you find all the parts (or think you have), take them back to the throne room on Galafry to win.

The program only needs to read in data once. You will only need to rewind the data tape if you type BREAK or answer no to the "Another game?" question.

- 000000000 -

***** LOWER CASE CONVERTER FOR BASIC PROGRAMS by D.M. Wright *****

After you have fitted a lower case conversion kit to your TRS-80/System 80 and have a Driver routine operating to your satisfaction, you will no doubt look back at all those BASIC programs you have produced with only capital letters displayed. While many programs have only a few statements in the way of instructions to modify, the thought of virtually retyping some of the long Adventure type programs is most daunting.

The following Assembly language program goes a long way to solving the problem as it converts the characters inside PRINT statements into lower case with the exceptions of the first letter in the quotation and the first letter after a period and two spaces which is assumed to be a new sentence. It only alters characters within the PRINT statement and does not convert program commands and statements.

It will be necessary to finally edit the program to capitalise people's names or titles which occur in PRINT statemments such as system 80/trs-80 ... or ... commander smith ...

Conditional statements may also need editing as follows:

IF A\$ = "Yes" THEN 100 would need to be altered back to, IF A\$ = "YES" THEN 100 or perhaps better still, IF A\$ = "YES" OR A\$ = "yes" THEN 100

So with a little care in final editing you too can have small letters in abundance throughout your programs and across your screen.

The program can be assembled with an Editor/Assembler and can be relocated by altering the ORG in line 140 of the source listing to \emptyset BEl \emptyset H for a 32K system or to \emptyset FEl \emptyset H for a 48K system. Alternatively, the HEX dump (for 16k system) that follows the source listing can be entered with a low memory monitor.

For this program to work you must have lowercase installed in your machine and a lower case driver active.

110 MEM 2 65535

To load from tape:

- 1. Answer MEMORY SIZE? (READY?) with: 32272.
- 2. Load the machine language using SYSTEM.
- 3. Type: /32272 (ENTER/NEWLINE) to start.

To load from distribution disk:

- 1. BASIC, 65040 (ENTER/NEWLINE)
- 2. CMD "CONVERT"
- 3. Type: /65040 (ENTER/NEWLINE) to start.

The program will initialize and display a copyright message. Then load the BASIC program you want converted and when ready, type:

SYSTEM (ENTER/NEWLINE) /32272 (ENTER/NEWLINE)

You can use the converter as many times as you like. The CMD file and EDTASM file supplied on the distribution DOS are for a 48k system. For a 32k system and suitably assembled machine language program, use 48656 as the memory size and entry address.

- 000000000 -

		00010	ş						49 S
		00020	4	LOWER CA	ASE CONVE	ERTOR V	/ER 2.0		
		00030	9						a second a set
		00040	7	CREATED		17 JUNE	<u> </u>		602.53
		00050	5	BY		DENNIS	WRIGHT		and the second
		00060	ş	1 SNEAD	COURT	DINGLEY	/ 3172	NES -	くんどり吊ささ
		00070	7	(C)	ALL R	IGHTS RE	ESERVED		a di kacamatan kacamatan di kacam Kacamatan di kacamatan di kacamat
		00080	5						
40A4		00090	BASBEG	EQU	40A4H				
40F9		00100	BASEND	EQU	40F9H				
06CC		00110	BASIC	EQU	06CCH				
4020		00120	CURSOR	EQU	4020H				
		00130	* *						
7E10		00140		ORG	7E10H				
7E10	CDC901	00150	INIT	CALL	01C9H				
7E13	21BC7E	00160		LD	HL, MESG:	1	;PRINT	MAIN MESSAGE	
7E16	11403C	00170		LD	DE, 3C40	н			
7E19	018000	00180		LD	BC,80H				
7E1C	EDBO	00190		LDIR					
7E1E	213C7F	00200		LD	HL, MESG	2			
7E21	11003D	00210		LD	DE, 3DOO	н			
7E24	014000	00220		LD	BC,40H				
7E27	EDBO	00230		LDIR					
7E29	21803D	00240		LD	HL,3D80	н	; PLACE	CURSOR	
7E2C	222040	00250		LD	(CURSOR),HL	;BELOW	MESSAGE	
7E2F	ED5BA440	00260		LD	DE, (BASI	BEG)			•
7E33	2AF940	00270		LD	HL, (BASI	END)			
7E36	B7	00280		OR	A		; CLEAR	FLAGS	
7E37	ED52	00290		SBC	HL,DE		;FIND	PR OG RAM LENGTH	1
7E39	7D	00200		LD	A,L		; ROUT I	NE CHECKS	
7E3A	FE02	00310		CP	2		;IF PR	DGRAM LENGTH	
7E3C	2005	00320		JR	NZ,CONT		;IS ON	LY TWO BYTES	

р • - -

•

.

•

7E3E 7C	00330	LD	A.H	;	LONG WHICH
7E3E EE00	00340	CP	ó		INDICATES
7E41 295D	00350	JR	7 - NOPROF	3	NO BASIC PROGRAM LOADED
7E41 2000	00360 00			- ,	PUT PROGRAM START ADDRESS
7E43 ED	00300 00			,	
7E44 23	00370	INC		5	
7E40 ZS	00380			9	BOTHE LINE
/E46 23	- 00340	INC.	HL	5	PUINTERS
7E47 CD997	E 00400 SE	EARCH CALL	NXIBYI	. 5	GET NEXT BYTE
7E4A FE22	00410	CP	22H	;	;QUOTE?
7E4C 20F9	00420	JR	NZ,SEARC	сн ;	TRY AGAIN
7E4E CD997	E 00430 FS	STCAP CALL	NXTBYT	;	ROUTINE LEAVES 1ST LETTER U
/C					
7E51 FE22	00440	CP	22H	;	2ND QUOTE?
7E53 28E2	00450	JR	Z.SEARCH	، ۱	SEARCH FOR NEXT QUOTATION
7E55 EE41	00460	CP	41H	-	IESS THAN "A"?
7557 7055	00470	JR		· ·	SEARCH FOR 1ST CAPITAL
750 550	00490	CP	500	•	$\frac{1}{2}$
	00400	10		· · · · ·	CEADELLEOD 1CT CADITAL
TEDB SVF1			NUTOVT	-1	SEARCH FUR IST CAFILAL
7E3D CD997	E 00200 FM	NDLEI CALL	NXIBYI		
7E60 FE22	00510	CP	22H		;2ND QUUIE?
7E62 28E3	00520	JR	Z, SEARCH		SEARCH FUR NEXT QUUTATION
7E64 FE2E	00530	CP	2EH		;PERIOD?
7E66 280D	00540	JR	Z, NUSENT	Γ;	CHECK FOR NEW SENTENCE
7E68 FE41	00550 CC	DNVRT CP	41H		;LESS THAN "A"?
7E6A 38F1	00560	JR	C, FNDLET	Г ;	FIND LETTER
7E6C FE5B	00570	CP	5BH		GREATER THAN "Z"?
7E6E 30ED	00580	JR	NC. ENDLE	т	
7E70 C620	00590		A. 20H		CONVRT TO LOWERCASE
7572 77	00600	1.02			STOPE BACK IN PROCEAM
7572 1050	00000			,	STORE DACK IN FROORAM
7E/3 18E8			FNDLET		
/E/5 CD99/	E 00620 NL	USENI CALL	NXIBYI		
7E78 FE22	00630	CP	22H		;2ND QUOTE?
7E7A 28CB	00640	JR	Z,SEARCH	ι;	SEARCH FOR NEXT QUOTATION
7E7C FE20	00650	CP	20H	1	; SPACE?
7E7E 20E8	00660	JR	NZ,CONVF	кт ;	THEN CONVERT
7E80 CD997	'E 00670	CALL	NXTBYT		
7E83 FE22	00680	CP	22H		2ND QUOTE?
7E85 28C0	00690	JR	7.SEARCH		SEARCH FOR NEXT QUOTATION
7E87 FE20	00700	CP	20H		2ND SPACE2
7500 2000	00710	TP		эт т	TUEN CONVERT
7607 2000	00710			`	, THEN CONVERT
7E0B 10C1	00720		FBICHE		JEARCH FUR FIRDI CHFIIAL
7E8D CI	00730 LI	INEND PUP	BL		JULEAR STALK
7E8E 23	00740	INC	HL	5	JUMP UVER
7E8F 23	00750	INC	HL	1	LINE POINTER
7E90 7E	00760	LD	A,(HL)		CHECK FOR
7E91 FE00	00770	CP	0	1	;END OF PROGRAM
7E93 2819	00780	JR	Z, PROENI) ;	RET TO BASIC
7E95 23	00790	INC	HĹ		JUMP OVER
7E96 23	00800	INC	HL		LINE NUMBER
7E97 18AE	00810	JR	SEARCH		
7599 23	00820 N	YTRYT INC	HI	:	
7500 75	00020 14/				I OAD NEYT BYTE
	00030		n, (nc/		FUR OF DRODAM LINED
7E78 FE00	00840				END OF PROGRAM LINE?
7E9D 28EE	00850	JR	Z,LINENI	ן נ	GU JUMP LINE PUINTERS
7EAF CA	00860	REI			
7EA0 217C7	F 00870 NG	OPROG LD	HL,MESG3	3	PRINT 'NO PROGRAM' MESSAGE
7EA3 11003	D 00880	LD	DE, 3DOOH	4	
7EA6 01400	0 00890	LD	BC,40H		
7EA9 EDBO	00900	LDIR			
7EAB C3CCC	6 00910	JP	BASIC	:	RETURN TO BASIC
7EAE 21BC7	F 00920 PF	ROEND LD	HL.MESG4	4	PRINT 'COMPLETED' MESSAGE
7EB1 11003	D 00930		DE. 3000		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
7EB4 01400	0 00940	10	BC 40H	•	
7507 5000	00950		20,401		
	00730		DACTO		
TEBY LOLL		JP 5004 5554	BHSIC		RETURN TO BASIC
TEBC 20	00970 ME	ESGI DEFM		LOWER CAS	SE CUNVERTER FUR BASIC PRUGR
AMS VERS	1UN 2.0	-			
7EFC 20	00 98 0	DEFM	2	CREATED I	BY DENNIS WRIGHT (
C) 17 JL	INE 1982	2			
7F3C 20	00 99 0 ME	ESG2 DEFM	,	PROGRAM	CONVERTING RESIDENT BASIC P
ROGRAM	-	,			
7F7C 20	01000 MF	ESG3 DFFM	,		NO BASIC PROGRAM LOADE
D	:	,			TO STOLD TROUTHIN COMPL
	01010 M	ESG4 DEEM	,		CONVERSION COMPLETE
		2001 DENN 7			CONVENCION CONFLETE
7E10	01020	END	TNITT		
1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 -	01020	END	11411		

7E10:

7E20:

7E30: 5B A4 40 2A F9 40 B7 ED 52 7D FE 02 20 05 7C FE 7E40: 00 28 5D EB 23 23 23 CD 99 7E FE 22 20 F9 CD 99 7E FE 22 28 F2 FE 41 38 F5 FE 5B 30 F1 CD 99 7E 7E50: 2E 28 0D FE 7E60: FE 22 28 E3 FE 41 38 F1 FE 5B 30 ED C6 20 77 18 E8 CD 99 7E FE 7E70: 22 28 CB FE 20 20 E8 7E80: CD 99 7E FE 22 28 CO FE 20 20 DD 18 C1 C1 23 23 7E FE 00 28 19 23 23 18 AE 23 7E FE 00 28 EE C9 7E90: 7EA0: 21 7C 7F 11 00 3D 01 40 00 ED B0 C3 CC 06 21 BC 7EB0: 7F 11 00 3D 01 40 00 ED B0 C3 CC 06 20 20 20 20 20 20 20 4C 4F 57 45 52 20 43 41 53 45 20 43 4F 7EC0: 7EDO: 4E 56 45 52 54 45 52 20 46 4F 52 20 42 41 53 49 43 20 50 52 4F 47 52 41 4D 53 20 20 20 56 45 52 **7EE0:** 7EF0: 53 49 4F 4E 20 32 2E 30 20 20 20 20 20 20 20 20 20 20 20 20 20 43 52 45 41 54 45 44 20 42 59 20 44 45 7F00: 7F10: 4E 4E 49 53 20 57 52 49 47 48 54 20 20 20 20 20 7F20: 20 20 20 20 20 20 28 43 29 20 20 20 31 37 20 4A 7F30: 55 4E 45 20 31 39 38 32 20 20 20 20 20 20 20 20 20 20 7F40: 20 20 20 50 52 4F 47 52 41 4D 20 20 43 4F 4E 56 45 52 54 49 4E 47 20 52 45 53 49 44 45 4E 54 20 7F50: 7F60: 42 41 53 49 43 20 50 52 4F 47 52 41 4D 20 20 20 7F70: 7F80: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 7F90: 20 4E 4F 20 42 41 53 49 43 20 50 52 4F 47 52 41 4D 20 4C 4F 41 44 45 44 20 20 20 20 20 20 20 20 20 20 7FA0: 7FB0: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 7FCO: 7FDO: 20 20 20 20 43 4F 4E 56 45 52 53 49 4F 4E 20 43 7FE0: 4F 4D 50 4C 45 54 45 20 7FF0:

COLOUR COMPUTER 10 ****VARIABLE WORKSHEET*** 20 '*****FOR THE TRS8CC***** 30 'ORIGINALLY BY S&₽ MILLER 40 'FOR THE TRS80 50 CLEAR 1000 60 CLS 70 ▶RINT @ 71, "VARIABLE WORKSHEE Т " 80 PRINT TAB(15) "BY 90 PRINT"BY S & P MILLER FOR THE TRS80CC 100 PRINT 110 PRINT 120 INPUT"MAX. NUMBER OF ARRAYS" ;A 130 INPUT"MAX. NUMBER OF SUBROUT INE LINES REQUIRED"; B 140 INPUT"MAX. NUMBER OF VARIABL E LINES REQUIRED";C 150 INPUT"NUMBER OF COPIES REQUI ";Y RED 160 FOR J=1 TO Y 170 PRINT#-2, CHR\$(27)CHR\$(14)"V ARIABLE WORKSHEET" 180 PRINT#-2, "PROGRAM : ",, "PROG RAMMER :",,"DATE : / /19 . . . 190 PRINT#-2," " 200 PRINT#-2,"MEMORY CLEARED -MEMORY PROTECTION AT -210 PRINT#-2," " 220 PRINT#-2," " 230 PRINT#-2," " 240 PRINT#-2, "LIST OF DIMENSIONE D ARRAYS :-" 250 FOR X=1 TO A 260 PRINT#-2," DIM () : N AME :"

**** VARIABLE WORKSHEET ****

270 NEXT X 280 PRINT#-2," " 290 PRINT#-2, "SUBROUTINE INFORMA TION :-" 300 PRINT#-2," " 310 PRINT#-2," START #: END # : COMMENTS " 320 PRINT#-2," " 330 FOR X=1 TO B 340 ₽RINT#-2," ---- ";": "; 350 PRINT#-2, STRING\$(41,"-") 360 NEXT X 370 ₽RINT#-2," " 380 PRINT#-2, "VARIABLE LIST :-" 390 PRINT#-2," " 400 PRINT#-2," NAME LABEL/USE 410 ₽RINT#-2," " 420 FOR X=1 TO C 430 PRINT#-2," ----" 440 ₱RINT#-2, STRING\$(44,"-") 450 NEXT X 460 PRINT#-2," " 470 PRINT#-2,"PERIPHERALS REQUIR ED :-" 480 PRINT#-2," " 490 PRINT#-2," CASSETTE () :PRI NTER () :DISKS () NUMBER REQUI RED --500 PRINT#-2,"NOTES:- " 510 PRINT#-2," " 520 PRINT#-2," " 530 PRINT#-2," " 540 NEXT J 550 CLEAR 50 560 VARIABLE LIST:-570 ' A: NO. OF ARRAY LINES 580 ' B: NO. OF SUBROUTINE LINES 590 ' C: NO. OF VARIABLE LINES 600 ' Y: NO. OF COPIES REQUIRED 610 ' J: LOOP COUNTER FOR COPIES 620 ' X: LOOP COUNTER FOR A,B,C

**** MILEAGE CALCULATOR ****

COLOUR COMPUTER

10 '***MILEAGE CALCULATOR*** 20 '****FOR THE TRS80CC***** 30 'ORIGINALLY BY FRANK GRECO 40 'FOR THE TRS80 50 ' 60 'DISK USERS CHANGE :-70 '#-1 TO #1 IN LINES 540-560, 80 'ADD :- '+STR\$(M)' TO 540, '+STR\$(MM) TO 1010. '+STR\$(M+1)' TO 1170 90 'LEAVE OUT THE 'AUDIO ON'S 100 'CHANGE 'TAPE'S TO 'DISK'S 110 ' 120 ' 130 CLS: PRINT@65, "KMS PER LITRE/ MILES PER GALLON CONVE RSION." 140 PRINT" WRITTEN BY FRANK GREC 0." 150 PRINT" MARIBYRNONG VICTORIA, AUSTRALIA 160 FORX=1 TO 3000:NEXTX 170 GOTO 680 180 SOUND200.1 190 CLS:PRINT: INPUT "MONTH NUMBER (1 TO 12)";M 200 IF M<1 OR M>12 THEN 190 210 INPUT"KILOMETERS AT THE STAR T";A 220 INPUT"KILOMETERS AT THE END" ; B 230 INPUT"HOW MANY LITRES HAVE Y OU PUT IN THE TANK";F 240 INPUT"COST";CC 250 C=B-A 260 PRINT:PRINT 270 PRINT"YOU TRAVELLED";C; "KIL OMETERS." 280 REM* CONV KM TO MILES * 290 D=C*0.62 300 PRINT"THAT'S ABOUT "; INT(D); " MILES." 310 PRINT: PRINT "NOW I AM WORKING OUT YOUR PETROL CONSUMPTION." 320 SOUND200,1 330 FORX=1 TO 1000:NEXTX 340 CLS

350 REM** PETROL CONS ** 360 E=C/F 370 G=E*0.62 380 H=G*4.5 390 REM * LITRES PER 100 KM * 400 FF=F/(C/100) 410 PRINT"DISTANCE TRAVELLED IS" ;C;"KMS." 420 PRINT"WHICH IS ABOUT"; INT(D) ;"MILES." 430 PRINT: PRINT"PETROL CONS FOR" ;C;"KMS IS ";:PRINT USING "##.#" ;E;:PRINT" KMS/LITRE." 440 PRINT:PRINT"-ABOUT"INT(H)"MI LES PER GALLON." 450 PRINT:PRINT THAT IS ";:PRINT USING "##.#";FF;:PRINT" LITRES PER 100 KM." 460 PRINT: PRINT WANT TO SAVE IT ON TAPE? <Y/N> "; 470 GOSUB 600 480 CLS 490 SOUND200,1 500 PRINT@162, "MAKE PREPARATION S FOR DUMPING" 510 PRINT@290, "";: INPUT "HIT ENTE R WHEN READY"; A 520 CLS:PRINT@266, "DUMPING DATA" 530 AUDIOON 540 OPEN "O".#-1. "MILES" 550 PRINT#-1, M, CC, A, B, F, D, E, H, C, FF 560 CLOSE #-1 570 CLS:PRINT@226, MAKE ANOTHER COFY 7 <Y/N> "; 580 GOSUB 600 590 GOT0520 600 SOUND200,1 610 C\$=INKEY\$:IF C\$=""THEN610 620 IF C\$="Y" THEN RETURN ELSE I F C\$="N" THEN CLS 630 CLS: PRINT:PRINT" ANY MORE DATA? <Y/N> "; 640 SOUND200.1 650 G\$="":G\$= INKEY\$:IF G\$="" TH EN 650 660 IF G\$="Y" THEN 190 670 IF G\$="N" THEN CLS:GOTO 680 680 CLS:PRINT@44. "M E N U " 690 PRINT:PRINT 700 FRINT" 1 - INPUT DATA ROUTI NE" 710 PRINT" 2 - PROJECTION CHAR

Т"

720 PRINT" 3 - SUMMARY 730 PRINT" 4 - END" 740 SOUND200.1 750 A\$="":A\$= INKEY\$:IF A\$="" TH EN 750 760 B= VAL(A\$):ON B GOTO 190 , 770 ,880 ,1350 770 CLS:PRINT@40, "PROJECTION CHA RT" 780 PRINT: PRINT"ON THE AVERAGE H OW MANY KM/LITRE DOES YOUR CAR D 0 "; 790 INPUT P 800 PRINT:PRINT"HOW MANY KM DO Y OU WANT IT PROJECTED FOR ** 810 INPUT PP 820 XX=PP/P 830 PRINT:PRINT"ON";PP;"KM YOU W OULD NEED"INT(XX)"LITRES OF FUEL 840 PRINT: PRINT: PRINT "PRESS ANY KEY TO RETURN TO MENU" 850 SOUND200,1 860 D\$= INKEY\$: IF D\$="" THEN 860 **ELSE 870** 870 GOTO 680 880 CLS: PRINT@44, "SUMMARY" 890 PRINT:PRINT:PRINT 900 PRINT" MONTHLY OR YEARLY (M OR Y) ?" 910 SOUND200,1 920 E\$="":E\$= INKEY\$:IF E\$="" TH EN 920 930 IF E\$="M"THEN950 ELSE1100 940 STOP 950 CLS: PRINT@44, "MONTHLY" 960 SOUND200.1 970 PRINT:PRINT:PRINT:INPUT" MON TH NUMBER = ";MM 989 M=0 990 CLS:PRINT@230, "READING -- MO NTH"; M+1 1000 AUDIO ON 1010 -1030, 1170-1190 1020 INPUT #-1, M, CC, A, B, F, D, E, H, C,FF 1030 CLOSE #-1 1040 IF M<>MM THEN 990 1050 CLS:PRINT@40, "MONTH NUMBER" :M:PRINT:PRINT 1060 GOSUB1360 1070 CLS:PRINT@470, "--- ANY MORE MONTHS ? <Y/N> ---"

	**** VARIABLE WORKSHEET ****
	HITACHI PEACH
2 ' ****	**************************************

1080 GDSUB 600 1090 -1030, 1170-1190 1100 PRINT:PRINT 1110 SOUND200,1 1120 INPUT"PRESS ENTER WHEN TAPE READY":A 1130 CLS:PRINT@102, "READING -- Y EARLY 1140 M=0 1150 PRINT@234, "MONTH #"; M+1 1160 AUDIO ON 1170 OPEN"I",#-1,"MILES" 1180 INPUT #-1, M, CC, A, B, F, D, E, H, C,FF 1190 CLOSE #-1 1200 Z=Z+CC 1210 Y=Y+F 1220 QQ=QQ+C 1230 RR=RR+E 1240 IF M<12 THEN 1150 1250 CLS:PRINT@44, "COSTS" 1260 PRINT:PRINT 1270 PRINT TOTAL LITRES", "TOTAL COST" 1280 PRINTY,Z 1290 PRINT: PRINT: PRINT "TOTAL KMS ","AVG KM/L" 1300 R=RR/12 1310 PRINTQQ,:PRINT USING"##.##" ;R 1320 PRINT:PRINT:PRINT:PRINT"PRE SS ANY KEY TO RETURN TO MENU" 1330 GOSUB 650 1340 RETURN 1350 CLS:PRINT@200, "--- THE END ---":PRINT@447," ";:END 1360 PRINT TOTAL LITRES", "TOTAL COST" 1370 PRINT:PRINTF,CC 1380 PRINT: PRINT: PRINT "TOTAL KMS ", "AVG KM/L" 1390 PRINT:PRINTC,:PRINT USING"# #.##":E 1400 PRINT:PRINT:PRINT"PRESS ANY KEY TO RETURN TO MENU"; 1410 GOT0650

80 × S & P MILLER : P.O.BOX 37-1 4 ' 30 × 5 2 STOKES VALLEY ; NEW ZEALAND 6 ' ********************************** **** 10 CLEAR 1000 20 CLS 30 LOCATE13,4:PRINT "VARIABLE WORKSHEET " 35 PRINT 40 PRINT TAB(38) "BY S & P MILLER - FOR T HE TRS-80" 50 PRINT 60 PRINT TAB(38)" - MODIFIED FOR HITACH I MB-6890" 65 PRINT 70 PRINT 80 INPUT"ENTER MAX. NUMBER OF ARRAYS "; Α 90 INPUT"ENTER MAX. NUMBER OF SUBROUTINE LINES REQUIRED ";B 100 INPUT"ENTER MAX. NUMBER OF VARIABLE LINES REQUIRED ";C 110 INPUT"ENTER NUMBER OF COPIES REQUIRE D ";Y 115 OPEN"O", 1, "LPT0:" 120 FOR J=1 TO Y 130 PRINT#1, "VARIABLE WORKSHEET" 140 PRINT#1, "PROGRAM : ",, "PROGRAMMER :" ,,"DATE : / /19 :" 150 PRINT#1," " 160 PRINT#1, "MEMORY CLEARED -MEMORY PROTECTION AT - " 170 PRINT#1." " 180 PRINT#1, "LIST OF TYPE DEFINITIONS :~ 190 PRINT#1," " 200 PRINT#1," DEFSTR ----- " 210 PRINT#1," DEFINT ---- " 220 PRINT#1," DEFSNG ---- " 230 PRINT#1," DEFDBL --- " 240 PRINT#1." " 250 PRINT#1," " 260 PRINT#1, "LIST OF DIMENSIONED ARRAYS :-" 270 FOR X=1 TO A 280 PRINT#1," DIM () : NAME :" 290 NEXT X 300 PRINT#1," " 310 PRINT#1, "SUBROUTINE INFORMATION :-" 320 PRINT#1," " 330 PRINT#1," START #: END # : COMMENTS

3 ' * VARIABLE WORKSHEET - FOR THE TRS-

410 PRINT#1," " 420 PRINT#1," NAME LABEL/ USE" 430 PRINT#1," " 440 FOR X=1 TO C 450 PRINT#1," ---", 460 PRINT#1,STRING\$(44,"-") 470 NEXT X 480 PRINT#1," " 490 PRINT#1, "PERIPHERALS REQUIRED :-" 500 PRINT#1," " 510 PRINT#1," CASSETTE # 1 ():CASSETTE # 2 ():PRINTER () :DISCS () NUMBER RE QUIRED -- " 520 PRINT#1, "NOTES:- " 530 PRINT#1," " 540 PRINT#1," " 550 PRINT#1," " 560 NEXT J 565 CLOSE 570 CLEAR 50 580 ' VARIABLE LIST:-590 ' A ; NUMBER OF ARRAY LINES 600 ' " " SUBROUTINE LINES в: " " VARIABLE LINES 610 C: Y: " COPIES REQUIRED 620 ' J : LOOP COUNTER FOR COPIES 630 ' X : " . . " A, B AND C 640 ' **** MILEAGE CALCULATOR **** HITACHI PEACH *** MILEAGE CALCULATOR *** 10 ' 12 ' FRANK GRECO × 13 ' RIVERVIEW COURT × 14 ' MARIBYRNONG VIC. ¥ 15 ' ********************** 20 CLS:LOCATE15,10:PRINT"KILOMETERS PER LITRE/MILES PER GALLON CONVERSION." 30 LOCATE25,12:PRINT"WRITTEN BY FRANK GR ECO."

360 PRINT#1," ----- ";":";" ----- ";":

340 PRINT#1." "

390 PRINT#1." "

";

380 NEXT X

350 FOR X=1 TO B

370 PRINT#1,STRING\$(41,"-")

400 PRINT#1, "VARIABLE LIST :-"

40 LOCATE20, 14: PRINT "MARIBYRNONG VICTORI A, AUSTRALIA, 3032" 50 FORX=1 TO 3000:NEXTX 60 GOTO 500 70 CLS: INPUT "PLEASE INPUT MONTH NUMBER ((1 TO 12))";M 80 INPUT"PLEASE INPUT KILOMETERS AT THE START": A 90 INPUT"PLEASE INPUT KILOMETERS AT THE END"; B 100 INPUT"PLEASE INPUT HOW MANY LITRES Y OU HAVE PUT IN THE TANK":F 110 INPUT"PLEASE INPUT COST";CC 120 C=B-A 130 PRINT:PRINT 140 PRINT"YOU HAVE TRAVELLED ";C; " KILOM ETERS." 150 PRINT:PRINT 160 REM* CONV KM TO MILES * 170 D=C*0.62 180 PRINT"THAT'S ABOUT ";D;" MILES." 190 PRINT:PRINT"NOW I AM WORKING OUT YOU R PETROL CONSUMPTION." 200 FORX=1 TO 1000:NEXTX 210 CLS 220 REM** PETROL CONS ** 230 E=C/F 240 G=E*0.62 250 H=G*4.5 260 REM * LITRES PER 100 KM * 270 FF=F/(C/100) 280 PRINT"--- TOTAL DISTANCE TRAVELLED I S ":C:" KMS." 290 PRINT:PRINT"--- WHICH IS ABOUT ";D;" MILES." 300 PRINT:PRINT"--- PETROL CONS FOR ";C; " KMS IS ";E;" KMS/LITRE." 310 PRINT:PRINT"--- THAT'S ABOUT ";H;" M ILES PER GALLON." 320 PRINT:PRINT"--- THAT IS ";FF;"LITRES PER 100 KM." 330 LOCATE19,14:PRINT"--- WANT TO SAVE I T ON TAPE? <Y/N> --- "; 340 GOSUB 440 350 CLS 360 LOCATE25,7:PRINT"MAKE PREPARATIONS F OR DUMPING" 370 LOCATE29, 10: PRINT"; PRINT"HIT ENTER WHEN READY" 375 G\$="":G\$=INKEY\$:IF G\$="" THEN 375 380 CLS:LOCATE29, 10:PRINT"DUMPING DATA" 385 OPEN"O",#1,"CAS0:DATA" 390 FORX=1 TO 1000:NEXT:PRINT#1,M,CC,A,B

,F,D,E,H,C,FF

75 CLOSE د 400 CLS:LOCATE29,10:PRINT"DATA HAS BEEN DUMPED" 410 LOCATE23,12:PRINT"--- MAKE ANOTHER C OPY <Y/N> ---"; 420 GOSUB 440 430 GOTO 380 440 C\$="":C\$=INKEY\$:IF C\$="" THEN 440 450 IF C\$="Y" THEN RETURN ELSE IF C\$="N" THEN CLS 460 CLS: LOCATE25, 10: PRINT --- ANY MORE DATA? <Y/N> ---"; 470 G\$="":G\$=INKEY\$:IF G\$="" THEN 470 480 IF G\$="Y" THEN 70 490 IF G\$="N" THEN CLS:GOTO 500 500 CLS:LOCATE31,3:PRINT"M E N U " 510 LOCATE31, 3: PRINT"SELECT 1" 520 LOCATE23, 6: PRINT"1 - INPUT DATA ROUT INE" 530 PRINTTAB(23); "2 - PROJECTION CHART" 540 PRINTTAB(23); "3 - SUMMARY 550 PRINTTAB(23); "4 - END" 560 A\$="":A\$=INKEY\$:IF A\$="" THEN 560 570 B=VAL(A\$):ON B GOTO 70 ,580 ,680 ,1070 580 CLS:LOCATE31, 3: PRINT"PROJECTION CHAR т" 590 PRINT: PRINT"ON THE AVERAGE HOW MANY KM/LITRE DOES YOUR CAR DO "; 600 INPUT P 610 PRINT"HOW MANY KM DO YOU WANT IT PRO JECTED FOR "; 620 INPUT PP 630 XX=PP/P 640 PRINT:PRINT"ON ";PP;"KM YOU WOULD NE ED ";XX;" LITRES OF FUEL" 650 LOCATE17,17:PRINT" PRESS ANY KEY TO RETURN TO MENU" 660 D\$=INKEY\$:IF D\$="" THEN 660 ELSE 67 Ø 670 GOTO 500 680 CLS:LOCATE31, 3: PRINT"SUMMARY" 690 PRINT: PRINTTAB(6) "READS PRERECORDED DATA TAPE CREATED BY INPUT DATA ROUTINE" : PRINT 700 LOCATE21, 7: PRINT "MONTHLY OR YEARLY --- M OR Y" 710 E\$="":E\$=INKEY\$:IF E\$="" THEN 710 720 IF E\$="M"THEN740 ELSE850 730 STOP 740 CLS:LOCATE31.3:PRINT"MONTHLY" 750 PRINT: PRINT: PRINT: INPUT "MONTH NUMBER = ";MM 755 LOCATE26, 9: PRINT "MAKE PREPARATIONS"

756 A=0:PRINT:PRINT TAB(23) "PRESS RETURN WHEN READY" 757 G\$="":G\$=INKEY\$:IF G\$="" THEN 757 760 M=0:CLS:LOCATE29,10:PRINT"SEARCHING FOR MONTH --#"; MM 770 OPEN "I".#1."CAS0:DATA" 780 INPUT #1, M, CC, A, B, F, D, E, H, C, FF 790 CLOSE:LOCATE29.12:PRINT"JUST READ MONTH --#"; M: IF M=MM THEN800 ELSE IF M<>MM THEN 770 800 CLS:LOCATE31, 3: PRINT "MONTH NUMBER"; M :PRINT:PRINT:PRINT 810 GOSUB1080 820 CLS:LOCATE29, 10:PRINT --- ANY MORE M ONTHS ? <Y/N> ---" 830 GOSUB 440 840 GOTO 740 850 CLS:LOCATE31, 3: PRINT"YEARLY" 860 LOCATE26, 7: PRINT "MAKE PREPARATIONS" 870 A=0:PRINT:PRINT TAB(23) "PRESS RETURN WHEN READY" 875 G\$="":G\$=INKEY\$:IF G\$="" THEN 875 880 CLS:LOCATE29, 1: PRINT "READING -- YEAR LY":PRINT:PRINT 890 M=0:Z=0:Y=0:QQ=0:RR=0:PRINT TAB(5)"M ONTH"," KM","LITRES","COST","KM/LITRE" 900 OPEN "I",#1,"CAS0:DATA" 910 INPUT #1, M, CC, A, B, F, D, E, H, C, FF 920 Z=Z+CC 930 Y=Y+F 940 QQ=QQ+C 950 RR=RR+E 955 CLOSE:PRINT TAB(6) M,C,F,CC,E 960 IF M=12 THEN 970 ELSE 900 970 PRINT TAB(5) "TOTAL", QQ, Y, Z, QQ/Y 980 PRINT:PRINT 990 PRINT TAB(20) "TOTAL LITRES", "TOTAL C OST" 1000 PRINT TAB(26) Y," \$";Z 1010 PRINT, "TOTAL KMS",,, "AVG KM/L" 1030 PRINT, QQ, ,, QQ/Y 1040 PRINT:PRINT:PRINT" PR ESS ANY KEY TO RETURN TO MENU" 1050 GOSUB 470 1060 RETURN 1070 CLS:LOCATE29,10:PRINT --- THE END ---":END 1080 PRINT TAB(20) "TOTAL LITRES", "TOTAL COST" 1090 PRINT TAB(26) F," \$";CC 1100 PRINT, "TOTAL KMS",,, "AVG KM/L" 1110 PRINT, C., E 1120 LOCATE21, 17: PRINT"PRESS ANY KEY TO RETURN TO MENU" 1130 GOSUB 470

MICRO-80

VOLUME

ω

8.

ω

(JULY,

198

Ň

VOLUME ω NO. ω (JULY, 1982)

PAGE 28

	340 LPRINTTAB(20)"M M I C C R R O
	350 LPRINTTAB(20)"M M III CCC R R
	360 LPRINT:LPRINT
	370 LPRINT" "+STRING\$(70,"*"):LPRINT
	380 FORI=0T09
**** CALENDAR ****	390 B\$=STRING\$(18.32)+X\$(I1.I)+" "+X\$
	"+X\$(T4_T)
	400 I PRINTB\$
	410 NEXTI
	420 LPRINT-LPRINT" "+STRING\$ (70, "*")
	430 RETURN
10 '**** CALENDAR ****	
20 'WRITTEN BY R.J.WIWATOWSKI	
40 CLEAR500	
50 DEFINID, I-N, W, Y	
60 DIM (11) VR(11 4 4) Ys(9 9)	
80 DATA" 000000 " "00000000" "00 00" "00 00" "00 00"."0	490 IFLEN(L\$)(3)HENL\$=" "+L\$
	300 B\$=B\$+C\$
	510 NEXIJ:NEXII
70 DHIH I, II , II , II , I - , - ,	520 LPRINIBS
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	530 NEXTW
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	540 LPRINT:RETURN
	550 CLS:PRINTTAB(23) "MICRO-80 CALENDAR":
110 DATA-33533335, "33333333", "33", "33", "33", "33", "33", "33", "33", "33", "33", "33", "33", "33", "33",",",",	-"):PRINT
333 "," 333","33 33","333333333"," 3333333	560 PRINT" THIS PROGRAM WILL GENERATE A
120 DATA" 4 "," 44 "," 444 "," 4444 "," 4444 4 ,	THE RANGE1901 - 1999. ALL YOU HAVE TO D
44 44 ", "44444444", "44444444", " 44 "," 44 "," 44 "," 44 "," 44 "," 44 "," 44 "," 44	Į "
130 DATA"55555555","5555555","55 ","55 ","55 ","55 ","55 ","55 ","55	570 PRINT" THE PROGRAM TAKES A FEW SECON
55555555"," 55","55 55","55555555"," 5555555	S AND YOU CAN SET THE PRINTER TO TOP OF
140 DATA" 666666 ","66666666","66 66","66 ","66666666	READY, A HARDCOPY OF THE CALENDAR WILL
66666666","66 66","66 66","66666666"," 6666666 "	580 A\$=" SU MO TU WE TH FR SA":A\$=A\$
150 DATA"7777777", "7777777", "70", "70",","70","",",","70","",",","70",	590 H1\$=STRING\$(11,32)+"JANUARY"+STRING\$
77 "," 77 "," 77 "," 77 "," 77 "	G\$(18,32)+"MARCH"
160 DATA" 888888 ","88888888","88 88","88 88"," 888888 ","	600 H2\$=STRING\$(12,32)+"APRIL"+STRING\$(2
888888 ","88 88","88 88","88888888"," 888888 "	2)+"JUNE"
170 DATA" 999999 ","9999999","99 99","99 99","99999999	610 H3\$=STRING\$(13,32)+"JULY"+STRING\$(20
9999999"," 99","99 <u>9</u> 9","99999999"," 999999 "	.32)+"SEPTEMBER"
180 FORI=0TO11:READL(I):NEXTI	620 H4\$=STRING\$(11.32)+"OCTOBER"+STRING\$
190 FORI=0TO9:FORJ=0TO9:READX\$(I,J):NEXTJ:NEXTI	G\$(17.32)+"DECEMBER"
200 B\$=CHR\$(27)+"G":LPRINTB\$;	630 INPUT"EOR WHICH YEAR DO YOU WANT A C
210 G0T0550	640 IFY<19010RY>1999THENPRINT:PRINT"OUT
220 FORM=0T011	650 IF4*INT(Y/4)=YTHENL(1)=29
230 W=0:DT=1	660 I=Y-1901; $J=INT(I/4)$; $I=I-4*J+2$
240 YR(M,D,W)=DT:DT=DT+1:D=D+1	670 K=5*(J-7*INT(J/7))+I:D=K-7*INT(K/7)
250 IFD>6THEND=0:W=W+1:IFW>4THENW=0	480 GOSUB220 PRINT PRINT PRESS 'NEW INF'
260 IFDT <l (m)+1then240<="" td=""><td>690 TEINKEY\$()CHR\$(13)THEN690</td></l>	690 TEINKEY\$()CHR\$(13)THEN690
270 NEXTM	700 GDSUB290
280 RETURN	710 M=0.1 PRINTH1\$.605UB440
290 Q\$=RIGHT\$(STR\$(Y),4):I1=VAL(MID\$(Q\$,1,1)):I2=VAL(MID\$(Q\$,2,1	720 M=3-1 PRINTH24-COCUPAA0
)):I3=VAL(MID\$(Q\$,3,1)):I4=VAL(MID\$(Q\$,4,1))	
300 LPRINT:LPRINT:LPRINT	
310 LPRINTTAB(20)"M M III CCC RRRR 000 88 00"	740 H-7:LFKINIG49:0000440 750 Dd-CUDd(27)+404-1 DDINIDd+-0017257 42
320 LPRINTTAB(20) "MM MM I C. C R R O O 8 8 0 0"	7.00 DP-UNRP(2/)+ N "LPKINIBP;:UU1203,12"

330 LPRINTTAB(20)"M M M I C 0 8 8 0 0" 88 000 00" \$(12,1)+" "+X\$(I3,I)+" :LPRINT:LPRINT:LPRINT TR\$(YR(M+I,J,W)) PRINTTAB(23)STRING\$(17," CALENDAR FOR ANY YEAR IN O IS TO SPECIFY THE YEAR NDS TO DO THE CALCULATION FORM. WHEN EVERYTHING IS BE PRINTED. ": PRINT 6+A\$+A\$ (18,32)+"FEBRUARY"+STRIN 21,32)+"MAY"+STRING\$(22,3),32)+"AUGUST"+STRING\$(17 (18,32)+"NOVEMBER"+STRIN CALENDAR ";Y OF RANGE": GOTO630 WHEN READY "

RRR O

0 *** 88 0 0"

**** HEX CONSTANTS **** LEVEL II	90 PRINT"INDUCTANCE CAN BE CALCULATED FOR AN INDUCTIVE CIRCUIT WITHOUT CAPACITANCE IF I, E, AND R, ARE KNOWN VALUES.": PRINT100 PRINT"FINALLY THE PROGRAMME WILL ALSO OPERATE A SI MPLE OHM'S LAW SOLUTION (I = E/R) BY INPUTTING 0 (ZERO) FOR B
<pre>10 'HEX CONSTANTS(C) COPYRIGHT ROGER BOWLER 1981 20 DEFINT H-N:PRINT"PLEASE WAIT":ON ERROR GOTO 110 30 MS=0:POKE VARPTR(MS),PEEK(16561) 'LOCATE 1ST BYTE 40 POKE VARPTR(MS)+1,PEEK(16562):MS=MS+3:HX=MS 'BEYOND MEM SIZE 50 READ A\$:IF A\$="*" GOTO 130 60 FORI=1TOLEN(A\$)-1STEP2 70 H=ASC(MID\$(A\$,I,1)):L=ASC(MID\$(A\$,I+1,1)) 80 IFH>64THENH=H-55ELSEH=H-48 90 IFL>64THENH=H-55ELSEH=H-48 100 J=H*16+L:POKE MS,J:IF PEEK(MS)=J GOTO 120 110 PRINT"YOU FORGOT TO SET MEMORY SIZE":END 120 MS=MS+1:NEXTI:GOTO50 130 POKE 16789,PEEK(VARPTR(HX)):POKE 16790,PEEK(VARPTR(HX)+1) 140 CLS:PRINT"YOU CAN NOW USE HEX CONSTANTS. HERE IS A DEMO" 150 FOR I=&H3C40 TO &H3CBF:POKE I,&H86:NEXTI 160 PRINT&HC0,"OK?" 170 DATA D7,CF48,0604,110000,7E,FE30,381F,FE3A,380A,FE41,3817 180 DATA FE47,3013,D607,E60F,48,0604,CB23,CB12,10FA,B3,5F,41 190 DATA D7,10DC,78,FE04,D29719,ED532141,3E02,32AF40,C9,*</pre>	<pre>FURTHER TWO SCROLLS OF THE SCREEN CAN BE MADE WHICH WILL DISPLA Y A CIR- CUIT DIAGRAM OF THE GENERAL SERIES CIRCUIT FOLLOWED BY A VECTOR DIAGRAM AND IMPEDANCE DIAGRAM": GOSUB 1530 120 GOSUB 980 130 GOSUB 1530 140 PRINT015;">>> INPUT OF KNOWN VALUES <<<" 150 INPUT"CURRENT";I 160 INPUT"CURRENT";I 160 INPUT "VOLTAGE ";E 170 INPUT"RESISTANCE";R 180 INPUT"CAPACITANCE";C 190 INPUT "INDUCTANCE";L 200 INPUT "FREQUENCY";F 210 INPUT"FREQUENCY";F 210 INPUT"EQUIV REACT";X 220 LET W = 2*3.1416*F 230 IF X=0 AND C=0 AND L=0 AND I=0 GOTO 840 240 IF X=0 AND C=0 AND L=0 AND R=0 GOTO 860 250 IF X=0 AND C=0 AND L=0 AND R=0 GOTO 880 260 IF X=0 AND C=0 AND L=0 GOTO 810 270 IF L=0 AND X=0 GOTO 790 280 IF C=0 GOTO 340 240 IF C=0 GOTO 340 </pre>
**** SERIES IMPEDANCE CIRCUIT **** LEVEL II	300 IF L=0 GDSUB 730 310 LET W=2*3.1416*F 320 X=(W*L)-1/(W*C) 330 PRINT"EQUIV REACT =",;X 340 G=W*L 350 PRINT "INDUCTIVE REACT =",;G 360 GDSUB 1530
10 CLS:PRINT TAB(16)">>> SERIES IMPEDANCE CIRCUIT <<<" 20 PRINT@(84),STRING\$(24,131);: PRINT: PRINT 30 PRINT" A WIDE VARIETY OF ELECTRICAL CIRCUIT PROBLEMS OF THIS CATEGORY ARE SOLVEABLE DEPENDING UPON THE KNOWN AND UNKN OWN VAR-IABLES AVAILABLE.": PRINT 40 'PREPARED BY W.G.HEATH 18 OXLEY ROAD. WARATAH NEWCASTLE . N.S.W. 2298.	370 IF C=0 AND L=0 GDTD 390 380 IF C=0 GDSUB 760 390 IF 0>X LETX=-X 400 IF Y=I GDTD 430 410 IF Y=E GDTD 450 420 IF Y=R GDTD 470 430 I= E/SQR((RL2)+(XL2)) 440 PRINT"CURRENT =".:I
50 PRINT" THE PROGRAMME WILL CALCULATE ALL THE VARIABLE S SHOWN IN THE FOLLOWING SCHEDULE SUBJECT TO APPROPRIATE INPUT VALUES.": PRINT 60 PRINT" IT WILL HANDLE TWO UNKNOWNS SIMULTANEOUSLY PR OVIDING ONE UNKNOWN IS FITHER I. F. OR R. AND THE SECOND IS FITH	450 E= I*SQR((R[2)+(X[2)) 460 PRINT"VOLTAGE =",;E 470 R=SQR(((E[2)/(I[2))-(X[2)) 480 PRINT"RESISTANCE =",;R
ER L, C OR X. MAKE NO ENTRY FOR THE UNKNOWN VALUES. I.E. PRESS < ENTER> ONLY WHEN INPUT? FOR THESE VARIABLES IS CALLED FOR."	490 T=ATN(X/R)*57.29578 500 PRINT"PHASE ANGLE =",;T 510 Q=COS(T*.01745329)
70 GOSUB 1530 80 PRINT:PRINT:PRINT" THE PROGRAMME CAN BE USED FOR PRO BLEMS WHERE NO CAPAC-ITANCE OR. ALTERNATIVELY, NO INDUCTANCE IS P	520 PRINT"POWER FACTOR =",;Q 530 P=E*I*Q
RESENT.": PRINT	540 PRINT"POWER (WATTS) =",;P .550 Z=SQR((RE2)+(XE2))

PAGE 29

560 PRINT"IMPEDANCE =",;Z 570 G=W*L 580 PRINT"INDUCTIVE REACT =",;G 590 VL=I*G 600 PRINT"INDUCTIVE VOLT DROP =",;VL 610 IF C = 0 GOTO 660620 J=1/(W*C) **630 PRINT"CAPACITIVE REACT** =",;J 640 VC=I*J 650 PRINT"CAPACITIVE VOLT DROP =",;VC 660 VR=I*R 670 PRINT"RESIST VOLT DROP =",;VR 680 IF C=0 GOTO 720 690 IF L=0 GOTO 720 700 M=1/(2*3.1416*SQR(L*C)) 710 PRINT"RESONANT FREQ. =",;M 720 GOTO 910 730 L=(X+(1/(W*C)))/(W) 740 PRINT"INDUCTANCE =",,;L 750 RETURN 760 C=(1/(W*L-X))/W770 PRINT"CAPACITANCE =",,;C 780 RETURN 790 X=1/(W*C): GOTO 390 800 X=W*L: GOTO 390 810 Z=E/I 820 X=SQR(Z[2-R[2) 830 L=X/W: PRINT "INDUCTANCE =",,;L: GOTO 390 840 I=E/R 850 PRINT "CURRENT ="...; I 860 E=I*R 870 PRINT "VOLTAGE =",,;E 880 R=E/I 890 PRINT "RESISTANCE =",,;R 900 PRINT "INDUCTANCE =",,;L: GOTO 330 910 GOSUB 1530 920 GOSUB 1280 930 GOSUB 1530 940 GOSUB 1100 950 GOSUB 1530 960 GOSUB 1410 970 END 980 PRINT TAB(18)">>> LIST OF VARIABLES USED <<<" 990 PRINT @(86),STRING\$(22,131);: PRINT :PRINT 1000 PRINT "I = CURRENT",,:PRINT "L = INDUCTANCE" 1010 PRINT "E = VOLTAGE",,: PRINT "C = CAPACITANCE" 1020 PRINT "R = RESISTANCE",,: PRINT "F = FREQUENCY" 1030 PRINT "Z = IMPEDANCE",,: PRINT "X = EQUIVALENT REACTANCE" 1040 PRINT "Q = POWER FACTOR",: PRINT "P = POWER (WATTS)" 1050 PRINT "G = INDUCTIVE REACTANCE", PRINT"VL = INDUCTIVE VOLT DROP" 1060 PRINT "J = CAPACITIVE REACTANCE", : PRINT VC = CAPACITIVE VOL T DROP" 1070 PRINT "M = RESONANT FREQUENCY", :PRINT"VR = RESISTIVE VOLT D ROP"

1080 PRINT "W = 2*3.14159* F",: PRINT " ? = UNKNOWN VALUE" 1090 PRINT TAB(10) "NOTE-- CAPACITANCE USUALLY MEASURED AS MICRO -FARADS AND INPUT AS PER EXAMPLE 600E-6 OR ALTERNATIVELY .0006" : RETURN 1100 CLS: PRINT@52, "E" -----";CHR\$(148) 1120 PRINT@143, CHR\$ (170): PRINT@179, ":" 1130 PRINT"VECTOR": PRINT@207, CHR\$(170): PRINT@243, ":" 1140 PRINT"DIAGRAM": PRINT@271, CHR\$(170): PRINT@307, ":" 1150 PRINT@335, CHR\$(170): PRINT@371, ":" 1160 PRINT@399,CHR\$(170): PRINT@408, "T=ANGLE OF LAG":PRINT@432," VR": PRINT@435,":" ----> I" 1180 LET X1=32:Y1=21:X2=102:Y2=4: GOSUB 1550 1190 PRINT@563,CHR\$(168) 1200 PRINT@627, CHR\$(170): PRINT@629, "X =" 1210 PRINT"IMPEDANCE": PRINT@691, CHR\$(170): PRINT@693, "LW-(1/CW). 1220 PRINT"DIAGRAM": PRINT@731, "Z": PRINT@755, CHR\$(170): PRINT@75 7,"LW > 1/CW" 1230 PRINT@819.CHR\$(170):PRINT@821."I.E. X IS +" 1240 PRINT@867, "R": PRINT@883, CHR\$(170) 1250 PRINT@909, "O": PRINT@912, STRING\$(35, 140); CHR\$(142) 1260 LET X1=33;Y1=42:X2=103:Y2=24: GOSUB 1550 1270 RETURN 1280 CLS: PRINT @66, "---":PRINT@ 70, CHR\$(172); CHR\$(186); CHR\$(17 1);CHR\$(186);CHR\$(171);CHR\$(186);CHR\$(171);CHR\$(186);CHR\$(171);CH R\$(186);CHR\$(171);CHR\$(186);CHR\$(131);"-----" 1290 PRINT092, STRING\$(12, 191); "-----"; CHR\$(170); "; CHR\$(1 49);"---->": PRINT@185,"CURRENT" 1300 PRINT@203, "RES.": PRINT@224, "IND.": PRINT@243, "CAP." 1310 PRINT0259, CHR\$(170); "<---- VR ---->"; CHR\$(138); "<-------- VL ----->";CHR\$(138);"<---- VC ---->";CHR\$(170) 1320 PRINT0323, CHR\$(170); "<----- APPLIED VOLTAGE ' E' ----->"; CHR\$(170) 1330 PRINT@475, CHR\$(170);" VL (LEAD)": PRINT@495, CHR\$(168);"------> I" 1340 PRINT@539, CHR\$(170): PRINT@559, CHR\$(170) 1350 PRINT@603, CHR\$(170): PRINT@623, CHR\$(170) 1360 PRINT@667, CHR\$(170): PRINT@687, CHR\$(170) 1370 PRINT@708, "-----> VR": PRINT@731, CHR\$(170): PRINT @751, CHR\$(170) 1380 PRINT0772, "-----> I": PRINT0795, CHR\$(130); "-----------> I": PRINT@815, CHR\$(138);" VC (LAG)" 1390 PRINT0854, "GENERAL SERIES CIRCUIT" 1400 PRINT@918, STRING\$(22,131):RETURN 1410 PRINT TAB(11)">>> REVIEW OF VALUES OF ALL VARIABLES <<<" 1420 PRINT@79,STRING\$(33,131) 1430 PRINT"CURRENT I ="; I,; "INDUCTANCE L =";L 1440 PRINT "VOLTAGE E =";E,;"FREQUENCY F =":F 1450 PRINT"RESISTANCE R =";R,; "CAPACITANCE C =":C X =";X,; "IMPEDANCE 1460 PRINT"EQUIV REACT Z =";Z T =";T,; "POWER FACTOR 1470 PRINT"PHASE ANGLE Q =":Q

1480 PRINT"IND. REACT. G =";G,; "IND. VOLT DROP VL =";VL 40 IFD<>-10PEN"0",1,"DRWH0/DAT" VOLUME 50 S="":FORI=1T0210:READJ:S=S+CHR\$(J):NEXTI:IFD<>-1THENPRINT#D,C 1490 PRINT"CAP. REACT. J =";J,; "CAP.VOLT DROP VC =";VC 1500 PRINT"R.VOLT DROP VR =";VR,;"RESONANT FREQ. M =";M HR\$(34);S;CHR\$(34)ELSEPRINT#-1,CHR\$(34);S;CHR\$(34) 1510 PRINT"POWER (WATTS) P =";P 60 DATA255, 159, 159, 255, 47, 243, 145, 255, 255, 31, 84, 255, 159, 246, 243, ω 247,111,63,100,102,87,86,255,255,255,255,241,66,255,240,104,152,1 1520 PRINT@848, -- ALL CALCULATIONS COMPLETED --": RETURN N 1530 PRINT@(964), "< ENTER >";: INPUT A\$ 52,85,50,255,21,63,255,21,242,255,255,243,255,17,50,253,112,152,2 23, 150, 152, 137, 112, 96, 7, 240, 118, 112 ω 70 DATA147,105,255,63,255,243,248,255,255,1,96,248,136,136,159,2 1550 IF ABS(X2-X1)<ABS(Y2-Y1) GOTO 1580 1560 DY = (Y2 - Y1) / ABS(X2 - X1)55, 246, 255, 115, 240, 255, 246, 249, 255, 68, 68, 50, 240, 112, 244, 255, 255, 1 (JULY, 1982) 27, 247, 255, 247, 255, 132, 159, 238, 238, 233, 119, 39, 255, 255, 247, 246, 255 1570 IF X2>X1 GOTO 1640 1580 FOR N=X1 TO X2 STEP-1 ,255,95,68,84,16,255,242,255,255,255,50 80 DATA146,134,253,146,51,255,24,48,253,37,17,253,238,238,246,83 ,117,253,120,112,79,105,86,253,96,146,253,23,128,253,62,228,254,4 1,120,255,113,54,111,231,226,242,126,9,247,142,158,249,232,46,248 1610 IF Y1<0 THEN Y1=0 ,52,33,63,101,25,95,24,69,79 90 DATA153,153,153,66,84,255,20,51,255,120,99,255,33,81,255,120, 70, 31, 255, 240, 255, 57, 136, 245, 55, 87, 255, 99, 99, 112 100 S="":FORI=1T070:READJ:S=S+CHR\$(J):NEXTI:IFD<>-1THENPRINT#D.C HR\$(34);S;CHR\$(34)ELSEPRINT#-1.CHR\$(34);S;CHR\$(34) 1670 IF Y1<0 THEN Y1=0 110 DATA99, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 55, 99, 14, 15, 10, 10, 10, 16, 17 ,23,19,18,21,22,20,18,24,25,32,27,28,29,30,31,26,33,99,35,36,35,3 5, 37, 35, 38, 35, 35, 35, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 50, 51, 52, 49, 49, 1700 DX = (X2 - X1) / ABS(Y2 - Y1)50.53.50.49.50 1710 IF Y2>Y1 GOTO 1780 120 S="":FORI=1T016:FORJ=1T04:READK:S=S+CHR\$(K):NEXTJ,I:IFD<>-1T 1720 FOR N=Y1 TO Y2 STEP -1 HENPRINT#D, CHR\$ (34); S; CHR\$ (34) ELSEPRINT#-1, CHR\$ (34); S; CHR\$ (34) 130 DATA99,1,99,8,6,2,5,9,58,3,5,8,12,4,2,8,19,5,10,66,23,6,1,8, MI CR0-80 25, 7, 2, 43, 22, 8, 1, 8, 30, 9, 3, 27, 35, 10, 1, 8, 40, 11, 2, 9, 44, 12, 1, 26, 48, 17 1750 IF X1<0 THEN X1=0 ,2,9,50,14,1,59,53,15,1,8,8,16,1,43 140 ST="":FORI=1T078:READX:ST=ST+CHR\$(X):NEXTI:IFD<>-1THENPRINT# D, CHR\$ (34); ST; CHR\$ (34) ELSEPRINT#-1, CHR\$ (34); ST; CHR\$ (34) 150 DATA31, 36, 36, 42, 45, 42, 20, 17, 38, 52, 45, 46, 51, 6, 50, 33, 59, 16, 6, 2 6,3,3,21,7,8,15,60,25 160 DATA35, 22, 55, 37, 27, 7, 9, 36, 39, 9, 40, 14, 38, 58, 56, 1, 1, 41, 22, 19, 8 1810 IF X1<0 THEN X1=0 .49.32.38.12 170 DATA37,23,33,10,39,2,40,99,11,46,57,43,99,2,35,41,17,48,53,5 4.5.5.4.4.44 180 PRINT"Initiallisation finished.":IFD<>-1CLOSED **** DR. WHO - LEVEL II **** **** DR. WHO - LEVEL II **** MAIN PROGRAM INITIALISER 10 GOTO640'Dr Who V3.1.James Smith.6 Northern Ave. TARRO 2322 20 PRINTS7"deep pit.":GOSUB580:RETURN 10 CLEAR5000: DEFINTA-Z: DEFSTRS' 30 PRINTS7"small chamber.":RETURN PROGRAM: Initialliser for Dr Who V1.1 40 PRINTSY"following a trail around a steep cliff.":RETURN AUTHOR: James Smith 50 PRINT"The path forks here.":RETURN 5th March. 1982 60 PRINTS8"top";:GOT0590 ADDRESS: 6 Northern Ave. TARRO 2322 Doctor Who data file initialliser 70 PRINTS8"base"::G0T0590 80 PRINTSY"lost in the maze.":RETURN 90 PRINT"The path widens for a bit.":RETURN

100 PRINTS7"secret room.":RETURN

110 PRINTS7"passage near some stairs.":RETURN

120 PRINTS8"junction of four underground passages.":RETURN

DATE:

20 CLS:PRINT"

Disk or Tape (D/T) ?

1540 CLS: RETURN

1590 SET(N, Y1)

1650 SET (N, Y1) 1660 Y1=Y1+DY

1640 FOR N=X1 TO X2

1600 Y1=Y1+DY

1620 NEXT N

1630 RETURN

1680 NEXT N

1690 RETURN

1730 SET (X1,N)

1740 X1=X1+DX

1780 FOR N=Y1 TO Y2 1790 SET (X1.N)

1760 NEXT N

1770 RETURN

1800 X1=X1+DX

1820 NEXT N

1830 RETURN

30 A\$=INKEY\$:IFA\$="D"THEND=1:PRINT"Creating disk file DRWHO/DAT. "ELSEIFA\$="T"THEND=-1:PRINT"Dumping data to tape."ELSE30

PAGE ω

450 GOSUB610:PRINT"into a 130 PRINTS7"big valley. To the south stands a vast black monolith. "S9"a thick haze to the north.":RETURN bottomless chasm filled with writhing yellow fog.":RETURN 460 PRINTSY"on the brim of a deep volcano.":RETURN 140 GOSUB570: PRINT"A message scrawled on it says: 'DIG HERE'":RE 470 PRINTS8"summit of a very tall mountain. "S9"a 150 PRINTS7"small cave of broken rocks.":RETURN thick foo here.":RETURN 480 GOSUB620: PRINT"4' chimney from which is pouring 160 PRINTS8"base of the monolith. "S9"a small grate ia very dense thick yellow smoq.":RETURN at your feet.":RETURN 490 GOSUB620:PRINT"small hole into which the fog is 170 GOSUB120:PRINT"Under a grate.":RETURN flowing.":RETURN 180 PRINTS6"Daleks. Passages lead off in all 500 GOSUB620:PRINT"fast flowing river. The water directions. I would not try to get past the Daleks, however.":R is a dull vellow shade.":RETURN 190 PRINTS8"top of a hill looking down upon a domed city.":RETUR 510 PRINTS8"intersection of two passages.":RETURN 520 GOSUB510: GOSUB580: RETURN 200 PRINTSY"in the city's drains. Tunnels lead off in all 530 PRINTS6"broken down tardises.":RETURN 540 GOSUB510:PRINT"A sign nearby says: directions.. ";:GOSUB580:RETURN ".:GOSUB1640:RETURN 210 PRINTS8"gates of the city. At your feet is a manhole 550 PRINTS7"magnificent chamber. "S9"a large throne here.":RETUR 220 PRINTSY"in the Dalek master strategy room. Red lights are Ν flashing and sirens screaming. I would leave.":RETURN 560 GOSUB630:PRINT", with a 230 PRINTSY"in an alcove that had been hidden behind a hole in it.":RETURN 570 GOSUB630:PRINT".":RETURN deactivated Dalek.":RETURN 580 PRINT"There are some stairs here.":RETURN 240 PRINTS6"broken machinery.":RETURN 250 PRINTS6"Bubbling Beakers, Burning Bunsens, 590 PRINT" of a steep incline.":RETURN Filter Funnels, Colourful Chemicals, etc.":RETURN 600 PRINT"Okay":GOT0780 260 PRINTSY"inside the Dalek experimentation room. I wouldn't 610 PRINTS8"top of a mountain looking down ";:RETURN linger, as the experiments look hungry.":GOSUB580:RETURN 620 PRINTS7"valley beside a ";:RETURN 630 PRINTS8"base of an unscalable wall of rock";:RETURN 270 PRINTS8"bottom of a deep dust-filled hole.":RETURN 640 CLEAR650: DEFINTA-Z: DEFSNGC: DEFSTRS 280 PRINTSY"just inside a vast crater. "S9"a ship to your 650 DIMSO(17), D(69,6), O(16,4), SI(6), SX(6), V(2), SW(5), L(6), PI(7) north, a deep hole to your south, and a set of rough stone steps leading up the crater wall near a cave entrance.":RETURN 660 DV=-1:SY="You are ":SH="You have ":S7="You are in a ":S8="Yo u are at the ":S9="There is ":S6="You are in a room full of ":CLS 290 PRINTSY"inside the ship near the airlock "RETURN 300 PRINTSH" just entered a room full of Cybermen. Back out very :PRINT0342, "Doctor Who"; 670 DATAAqqador.a pick.a desionating renticulator.a sonic screwd slowly and they might not see you. Be carefull!":RETURN river, a giant blue spider !!!, a bag of jelly babies, a Dalek ray-gu 310 PRINTS8"base of the Cybership.":RETURN n,a copy of 'Playdalek',a large rock,a long scarf,a strange looki 320 PRINTS7"cave in the cliff face.":RETURN ng object, a bunch of bananas, a dead spider 330 PRINTSY"beneath the Cybership.":RETURN 340 PRINTS8"top of a cliff overlooking a spaceship.":RETURN 680 DATAa white crystal, a blue crystal, a sionated cumquat, a skul 350 PRINTS6"heavy machinery.":RETURN 1 360 PRINTS6"television screens showing views of 690 DATAgroup of Peladonians, trio of ugly mutants, group of Dalek different areas. For example, one shows a white crystal lying s,Cyberman, single loathesome horrible creature, Troglodyte, Timelor d, 49, 56, 57, 58, 27, 59, 60, 2, 3, 4, 6, 5, 99, 1 beside a mountain path.":RETURN 370 PRINTSY"lost in the slime. "S9"a tree in the distance.":RET 1131122111113511111313331211131112131223111113111131113" 380 PRINTSY"standing at the high point on a vast plain of slime. 710 SC="AGGBABBAGBANBLUBUNCLOCOMCOPCRECRYCUMCYBD DALDESDIEDIGDO "S9"a lone tree to your north.":RETURN WDROE EASEATENTEXIFIRGALGETGIAGIYGRAGUNHIDIN INVJELLARLISLONLOOM 390 PRINTS8"top of a tall tree. The tardis can be seen to AGMOOMUTN NOROBJOFFOPEOUTPELPICPLAQUIRAYREARENRESROCS SCASCRSEA the south.":RETURN 400 PRINTSY at a clearing in the slime. "S9" a tall tree here.": 410 PRINTSY"following a narrow path along the edge of the cliff. 420 PRINTSY"in the midst of a thick fog.":RETURN

430 PRINTS7"valley between two mountains. A thick fog is rolling in from the north.":RETURN

440 GOSUB610:PRINT"upon a vast expanse of ocean.":RETURN

TURN

ETURN

cover.":RETURN

N

URN

RETURN

":RETURN

SIOSKASKUSONSOUSPISTRTALTARTIMTROU UP W WESWHI" 720 REM OPEN FILE HERE FOR DISK 730 INPUT#-1,SP:INPUT#-1,SD:INPUT#-1,SO:INPUT#-1,S1:S1=LEFT\$(S1, 60 + CHR\$ (34) + MID\$ (S1.62.4) + CHR\$ (34) + RIGHT\$ (S1.12) 740 FORI=1T016:FORJ=1T04:O(I,J)=ASC(MID\$(SO,(I-1)*4+J,1)):NEXTJ, I:FORI=0T069:FORJ=1T05STEP2:X=ASC(MID\$(SP.I*3+(J+1)/2.1)):D(I.J)= INT (X/16):D(I, J+1)=X-INT (X/16) *16:NEXTJ, I:FORI=0T069:D(I,0)=ASC(M ID\$(SD.I+1.1)):NEXTI

750 RESTORE: FORI=1T017: READSO(I): NEXT: FORI=0T06: READSI(I): NEXT: F ORI=OTO6:READL(I):NEXT:FORI=1T07:READPI(I):NEXT

760 RANDOM: Q9=0: C=0: WT=0: TI=0: L=6: PD=1: P=61: TL=20+RND(20): CLS 1060 IFPO=99PRINTSY"inside the tardis.":GOT01090 770 D(14,0)=13:D(39,0)=34:D(63,3)=2:D(63,4)=2:O(1,1)=0:D(1,2)=1:1070 OND(P,0)+160SUB20,30,40,50,60,70,80,90,100,110,120,130,140, D(0,0)=0:D(1,3)=0:D(69,1)=0:D(69,2)=0:D(69,3)=0:D(69,4)=0:GOT0107 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 3 10, 320, 330, 340, 350, 360, 370, 380, 390, 400, 410, 420, 430, 440, 450, 460, 47 0,480,490,500,510,520,530,540,550,560,570,580,590,600 1080 IFPO=1PRINT"The tardis is sitting off to one side." 1090 GOSUB1820:NO=0:FORI=1T016:IFP=0(I,1)AND(0(I,4)AND1)=0NO=NO+ 1: IFNO=1THENPRINT"Around you you see:":PRINT, SO(O(I,2))ELSEPRINT, SO(O(I,2)) 1100 NEXTI: GOT0780 1110 IFPO=99PRINTSY"still in the tardis.":GOTO780ELSEY=D(P,V(0)) : IFP=19ANDD(19.1)=15THEN1680ELSEIFY=15THEN1530ELSEIFY=130NLG0T017 00,1110,1110,1740ELSEIFY=140NL-2G0T01710,1720,1730 1120 IFL<>6ANDC>2.5ANDRND(0)>.5IFL>2THENPRINT"He has"::GOT01690E LSEPRINT"They have";:GOT01690 1130 IFP=36ANDRND(0)>.3PRINT"The ship just blasted off. As you were underneath it at the time, "SY"now slightly incinerated.":GOT01420 1140 IFP=62ANDRND(0)>.3PRINTSH"been hit by a low flying tardis." :GOT01420 1150 IFP=6ANDRND(0)<.9Y=P0 1160 PO=Y:P=L*10+PO:C=-(P=200RP=330RP=66):GOT01060 1170 F=0:IFPO<>99S="already outside"ELSES="now outside":PO=1:F=-1:P=1+L*10 1180 PRINTSY;S" the tardis.":IFFTHEN1060ELSE780 1190 F=0:C=0:IFPO=99S="already inside":GOTO1180ELSEIFPO<>1THEN15 50ELSES="now inside":PO=99:P=PO:GOT01180 1200 IFPO<>99THEN1540ELSETI=TI+1:IFTI>TLTHENPRINT"Your tardis ha s run out of power .. "SY"'Lost in Space'!!":GOT01420 1210 FORI=OTO6:IFV(1)=L(I)D=IELSENEXTI:D=RND(7)-1 1220 IFRND(2)=1D=RND(10)-1:IFD>6PRINTSH"materialised in space. I will reset the tardis for vou.":D=RND(7)-1 1230 L=D:PRINTSH"materialised on ";:IFL=3PRINT"the dark side of the Moon":GOT0780ELSEPRINTMID\$("Peladon Mutos Hidaous DiethylamideGalafry",L*12+1,12):GOT078 0 1240 PRINTSH"found:":F=0:IFP=6ANDRND(0)>.30RL=4ANDPO<>6ANDPO<>4A NDPO<>1ANDRND(0)>.7THEN1260ELSEFORI=1T016:IFO(I.1)=PTHENF=F+1:PRI NT, SO(O(I, 2)):O(I, 4) = (O(I, 4)OR5) - 11250 NEXTI: IFP=26F=F+1: PRINT" a secret passage hidden behind a deactivated Dalek.":D(26,3)=5 1260 IFFTHEN780ELSEPRINT, "nothing":GOT0780 1270 IFV(1)=360RV(1)=420RV(1)=46IF0(V(1)-30.1)=90THEN0(V(1)-30.1))=91:PRINT"Delicious!!":GOT0780ELSE1500ELSE1450 1280 IFO(2,1)<>90THENPRINTSH"nothing to dig with.":GOT0780ELSEIF P=12ANDD(12,4)=15THEND(12,4)=6:D(12,0)=54:GOT01300ELSEIFP=13ANDD(13,3)=15THEND(13,3)=4:D(13,0)=54:GOT01300

1290 PRINT"You haven't dug up anything useful.":GOTO780

1300 PRINTSH"broken a hole in the wall with your pick. You can see

a vague chamber behind it.":GOT0780

1310 IFV(1)<>55ANDV(2)<>55THEN1450ELSEIFL<>10RP0<50RP0>6THEN1480 ELSEIFO(4,1)<>90THEN1450ELSEIFV(0)=20THEN1330 1320 D(15.6)=6:D(16.5)=5:GOT0600

1330 D(15,6)=13:D(16,5)=13:GOT0600

940 O(X1,1)=90:WT=WT+O(X1,3):O(X1,4)=(O(X1,4)OR1)-1:GOTO600

800 IFC>0C=C+RND(0)

810 IF (P=660RP=7) ANDC=0ANDRND(0) >.40RRND(0) >.9ANDC=0C=1

790 IFP=60RP=80RP=250RP=440RP=990RL=4ANDQ9THEN840

820 IFP=19ANDO(5,2)=5IFC<4THEN840ELSEPRINT"The spider has eaten you.":GOT01420

830 IFC=1PRINT"Ohoh, here comes a "SI(L)"!"ELSEIFCPRINTS9"a "SI(L)" here!"

840 A\$="":INPUTA\$:IFA\$=""140

780 IFP0=99P=99ELSEP=L*10+P0

0

850 V(0)=100:V(1)=100:V(2)=100:FORI=1T05:SW(I)="":NEXTI

860 J=0:K=-1:FORI=1TOLEN(A\$):X\$=MID\$(A\$,I,1):IFX\$=" "THENIFKTHEN NEXTIELSEK=-1:NEXTIELSEIFNOTKTHENSW(J)=SW(J)+X\$:NEXTIELSEIFJ<5THE NK=0:J=J+1:SW(J)=X\$:NEXTI

870 FORI=1TOJ:SW(I)=LEFT\$(SW(I)+" ",3):NEXTI:K=0:FORI=1TOJ:A1= 1:AN=78:FORI1=1T078:AI=INT((A1+AN)/2):X\$=MID\$(SC.AI*3-2.3):IFSW(I)>X\$A1=AI+1ELSEAN=AI-1

880 IFX\$<>SW(I)ANDA1<=ANNEXTI1ELSEIFX\$=SW(I)V(K)=ASC(MID\$(S1,AI, 1)):K=K+1:IFK>2THENGOSUB1840ELSENEXTI:GOSUB1840ELSENEXTI:GOSUB184 0

890 IFRND(0)>.998ANDWTANDPS<>99PRINTSY"hit over the back of the head and everything goes black

...You awake to find that everything you were holding is gone.": FORI=2T016: IFO(I.1)=90THENO(I.1)=6:NEXTI:GOT0840ELSENEXTI:GOT0840 900 IFV(0)=22ANDV(2)=1000RV(1)=100AND(V(0)>17ANDV(0)<27ANDV(0)<> 22) PRINT"Please be more specific.":GOT0840

910 IFV(0)=33ANDV(1)=46THENIF0(3,1)=90AND0(16,1)=900(16,1)=91:PR INTSH"desionated the cumquat into";:GOT01410ELSE1500

920 ONV(0)G0T01110.1110.1110.1110.1110.1110.1170.1020.1200. 1240, 1440, 840, 1060, 1370, 1280, 970, 840, 1310, 1310, 1270, 1350, 1340, 840 ,930,950,1040:PRINT"What?":GOT0840

930 IFV(1)<310RV(1)>46THEN1450ELSEX1=V(1)-30:IF0(X1,1)=90THEN146 OELSEIFO(X1,1) <>PTHEN1480ELSEIF(O(X1,4)AND8)=OTHEN1450ELSEIFO(X1,)3)+WT>20THEN1490

950 IFV(1)<310RV(1)>46THEN1450ELSEX1=V(1)-30:IF0(X1,1)<>90THENIF $O(X1,1) \leq PORO(X1,1) = 91$ THEN1480ELSE1500ELSEIFV(1) = 31THEN1510 960 WT=WT-O(X1,3):O(X1,1)=P:GOTO600

970 IFC=0THEN1520ELSEIFP=7THEN1570ELSEIFL=4ANDC<0THEN1620ELSEIFP =66G0SUB1000: IFN0<6THEN1750ELSE1430

980 ONL+160T01580,1520,1590,1600,1610,990,1630

990 ONRND(3)GOT01650,1660,1670

1000 QC=0:ND=0:FORI=1T016:IF(0(I,4)AND2)AND0(I,1)=90N0=N0+1:QC=Q C+(O(I.4)AND240)/16

1010 NEXTI:RETURN

1020 PRINTSY"presently holding:":NO=0:FORI=1T016:IFO(I,1)=90PRIN T,SO(O(I,2)):NO=NO+1

1030 NEXTI: IFNO=OPRINT, "nothing": GOTO780ELSE780

1040 IFL=60RC=0PRINTS9"no need to hide!":GOT0780ELSEIFP=19ANDD(1

9,1)=15THEN1450ELSEIFRND(0)>.95PRINT"The creature that you bumped as you rushed desperately past was faster than you. He caught and killed you.":GOT01420ELSEC=0

1050 IFL=5P0=1:P=51ELSEIFL=4P0=4:P=44ELSEP0=6:P=L*10+P0

PAGE

ω

Skaro

MI CR0-80

1340 IFV(1)=38IFO(8,1)=90THENPRINT"This does nothing for me, you 1630 GOSUB1640:GOT0780 know.":GOT0780ELSE1500ELSEIFV(1)=33IF0(3.1)=90THENPRINT"It says" 1640 PRINT"'BEWARE OF LOW FLYING TARDISES'":RETURN ::GOSUB1000:PRINTABS(NO+RND(3)-2+RND(0)):GOT0780ELSE1500ELSE1450 1350 IFV(1)=35ANDV(2)=45ANDD(15,1)=90ANDP=19ANDD(19,1)=15C=0:D(1 T0780 9.1 = 7:0 (5.1) = 14:0 (5.4) = 0 (5.4) OR8:0 (5.2) = 13:60T01760 1660 PRINT"The fog muffles your voice so that he can't hear you. 1360 IFV(1)<310RV(1)>460RV(2)<490RV(2)>56THEN1450ELSEY=PI(V(2)-4 ":GOT0780 9): IFY=990RY<>LORC=00R0(V(1)-30.1)<>90THEN1450ELSEY=VAL(MID\$(SF.(1670 PRINT"He has spoken of a strange crystal on a mountain many V(1)-30)+16*Y,1)):ONYGOTO1770,1780,1790,1800,1810 miles away ...":GOT0780 1370 IFV(1)<>37THEN1450ELSEIF0(7,1)<>90THEN1500ELSEIFC=0PRINTS9" nothing here to fire at.":GOT0780ELSEC=0 1680 PRINT"As you approached the spider, it turned red and start 1380 IFL=6PRINTS9"a loud explosion and ... Hey that was a Timelo ed spittingvellow poison. I wouldn't try that again!":GOTO780 rd! 1690 PRINT" taken an intense dislike to you and torn Just whose side are you on? That's it for you, buster!":GOT0142 you to pieces.":GOT01420 Ω 1700 PRINTSH"to open the grate first.":GOT0780 1390 IFP=19D(19.1)=7:0(5.1)=91 1710 PRINTSH"been strangled by a Cyberman.":GOT01420 1400 PRINTS9"a loud explosion and a brilliant flash, and you fin 1720 PRINT"You jumped from the tree and broke your neck. Silly ы you.":GOT01420 yourself looking at";:IFL=409=1 1730 PRINTSH" jumped to your doom. ":GOT01420 1410 PRINT" a small pile of grey powder 1740 PRINTSH"taken a mouth full of slime. (Uggh!!!)":GOT0780 which slowly dissipates.":GOT0780 1750 PRINT"'These are not the right parts. Take them from my si 1420 PRINTSH"failed ... the universe will be destroyed ... and a aht!'":GOT0780 11 1760 PRINT"The spider looked deep into the crystal, turned green the blame lies on your shoulders ...":GOT01440 and rose 1430 PRINTSH"found all six parts!! We are saved!!! slowly up through the roof. I think it was dead!":GOT0780 ", "Congratulations!!!!" 1770 PRINT"He doesn't seem interested in it.":GOT0780 1440 GOSUB1000:QC=(QC*NO*100)*TL/(TI+1):PRINT"You scored"QC:INPU 1780 PRINT"He takes it and scarpers.":C=0:O(V(1)-30,1)=L*10+RND(T"Want to try again":X\$:IFLEFT\$(X\$.1)="N"THENCLS:ENDELSE740 10) - 1:60T07801450 PRINT"I can't guite see how to do that.":GOT0780 1790 PRINT"He eats it.": O(V(1)-30,1)=91:C=.01:GOT0780 1460 PRINT"You're already carrying it.":GOT0780 1800 PRINT"The Daleks take one look at your copy of 'Playdalek' 1470 PRINT"You can't do that!":GOT0780 and freak 1480 PRINT"I can't see that here.":GOT0780 out down separate tunnels doing cartwheels and blasting the 1490 PRINT"You can't carry all this and that too. You'll have t walls in strange ways. (This must be hot stuff!!)":C=0:GOT0780 o drop 1810 PRINT"The creature has calmed down a little and looks frien something first.":GOT0780 dly (ugly, 1500 PRINT"You aren't carrying it.":GOT0780 but friendly)...even talkative!":C=-1:GOT0780 1510 PRINT"He doesn't seem to want to go away.":GOT0780 1820 IFP=99RETURNELSEPRINT"You can go";:FORI=1T06:IFD(P,I)<>15IF 1520 PRINTS9"nobody else here who can talk.":GOTO780 I<3PRINTMID\$(" north, south,", I*7-6,7); ELSEIFI=5PRINT" up, "; ELSEP 1530 PRINTS9"no way to go in that direction.":GOT0780 RINTMID\$(" east, west, , down, ", I*6-17,6); 1540 PRINTSH"to be in the tardis to use it.":GOT0780 1830 NEXT:PRINTCHR\$(8)".":RETURN 1550 PRINTS9"no tardis here.":GOT0780 1840 IFV(0)>V(1)ORV(0)>V(2)T=-((V(1)<=V(2))*1+(V(1)>V(2))*2):T1= 1560 PRINT"You can't get through the grate without opening it fi V(0):V(0)=V(T):V(T)=T1rst.":GOT0780 1850 IFV(1)>V(2)T=V(2):V(2)=V(1):V(1)=T:RETURNELSERETURN 1570 PRINT"They have spoken of a secret doorway behind a nearby 1860 SAVE"DRWHO/TBS" curtain.":D(7.3)=8:GOT0780 1580 PRINT"One of them poked you with a spear, but you were not killed. just mortally wounded.":GOT0780 **** DR. WHO - LEVEL II **** 1590 PRINT"'E x t e r m i n a t e' ... it zapped you. "SY"now a DISK CHANGES slowly evaporating pile of grey powder.":GOT01420 1600 PRINT"'Have you got a spot of oil on you?'":GOT0780 1610 PRINT"'Creature hungry ... can creature eat you?'":GOT0780 1620 PRINT"'I say, these sweeties are simply spiffing. Rather. 720 OPEN"I",1,"DRWHO/DAT Not at 730 INPUT#1, SP: INPUT#1, SD: INPUT#1, SO: INPUT#1, S1: S1=LEFT\$(S1, 60)+ all like those bananas. Crunchy bananas are just simply not CHR\$(34)+MID\$(S1,62,4)+CHR\$(34)+RIGHT\$(S1,12):CLOSE1

on, what?'":C=.01:GOT0780

***** 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.

** JUMP THE RAPIDS LII/16K (80) **

This game makes use of the graphic utility (MOVIE) published in MICRO - 80 issue 22. In this game you have to cross a river by jumping from log to log, but with the graphic utility Movie driving the graphics, it's not easy.

** MORSE CODE TRANSMITTER LII/4K-16K **

This morse code program will transmit morse code out to the cassette port. The code can be random or can be supplied by you. As actual morse code is transmitted by this program it is conceivable that the computer output could be hooked up to a transmitter for live transmission.

** CHEQUE BOOK DATA FILE (COLOUR) **

This program enables you to save all your cheque book transactions on a cassette data file and retrieve them at will. A better way to check queries than to search through the stubs!

** FAULT FINDER LII/16K (80) **

This program is problem in deductive a The scenario is of a machine (forlogic. tunately which fictitious), is subject to three particularly undesirable faults when combinations of actions are taken. The computer will act as your agent in performing the actions that you order and reporting the results. You are required to determine what the common features of combinations of actions which produce the faults are and, hence, by trial and error, how to avoid the faults. (Huh! - Ed.)

** CHEQUE ACCOUNT MANAGER LII/16K (80) **

This program will store up to 50 transactions of your cheque account. It maintains a balance of the account. Being fully interactive, it will check bank statements, edit records, sort records and run record searches.

** PAYROLL (COLOUR) **

A simple payroll program originally designed to operate on a Level II, 16K cassette system. This program will allow you to manage the payroll for a small staff on your Colour Computer.

anvergammes power suppry as musurated, two drive cable and the ver-single drive-cablines power suppry as indistrated. sion of DOSPLUS indicated.
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 dualdrive 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.

MPI B51 40 track, single-head, 100K MPI B52 40 track, dual-head, 200K MPI B92 80 track, dual-head, 400K Simple, wrap-around cabinet	Price \$399 Reduced Price \$449 \$619 \$12	Freight \$5.00 \$5.00 \$5.00 \$2.00	Self-contained, single drive cabinet/power supply Self-contained, dual-drive cabinet/power supply Two drive cable Fan drive cable DOSPLUS 3.3	Price \$99 \$135 \$39 \$49 \$99 95	Freight \$5.00 \$5.00 \$2.00 \$2.00 \$2.00
Separate, dual-drive power supply	\$85	\$8.00	DOSPLUS 3.3 DOSPLUS 3.4	\$99.95 \$149.95	\$2.00 \$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.

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 \$19.951 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