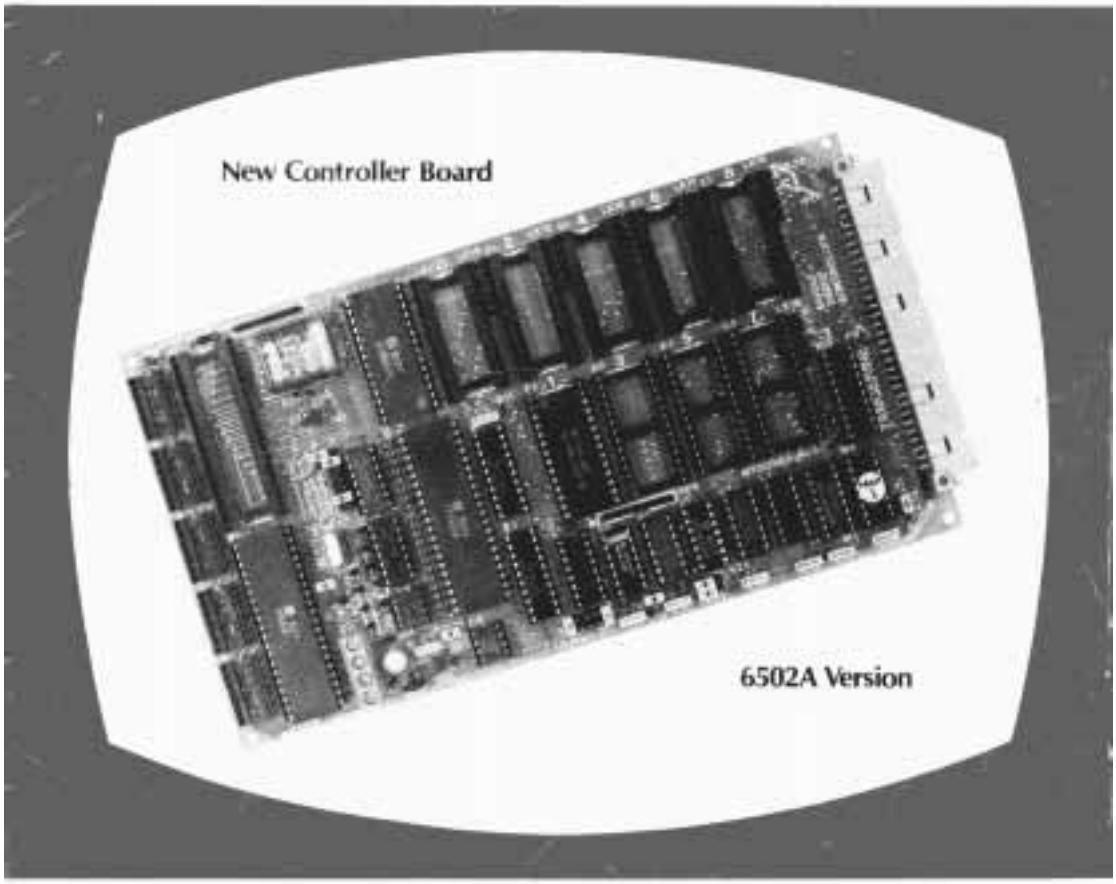
MICROTAN WORLD

Volume No. 1 | Feb.-Mar. 1984 | Issue No. 6

Programs · Reviews · Your Letters



SYSTEM NINE - UPDATE

PUBLISHERS
MICROTANIC COMPUTER SYSTEMS LIMITED
16 UPLAND ROAD, DULWICH, LONDON SE22. 01 693 1137

SOFTWARE FOR THE MICROTAN SYSTEM

REGISTERED OFFICE
235 FRIERN ROAD
DULWICH
LONDON SE22

MICRO COMPUTERS
CONSULTANTS
SOFTWARE
SERVICES
RENTALS
HARDWARE

SHOWROOMS
16 UPLAND ROAD
DULWICH
LONDON SE22
TEL 01 693 1137

MICROTANIC SOFTWARE PRODUCT LIST

ADVENTURE 1	7K M/C	5.95
ADVENTURE 2	7K BAS	6.95
ADVENTURE 3	16K M/C	9.95
ADVENTURE PACK	3 x 7K BAS	14.95
THE DEFENDER	7K M/C	6.95
TANK RAID	16K M/C	9.95
EARTH ATTACK	7K M/C	6.95
SPACE INVADER	3K M/C	5.95
THE GOBBLER	6K M/C	6.95
SUB STRIKE	5K BAS	5.95
STAR CHESS	3K M/C	6.95
3D MAZE	7K BAS	6.95
3D OXO	7K BAS	6.95
GRAPHIC PUZZLES	7K M/C	6.95
MOLE SWAT	7K BAS	6.95
GAMES PACK 1	3 x 7K BAS	8.95
GAMES PACK 2	3 x 7K BAS	8.95
GAMES PACK 3	3 x 7K BAS	8.95
GAMES PACK 4	4 x 7K M/C	8.95
AUTHOR	7K M/C	6.95

TOOLKIT 1	2K EPROM	22.50
MICROTANTEL	2K EPROM	16.95
HI RES TOOLKIT	2K EPROM	16.95
FORTH LANGUAGE	IN EPROMS	34.95
2 PASS ASSEMBLER	EPROM	34.95
TOOLKIT 2	2K EPROM	16.95
ASIMOV WORD/PROC	EPROMS	49.95
FILE UTILITIES BAS TAPE		9.95
M/C RELOCATOR M/C TAPE		5.95
EPROM PROGRAMMER SELF BUILD		9.95
TEXT PROCESSOR M/C TAPE		14.95
BAS LINKED ASSEMBLER TAPE		6.95
MORSE CODE GENERATOR TAPE		4.95
SINCLAIR PRINTER INTERFACE		29.95
DISPLAY PLANNER M/C TAPE		12.95
E2 EPROM EXTENSION CARD		22.50
H2 EPROM EXTENSION CARD		24.50
25 x 64 COLOUR VDU BOARD		85.00
LOOSELEAF DATA BASE TAPE		19.95
NEW SOUND BOARD		19.95

UNIVERSAL EPROM ERASER FULLY BUILT TO ERASE UP TO 8 EPROMS AT A TIME 34.95

HI RESOLUTION TOOLKIT 16.95

TOOLKIT - ADVENTURE GAME - 3 PASS ASSEMBLER AVAILABLE FOR NEW SCREEN BRD POA

PILOT LANGUAGE NOW AVAILABLE FOR THE MICROTAN – THIS EXTENSION TO BASIC WILL ENABLE YOU TO WRITE AND RUN COMPUTER ASSISTED LEARNING (CAL) AND COMPUTER ASSISTED INSTRUCTION(CAI) PROGRAMS ON YOUR MICROTAN – (TAPE OR EPROM) 19.95

LARGE SELECTION OF COMPUTER BOOKS NOW AVAILABLE FROM STOCK – WRITE FOR DETAILS

If You have any programs or Hardware ideas please contact Mr. David Northway

ALL PRICES INCLUDE V.A.T. BUT PLEASE ADD 00.60p FOR POST AND PACKING

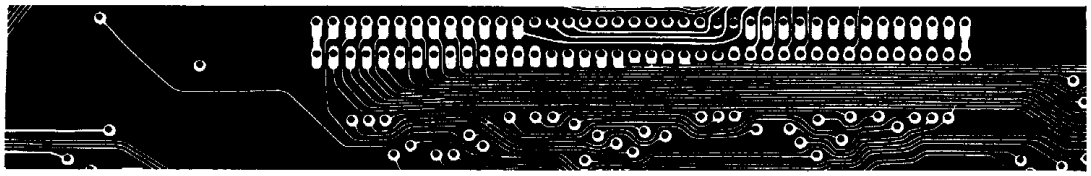
MICROTANIC COMPUTER SYSTEMS LTD.

Computer Manufacturer
Software · Hardware · Books

16 Upland Road
Dulwich, London SE 22

Registration No: 1668843 V.A.T No. 237794718 Telephone: 01-693 1137

Contents



WHO'S WHO AND WHAT'S WHAT

The Company

Microtanic Computer Systems Ltd. 16 Upland Road,
Dulwich, London SE22. 01 693 1137

Managing Director	David Northway
Director	Charles Cooper
Director & General Manager	Robin Northway
Design Consultant	Dr. P.N. Gaunt

Microtan World

Published by	Microtanic Computer Systems Ltd
Editor	Deryck M. Sutton
Repairs & Technical	Andy Parnell & Clive Reynolds

The Microtan System

1K Computer in Machine Code

Microtan 65 + Keypad
+ u/l case + Graphics

8K Computer Expansion

As above + Tanex +
Mini Motherboard +
Power Supply + Full
ASCII Keyboard + Basic

48K Computer Expansion

As 8K + Tanram + MPS2
Power Supply + System
Motherboard.

Additional Expansion

HI RES Board. VDU
Colour Board. Sound
Board. DOS Board.
MASS Storage Eprom
Board

Languages

Basic. Forth. Pilot
Assembler

Also Available

Microtutor
Controller Board

SHOP WINDOW

Main Microtan Dealers

Waltham Forest Service Centre
889 Leabridge Road,
Nr. Whipps Cross,
Walthamstow. E 17.
Tel: 01 520 7747

Micron Computer Equipment Ltd.
Devonshire House,
Devonshire Square,
Loughborough,
Leicester. LE11 3DW

Overseas Dealers

Gloor Instruments Ltd,
Schwizerstrasse 2,
8610 Uster,
Switzerland.

J Muller,
Korblumeinweg,
Ebersheim,
West Germany.

Micromation Systems Ltd,
21 Elma Park Centre,
Edendale Road,
Edonvale 1610,
South Africa.

Mr Beni
Mayost
Rehov Hashoftim 29
Mercaz Rasco
Kiryat Motzkin
Israel.

The Inside Story

by David Northway

I am pleased to tell you that the problems we have experienced with hardware are virtually over and we are now concentrating on software. We are reviewing the programs we have received. We are also looking for more programs, so if you have any that you think would be suitable for reproduction and sale, send them to me at M.C.S. Ltd., for evaluation. We are interested in all programs, games and business etc.

We are currently looking at a Modem which we hope to be able to offer shortly. This has recently received Post Office approval. More details in the next issue.

The article on Flex (TM) and System Nine by Dr. Gaunt is elsewhere in this issue and in issue 7 I confidently expect to be able to publish full details of this new system and the software which will be available.

I am hoping that by issue 7 we will also have available a bank of music data for those of you who have the Sound Board and The Music Translator Program. This data will be available on either tape or disc and will give you a library of tunes to play on your Sound Board.

I have asked the Editor if it is possible to produce an index to cover the first six issues of the magazine for publication in issue 7 to help readers keep track of the main articles.

So as we head for our second year the future looks very exciting and I am sure that with your continued help we shall take great strides forward.

David Northway.

David and team at Bread-Board '83

ED'S Page

Issue No 6 already . . . it does not seem possible that nearly a year has gone by since Microtan World was started, somewhat hesitantly. Now here we are with contributions flowing in and some really great ideas building up.

The next issue will take us into year two and we look forward to some exciting new advanges on several fronts. We will, however, continue to need lots of good articles, hints and tips and of course your letters with all your ideas and comments.

In issue 5 Peter Chamberlain referred to the Simple Simon articles and his view echoed several others. If you do not agree with this view, that is to say, if you do want to see simple articles like that, then put pen to paper and let us know. Whilst writing also let us know what subjects you would like to see treated in this manner.

Niel Mackay queries System Nine and there is the latest article on this subject in this issue. In issue 7 we will be looking more closely at System Nine and also looking at other aspects of the use of the 6809 processor.

Niel Mackay also refers to a MOD on Tanex and there is a comment by a contributor in this issue. If however you require more information Niel, let us know your problem in detail and we will deal with it.

As we move into year two we would like to include articles on your systems. What boards have you got? What power supply? What keyboard? What printer? What monitor? etc. and then what use you make of your system. We will publish the best of articles received with the usual voucher in return.

I hope like me you are looking forward to the 2nd year, until then . . . . .

Happy computing . . . . Ed

OK. Do you have and use a modem?
Send in details as above and we will publish . . . . Ed.

Vouchers now being issued.
Sorry for delay. Ed.

(The error in the article was under circuit operation - Para 2)

(Apologies for detaching the above from Andre Margas letter - see opp. Printer)

Question

As I see it, if you suffer a mains power failure whilst you have a disk in use, or you forget to remove a disk before switching off, you lose all the information on the disk. Am I right? If I am right does anyone have any suggestions to safeguard against this . . . . Ed.

Note: Subscribers who received Issue 1 of Volume 1 need to review their subscription. Please see the order form

VIEWPOINT

Philip J. Vincent
Cambridge

Dear Mr. Northwood,

With regard to the letter by Peter Chamberlin published in issue 5. While I agree that a five page spread of photos, of the Microtan system boards may be a waste of space, I found them interesting and useful, to be able to see the quality of what one's prospective purchases. So, can I see more photos, or perhaps a catalogue the standard of Microtan World, with more photos, technical descriptions and perhaps circuits, ARRRRRRRRRR! Circuits what a blessing they would be, saving me much time and aggravation in track tracing and trying to compile a circuit.

Now for Niel Mackay's, and other's, he might find the following extracts from the Tansoft Gazette of interest. I don't accept any responsibility for the following . . . . .

Ref: Tansoft Gazette Feb/Mar 82,
Problem, bit dropout on Tanram (issue unspecified).
Cure, isolate pin 2 of IC M2, connect 470 Ohm resistor to pin of IC M2 and the other end to pin 11 of IC D8, connect a 330pf (n33 ceramic, preferably) from pin 2 of IC M2 and the outer end to ground.

To be able to use a video monitor with the Microtan 65, a 1 volt peak to peak composite video is available across R7 (75 ohms).

Ref' Transoft Gazette Jun/Jul 82,
To improve operation of crystal oscillator on Tanex serial 10.
R55 is now 1K changes from 10K
R56 is now 1K changes from 10K

To improve the performance of the cassete interface, R5 is now 10K.

Ref: Tansoft Gazette Dec 81/Jan 82.
Problem. Screen jitter on hires graphics board when used with issue 1. Microtan 65 (Note they can be identified easily, as they do not have an EPROM but two PROM's for TANBUG).
Cure, connect a 120 pf (n12 ceramic type) from pin 3 of IC L4 to pin 7 of IC L4.

Issue 1 and 2 Tanex will not work with issue 6 back planes.
Cure, locate and cut the track which join 17a to 17b of the DIN connector, both top and bottom of the board.

I have not done all the modifications to my system, only those which were definitely needed, or cured the problem if it showed up.

Alan Murphy
Camberley

Dear Deryck,

I have produced two more articles which I have enclosed for your inspection (see Programs pages. Ed)

One is concerned with the CATALOG program that I sent you last time, and the other is a new one that produces Banner headers, and could be used for things such as newsletter titles etc in local area computer clubs.

I hope that you will find these of sufficient standard and interest for publication in Microtan world.

Keep on the good work on the magazine.

Andre C. Margas
Cambridge

Firstly let me apologise for the briefness of this letter, but I'm afraid I am rather pushed for time at this time (long-term). I have the following points to make:

1. Many thanks for printing my article on the auto-repeat for Tangerine ASCII keyboards. The quality of the drawings was extremely good. However there was just one typing-error (yours not mine!) in the text. (Sorry - correction below . . . . Ed.)

> With no key depressed, STROBE INPUT is LO, hence pin 1 of N2 is normally LO. Pin 11 of N3 is normally H1. Pin 8 of N4 is always the inverse of pin 11 N3.

2. To improve your sales, why not try and get mail-order companies like Ambit International and Maplin to stock the Tangerine system. Ambit will probably do a review in 'Radio & Electronics World' free, although being journalists, they might expect to keep the 'loaned' review equipment.

3. To help you fill up your pages, why not set up an index of Tangerine owners who have telephone modems. The most popular is the European CCITT V21 standard, which is 300/300 full duplex. Several magazines and Maplin have produced designs to operate on this standard. I have also the capability of the Prestel standard (1200/75 & 75/1200 baud full-duplex). Unfortunately my rack at this moment is 'down' because of PSU problems (blowing up 4118 dynamic RAM's) and I cannot foresee it being fixed much before Christmas. Anyway, my suggestion is that you could print a list of people's names and telephone numbers and city that have the capability to communicate by modem. Someone out there, with a System Rack with Discs, might be encouraged to set up a Tangerine Bill Board net!

P.S. Got any disc drives going cheap?

4. Oh yes. Do I qualify for the award(s) for sending in an article as other writers are being promised in Issue 2, namely a discount voucher/cash.

Keep up the good work.

Best Wishes

FLEX - a disc operating system for System Nine

Dr P.N. Gaunt

In a previous article I promised to describe the disc operating system (DOS) called FLEX (TM) which we have configured to run on the 6809 based system Nine. This powerful DOS allows many commercially available programs to be run without any modification on System Nine, indeed FLEX is one of the few machine-independent operating systems available (one of the other being OS-9 which is also for 6809 based systems). Before I describe FLEX and its capabilities in detail I should perhaps define what a DOS does for those of you unfamiliar with disc-based computers.

A DOS is a complex program, or series of program modules, designed to manage the storage and retrieval of programs and data on discs. Information is stored on the disc in blocks of 256 bytes called sectors, several of which may be required to hold one program. The DOS relieves the user of the burden of allocating sectors to a file by allowing reference to programs and data by a file name. Details of each file, its name, size, position on the disc and date of creation are also stored in a disc directory. The DOS must also provide the essential utilities which enable the user to save and load such files from disc, to examine the directory and manipulate files on the disc. These functions are called through a 'command processor', a part of the DOS that accepts commands typed at the keyboard, executes them and issues any messages generated by the utility or DOS itself to the console. FLEX includes all these features and many more, producing a very powerful operating environment in which to develop and run programs. The following is a summary of some features of the FLEX operating system.

(TM) FLEX is a trademark of Technical Systems Consultants Inc.

Hardware requirements

6809 Single Board Controller with 16K bytes RAM, TVBUG 4K ROM monitor and MAP4 mapping PROM. Both 6522 VIAs and the 6551 ACIA should be installed.

Tanram 1 (1MHz version) is recommended to provide 47K bytes of user memory.

Microtan Disc Controller board (no on-board memory required).

Either - 80 column terminal connected to serial port

Or VDU card (Mousepacket Design at present)
Microtan ASCII keyboard
Monochrome monitor

System Motherboard, MPSIIA and system rack.

One to four 5" mini-floppy disc drives - two are recommended for a full FLEX system.

A printer with Centronics-like interface may be connected to one of the ports on the Single Board Controller.

Memory requirements:

FLEX occupies 8K of RAM on the single board controller from address SC000 to SDFFF, into which it is loaded from disc by the bootstrap loader command in TVBUG (the 6809 monitor). The same 8K block of memory is also used by FLEX for data buffers, a transient command area and for system variables, so the DOS cannot be transferred into ROM. User programs and utilities too large to fit in the transient command area may be run anywhere in memory from S0000 to SBFFF. With Tanram 1 fully populated the top of user memory is in fact at SBBFF, giving 47K of available memory. The monitor ROM occupies 4K from SF000 to SFFFF and I/O space is between SEC00 and SEFFF. The 'gap' in memory at SE000 to SEBFF will be occupied by the VDU card if used, but is otherwise reserved for future use. The memory map is as follows:

0000	User RAM
BBFF	Top of Tanram 1
C000	FLEX DOS
E000	VDU screen memory
EC00	I/O space
F000	TVBUG 4K monitor

FLEX command processor:

When FLEX is booted from disc (by using the TVBUG 5" disc boot command) the user leaves TVBUG command mode with its prompt and enters the FLEX command processor where the prompt is +++ . Whenever this prompt appears FLEX is ready for a command. A command will consist of a command name followed by whatever parameters that command requires. There are only two commands resident in FLEX - these are MON (return control to TVBUG) and GET (load a binary file from disc). All other commands are stored on the FLEX system disc which will always be in one of the drives. Some of these are standard 'utilities' which come as a part of FLEX itself, while the rest are the many utilities and programs that have been written for FLEX. I will mention some of these later but first I will outline those commands that are a part of the FLEX package:

APPEND	Concatenate two or more files.
ASMB	Marco assembler
ASN	Specify which drive is to be system or work
BUILD	Build a text file line at a time
CAT	List the disc directory
COPY	Copy one or more disc files
DATE	Display and/or set FLEX date (set on startup)
DELETE	Delete a file
EDIT	Full function line editor
EXEC	Execute a text file containing commands
I	Input characters from disc rather than keyboard
JUMP	Jump to a location in memory
LINK	Link FLEX to loader (used in making system disc)
LIST	List a text file to screen
NEWDISK	Format a blank disc
O	Route output to disc rather than screen
P	Route output to printer rather than screen
PRINT	Send text file to printer spooler
PROT	Protect file on disc
QCHECK	Display/modify print queue of spooler
RENAME	Rename file
SAVE	Save binary file from memory
STARTUP	Startup command file called on booting FLEX
TTYSET	Examine/change system parameters
VERIFY	Enable/disable auto-verify after disc writes
VERSION	Display utility program's version number
XOUT	Delete all output files after print spooling

As you can see, there are many commands within FLEX itself

including a powerful macro assembler and an excellent text editor. I do not have space to explain any of the commands here but all are well documented in the FLEX manuals.

Other programs available for FLEX:

One of the delights of a machine-independent DOS such as FLEX is the availability of good general purpose software written for the DOS often running on a completely different 6809 system. This is the case for nearly all FLEX software, since System Nine has only been in existence for a few months! The company that supplies FLEX - TSC in North Carolina USA - also supports the system with several excellent packages:

FLEX Utilities - yet more very useful utilities

DEBUG A complete assembler language program debugging tool capable of full software simulation of 6809.

Diagnostics - a series of utilities for diagnosis of disc and memory failures. Can be useful even on System Nine!

XBASIC A 20K disc BASIC with high precision maths and good disc file handling.

PASCAL Native (compiled) PASCAL - the standard version. Very fast!

Text processor - for creating beautiful formatted documents. Allows macro commands for greatest versatility.

68000 macro cross assembler - for our next generation controller card (see next month's article!)

Many other companies supply software, a selection of the best follows:

Stylograph word processor - used to write this article.

Screditor III - another very good word processor.

RMS record management system - a data base manager.

Dynacalc - a Visicalc style speadsheet calculator.

Super Sleuth Disassembler - disassemble binary code into source code from 6809, 6800, 6502, 8080, Z80.

Cross assemblers for 6502, 6800, 6801, 6805, 8080, 8085, Z80

PL/9 a very useful compiled language for control. Based on PL/M and PASCAL with a touch of C.

C The ultimate in structured 'clean' languages. (Forget all the others!).

FORTH an enhanced FORTH with many features.

I think that should be enough to whet your appetite for FLEX. I should say that there are many more packages available, most of which are of the highest quality.

How do you install FLEX on a System Nine? Simply order a copy of FLEX configured for Microtan from Microtanic and you will receive two copies of the FLEX operating system ready to run on your System Nine. The selection of software for FLEX summarized above may be obtained from some or all of the software suppliers listed below.

I hope that you will take the opportunity to invest in this excellent operating system and have a great deal of enjoyment from its use on the System Nine. The two are very well matched.

Using Azimov Text Processor

USING THE AZIMOV TEXT PROCESSOR WITH TANDOS 65

Programs will shortly be available to adapt Azimov to the Microtan's floppy disc system. Meanwhile, texts can be stored on floppt discs and retrieved using the following method.

1. Saving text on disc.

Azimov stores text in the Microtan's memory from address S0D00 on up. The address of the end-of-text marker is kept in zero page locations B2 and B3. Get back to TANBUG with a reset and use the M command to determine this address (remember that the LOW Address byte is in B2, the HIGH byte in B3). Then use DSAVE from start address D00 to this address. If you save with a transfer address of C003 this will make retrieving the text file easier. E.9.

If, using the M command, we get

MB2,38,
M00B3,10,

Then to save the text use

DSAVE ANAME. TXT T0003
START: D00
END: 1038

2. Retrieving text files.

This is quite easy since it involves only Azimov functions. First enter Azimov with GC000, Press 'W' and answer 'Y' to destroy old text, this clears any previous text from the memory, which might otherwise cause problems. Get out of Azimov with a reset and load the text file by typing its name, and then warm-start Azimov with GC003. (If you have saved the text with a transfer address of C003 this will happen automatically). Press 'W' and when the message 'DESTROY old text? Y " appears press the RETURN key (or, in fact, any key except Y). This will set a flag to indicate that there is text in the MICROTAN'S memory. Now press 'O' (i.e. CONTINUE). This will locate the end-of-text marker and put its address in locations B2 and B3 of memory. Azimov now has all the information it needs to process the text. You can carry on directly adding new text, or leave the CONTINUE command, which has performed its essential function of finding the end of the text, by pressing BREAK or CONTROL-Q to get back to the Mian Command Level. If you have a later version of Azimov and you are going to print the text on a printer other than the EPSOM MX80, don't forget to set the layout to non-MX80 first.

EASY CHUNKIES

Forget all that tedious plotting with this easy to use chunky graphics routine.

Doug Deedman

When I first started to work with the Microtan, back in 1980, I soon became frestrated with the awkward business of composing screens of graphics. It seemed to me that what was wanted was some means of drawing directly on the screen the desired display and then saving it for later use. Having written a program that would enable me to do this, I used it for a while and then lost interest, since I was making little use of the graphics by then anyway. However, I was prompted by Alan Murphy's 'Chunky Graphics Screen Printer' article in issue 4 to dig out the source code, painfully written using the translator in XBUG, and look at it afresh. I have rewritten part of the program so as to include a 'retrieve and print' option which, as you can see from the listing given, merely includes a jump to the start address of Alan's routine.

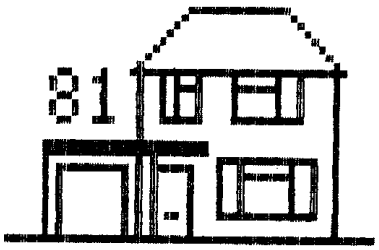
On entry to the program, you are asked to select from the presented menu the mode of operation:

1. DRAW & SAVE You will be asked for the base address of the block of memory where you wish to save the screen contents. On entering this, a flashing pixel will appear in the middle of an empty screen. This pixel can be steered around the screen using the numeric pad on the Tangerine keyboard. Pressing key (5) will leave a pixel at any selected position and pressing (0) will erase one. The other numbers are used to shift the plotting pixel in the manner of a joystick: (7) moves one position up and to the left, for example, (2) moves the plotting pixel down one place. A continuous trace will be left if key (L) is pressed (this key acts as a 'toggle', turning off the facility on the next press) and, making use of the repeat key near the numeric pad, fast line drawing can be accomplished. It should be noted that this 'Line' mode must be turned off before key (0) can be used to erase pixels. When the display is complete, press key (F). This will save the display to the desired location and return you to the menu. At this point you may decide to exit to Tanbug, in order to dump the screen data to backing store, or select the print screen option.

2. RETRIEVE & PRINT You will be asked for the base address of the screen data (this could have been loaded from backing store). On entering this address, the screen is retrieved, printed and, once done, a return is made to the menu.

3. EXIT to Tanbug The much rewritten code of GRAPIX follows. It includes a subroutine call to Alan's print routine at its original address, although this could be assembled so that it is tagged on the end of GRAPIX.

For those that might be interested, I have written this short article using Mousepacket's word processor, WORD PERFECT V.2. When I have had more time to work with WORD PERFECT I will write a more comprehensive review of it but, for the moment, let it suffice for me to say that it is a joy to use.




```
0001 ;****** GRAPIX ******      0400
0002 ;                         0400
0003 ;Microtan Chunky Graphics 0400
0004 ;Sketch and Save Program. 0400
0005 ;D. P. Deedman            0400
0006 ;                         0400
0007 ICHAR  EPZ $01            0400
0008 HXPKL  EPZ $13            0400
0009 HXPKH  EPZ $14            0400
0010 PIXMOD EPZ $40            0400
0011 XCOORD EPZ $41            0400
0012 YCOORD EPZ $42            0400
0013 VDULO  EPZ $43            0400
0014 VDUHI  EPZ $44            0400
0015 STORE  EPZ $45            0400
0016 FINISH EPZ $46            0400
0017 CHRSCL EPZ $47            0400
0018 CHRSCH EPZ $48            0400
0019 CHRDSL EPZ $49            0400
0020 CHRDSH EPZ $4A            0400
0021 DRAW   EPZ $4B            0400
0022 PRINT  EQU $1F08          0400
0023 GON    EQU $BFF0          0400
0024 GROFF  EQU $BFF3          0400
0025 OUTRET EQU $F80C          0400
0026 OUTALL EQU $F80E          0400
0027 JPLKB  EQU $F81D          0400
0028 JHXPK  EQU $F817          0400
0029 JHXPN  EQU $F81A          0400
0030 TANBUG EQU $FC00          0400
0031 ;                         0400
0032 ;Clear screen             0400
0033 MENU   JSR CLRVDU         0400 20 F3 04
0034 ;Display menu             0403
0035        LDX #$00           0403 A2 00
0036 GETCH1 LDA MES1,X         0405 BD B9 06
0037        BEQ GETOPT         0408 F0 06
0038        JSR OUTALL         040A 20 0E F8
0039        INX                040D E8
0040        BNE GETCH1         040E D0 F5
0041 GETOPT CLI                0410 58
0042        JSR JPLKB          0411 20 1D F8
0043        LDA ICHAR          0414 A5 01
0044        CMP #$31           0416 C9 31
0045        BEQ DRAWIT         0418 F0 0E
0046        CMP #$32           041A C9 32
0047        BEQ RETRVE         041C F0 07
0048        CMP #$33           041E C9 33
0049        BNE GETOPT         0420 D0 EE
0050        JMP TANBUG         0422 4C 00 FC
0051 RETRVE JMP DSPLAY         0425 4C C1 04
0052 ;Sketch & save routine.   0428
0053 ;Ask for base address of  0428
0054 ;screen save area         0428
0055 DRAWIT JSR CLRVDU         0428 20 F3 04
0056        LDX #$00           042B A2 00
0057 GETCH2 LDA MES2,X         042D BD 0F 07
0058        BEQ GETADD         0430 F0 06
0059        JSR OUTALL         0432 20 0E F8
0060        INX                0435 E8
0061        BNE GETCH2         0436 D0 F5
0062 ;Now go fetch & pack addr.0438
0063 GETADD JSR PAKADD         0438 20 09 05
0064 ;Request "G" for start    043B
0065        LDX #$00           043B A2 00
0066 GETCH3 LDA MES3,X         043D BD 4E 07
0067        BEQ GETGO          0440 F0 06
0068        JSR OUTALL         0442 20 0E F8
0069        INX                0445 E8
0070        BNE GETCH3         0446 D0 F5
0071 ;and now wait for it      0448
0072 GETGO  JSR JPLKB          0448 20 1D F8
0073        LDA ICHAR          044B A5 01
0074        CMP #$47           044D C9 47
0075        BNE GETGO          044F D0 F7
0076 ;Display for a while      0451
0077        JSR OUTALL         0451 20 0E F8
0078        JSR DELAY          0454 20 1E 05
0079        JSR DELAY          0457 20 1E 05
0080        JSR DELAY          045A 20 1E 05
0081        JSR DELAY          045D 20 1E 05
0082 ;Turn on graphics and     0460
0083 ;clear screen with nulls  0460
0084        JSR GRACLR         0460 20 F9 04
```

```
0085 ;0 in IX.Use it to clear  0463
0086 ;locs $45,$46 & $4B       0463
0087         STX STORE         0463 86 45
0088         STX FINISH        0465 86 46
0089         STX DRAW          0467 86 4B
0090 ;Put pixel in centre      0469
0091 ;of screen at 31,31       0469
0092         LDA #$1F          0469 A9 1F
0093         STA XCOORD        046B 85 41
0094         STA YCOORD        046D 85 42
0095 ;and set up sio.int.link  046F
0096         LDA #$4C          046F A9 4C
0097         STA $10           0471 85 10
0098         LDA #INTSRV       0473 A9 8A
0099         STA $11           0475 85 11
0100         LDA #INTSRV)      0477 A9 05
0101         STA $12           0479 85 12
0102 ;Flashing pixel loop      047B
0103 PIXON   LDA #$01          047B A9 01
0104         STA PIXMOD        047D 85 40
0105         SEI               047F 78
0106         JSR PIXEL         0480 20 29 05
0107         CLI               0483 58
0108         JSR DELAY         0484 20 1E 05
0109         JSR DELAY         0487 20 1E 05
0110         DEC PIXMOD        048A C6 40
0111         SEI               048C 78
0112         JSR PIXEL         048D 20 29 05
0113         CLI               0490 58
0114         LDA FINISH        0491 A5 46
0115         BNE LNKOFF        0493 D0 06
0116         JSR DELAY         0495 20 1E 05
0117         JMP PIXON         0498 4C 7B 04
0118 ;FINISH set-leave loop    049B
0119 ;and copy screen contents 049B
0120 LNKOFF LDA #$40           049B A9 40
0121         STA $10           049D 85 10
0122         LDA STORE         049F A5 45
0123         BEQ COPY          04A1 F0 05
0124         STA PIXMOD        04A3 85 40
0125         JSR PIXEL         04A5 20 29 05
0126 ;start of copy routine    04A8
0127 COPY    LDA #$00          04A8 A9 00
0128         STA CHRSCL        04AA 85 47
0129         LDA #$02          04AC A9 02
0130         STA CHRSCH        04AE 85 48
0131         LDA HXPKL         04B0 A5 13
0132         STA CHRDSL        04B2 85 49
0133         LDA HXPKH         04B4 A5 14
0134         STA CHRDSH        04B6 85 4A
0135         JSR SCRMOV        04B8 20 77 05
0136 ;Screen contents safe     04BB
0137 ;Graphics off             04BB
0138         STA GROFF         04BB 8D F3 BF
0139 ;Return to menu           04BE
0140         JMP MENU          04BE 4C 00 04
0141 ;                         04C1
0142 ;Retrieve screen routine  04C1
0143 ;Ask for base addr.of     04C1
0144 ;stored screen data       04C1
0145 DSPLAY JSR CLRVDU         04C1 20 F3 04
0146         LDX #$00          04C4 A2 00
0147 GETCH4 LDA MES4,X         04C6 BD 6B 07
0148         BEQ GETSRC        04C9 F0 06
0149         JSR OUTALL        04CB 20 0E F8
0150         INX               04CE E8
0151         BNE GETCH4        04CF D0 F5
0152 GETSRC JSR PAKADD         04D1 20 09 05
0153 ;O.K.,now copy back       04D4
0154         LDA HXPKL         04D4 A5 13
0155         STA CHRSCL        04D6 85 47
0156         LDA HXPKH         04D8 A5 14
0157         STA CHRSCH        04DA 85 48
0158         LDA #$00          04DC A9 00
0159         STA CHRDSL        04DE 85 49
0160         LDA #$02          04E0 A9 02
0161         STA CHRDSH        04E2 85 4A
0162         JSR GRACLR        04E4 20 F9 04
0163         JSR SCRMOV        04E7 20 77 05
0164 ;Now go print it          04EA
0165         JSR PRINT         04EA 20 08 1F
0166         STA GROFF         04ED 8D F3 BF
0167 ;and loop back to menu    04F0
0168         JMP MENU          04F0 4C 00 04
0169 ;                         04F3
0170 ;Subr.clear screen alpha  04F3
0171 CLRVDU LDA #$0C           04F3 A9 0C
0172         JSR OUTALL        04F5 20 0E F8
0173         RTS               04F8 60
0174 ;                         04F9
0175 ;Subr.clear screen graph. 04F9
0176 GRACLR LDA GON            04F9 AD F0 BF
0177         LDX #$00          04FC A2 00
0178         TXA               04FE 8A
0179 NULOUT STA $200,X         04FF 9D 00 02
0180         STA $300,X        0502 9D 00 03
0181         DEX               0505 CA
0182         BNE NULOUT        0506 D0 F7
0183         RTS               0508 60
0184 ;                         0509
0185 ;Pack input address to    0509
0186 ;HXPKL & HXPKH subroutine 0509
0187 PAKADD JSR JPLKB          0509 20 1D F8
0188         LDA ICHAR         050C A5 01
0189         CMP #$0D          050E C9 0D
0190         BEQ GOPACK        0510 F0 06
0191         JSR OUTALL        0512 20 0E F8
0192         JMP PAKADD        0515 4C 09 05
0193 GOPACK LDY #$FF           0518 A0 FF
0194         JSR JHXPK         051A 20 17 F8
0195         RTS               051D 60
0196 ;                         051E
0197 ;Delay subroutine         051E
0198 DELAY   LDX #$40          051E A2 40
0199         LDY #$00          0520 A0 00
0200 WAIT    DEY               0522 88
0201         BNE WAIT          0523 D0 FD
0202         DEX               0525 CA
0203         BNE WAIT          0526 D0 FA
0204         RTS               0528 60
0205 ;                         0529
0206 ;Pixel address and pattern0529
0207 ;subroutine as in manual  0529
0208 PIXEL   LDA #$3F          0529 A9 3F
0209         CMP XCOORD        052B C5 41
0210         BMI EXIT          052D 30 33
0211         SEC               052F 38
0212         SBC YCOORD        0530 E5 42
0213         PHA               0532 48
0214         AND #$03          0533 29 03
0215         TAX               0535 AA
0216         PLA               0536 68
0217         AND #$3C          0537 29 3C
0218         ASL @             0539 0A
0219         ASL @             053A 0A
0220         ASL @             053B 0A
0221         STA VDULO         053C 85 43
0222         LDA #$02          053E A9 02
0223         STA VDUHI         0540 85 44
0224         BCC NOINC         0542 90 02
0225         INC VDUHI         0544 E6 44
0226 NOINC   LDA XCOORD        0546 A5 41
0227         LSR @             0548 4A
0228         TAY               0549 A8
0229         LDA #$01          054A A9 01
0230         BCC NOSF2         054C 90 01
0231         ASL @             054E 0A
0232 NOSF2   DEX               054F CA
0233         BMI STPATT        0550 30 04
0234         ASL @             0552 0A
0235         ASL @             0553 0A
0236         BNE NOSF2         0554 D0 F9
0237 ;PIXMOD checked to see    0556
0238 ;if test or delete of     0556
0239 ;pixel is wanted          0556
0240 STPATT TAX                0556 AA
0241         LDA PIXMOD        0557 A5 40
0242         BMI TEST          0559 30 08
0243         BEQ DELETE        055B F0 10
0244         TXA               055D 8A
0245 ;Combine with what is     055E
0246 ;there and write back     055E
0247         ORA (VDULO),Y     055E 11 43
0248         STA (VDULO),Y     0560 91 43
0249 EXIT    RTS               0562 60
0250 TEST    TXA               0563 8A
0251         AND (VDULO),Y     0564 31 43
0252 ;If matching bit present  0566
0253 ;set STORE to 1           0566
0254         BEQ RET1          0566 F0 04
0255         LDA #$01          0568 A9 01
0256         STA STORE         056A 85 45
0257 RET1    RTS               056C 60
0258 DELETE TXA                056D 8A
0259         AND (VDULO),Y     056E 31 43
0260         BEQ RET2          0570 F0 04
0261         EOR (VDULO),Y     0572 51 43
0262 ;Result zero where match  0574
0263 ;bit found-amend display  0574
0264         STA (VDULO),Y     0574 91 43
0265 RET2    RTS               0576 60
0266 ;                         0577
0267 ;Subr.to copy 2 pages of  0577
0268 ;data within RAM. Used to 0577
0269 ;move screen out or back  0577
0270 SCRMOV LDX #$02           0577 A2 02
```

```
0271          LDY #$00              0579 A0 00
0272 NEXCHR   LDA (CHRSCL),Y        057B B1 47
0273          STA (CHRDSL),Y        057D 91 49
0274          DEY                   057F 88
0275          BNE NEXCHR            0580 D0 F9
0276          INC CHRSCH            0582 E6 48
0277          INC CHRDSH            0584 E6 4A
0278          DEX                   0586 CA
0279          BNE NEXCHR            0587 D0 F2
0280          RTS                   0589 60
0281 ;                              058A
0282 ;Start of int.service          058A
0283 ;routine.Save regs.            058A
0284 INTSRV   PHA                   058A 48
0285          TXA                   058B 8A
0286          PHA                   058C 48
0287          TYA                   058D 98
0288          PHA                   058E 48
0289 ;Pick up input and examine     058F
0290          LDA ICHAR             058F A5 01
0291          CMP #$38              0591 C9 38
0292          BEQ UP                0593 F0 47
0293          CMP #$32              0595 C9 32
0294          BEQ DOWN              0597 F0 54
0295          CMP #$34              0599 C9 34
0296          BEQ LEFT              059B F0 5F
0297          CMP #$36              059D C9 36
0298          BEQ RIGHT             059F F0 6A
0299          CMP #$37              05A1 C9 37
0300          BEQ JUL               05A3 F0 1F
0301          CMP #$39              05A5 C9 39
0302          BEQ JUR               05A7 F0 1E
0303          CMP #$31              05A9 C9 31
0304          BEQ JDL               05AB F0 1D
0305          CMP #$33              05AD C9 33
0306          BEQ JDR               05AF F0 1C
0307          CMP #$35              05B1 C9 35
0308          BEQ JKEEP             05B3 F0 1B
0309          CMP #$30              05B5 C9 30
0310          BEQ JERASE            05B7 F0 1A
0311          CMP #$4C              05B9 C9 4C
0312          BEQ JLINE             05BB F0 19
0313          CMP #$46              05BD C9 46
0314          BEQ JSTOP             05BF F0 18
0315          JMP RESREG            05C1 4C 91 06
0316 JUL      JMP UL                05C4 4C 1C 06
0317 JUR      JMP UR                05C7 4C 33 06
0318 JDL      JMP DL                05CA 4C 4A 06
0319 JDR      JMP DR                05CD 4C 5F 06
0320 JKEEP    JMP KEEP              05D0 4C 76 06
0321 JERASE   JMP ERASE             05D3 4C 7D 06
0322 JLINE    JMP LINE              05D6 4C 84 06
0323 JSTOP    JMP STOP              05D9 4C 8D 06
0324 UP       LDA YCOORD            05DC A5 42
0325          CMP #$3F              05DE C9 3F
0326          BEQ JRES1             05E0 F0 08
0327          JSR MOVPIX            05E2 20 97 06
0328          INC YCOORD            05E5 E6 42
0329          JSR PIXCHK            05E7 20 B1 06
0330 JRES1    JMP RESREG            05EA 4C 91 06
0331 DOWN     LDA YCOORD            05ED A5 42
0332          BEQ JRES2             05EF F0 08
0333          JSR MOVPIX            05F1 20 97 06
0334          DEC YCOORD            05F4 C6 42
0335          JSR PIXCHK            05F6 20 B1 06
0336 JRES2    JMP RESREG            05F9 4C 91 06
0337 LEFT     LDA XCOORD            05FC A5 41
0338          BEQ JRES3             05FE F0 08
0339          JSR MOVPIX            0600 20 97 06
0340          DEC XCOORD            0603 C6 41
0341          JSR PIXCHK            0605 20 B1 06
0342 JRES3    JMP RESREG            0608 4C 91 06
0343 RIGHT    LDA XCOORD            060B A5 41
0344          CMP #$3F              060D C9 3F
0345          BEQ JRES4             060F F0 08
0346          JSR MOVPIX            0611 20 97 06
0347          INC XCOORD            0614 E6 41
0348          JSR PIXCHK            0616 20 B1 06
0349 JRES4    JMP RESREG            0619 4C 91 06
0350 UL       LDA XCOORD            061C A5 41
0351          BEQ JRES5             061E F0 10
0352          LDA YCOORD            0620 A5 42
0353          CMP #$3F              0622 C9 3F
0354          BEQ JRES5             0624 F0 0A
0355          JSR MOVPIX            0626 20 97 06
0356          DEC XCOORD            0629 C6 41
0357          INC YCOORD            062B E6 42
0358          JSR PIXCHK            062D 20 B1 06
0359 JRES5    JMP RESREG            0630 4C 91 06
0360 UR       LDA #$3F              0633 A9 3F
0361          CMP XCOORD            0635 C5 41
0362          BEQ JRES6             0637 F0 0E
0363          CMP YCOORD            0639 C5 42
0364          BEQ JRES6             063B F0 0A
0365          JSR MOVPIX            063D 20 97 06
0366          INC XCOORD            0640 E6 41
0367          INC YCOORD            0642 E6 42
0368          JSR PIXCHK            0644 20 B1 06
0369 JRES6    JMP RESREG            0647 4C 91 06
0370 DL       LDA XCOORD            064A A5 41
0371          BEQ JRES7             064C F0 0E
0372          LDA YCOORD            064E A5 42
0373          BEQ JRES7             0650 F0 0A
0374          JSR MOVPIX            0652 20 97 06
0375          DEC XCOORD            0655 C6 41
0376          DEC YCOORD            0657 C6 42
0377          JSR PIXCHK            0659 20 B1 06
0378 JRES7    JMP RESREG            065C 4C 91 06
0379 DR       LDA YCOORD            065F A5 42
0380          BEQ JRES8             0661 F0 10
0381          LDA XCOORD            0663 A5 41
0382          CMP #$3F              0665 C9 3F
0383          BEQ JRES8             0667 F0 0A
0384          JSR MOVPIX            0669 20 97 06
0385          INC XCOORD            066C E6 41
0386          DEC YCOORD            066E C6 42
0387          JSR PIXCHK            0670 20 B1 06
0388 JRES8    JMP RESREG            0673 4C 91 06
0389 KEEP     LDA #$01              0676 A9 01
0390          STA STORE             0678 85 45
0391          JMP RESREG            067A 4C 91 06
0392 ERASE    LDA #$00              067D A9 00
0393          STA STORE             067F 85 45
0394          JMP RESREG            0681 4C 91 06
0395 LINE     LDA DRAW              0684 A5 4B
0396          EOR #$01              0686 49 01
0397          STA DRAW              0688 85 4B
0398          JMP RESREG            068A 4C 91 06
0399 STOP     LDA #$01              068D A9 01
0400          STA FINISH            068F 85 46
0401 ;Restore regs.and return       0691
0402 RESREG   PLA                   0691 68
0403          TAY                   0692 A8
0404          PLA                   0693 68
0405          TAX                   0694 AA
0406          PLA                   0695 68
0407          RTI                   0696 40
0408 ;MOVPIX checks to see if       0697
0409 ;pixel should be on.If so,     0697
0410 ;the pixel is set,even if      0697
0411 ;it is already on.It is        0697
0412 ;cleared otherwise             0697
0413 MOVPIX   LDA DRAW              0697 A5 4B
0414          BNE STAPIX            0699 D0 04
0415          LDA STORE             069B A5 45
0416          BEQ PXOFF             069D F0 08
0417 STAPIX   STA PIXMOD            069F 85 40
0418          JSR PIXEL             06A1 20 29 05
0419          JMP CLSTR             06A4 4C AC 06
0420 PXOFF    STA PIXMOD            06A7 85 40
0421          JSR PIXEL             06A9 20 29 05
0422 CLSTR    LDA #$00              06AC A9 00
0423          STA STORE             06AE 85 45
0424          RTS                   06B0 60
0425 ;PIXCHK examines new           06B1
0426 ;location to see if pixel      06B1
0427 ;set. If so,STORE set by       06B1
0428 ;the subroutine PIXEL          06B1
0429 PIXCHK   LDA #$FF              06B1 A9 FF
0430          STA PIXMOD            06B3 85 40
0431          JSR PIXEL             06B5 20 29 05
0432          RTS                   06B8 60
0433 ;start of text store           06B9
0434 MES1     BYT $0D,$0D,$0D,$0D   06B9 0D 0D 0D
0435          BYT 'S,'e,'l          06BC 53 65 6C
0436          BYT 'e,'c,'t          06BF 65 63 74
0437          BYT ' ,'o,'p          06C2 20 6F 70
0438          BYT 't,'i,'o          06C5 74 69 6F
0439          BYT 'n,' ,'b          06C8 6E 20 62
0440          BYT 'y,' ,'n          06CB 79 20 6E
0441          BYT 'u,'m,'b          06CE 75 6D 62
0442          BYT 'e,'r,'           06D1 65 72 20
0443          BYT '-                06D4 2D
0444          BYT $0D,$0D,$0D       06D5 0D 0D 0D
0445          BYT $20,$20,$20       06D8 20 20 20
0446          BYT '1,' ,'           06DB 31 20 20
0447          BYT 'D,'R,'A          06DE 44 52 41
0448          BYT 'W,' ,'&          06E1 57 20 26
0449          BYT ' ,'S,'A          06E4 20 53 41
0450          BYT 'V,'E             06E7 56 45
0451          BYT $0D,$0D           06E9 0D 0D
0452          BYT $20,$20           06EB 20 20
0453          BYT $20,'2,'          06ED 20 32 20
0454          BYT ' ,'R,'E          06F0 20 52 45
0455          BYT 'T,'R,'I          06F3 54 52 49
0456          BYT 'E,'V,'E          06F6 45 56 45
```

```
0457          BYT ' ,'&,'          06F9 20 26 20
0458          BYT 'P,'R,'I         06FC 50 52 49
0459          BYT 'N,'T,$0D,$0D    06FF 4E 54 0D 0D
0460          BYT $20,$20,$20      0703 20 20 20
0461          BYT '3,' ,'          0706 33 20 20
0462          BYT 'E,'X,'I         0709 45 58 49
0463          BYT 'T,$0D,$00       070C 54 0D 00
0464 MES2     BYT $0D,$0D,$0D      070F 0D 0D 0D
0465          BYT 'F,'r,'o         0712 46 72 6F
0466          BYT 'm,' ,'w         0715 6D 20 77
0467          BYT 'h,'i,'c         0718 68 69 63
0468          BYT 'h,' ,'b         071B 68 20 62
0469          BYT 'a,'s,'e         071E 61 73 65
0470          BYT ' ,'a,'d         0721 20 61 64
0471          BYT 'd,'r,'e         0724 64 72 65
0472          BYT 's,'s,'          0727 73 73 20
0473          BYT 'd,'o,'          072A 64 6F 20
0474          BYT 'y,'o,'u         072D 79 6F 75
0475          BYT ' ,' ,'w         0730 20 20 77
0476          BYT 'i,'s,'h         0733 69 73 68
0477          BYT ' ,'t,'o         0736 20 74 6F
0478          BYT ' ,'s,'t         0739 20 73 74
0479          BYT 'o,'r,'e         073C 6F 72 65
0480          BYT ' ,'t,'h         073F 20 74 68
0481          BYT 'e,' ,'d         0742 65 20 64
0482          BYT 'i,'s,'p         0745 69 73 70
0483          BYT 'l,'a,'y         0748 6C 61 79
0484          BYT '?,$0D,$00       074B 3F 0D 00
0485 MES3     BYT $0D,$0D,$0D      074E 0D 0D 0D
0486          BYT 'O,'.,'K         0751 4F 2E 4B

0487          BYT ',,',,'t         0754 2E 2C 74
0488          BYT 'y,'p,'e         0757 79 70 65
0489          BYT ' ,'G,'          075A 20 47 20
0490          BYT 't,'o,'          075D 74 6F 20
0491          BYT 's,'t,'a         0760 73 74 61
0492          BYT 'r,'t,'          0763 72 74 20
0493          BYT '-,$0D,$0D       0766 2D 0D 0D
0494          BYT $0D,$00          0769 0D 00
0495 MES4     BYT $0D,$0D,$0D      076B 0D 0D 0D
0496          BYT 'E,'n,'t         076E 45 6E 74
0497          BYT 'e,'r,'          0771 65 72 20
0498          BYT 't,'h,'e         0774 74 68 65
0499          BYT ' ,'b,'a         0777 20 62 61
0500          BYT 's,'e,'          077A 73 65 20
0501          BYT 'a,'d,'d         077D 61 64 64
0502          BYT 'r,'e,'s         0780 72 65 73
0503          BYT 's,' ,'o         0783 73 20 6F
0504          BYT 'f,$0D           0786 66 0D
0505          BYT 't,'h,'e         0788 74 68 65
0506          BYT ' ,'s,'c         078B 20 73 63
0507          BYT 'r,'e,'e         078E 72 65 65
0508          BYT 'n,' ,'y         0791 6E 20 79
0509          BYT 'o,'u,'          0794 6F 75 20
0510          BYT 'w,'i,'s         0797 77 69 73
0511          BYT 'h,' ,'t         079A 68 20 74
0512          BYT 'o,' ,'r         079D 6F 20 72
0513          BYT 'e,'t,'r         07A0 65 74 72
0514          BYT 'i,'e,'v         07A3 69 65 76
0515          BYT 'e,$0D,$0D       07A6 65 0D 0D
0516          BYT $00              07A9 00
```

RS-232C FOR THE MICROTAN

By Andy Michael

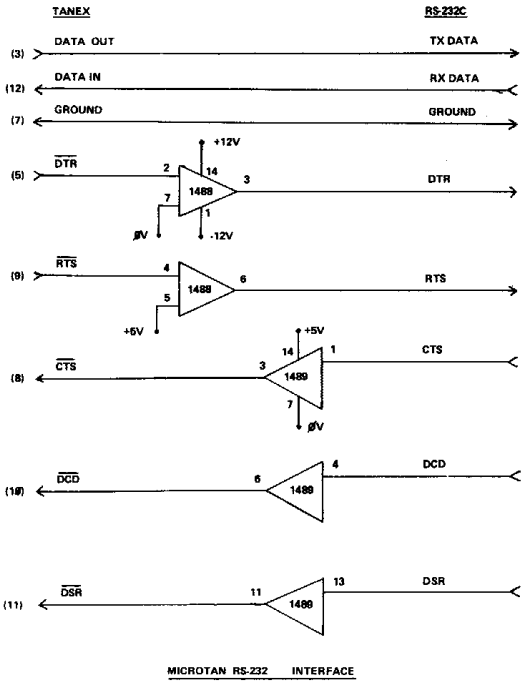
In my article in issue 2 of Microtan World, I described how to use the serial port on Tanex. Those users who wish to connect their Microtan to professional serial printers or modems will meet another problem in that the serial port does not support a proper RS-232C interface. This article should resolve the problem.

Firstly, some definitions. The E1A RS-232C standard states that data signals on the interface represent a logic 1 with a voltage between -3 and -25 volts. A logic 0 is represented by a voltage greater than +3 volts and less than +25 volts. However just to confuse things, the control signals work with the opposite logic, so that a signal is on if it is between +3 and +25V. Voltages between -3 and +3V are undefined. Although the RS-232C interface allows the use of longer cables than a parallel connection, cable lengths over 15 metres should only be used with care.

The 6551 on Tanex offers data rates from 50 to 19200. In practice, the processor clock speed sets an upper limit of about 9600 baud. The data in and data out lines already feature RS-232C levels, but the control lines are at TTL levels straight from the 6551. For a lot of applications, only the data lines need be used provided that care is taken to ground the CTS, DCD and DSR lines as mentioned in the last article. For a full interface, the control lines need buffering to RS-232C levels.

Fortunately, suitable low cost buffers are available in the form of the MC1488 and MC1489 chips. These are obtainable from most suppliers for about 60p each, and because they are quad devices, only one of each is needed. A suitable circuit is shown below, which may be built on a piece of veroboard or similar. Note that the 1488 and 1489 also take care of the required control signal inversion.

No details are given for the interface connector; a 25 pin 'D' type plug is usual for professional equipment, but it is left to the user to attach a suitable connector for the equipment in use.



MICROTAN RS-232 INTERFACE

Finally, do not forget that if you need more than one RS-232C port, a serial I/O board is available from Microtanic which offers another eight serial ports similar to that on Tanex.

SOUND BOARD ROUTINES

As promised in the last issue here are some ideas to assist you in using the sound board.

This first program, in Basic, has instructions on screen so it needs no explanation.

The second program gives two bleeps and can be inserted into a program as a subroutine. If lines 1005 and 1067 are omitted it will produce a single bleep. Similarly by changing the registers the sound can be altered. (See instructions with the board).

For those of you who have the Translator/Compiler program we are hoping to be able to offer one or more libraries of pre-programmed music. This will be through MCS ltd., more information in the next issue.

```
100 PRINTCHR$(12)
110 PRINT"            SOUNDS      "
120 FORF=1TO500:NEXTF
130 PRINTCHR$(12)
140 PRINT"L = Laser Gun........."
150 PRINT"B = Bomb............."
160 PRINT"E = Space Ship Engine."
170 A=48128:B=48129
180 GETS$
190 IFS$="L"THEN230
200 IFS$="B"THEN310
210 IFS$="E"THEN 450
220 IFS$<>"L"ORS$<>"B"ORS$<>"E"THEN180
230 POKEA,7:POKEB,62
240 POKEA,8:POKEB,13
250 FORZ=48TO100
260 POKEA,0:POKEB,Z
270 POKEA,1:POKEB,0
280 NEXTZ
290 POKEA,8:POKEB,0
300 GOTO130
310 POKEA,7:POKEB,62
320 POKEA,8:POKEB,13
330 FORX=1TO200
340 POKEA,0:POKEB,X
350 POKEA,1:POKEB,0
360 NEXTX
370 POKEA,6:POKEB,0
380 POKEA,7:POKEB,7
390 POKEA,8:POKEB,16
400 POKEA,9:POKEB,16
410 POKEA,10:POKEB,16
420 POKEA,12:POKEB,16
430 POKEA,13:POKEB,0
440 GOTO130
450 PRINTCHR$(12)
460 PRINT"< = Accelerate..."
470 PRINT"> = Deaccelerate."
480 PRINT"S = Stop Engines."
490 V=200:VV=2
500 Q$=CHR$(PEEK(1))
510 IFQ$=","THENV=V-2:GOTO560
520 IFQ$="."THENV=V+2:GOTO560
530 IFQ$=" "THEN 560
540 IFQ$="S"THEN670
550 GOTO490
560 IFV>244THENVV=VV+1:V=0
570 IFV<0THEN VV=VV-1:V=244
580 IFVV=2ANDV=200THEN640
590 POKEA,1:POKEB,VV:POKEA,3:
    POKEB,VV:POKEA,5:POKEB,VV
600 POKEA,8:POKEB,13:POKEA,9:
    POKEB,13:POKEA,10:POKEB,13
610 POKEA,0:POKEB,V:POKEA,2:
    POKEB,V+5:POKEA,4:POKEB,V+10
620 POKEA,7:POKEB,56
630 GOTO500
640 FOR Y=8TO10:POKEA,Y:POKEB,16:NEXTY
650 POKEA,7:POKEB,7:POKEA,6:POKEB,0:
    POKEA,12:POKEB,50:POKEA,13:POKEB,0
660 GOTO130
670 POKEA,8:POKEB,0:POKEA,9:POKEB,0:
    POKEA,10:POKEB,0:GOTO130
OK
```

BLEEP-BLEEP.

```
1005 FORI=1TO2
1010 POKE48128,0
1015 POKE48129,20
1020 POKE48128,1
1025 POKE48129,0
1030 POKE48128,8
1035 POKE48129,10
1040 POKE48128,7
1045 POKE48129,254
1050 FOR Q=1TO60:NEXT
1060 POKE48128,7
1065 POKE48129,255
1067 NEXTI
1070 RETURN
OK
```

REPAIRING FAULTY DISCS

By Peter J.F. Clews
Malaga
Spain.

From time to time, as you try to read a file from Microtan's floppy discs, you may get error messages. Often enough it is sufficient to turn the system off and on again . . . many faults seem to be caused by corruption of data in the disc operating system (If this happens a lot you should look for the source of the interference. It may indicate a fault in your Microtan system, or it may be that you need a mains interference suppressor). However, if your disc system is basically relaible, but you persistently get '08 HARD ERROR' when reading one particular file, it probably means that that file has been recorded badly. As long as it is only a recording fault, deleting the file will clear the condition, since it is highly unlikely that another fault will occur the next time something is recorded on that sector of the disc. (Physical faults on the surface of the disc are another matter). But deleting the file means losing its contents, which will certainly be inconvenient and may be costly, so if it is possible to recover the file it is well worth theeffort. The following method, using the SECDMP utility included on the TANDOS distribution disc, has given very good results.

Basically, the idea is to identify the sector on the disc where the error occurs, and rewrite this sector onto the disc, preferably with the correct contents. Normally only one byte is bad, and this can often be identified and corrected. Indeed, it commonly happens that there is no bad byte . . . the fault can be in a check sum or a parity bit, or simply caused by a slight timing error, large enough to be detected by the disc controller circuit but not so large as to corrupt the data. In any case, once the file can be read, its contents can be examined and corrected if necessary. The complicated part of this is tracing through the disc to find the faulty sector.

The formats used by TANDOS are explained below in an appendix. The information given in the TANDOS users manual is far from complete in this area. The SECDMP program itself is reasonably well explained, although I will attempt to make this description complete enough to be used without reference to the manual.

First call SECDMP by typing its name. If the faulty disc does not include SECDMP don't worry . . . insert a disc which does have it, call it, and then change the discs.

Now read the first directory sector. This will be on track 0, sector 4, so type:

R0,4 CR

The disc will be set into motion and within a few seconds the bottom half of the screen will fill with the contents of this sector. The file names can be read directly off the screen, interspersed with graphics characters which correspond to the track and sector numbers, etc., where the files are stored.

Examine the filenames. If the faulty file is not included, then it must be on a subsequent directory sectory, and we must bring that sector onto the screen. The track and sector numbers of the next directory sector are given, in that order, in the first two bytes of this sector. To see what they are we use SECDMP's P command, Type:

P CR

and a number will appear on the screen immediately to the right of the P, at the same time as the cursor (a horizontal bar), moves to the second character position on the bottom half of the screen.

Use the P command again to print the second byte of the sector. These two bytes are the track and sector numbers respectively of the next directory sector. Use them with the R command separating them with a comma, to read the next sector of the directory. E.g . if after the first R command and the two P commands the top of the screen says:

R0,4
P06
P04

then use

R6,4

to read the next directory sector.

If the faulty file is still not to be seen, continue in this way, printing the first two bytes of each sector and then using them with the R command, until you come to the directory sector which contains the faulty file name.

Once the filename is located, the next step is to find where the file is recorded on the disc. The cursor should be moved to the last character of the filename's extension using the M command in conjunction with D for Down, U for Up, L for Left and R for Right. e.g.

MD CR moves the cursor down one line
MOD CR moves the cursor down two lines
MRRRRR CR moves the cursor right 5 characters.

When the cursor is on the third character of the filename extension (be careful if the filename contains spaces . . . remember that every filename is allotted 9 characters, 6 for the name and 3 for the extension), type the command:

MRRR CR

This will put the cursor on the SECTOR number where the start of the file is stored. Use P CR to read this number, then again to read the following number, which is the TRACK numbe. where the file is stored.

Use the R command with these numbers in the OPPOSITE order, i.e. track number first, to read the first sector of the faulty file. (Note that it is advisable to move the input cursor to the top of the screen before reading any sector of a faulty file, since the error messages do not scroll round in the top half of the screen, they will continue on into the bottom half if they start low enough. Pressing the return key several times will achieve this. If the message HARD ERROR appears then this is the faulty sector; if not then we must trace through the file, sector by sector, until we find the faulty one.

This is done in the same way as we traced through the directory sectors, i.e. Print the first two bytes of this sector onto the screen using the P command, then use these numbers (in the SAME order, this time), with the R command, separating them with a comma, to read the next sector in the file. Continue in this way until a HARD ERROR is produced. Nov examine the bottom half of the screen. If the file is a text file, basic program, or basic data file, it will be more or less intelligible and you should be able to decide if it appears correct or is corrupted. If it is a machine code program it will be best to just re-record it as it is, and then load it and check the code with TANBUG's I command. In any case, one of two things will generally happen.

1. The DOS has managed to read the sector, maybe with one or two bytes faulty. In this case the bottom half of the screen will be filled with random-seeming characters. Note that the last sector of a file will have the space between the

end of the file and the end of the sector filled with 0's, which appear on the screen as square boxes. The last sector of a file can normally be identified because it has two 0's in the first two bytes, meaning that there is no next sector, although this is not the case for merged file, complete with its padding of zeros, is linked to the first sector of the next file that was merged.

2. If a gross timing error has occurred, only the first part of the sector will have been read. The DOS will fill up the rest of the screen with FF bytes, which appear as grey squares, the same character, in fact, that is used in the top half of the screen to indicate the keyboard input position. If this happens, repeat the R command, several times if necessary, and there is a good chance that the sector will eventually be read correctly. Even though HARD ERROR may still be indicated, the screen will be full of assorted characters instead of grey squares.

When it seems that no more improvement can be made to the contents of the sector, write it back onto the disc using the W command with the same track, sector numbers that were used with the R command to read the sector. Now try reading this sector again. If all has gone well, no error message will appear this time.

Type X CR to exit from SECDMP back to TANBUG. Try to load the file. It should load without problems, and you can check its contents for errors . . . if it is a basic program, list it, if it is a data file, run the program it serves, etc.

The above procedure may sound complicated, but as often happens, a lot of the complication comes from having to explain clearly and unambiguously (I hope) operations which are actually very simple once you have seen them carried out. Try experimenting with SECDMP. As long as you don't use the W (Write) command nothing will actually happen to the disc . . . the other commands only change the data in the screen memory. If you really want to be cautious, put a write-protect tab over the rectangular notch in the side of the disc (see the silver tab on the distribution disc). Now even the W command cannot harm the disc, since it will produce a protected disc error.

Over the past few months I have had several files corrupted in this way, probably due to the harsh conditions in which I have to use the Microtan (dust, moisture etc) but since I discovered this method of repairing discs I have not had to write-off a single file. I have also been able to repair discs which were refusing to write more files, by going through the free space chain with SECDMP until I got a hard error, and then rewriting the offending segment with whatever was on the screen. So if it happens to you, don't resign yourself to the loss of the file. Give it a try, don't be afraid . . . . and good luck.

APPENDIX

MICROTAN DOS FORMATS

The system sector format is adequately explained in the DOS users manual.

The directory format seems to have got lost in printing. The directory is organised as follows:

The position of the first directory sector is given in bytes 18 (sector) and 19 (track) of the system sector. This is normally track 0, sector 4.

When the directory consists of more than one sector, the first two bytes of each sector indicate the track and sector numbers where the next sector of the directory has been stored. The last sector of the directory has these bytes both set to 0.

The third byte in a directory sector indicates how many filenames it contains. A full directory sector holds 15 entries, but if files have been deleted, the directory entry is over-written with zeros. The other entries are not 'closed up' to remove the gap.

The first directory entry starts in the next byte. Each entry has 16 bytes, the first 6 for the filename, and the next 3 for the extension. After these come a series of numbers. First, the file length (in sectors), given as two bytes, low byte first (A file can occupy more than 256 sectors). Then come the sector number, followed by the track number, where the first sector of the file is saved. Next come sector and track numbers of the last sector of the file, And finally comes a single byte which holds the file attributes. This is 80 (Hex) for a write-protected file and 0 for non-protected files.

This sequence is repeated for successive entries, and finally the last 13 bytes of the directory sector are filled with zeros.

So, in summary:-

Byte 0 TRACK Next directory sector
Byte 1 SECTOR
Byte 2 Number of entries in this sector

The rest of the sector is filled with file entries in the following format.

6 bytes Filename
3 bytes Filename extension
2 bytes File length (sectors). Low byte first.
1 byte SECTOR First Sector of file
1 byte TRACK
1 byte SECTOR Last sector of file
1 byte TRACK
! byte ATTRIBUTE

FILES

Files are saved in the following format.

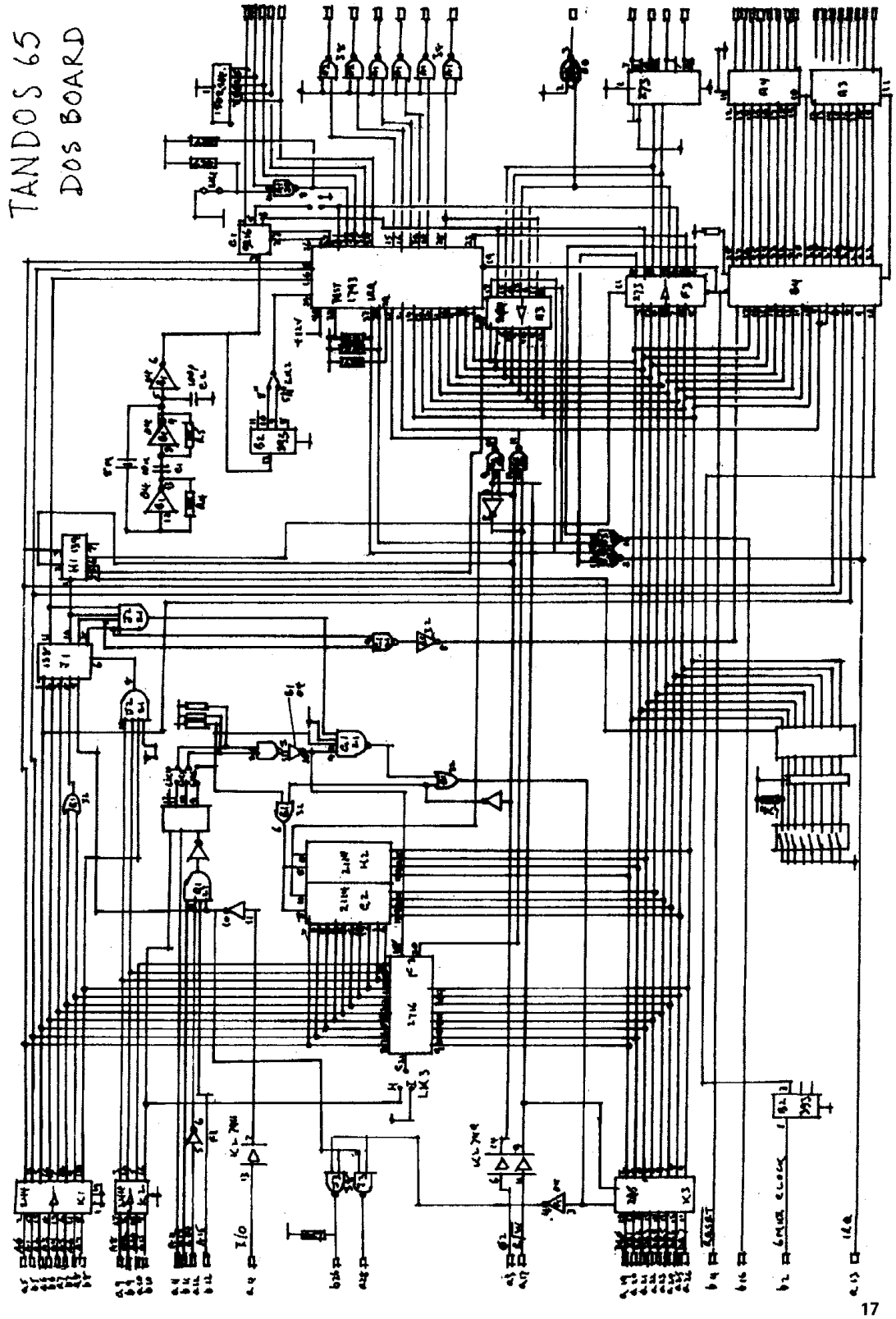
The first two bytes hold the track and sector numbers of succeeding sectors, or 0 if this is the last (or the only) sector of a file.

The next byte holds the length of the first record in this sector. In the first sector, the first record will normally be an address record, see DOS users manual page 5.6. This occupies 8 bytes, making a total of 10. So there are 246 bytes left in the sector. Using one of these for the record length, there are 245, or F5 in hex, bytes left for saving data. So unless a file is shorter than 245 bytes of the file.

Succeeding sectors will have track and sector pointers to the following sector, then a record length byte, leaving 253 bytes for saving data. So the third byte of the other sectors in a file is normally FD (253 in hex), and the rest of the sector is filled with data.

The last sector in a file is not normally occupied completely by data, so the record length byte is less than FD. The unused part of the sector is filled out with zeros in normal files, i.e. files saved with a DSAVE command. Data files saved under basic have a different structure. In particular, they do not have address records, since data files have no fixed address. Rather, they appear to be stored as a series of records on the disc, each record with just a length byte, leaving it to DBASIC to decide the addressing. For this reason, basic data files cannot be read by normal DOS commands.

TANDOS 65
DOS BOARD



Description and use of Tandos routines

- Part 1

ZSAVE (SB7B5), ESCAPE (SB7B2), ERRRET (SB7C7) and ZRSTR (SB492)

These routines are used in conjunction with addresses S40 and S41.

ZSAVE and the associated routine ESCAPE allow the user to enter and leave programs safely, i.e., by using these routines, the original conditions which existed before the routine was entered are restored when the program is left.

ZSAVE saves the current values of store locations S40 and S41, and the value of the stack pointer BEFORE ZSAVE was called (i.e. the value that the stack pointer will have after ZSAVE returns).

Locations S40 and S41 can now be used freely, for example, the TANDOS routines themselves use these addresses, among other things, as a pointer to transfer messages to the screen. (get the message, character by character, with LDA (S40),Y, incrementing either Y or S40/41, and print the characters by calls to Tanbug's OPCHR routine at SFE75, or some other suitable routine. OPCHR will, in fact, print to all active outputs, screen, printer or RS232).

ESCAPE returns to Tanbug after restoring the original values of these locations and, most importantly, restoring the original value of the stack pointer. It can therefore be called from within subroutines, even from within nested subroutines. The programmer does not have to keep count of the levels of subroutine calls, and nevertheless can leave a program from any position in it, for example, from within an error detection subroutine.

An alternative to this is to return via ERRET (SB7C7). This first calls a routine to print "?" followed by a carriage return, and then enters ESCAPE to return to tanbug.

Both ESCAPE and ERRRET use subroutine ZRSTR at SB492 to reset the stack pointer and locations S40 and S41 (in fact, ESCAPE consists of the two instructions JSR SB492 and JMP SF823). This routine could be used to reset these locations without returning to Tanbug, but note that it is not defined as a user-accessible subroutine and its address could change in future versions of TANDOS. If this seems to be the case, disassemble ESCAPE with X-bug's disassembler, or by any other convenient method, to get its new address. Note, by the way, that ZRSTR corrupts the Acc and the X register.

In executable programs, (i.e. programs saved with a transfer address), it is normal to print a carriage return to the screen when the program is entered, but if the program has to extract any information from the command line this should be done. first (for example, DIR looks for filenames, see the TANDOS manual. See also 'Auto-loading your own programs' on page 7.4 of the manual). Also, if the program produces any output, it is advisable to do the same just before calling ESCAPE. Tanbug's OUTPCR routine at SFE73 will do this to all active outputs, screen, printer, and RS232.

So a typical executable program will have the following structure.

```
JSR SB7B5      : ZSAVE
  "
  "            : Extract info from command line
  "
JSR SFE73      : Print CR
  "
  "            : Main body of program
  "
JSR SFE73      : CR . Advisable if any printing done
JMP SB7B2      : ESCAPE
  "            : Error checking subroutine
BEQ RETURN     : No error
JMP ERRRET     : Prints "?" CR and ESCAPE's
RETURN: RTS
```

The 'BEQ' instruction in the error checking routine is purely illustrative, of course. In fact, many programs will not even have such a routine.

Obviously, these routines can be used perfectly well within subroutines which will be called from within a program: - there is no reason which restricts their use to disc programs called from the keyboard via Tanbug. But it should be born in mind that ZSAVE only saves the stack pointer and the two memory addresses. If the Acc or the X or Y registers need to be preserved this must be done separately, and in the case of the Acc and IX it must be done before calling ZSAVE, since ZSAVE itself corrupts the Acc and the X register. It will also be necessary to restore the Acc and the index registers, AFTER calling ZRSTR to restore the stack pointer. The Y register can be saved on the stack, it must be restored at the correct moment, i.e., if it was stored AFTER calling ZSAVE, it must be recovered BEFORE calling ZRSTR, and vice versa. If the Y register is saved in a memory location instead of on the stack this does not apply.

VIEWPOINT Contd.

Claude Gasquet
Switzerland.

Dear Sir,

I have well received this week the issue No. 5 of Microtan World. Thanks! **But I never received the issue No. 4.** Please could you see and send me one? Because of my interest for the Microtan system I should not like to lose information about it.

Now some ideas:

- Is MCS investigating something with the Mp 65C02? It could be real improvement for the system without leave of its philosophy.
- Could you explain in Microtan World how works the character generator on the microtan 65 board?
- I would be interested to implant french characters but how to do? When I wish a 'e' on my printer (RX80 Epson) I have to type a " " on the screen. Useful not at all!
- It could be a sale support overseas to propose other sets of characters (Germain, Spanish, French, etc. . .) English speaking countries are not all the world. Maybe for a future system but why not for microtan 65?
- Herewith 3 little different but identical programs. 2 give error. One is correct. Why?

(Please excuse my bad English)

Would readers like to comment . . . Ed.

THE TWO PASS PIECE

Graham Davies M.Sc., B.Sc., C. Eng., M.I. Mech.E.

Recently, my programs 'OVAR' and 'HULTAN', demonstrated that M'AN CANNOT FILE BY READ ALONE. I have continued to work on disc file handling and I am now able to present 'TPAUG'; a patch that allows the Microtanic Two Pass Assembler (TPA) to support disc handling of source code files.

The source code for 'TPAUG starts at the rad address S2000 and its printer listing requires some fourteen fan fold pages at sixty lines of code per page. Because of its length and as a generous gesture to users wishing to avoid the tedium of manual entry, David Northway has kindly offered to supply the source code on diskette at a nominal charge. This file is designated 'TPAUG.T' and should be 'DLOADED' to ram using the 'S' parameter if a source start other than S2000 is required. The TPA should be then entered using a cold start, followed, typically, by 'ES' <CR>, <BKSPC> and <A1> <CR> when the parameters for assembly etc. will be set.

IMPLEMENTING TPAUG

'TPAUG.T' is most conveniently implemented when users have ram from $C000 to beyond $D000. In this case the patch should be assembled in the usual way when the 'TPA' will be automatically modified to include the jump vectors for the patch. The assembled object code can then be saved to disc using the 'T' parameter when the user becomes the proud owner of an updated disc file handling Microtanic Two Pass Assembler.

For Tanfolk who continue to operate the 'TPA' out of eprom there is no real problem because most users have access to an eprom programmer. Before blowing a new eprom however, it will be necessary to decide where in ram the patch is to be located and then alter the 'ORG' addresses in the patch to suit. With the patch assembled, examination of the source/object code listing will reveal the altered jump addresses to be inserted into the eprom. It will also be seen that the p .tch at $C1F7 corresponds to the new length of the 'Action' command tables whilst the last patch is used to create a new default source code start address of $2000. Please note that the patch has to be in ram because one byte within the patch may be subject to change during run time.

FILE CREATION

This consists of inputting 'DD' <CR> to the 'Action' space. The top half of the screen will be cleared and the following message will appear:

```
WRITING TO DISC
PLEASE ENTER AS:
DRV:FILENAME
```

For example, to file 'TPAUG' to drive '0', simply enter

```
TPAUG <CR>
```

and this is echoed on the bottom line of screen. The number of characters allowed here is a maximum of six.

The curious may now exit from the 'TPA' and access 'DIR' to, hopefully, find the file 'TPAUG.TWP' resident; the file extension having been added automatically It should be noted such a file cannot be loaded using 'DLOAD'.

For those with multiple drives, it may be appropriate to enter the filename as:

```
2:TPAUG <CR>
```

in which case drive '2' is activated and the filename 'TPAUG.TWP' will be found in the corresponding directory. In a sense the drive number is optional but if given must be accompanied by ':'. Here the number of characters is limited to eight providing ':' accompanies the drive number. In both cases correction is possible using the delete key.

Care must be taken here as escape from file creation or retrieval at this point requires a 'RESET'. Further, each file created must have its own unique filename. Consequently, when a file is updated and returned to disc it must be given a new name. The original file must then be deleted in the usual way. I have plans in hand to overcome this problem.

The last record, which in 'OVAR' contained only the end-of-text marker; here in 'TPAUG' contains all the parameters required for immediate assembly when the file has been recovered from disc (Warm Start - section 35 of the TPA manual). I hasten to add that this does not mean assembly during file retrieval.

The problem of end-of-text marker now rears its head and I here reinforce remarks made elsewhere; my use of $1A on its own can be disastrous as $1A could legitimately appear in the source code. Actually, the very first byte of 'TPAUG' source code proved to be $1A and the file could not be recovered from disc by 'HULTAN ' or 'LIST'. From Doug Deedman's remarks in issue 4 and discussions I have had with Andy Michael, it is clear that some thought will have to be put into end-of-file identification. I have opted for two nulls preceding $1A. Other opinions would be welcome.

Before outputting the final record, it is 'assembled' into a temporary buffer PaRAMeterSaVe. This serves two purposes, firstly the buffer may be output in its entirety as a last record to provide a warm start on file recovery and secondly, it provides a home for the parameters should the disc operation be aborted and the normal parameter locations be corrupted.

FILE RETRIEVAL

With 'FD' <CR> input at 'Action' the ensueing screen activity corresponds to that of 'DD' except that the first line of the upper screen message reads:

READING FROM DISC

Inputting the filename and a <CR> will cause the file to be loaded to ram from $2000. Termination of 'FD' or 'DD' action is indicated in the customary 'TPA' manner; the cursor re-appears in the 'Action' box. The last record subroutine, RECRDA allows the parameters to be placed so as to enable immediate assembly, if that is required. Printing of the source code is also possible on file recovery. However a cold may be necessary if the printer type changes from say parallel to serial between file creation and file recovery.

In both cases 'ERREX' has been used to vector TANDOS error messages to the bottom line of screen. This is accompanied in the upper portion of screen by the TPAUG message:

```
DISC ERROR
Press 'R' to RESTART
```

The error message may then be viewed by the user before typing 'R' to clear the screen for the reconstruction of the screen table and recovery of the 'WARM START' parameters from PRAMSV. These are then entered into the table and the cursor reappears in the 'ACTION' box. It should be noted that this action is taken following disc error in both 'READ' and 'WRITE' modes. The

parameters are also restored following a successful 'WRITE' operation as disc activity corrupts locations $40 and $41.

FURTHER DEVELOPMENTS

With the command tables now in ram and the relative jump technique understood, it should be possible to insert wedges into any of the 'TPA' routines. For example, the recently published 'printer utilities' might benefit from this. My personal priority is to output the line 'ESC'; '1'; '8' to my FX80 to provide a reasonable margin for filing without destroying and/or hiding the line numbers of source listings. Access to the command tables will be of interest to non-disc users so perhaps David can be persuaded to provide the source listing on cassette!

FINAL REMARKS

Please remember that 'TPAUG' is, to some extent, experimental. In particular I am hoping for some response to the need for standardisation of end-of-text marking. I am also looking forward with interest to reports of the success or otherwise of 'TPAUG'. I have used the program with success for some weeks now and am therefore confident that the bugs have been exorcised.

Department of Mechanical Engineering
Bolton Institute of Higher Education
30 November 1983

```
0001 ;*************************1400
0002 ;**        PROGRAM        **1400
0003 ;**         TPAUG         **1400
0004 ;** DISC FILE  FACILITY **1400
0005 ;**    FOR  MICROTANIC    **1400
0006 ;** TWO PASS  ASSEMBLER **1400
0007 ;**                       **1400
0008 ;** Copyright  G.Davies **1400
0009 ;**  November     1983    **1400
0010 ;*************************1400
0011 ;                           1400
0012 ;                           1400
0013         ORG $D000           1400
0014 ;                           D000
0015 ;                           D000
0016 ;TRANSFER ACTION ADDRESS    D000
0017 ;TABLES TO PROVIDE ROOM     D000
0018 ;FOR ADDITIONAL COMMANDS    D000
0019 ;                           D000
0020 TABL1   BYT 'O,'E,'D,'F     D000 4F 45 44 46
0021         BYT 'E,'F,'A,'F     D004 45 46 41 46
0022         BYT 'A,'O,'O,'O     D008 41 4F 4F 4F
0023         BYT 'O,'O,'L,'E     D00C 4F 4F 4C 45
0024         BYT 'D,'F,'P        D010 44 46 50
0025 ;                           D013
0026         ORG $D020           D013
0027 ;                           D020
0028 TABL2   BYT 'S,'S,'F,'F     D020 53 53 46 46
0029         BYT 'F,'1,'1,'2     D024 46 31 31 32
0030         BYT '2,'M,'P,'N     D028 32 4D 50 4E
0031         BYT 'L,'R,'L,'X     D02C 4C 52 4C 58
0032         BYT 'D,'D,'P        D030 44 44 50
0033 ;                           D033
0034         ORG $D040           D033
0035 ;                           D040
0036 TABL3   BYT 0,4,7,4         D040 00 04 07 04
0037         BYT 0,$97,$97,$97   D044 00 97 97 97
0038         BYT $97,0,0,0       D048 97 00 00 00
0039         BYT 0,0,0,0         D04C 00 00 00 00
0040         BYT 0,0,0           D050 00 00 00
0041 ;                           D053
0042         ORG $D060           D053
0043 ;                           D060
0044 ;LOW BYTE OF (JMP)          D060
0045 ;ADDRESSES                  D060
0046 ;                           D060
0047 TABL4   BYT $B,$AE,$77      D060 0B AE 77
0048         BYT $21,$28,$31     D063 21 28 31
0049         BYT $EC,$31,$EC     D066 EC 31 EC
0050         BYT 8,$E,$10,7      D069 08 0E 10 07
0051         BYT $C,$B9,0,$A0    D06D 0C B9 00 A0
0052         BYT $A3,$A6         D071 A3 A6
0053 ;                           D073
0054         ORG $D080           D073
0055 ;                           D080
0056 ;HIGH BYTE OF (JMP)         D080
0057 ;ADDRESSES                  D080
0058 ;                           D080
0059 TABL5   BYT $C7,$C3,$C5     D080 C7 C3 C5
0060         BYT $C6,$C6,$C6     D083 C6 C6 C6
0061         BYT $C6,$C6,$C6     D086 C6 C6 C6
0062         BYT $C7,$C7,$C7     D089 C7 C7 C7
0063         BYT $C7,$C7,$CD     D08C C7 C7 CD
0064         BYT $FC,$D0,$D0     D08F FC D0 D0
0065         BYT $D0             D092 D0
0066 ;                           D093
0067         ORG $D0A0           D093
0068 ;                           D0A0
0069 ;                           D0A0
0070 ;JUMP VECTORS               D0A0
0071 ;                           D0A0
0072 JWRITE JMP WRITEO           D0A0 4C 19 D3
0073 JREAD   JMP READO           D0A3 4C 8A D3
0074 ;                           D0A6
0075 ;                           D0A6
0076 ;TANDOS ADDRESSES           D0A6
0077 ;                           D0A6
0078 GETFIL EQU $B7F7            D0A6
0079 INIIO  EQU $A806            D0A6
0080 OPENR  EQU $A809            D0A6
0081 OPENW  EQU $A80C            D0A6
0082 CLOSER EQU $A80F            D0A6
0083 CLOSEW EQU $A812            D0A6
0084 RECIN  EQU $A815            D0A6
0085 RECOUT EQU $A818            D0A6
0086 UNITO  EQU $B92D            D0A6
0087 FNAMO  EQU $B92E            D0A6
0088 TEXTO  EQU $B934            D0A6
0089 ERREX  EQU $B94F            D0A6
0090 WBUF   EQU $BBFC            D0A6
0091 WREC   EQU $BBFD            D0A6
0092 RBUF   EQU $BBFE            D0A6
0093 RREC   EQU $BBFF            D0A6
0094 WSCBUF EQU $400             D0A6
0095 WRCBUF EQU $500             D0A6
0096 RSCBUF EQU $600             D0A6
0097 RRCBUF EQU $700             D0A6
0098 ;                           D0A6
0099 ;                           D0A6
0100 ;TANBUG ADDRESSES           D0A6
0101 ;                           D0A6
0102 ICHAR  EPZ $01              D0A6
0103 VDUIND EPZ $03              D0A6
0104 ICURS  EPZ $0A              D0A6
0105 JPLKB  EQU $FB1D            D0A6
0106 ;                           D0A6
0107 ;TPAUG ADDRESSES            D0A6
0108 ;                           D0A6
0109 MSADL  EPZ $38              D0A6
0110 PENFLG EPZ $3A              D0A6
0111 YCOUNT EPZ $3B              D0A6
0112 CRNTLO EPZ $3C              D0A6
0113 DTLFTL EPZ $3E              D0A6
0114 STRTLO EPZ $66              D0A6
0115 ENDLO  EPZ $68              D0A6
0116 LINOO  EPZ $00              D0A6
0117 LINO1  EPZ $20              D0A6
0118 LINO2  EPZ $40              D0A6
0119 LINO3  EPZ $60              D0A6
0120 LINO4  EPZ $80              D0A6
0121 LINO5  EPZ $A0              D0A6
0122 LINO6  EPZ $C0              D0A6
0123 LINO7  EPZ $E0              D0A6
0124 PRAMSV EQU $100             D0A6
0125 ;                           D0A6
0126 ;                           D0A6
0127 ;MESSAGES RELATING TO       D0A6
0128 ;DISC FILE HANDLING         D0A6
0129 ;                           D0A6
0130 MSG1    BYT 'D,'I,'S,'C     D0A6 44 49 53 43
0131         BYT 0               D0AA 00
0132 ;                           D0AB
0133 MSG2    BYT 'D,'U,'M,'P     D0AB 44 55 4D 50
0134         BYT 0               D0AF 00
0135 ;                           D0B0
0136 MSG3    BYT 'W,'R,'I,'T     D0B0 57 52 49 54
0137         BYT 'I,'N,'G,'      D0B4 49 4E 47 20
```

```
0138          BYT 'T,'O,'
0139          BYT 0
0140 ;
0141 MSG4     BYT 'C,'R,'E,'A
0142          BYT 'T,'E,'D,0
0143 ;
0144 MSG5     BYT 'R,'E,'T,'R
0145          BYT 'I,'E,'V,'E
0146          BYT 'D,0
0147 ;
0148 MSG6     BYT 'O,'N,0
0149 ;
0150 MSG7     BYT 'F,'R,'O,'M
0151          BYT ' ,0
0152 ;
0153 MSG8     BYT 'F,'I,'L,'E
0154          BYT 0
0155 ;
0156 MSG9     BYT 'R,'E,'A,'D
0157          BYT 'I,'N,'G,'
0158          BYT 0
0159 ;
0160 MSG10    BYT 'P,'L,'E,'A
0161          BYT 'S,'E,'
0162          BYT 'E,'N,'T,'E
0163          BYT 'R,' ,'A,'S
0164          BYT ':,0
0165 ;
0166 MSG11    BYT 'N,'A,'M,'E
0167          BYT 0
0168 ;
0169 MSG12    BYT 'D,'R,'V,':
0170          BYT 0
0171 ;
0172 MSG13    BYT ' ,' ,0
0173 ;
0174 MSG14    BYT 'T,'W,'P,0
0175 ;
0176 MSG15    BYT 'E,'R,'R,'O
0177          BYT 'R,0
0178 ;
0179 MSG16    BYT 'P,'r,'e,'s
0180          BYT 's,' ,'',R
0181          BYT '',' ,'t,'o
0182          BYT ' ,'R,'E,'S
0183          BYT 'T,'A,'R,'T
0184          BYT 0
0185 ;
0186 ;
0187 ;MESSAGE ADDRESS TABLE
0188 ;
0189 MSX0     WOR MSG1
0190 MSX2     WOR MSG2
0191 MSX4     WOR MSG3
0192 MSX6     WOR MSG4
0193 MSX8     WOR MSG5
0194 MSX10    WOR MSG6
0195 MSX12    WOR MSG7
0196 MSX14    WOR MSG8
0197 MSX16    WOR MSG9
0198 MSX18    WOR MSG10
0199 MSX20    WOR MSG11
0200 MSX22    WOR MSG12
0201 MSX24    WOR MSG13
0202 MSX26    WOR MSG14
0203 MSX28    WOR MSG15
0204 MSX30    WOR MSG16
0205 ;
0206 ;
0207 ;SUBROUTINES:
0208 ;
0209 ;1) SET UP ZERO PAGE BYTES
0210 ;OF MESSAGE ADDRESSES
0211 ;
0212 MSGX     LDA MSX0,X
0213          STA MSADL
0214          LDA MSX0+1,X
0215          STA MSADL+1
0216          RTS
0217 ;
0218 ;2) OUTPUT THE MESSAGE
0219 ;TO SCREEN
0220 ;
0221 MSOUT1 JSR MSGX

D0B8 54 4F 20
D0BB 00
D0BC
D0BC 43 52 45 41
D0C0 54 45 44 00
D0C4
D0C4 52 45 54 52
D0C8 49 45 56 45
D0CC 44 00
D0CE
D0CE 4F 4E 00
D0D1
D0D1 46 52 4F 4D
D0D5 20 00
D0D7
D0D7 46 49 4C 45
D0DB 00
D0DC
D0DC 52 45 41 44
D0E0 49 4E 47 20
D0E4 00
D0E5
D0E5 50 4C 45 41
D0E9 53 45 20
D0EC 45 4E 54 45
D0F0 52 20 41 53
D0F4 3A 00
D0F6
D0F6 4E 41 4D 45
D0FA 00
D0FB
D0FB 44 52 56 3A
D0FF 00
D100
D100 20 20 00
D103
D103 54 57 50 00
D107
D107 45 52 52 4F
D10B 52 00
D10D
D10D 50 72 65 73
D111 73 20 27 52
D115 27 20 74 6F
D119 20 52 45 53
D11D 54 41 52 54
D121 00
D122
D122
D122
D122
D122 A6 D0
D124 AB D0
D126 B0 D0
D128 BC D0
D12A C4 D0
D12C CE D0
D12E D1 D0
D130 D7 D0
D132 DC D0
D134 E5 D0
D136 F6 D0
D138 FB D0
D13A 00 D1
D13C 03 D1
D13E 07 D1
D140 0D D1
D142
D142
D142
D142
D142
D142 BD 22 D1
D145 85 38
D147 BD 23 D1
D14A 85 39
D14C 60
D14D
D14D
D14D
D14D 20 42 D1

0222          LDY #0
0223 MSOUT2 LDA (MSADL),Y
0224          BEQ MSOUT4
0225          CPY #0
0226          BNE MSOUT3
0227          PHA
0228          LDA ICURS
0229          ADC VDUIND
0230          STA ICURS
0231          PLA
0232 MSOUT3 STA (ICURS),Y
0233          INY
0234          JMP MSOUT2
0235 MSOUT4 DEY
0236          STY VDUIND
0237          RTS
0238 ;
0239 ;3) FIRST WORD OF MESSAGE
0240 ;
0241 WRD1S    LDY #0
0242          STY VDUIND
0243          STA ICURS
0244          JSR MSOUT1
0245          RTS
0246 ;
0247 ;4) CLEAR A LINE
0248 ;
0249 CLRLNO LDY #0
0250          STY VDUIND
0251          STA ICURS
0252          LDA #'
0253 CLRLN1 STA (ICURS),Y
0254          INY
0255          CPY #$20
0256          BNE CLRLN1
0257          RTS
0258 ;
0259 ;5) CLEAR TOP OF SCREEN
0260 ;
0261 CLRTPO LDA #'
0262          LDY #$0
0263 CLRTP1 STA $200,Y
0264          DEY
0265          BNE CLRTP1
0266          RTS
0267 ;
0268 ;6) OUTPUT THE FILENAME
0269 ;AND DRIVE NUMBER TO
0270 ;BOTTOM LINE
0271 ;
0272 CMLINO LDY #0
0273          LDA #$7
0274          STA LENLN+1
0275 CMLIN1 JSR JPLKB
0276          LDA ICHAR
0277          CMP #$0D
0278          BNE CONCM1
0279          RTS
0280 CONCM1 CMP #':
0281          BNE CONCM2
0282          PHA
0283          LDA #$9
0284          STA LENLN+1
0285          PLA
0286 CONCM2 CMP #$7F
0287          BEQ DELET1
0288 ALPHWT STA (ICURS),Y
0289          INY
0290 LENLN    CPY #$9
0291          BPL DELET2
0292          BNE CMLIN1
0293 DELET1 CPY #$0
0294          BEQ CMLIN1
0295          CPY #$01
0296          BNE DELET2
0297          LDA #7
0298          STA LENLN+1
0299 DELET2 LDA #'
0300          STA (ICURS),Y
0301          DEY
0302          LDA #$7F
0303          STA (ICURS),Y
0304          JMP CMLIN1
0305 ;

D150 A0 00
D152 B1 38
D154 F0 12
D156 C0 00
D158 D0 08
D15A 48
D15B A5 0A
D15D 65 03
D15F 85 0A
D161 68
D162 91 0A
D164 C8
D165 4C 52 D1
D168 88
D169 84 03
D16B 60
D16C
D16C
D16C
D16C A0 00
D16E 84 03
D170 85 0A
D172 20 4D D1
D175 60
D176
D176
D176
D176 A0 00
D178 84 03
D17A 85 0A
D17C A9 20
D17E 91 0A
D180 C8
D181 C0 20
D183 D0 F9
D185 60
D186
D186
D186
D186 A9 20
D188 A0 00
D18A 99 00 02
D18D 88
D18E D0 FA
D190 60
D191
D191
D191
D191
D191 A0 00
D193 A9 07
D195 8D B5 D1
D198 20 1D FB
D19B A5 01
D19D C9 0D
D19F D0 01
D1A1 60
D1A2 C9 3A
D1A4 D0 07
D1A6 48
D1A7 A9 09
D1A9 8D B5 D1
D1AC 68
D1AD C9 7F
D1AF F0 09
D1B1 91 0A
D1B3 C8
D1B4 C0 09
D1B6 10 0F
D1B8 D0 DE
D1BA C0 00
D1BC F0 DA
D1BE C0 01
D1C0 D0 05
D1C2 A9 07
D1C4 8D B5 D1
D1C7 A9 20
D1C9 91 0A
D1CB 88
D1CC A9 7F
D1CE 91 0A
D1D0 4C 98 D1
D1D3
```

```
0306 ;7) CALL 'GETFIL' TO       D1D3
0307 ;COMMAND LINE              D1D3
0308 ;                          D1D3
0309 FINDFL LDY #$0             D1D3 A0 00
0310        JSR GETFIL          D1D5 20 F7 B7
0311        RTS                 D1D8 60
0312 ;                          D1D9
0313 ;8) ADD EXTENSION 'TWP'    D1D9
0314 ;TO FILENAME               D1D9
0315 ;                          D1D9
0316 FILXT  LDX #$0             D1D9 A2 00
0317 FILXN1 LDA MSG14,X         D1DB BD 03 D1
0318        BEQ RET8            D1DE F0 07
0319        STA TEXTO,X         D1E0 9D 34 B9
0320        INX                 D1E3 E8
0321        JMP FILXN1          D1E4 4C DB D1
0322 RET8   RTS                 D1E7 60
0323 ;                          D1E8
0324 ;9) INITIALISE FLAGS AND   D1E8
0325 ;COUNTERS                  D1E8
0326 ;                          D1E8
0327 INIT1  LDA #$0             D1E8 A9 00
0328        STA YCOUNT          D1EA 85 3B
0329        STA PENFLG          D1EC 85 3A
0330        LDA STRTLO          D1EE A5 66
0331        STA CRNTLO          D1F0 85 3C
0332        LDA STRTLO+1        D1F2 A5 67
0333        STA CRNTLO+1        D1F4 85 3D
0334        RTS                 D1F6 60
0335 ;                          D1F7
0336 ;10) SET UP ERROR VECTOR   D1F7
0337 ;POINTER FOR TANDOS ERROR  D1F7
0338 ;MESSAGES                  D1F7
0339 ;                          D1F7
0340 ERROR1 LDA #ERROR2         D1F7 A9 EB
0341        STA ERREX           D1F9 8D 4F B9
0342        LDA #ERROR2>        D1FC A9 D3
0343        STA ERREX+1         D1FE 8D 50 B9
0344        RTS                 D201 60
0345 ;                          D202
0346 ;11) OUTPUT RECORDS TO     D202
0347 ;DISC                      D202
0348 ;                          D202
0349 TRNSFW LDY YCOUNT          D202 A4 3B
0350 RCSTRT LDX #0              D204 A2 00
0351        SEC                 D206 38
0352        LDA DTLFTL          D207 A5 3E
0353        SBC #$FF            D209 E9 FF
0354        STA DTLFTL          D20B 85 3E
0355        BNE DATHI           D20D D0 06
0356        LDA DTLFTL+1        D20F A5 3F
0357        BEQ PENREC          D211 F0 09
0358        BNE FULREC          D213 D0 16
0359 DATHI  DEC DTLFTL+1        D215 C6 3F
0360        BMI PENREC          D217 30 03
0361        JMP FULREC          D219 4C 2B D2
0362 PENREC LDA #1              D21C A9 01
0363        STA PENFLG          D21E 85 3A
0364        CLC                 D220 18
0365        LDA DTLFTL          D221 A5 3E
0366        ADC #$FF            D223 69 FF
0367        STA WRCBUF          D225 8D 00 05
0368        JMP FILL1           D228 4C 30 D2
0369 FULREC LDA #$FF            D22B A9 FF
0370        STA WRCBUF          D22D 8D 00 05
0371 FILL1  INX                 D230 E8
0372        LDA (CRNTLO),Y      D231 B1 3C
0373        STA WRCBUF,X        D233 9D 00 05
0374        CPY #$FF            D236 C0 FF
0375        BNE FILL2           D238 D0 02
0376        INC CRNTLO+1        D23A E6 3D
0377 FILL2  INY                 D23C C8
0378        CPX WRCBUF          D23D EC 00 05
0379        BNE FILL1           D240 D0 EE
0380        STY YCOUNT          D242 84 3B
0381        JSR RECOUT          D244 20 18 A8
0382        LDA PENFLG          D247 A5 3A
0383        BNE LASREC          D249 D0 03
0384        JMP TRNSFW          D24B 4C 02 D2
0385 ;                          D24E
0386 LASREC LDY PRAMSV          D24E AC 00 01
0387 WLAS1  LDA PRAMSV,Y        D251 B9 00 01
0388        STA WRCBUF,Y        D254 99 00 05
0389        STA RRCBUF,Y        D257 99 00 07
0390        DEY                 D25A 88
0391        BPL WLAS1           D25B 10 F4
0392        JSR RECOUT          D25D 20 18 A8
0393        JSR CLOSEW          D260 20 12 A8
0394        LDA #$FF            D263 A9 FF
0395        STA PENFLG          D265 85 3A
0396        JSR RECRDB          D267 20 72 D2
0397        RTS                 D26A 60
0398 ;                          D26B
0399 ;12) RETRIEVE A FILE       D26B
0400 ;FROM DISC                 D26B
0401 ;                          D26B
0402 RECRDA JSR RECIN           D26B 20 15 A8
0403        LDA #$0             D26E A9 00
0404        STA PENFLG          D270 85 3A
0405 RECRDB LDY #$0             D272 A0 00
0406        TYA                 D274 98
0407 RLAS1  CMP RRCBUF+1,Y      D275 D9 01 07
0408        BNE TRNCN1          D278 D0 44
0409        INY                 D27A C8
0410        CPY #$2             D27B C0 02
0411        BNE RLAS1           D27D D0 F6
0412        LDA #$1A            D27F A9 1A
0413        CMP RRCBUF+1,Y      D281 D9 01 07
0414        BNE TRNCN1          D284 D0 38
0415        LDX #$0             D286 A2 00
0416        INY                 D288 C8
0417 RLAS1A LDA RRCBUF+1,Y      D289 B9 01 07
0418        STA $40,X           D28C 95 40
0419        INY                 D28E C8
0420        INX                 D28F E8
0421        CPX #$05            D290 E0 05
0422        BNE RLAS1A          D292 D0 F5
0423        LDA RRCBUF+1,Y      D294 B9 01 07
0424        STA $FF             D297 85 FF
0425        LDX #$0             D299 A2 00
0426        INY                 D29B C8
0427 RLAS2  LDA RRCBUF+1,Y      D29C B9 01 07
0428        CMP #$1A            D29F C9 1A
0429        BNE RLAS3           D2A1 D0 09
0430        INY                 D2A3 C8
0431        LDA RRCBUF+1,Y      D2A4 B9 01 07
0432        CMP #$1A            D2A7 C9 1A
0433        BEQ RET12A          D2A9 F0 08
0434        DEY                 D2AB 88
0435 RLAS3  STA $62,X           D2AC 95 62
0436        INX                 D2AE E8
0437        INY                 D2AF C8
0438        JMP RLAS2           D2B0 4C 9C D2
0439 RET12A LDA PENFLG          D2B3 A5 3A
0440        BNE RET12B          D2B5 D0 03
0441        JSR CLOSER          D2B7 20 0F A8
0442 RET12B JSR $C152           D2BA 20 52 C1
0443        RTS                 D2BD 60
0444 TRNCN1 LDY YCOUNT          D2BE A4 3B
0445        LDX #$0             D2C0 A2 00
0446 TRNSFR INX                 D2C2 E8
0447        LDA RRCBUF,X        D2C3 BD 00 07
0448        STA (CRNTLO),Y      D2C6 91 3C
0449        CPY #$FF            D2C8 C0 FF
0450        BNE NXTY            D2CA D0 02
0451        INC CRNTLO+1        D2CC E6 3D
0452 NXTY   INY                 D2CE C8
0453        CPX RRCBUF          D2CF EC 00 07
0454        BNE TRNSFR          D2D2 D0 EE
0455        STY YCOUNT          D2D4 84 3B
0456        JMP RECRDA          D2D6 4C 6B D2
0457 RET12  RTS                 D2D9 60
0458 ;                          D2DA
0459 ;13) TEMPORARY STORE FOR   D2DA
0460 ;PRINT AND ASSEMBLE        D2DA
0461 ;PARAMETERS FOLLOWING      D2DA
0462 ;DISC WRITE AND READ       D2DA
0463 ;                          D2DA
0464 PRMSV1 LDY #$00            D2DA A0 00
0465        TYA                 D2DC 98
0466 PRMSV2 STA PRAMSV+1,Y      D2DD 99 01 01
0467        INY                 D2E0 C8
0468        CPY #$02            D2E1 C0 02
0469        BNE PRMSV2          D2E3 D0 F8
0470        LDA #$1A            D2E5 A9 1A
0471        STA PRAMSV+1,Y      D2E7 99 01 01
0472        LDX #$0             D2EA A2 00
0473        INY                 D2EC C8
```

```
0474 PRMSV3 LDA $40,X          D2ED B5 40
0475        STA PRAMSV+1,Y     D2EF 99 01 01
0476        INY                D2F2 C8
0477        INX                D2F3 E8
0478        CPX #$05           D2F4 E0 05
0479        BNE PRMSV3         D2F6 D0 F5
0480        LDA $FF            D2F8 A5 FF
0481        STA PRAMSV+1,Y     D2FA 99 01 01
0482        LDX #$0            D2FD A2 00
0483        INY                D2FF C8
0484 PRMSV4 LDA $62,X          D300 B5 62
0485        STA PRAMSV+1,Y     D302 99 01 01
0486        INY                D305 C8
0487        INX                D306 E8
0488        CPX #$11           D307 E0 11
0489        BNE PRMSV4         D309 D0 F5
0490        LDA #$1A           D30B A9 1A
0491        STA PRAMSV+1,Y     D30D 99 01 01
0492        INY                D310 C8
0493        STA PRAMSV+1,Y     D311 99 01 01
0494        INY                D314 C8
0495        STY PRAMSV         D315 8C 00 01
0496        RTS                D318 60
0497 ;                         D319
0498 ;**** MAIN  PROGRAM ****  D319
0499 ;************************ D319
0500 ;                         D319
0501 ;a) FILE CREATION         D319
0502 ;************************ D319
0503 ;                         D319
0504 ;                         D319
0505 ;SAVE PARAMETERS TO       D319
0506 ;PREVENT CORRUPTION       D319
0507 ;BY DISC ACTIVITY         D319
0508 ;                         D319
0509 WRITE0 JSR PRMSV1         D319 20 DA D2
0510 WRITE1 JSR CLRTP0         D31C 20 86 D1
0511        LDA #2             D31F A9 02
0512        STA ICURS+1        D321 85 0B
0513        LDX #4             D323 A2 04
0514        LDA #LIN02         D325 A9 40
0515        JSR WRD1S          D327 20 6C D1
0516 ;                         D32A
0517 WRITE2 LDX #0             D32A A2 00
0518        JSR MSOUT1         D32C 20 4D D1
0519 ;                         D32F
0520 WRITE3 LDX #18            D32F A2 12
0521        LDA #LIN04         D331 A9 80
0522        JSR WRD1S          D333 20 6C D1
0523 ;                         D336
0524 WRITE4 LDX #22            D336 A2 16
0525        LDA #LIN05         D338 A9 A0
0526        JSR WRD1S          D33A 20 6C D1
0527 ;                         D33D
0528 WRITE6 LDX #14            D33D A2 0E
0529        JSR MSOUT1         D33F 20 4D D1
0530 ;                         D342
0531 WRITE7 LDX #20            D342 A2 14
0532        JSR MSOUT1         D344 20 4D D1
0533 ;                         D347
0534 ;OUTPUT THE COMMAND LINE  D347
0535 ;                         D347
0536 JCMND  CLI                D347 58
0537        LDA #3             D348 A9 03
0538        STA ICURS+1        D34A 85 0B
0539        LDA #LIN07         D34C A9 E0
0540        JSR CLRLN0         D34E 20 76 D1
0541        JSR CMLIN0         D351 20 91 D1
0542        SEI                D354 78
0543 ;                         D355
0544 ;                         D355
0545        JSR FINDFL         D355 20 D3 D1
0546        JSR FILXT          D358 20 D9 D1
0547        JSR INIT1          D35B 20 E8 D1
0548 ;                         D35E
0549 ;                         D35E
0550 ;COMPUTE NUMBER OF BYTES  D35E
0551 ;IN SOURCE-CODE FILE      D35E
0552 ;                         D35E
0553 LENSRC CLD                D35E D8
0554        SEC                D35F 38
0555        LDA ENDLO          D360 A5 68
0556        SBC STRTLO         D362 E5 66
0557        STA DTLFTL         D364 85 3E
0558        LDA ENDLO+1        D366 A5 69
0559        SBC STRTLO+1       D368 E5 67
0560        STA DTLFTL+1       D36A 85 3F
0561        INC DTLFTL         D36C E6 3E
0562        BNE CRCTLN         D36E D0 02
0563        INC DTLFTL+1       D370 E6 3F
0564 CRCTLN NOP                D372 EA
0565 ;                         D373
0566 ;                         D373
0567 ;SET UP SECTOR/RECORD     D373
0568 ;BUFFERS FOR WRITE        D373
0569 ;                         D373
0570        LDA #$04           D373 A9 04
0571        STA WBUF           D375 8D FC BB
0572        LDA #$05           D378 A9 05
0573        STA WREC           D37A 8D FD BB
0574 ;                         D37D
0575 ;                         D37D
0576        JSR ERROR1         D37D 20 F7 D1
0577        JSR INIIO          D380 20 06 A8
0578        JSR OPENW          D383 20 0C A8
0579 ;                         D386
0580        JSR TRNSFW         D386 20 02 D2
0581        RTS                D389 60
0582 ;                         D38A
0583 ;                         D38A
0584 ;b) FILE RETRIEVAL ***    D38A
0585 ;                         D38A
0586 READ0  JSR PRMSV1         D38A 20 DA D2
0587        JSR CLRTP0         D38D 20 86 D1
0588        LDA #2             D390 A9 02
0589        STA ICURS+1        D392 85 0B
0590        LDX #16            D394 A2 10
0591        LDA #LIN02         D396 A9 40
0592        JSR WRD1S          D398 20 6C D1
0593 ;                         D39B
0594 READ1  LDX #12            D39B A2 0C
0595        JSR MSOUT1         D39D 20 4D D1
0596 ;                         D3A0
0597 READ2  LDX #0             D3A0 A2 00
0598        JSR MSOUT1         D3A2 20 4D D1
0599 ;                         D3A5
0600 READ3  LDX #18            D3A5 A2 12
0601        LDA #LIN04         D3A7 A9 80
0602        JSR WRD1S          D3A9 20 6C D1
0603 ;                         D3AC
0604 READ4  LDX #22            D3AC A2 16
0605        LDA #LIN05         D3AE A9 A0
0606        JSR WRD1S          D3B0 20 6C D1
0607 ;                         D3B3
0608 READ5  LDX #14            D3B3 A2 0E
0609        JSR MSOUT1         D3B5 20 4D D1
0610 ;                         D3B8
0611 READ6  LDX #20            D3B8 A2 14
0612        JSR MSOUT1         D3BA 20 4D D1
0613 ;                         D3BD
0614 ;OUTPUT THE COMMAND LINE  D3BD
0615 ;                         D3BD
0616 JCMNDR CLI                D3BD 58
0617        LDA #3             D3BE A9 03
0618        STA ICURS+1        D3C0 85 0B
0619        LDA #LIN07         D3C2 A9 E0
0620        JSR CLRLN0         D3C4 20 76 D1
0621        JSR CMLIN0         D3C7 20 91 D1
0622        SEI                D3CA 78
0623 ;                         D3CB
0624        JSR FINDFL         D3CB 20 D3 D1
0625        JSR FILXT          D3CE 20 D9 D1
0626        JSR INIT1          D3D1 20 E8 D1
0627 ;                         D3D4
0628        LDA #$6            D3D4 A9 06
0629        STA RBUF           D3D6 8D FE BB
0630        LDA #$7            D3D9 A9 07
0631        STA RREC           D3DB 8D FF BB
0632        JSR ERROR1         D3DE 20 F7 D1
0633        JSR INIIO          D3E1 20 06 A8
0634        JSR OPENR          D3E4 20 09 A8
0635        JSR RECRDA         D3E7 20 6B D2
0636 EXIT   RTS                D3EA 60
0637 ;                         D3EB
0638 ERROR2 LDX #$FF           D3EB A2 FF
0639        TXS                D3ED 9A
0640        JSR CLRTP0         D3EE 20 86 D1
0641 ;                         D3F1
```

```
0642          LDA #$2
0643          STA ICURS+1
0644          LDX #$0
0645          LDA #LIN02
0646          JSR WRD1S
0647 ;
0648          LDX #24
0649          JSR MSOUT1
0650 ;
0651          LDX #28
0652          JSR MSOUT1
0653 ;
0654          LDX #30
0655          LDA #LIN04
0656          JSR WRD1S
0657          LDA #$3
0658          STA ICURS+1
0659 ;
0660          LDA #LIN05
0661          JSR CLRLNO
0662 ;
0663          CLI
0664 POLERR   JSR JPLKB
0665          LDA ICHAR
0666          CMP #'R
0667          BNE POLERR
0668          SEI
0669          LDY PRAMSV
0670 REST1    LDA PRAMSV,Y
0671          STA RRCBUF,Y
0672          DEY
0673          BPL REST1
```

```
D3F1 A9 02
D3F3 85 0B
D3F5 A2 00
D3F7 A9 40
D3F9 20 6C D1
D3FC
D3FC A2 18
D3FE 20 4D D1
D401
D401 A2 1C
D403 20 4D D1
D406
D406 A2 1E
D408 A9 80
D40A 20 6C D1
D40D A9 03
D40F 85 0B
D411
D411 A9 A0
D413 20 76 D1
D416
D416 58
D417 20 1D F8
D41A A5 01
D41C C9 52
D41E D0 F7
D420 78
D421 AC 00 01
D424 B9 00 01
D427 99 00 07
D42A 88
D42B 10 F7
```

```
0674          LDA #$FF
0675          STA PENFLG
0676          JSR RECRDB
0677          JMP $C938
0678 ;
0679 ;TWO PASS PATCHES
0680 ;
0681          ORG $C1F7
0682          LDX #$12
0683 ;
0684          ORG $C1FC
0685          CMP TABL1,X
0686 ;
0687          ORG $C204
0688          CMP TABL2,X
0689 ;
0690          ORG $C216
0691          LDA TABL3,X
0692 ;
0693          ORG $C21B
0694          LDA TABL4,X
0695 ;
0696          ORG $C220
0697          LDA TABL5,X
0698 ;
0699          ORG $CFDC
0700          BYT $20
0701 ;
0702 ;******************
0703 ;** END OF 'TPAUG'
0704 ;******************
```

```
D42D A9 FF
D42F 85 3A
D431 20 72 D2
D434 4C 38 C9
D437
D437
D437
D437
C1F7 A2 12
C1F9
C1F9
C1FC DD 00 D0
C1FF
C1FF
C204 DD 20 D0
C207
C207
C216 BD 40 D0
C219
C219
C21B BD 60 D0
C21E
C21E
C220 BD 80 D0
C223
C223
CFDC 20
CFDD
CFDD
CFDD
CFDD
```

MICRON PROBLEMS

Symtoms

1. For about 1 minute after switching on
   RESET gave
   PRINT ERROR
   TANBUG

Otherwise system was dead to all keys except RESET.

2. After about a minute or so, commands such as
   M L etc.
were OK but 'BAS' frequently gave a locked up condition, RESET was the only key to give a response.

3. After a further few minutes BASIC could be entered and commands typed in but rhe program was stored in a garbled way.

4. At this stage the 'COPY' command copied a pattern such as OO,FF satisfactorily in the first 1K of memory (on the Microtan board) but the pattern was changed to OO,OF in the Tanex memory region.

5. A test program gave lock up when 2 write instructions were placed within about 10 cycles but seemed OK if the write cycles were widely separated.

Fault

A faulty 2114 in N12(Tanex) caused the problems. This chip presumably loaded the MS data bus bits accounting for the COPY behaviour and must have affected memory usage for BASIC.

Method of tracing fault

A short test program, pattern COPY and BASIC command entry were used.

1. Data buffer ICs were swapped - fault stayed on
2. VIAs (6522) and AC1As(6551) removed - fault stayed on
3. Memory ICs were removed and replaced one at a time. Fault located.

Assembler Source Tape Catalogue

This program produces a catalogue listing of a tape containing files created by the Microtanic two pass assembler. The listing contains one line for each file found. It consists of the following information'

a) The filename
b) The number of lines of source in the file
c) The total number of bytes of store that would be occupied by the source.
d) The number of blocks (or screens) of source that were read in total
e) The number of blocks of the source which had read errors in them.

Before the program is run, the XBUG F or C commands should be used to select the speed at which the files were recorded. Im addition if a printed catalogue listing is required, then the TAN-BUG CTRL P or V commands should be used.

To stop the program use the 'Break' key.

How It Works

The program uses routines in both XBUG and the assembler itself, so both of these EPROMS should be installed in their normal positions.

The program starts by looking for a file header block, and then begins processing the source which is placed in the top half of the screen by the assembler read routine. The processing continues with successive screens until an end of file block is found. The line of file statistics is then printed.

All the actual reading is performed by the assembler routine used.

Restrictions

The program prints out the statistics information when it identifies the end of file block. Your printer must be capable of taking the whole line in the gap between files otherwise the program may miss the next file. The same applies to the heading which is output when the program is started.

An example output is given at the end of the assembly listing.

```
0001 ;                                0400
0002 ;*************************0400
0003 ;      C A T A L O G      0400
0004 ;*************************0400
0005 ;                                0400
0006 ; (C) A.S. Murphy 1983.    0400
0007 ;                                0400
0008 ;This program produces a   0400
0009 ;catalog listing of all    0400
0010 ;the files on a tape which0400
0011 ;have been produced by the0400
0012 ;Microtanic two pass       0400
0013 ;assembler.                0400
0014 ;This program requires     0400
0015 ;the assembler EPROM to be0400
0016 ;installed.                0400
0017 ;As this uses itself uses 0400
0018 ;XBUG cassette handing     0400
0019 ;routines, this must also 0400
0020 ;be installed.             0400
0021 ;                         0400
0022 ;To exit from the program 0400
0023 ;use the RESET key.        0400
0024 ;If output to a printer is0400
0025 ;selected before running   0400
0026 ;the program output will   0400
0027 ;be printed, but your      0400
0028 ;printer must be capable   0400
0029 ;of taking a lines data    0400
0030 ;in the inter file gap.    0400
0031 ;                         0400
0032 ;       PAGE ZERO         0400
0033 ;                         0400
0034 NBLKS  EPZ $80             0400
0035 NLINES EPZ $82             0400
0036 NBYTES EPZ $84             0400
0037 ERRS   EPZ $86             0400
0038 ;                          0400
0039 BYTCNT EPZ $1D             0400
0040 COPL   EPZ $1E             0400
0041 COPH   EPZ $1F             0400
0042 SPEED  EPZ $50             0400
0043 ;                          0400
0044 ;External Subroutines      0400
0045 ;                          0400
0046 OUTALL EQU $F80E           0400
0047 JHXPN  EQU $F81A           0400
0048 INIT   EQU $F4AA           0400
0049 RDBLK  EQU $C103           0400
0050 ;                          0400
0051 ;This is the programs      0400
0052 ;entry point.              0400
0053 ;Cassette speed should be  0400
0054 ;set using XBUG commands   0400
0055 ;before starting.          0400
0056 ;                          0400
0057 CATLOG SEI                 0400 78
0058 ;                          0401
0059 ;Find out tape speed for   0401
0060 ;heading.                  0401
0061 ;                          0401
0062        LDA SPEED           0401 A5 50
0063        AND #1              0403 29 01
0064        TAX                 0405 AA
0065        LDY #3              0406 A0 03
0066 SPED1  LDA SPDTXT,X        0408 BD 94 04
0067        STA SPDMSG,Y        040B 99 26 04
0068        INX                 040E E8
0069        INX                 040F E8
0070        DEY                 0410 88
0071        BPL SPED1           0411 10 F5
0072 ;Print heading line        0413
0073        JSR MESAGE          0413 20 09 05
0074        BYT $D,'T,'a,'p     0416 0D 54 61 70
0075        BYT 'e,' ,'C,'a     041A 65 20 43 61
0076        BYT 't,'a,'l,'o     041E 74 61 6C 6F
0077        BYT 'g,' ,'-,'      0422 67 20 2D 20
0078 SPDMSG BYT 'X,'X,'X,'X     0426 58 58 58 58
0079        BYT $D              042A 0D
0080        BYT $D,'F,'i,'l     042B 0D 46 69 6C
0081        BYT 'e,' ,' ,'N     042F 65 20 20 4E
0082        BYT 'o,'.,'L,'i     0433 6F 2E 4C 69
0083        BYT 'n,'e,'s,'      0437 6E 65 73 20
0084        BYT 'B,'y,'t,'e     043B 42 79 74 65
0085        BYT 's,' ,'B,'l     043F 73 20 42 6C
0086        BYT 'o,'c,'k,'s     0443 6F 63 6B 73
0087        BYT ' ,'E,'r,'r     0447 20 45 72 72
0088        BYT 's,$D,0         044B 73 0D 00
0089 ;                          044E
0090        JSR INIT ;Init VIA  044E 20 AA F4
0091 ;                          0451
0092 ;Clear stats area          0451
0093 ;                          0451
0094 NXTFIL LDX #6              0451 A2 06
0095        LDA #0              0453 A9 00
0096 CAT1   STA NBLKS,X         0455 95 80
0097        DEX                 0457 CA
0098        BPL CAT1            0458 10 FB
0099 ;                          045A
0100 ;Search for file header    045A
0101 ;                          045A
0102 CAT2   JSR RDBLK           045A 20 03 C1
0103        BNE CAT2            045D D0 FB
0104        JSR CHKHED          045F 20 EE 04
0105        BCC CAT2            0462 90 F6
0106 ;                          0464
0107 ;Read file contents and    0464
0108 ;calc stats                0464
0109 ;                          0464
0110 NXTBLK JSR RDBLK           0464 20 03 C1
0111        BNE RDERR           0467 D0 33
```

```
0112 ;Check for end of file    0469
0113         LDX #$3E          0469 A2 3E
0114         TXA               046B 8A
0115 EOF1    CMP $200,X        046C DD 00 02
0116         BNE NOTEOF        046F D0 35
0117         DEX               0471 CA
0118         BPL EOF1          0472 10 F8
0119         JSR MESAGE        0474 20 09 05
0120 FNAME   BYT 'F,'N,'A,'M   0477 46 4E 41 4D
0121         BYT 'E,'X         047B 45 58
0122         BYT ' ,'          047D 20 20
0123         BYT $83,$82;Lines 047F 83 82
0124         BYT ' ,' ,' ,'    0481 20 20 20 20
0125         BYT $85,$84;Bytes 0485 85 84
0126         BYT ' ,'          0487 20 20
0127         BYT $81,$80;Blocks 0489 81 80
0128         BYT ' ,' ,'       048B 20 20 20
0129         BYT $86;Errors    048E 86
0130         BYT $D,0          048F 0D 00
0131         JMP NXTFIL        0491 4C 51 04
0132 ;                         0494
0133 ;Tape speed message text  0494
0134 ;                         0494
0135 SPDTXT BYT 't,'s,'s,'t    0494 74 73 73 74
0136         BYT 'a,'u,'F,'C   0498 61 75 46 43
0137 ;                         049C
0138 ;Block read error         049C
0139 ;                         049C
0140 RDERR   LDA ERRS          049C A5 86
0141         JSR AD1BCD        049E 20 E8 04
0142         STA ERRS          04A1 85 86
0143         JMP ADDBLK        04A3 4C D5 04
0144 ;                         04A6
0145 ;Good block. Calc number  04A6
0146 ;lines and bytes in block 04A6
0147 ;                         04A6
0148 NOTEOF LDY #0             04A6 A0 00
0149 NOTEF2 LDA NLINES         04A8 A5 82
0150         JSR AD1BCD        04AA 20 E8 04
0151         STA NLINES        04AD 85 82
0152         BCC NOTEF1        04AF 90 07
0153         LDA NLINES+1      04B1 A5 83
0154         JSR AD1BCD        04B3 20 E8 04
0155         STA NLINES+1      04B6 85 83
0156 ;                         04B8
0157 ;Find start of next line  04B8
0158 ;                         04B8
0159 NOTEF1 CLC                04B8 18
0160         TYA               04B9 98
0161         ADC $200,Y        04BA 79 00 02
0162         BCS ENDBLK        04BD B0 0B
0163         TAY               04BF A8
0164         INY               04C0 C8
0165         CPY BYTCNT        04C1 C4 1D
0166         BEQ ENDBLK        04C3 F0 05
0167         BCS ENDBLK        04C5 B0 03
0168         JMP NOTEF2        04C7 4C A8 04
0169 ;                         04CA
0170 ;End of block             04CA
0171 ;                         04CA
0172 ENDBLK LDA BYTCNT         04CA A5 1D
0173         CLC               04CC 18
0174         ADC NBYTES        04CD 65 84
0175         STA NBYTES        04CF 85 84
0176         BCC ADDBLK        04D1 90 02
0177         INC NBYTES+1      04D3 E6 85
0178 ;                         04D5
0179 ADDBLK LDA NBLKS          04D5 A5 80
0180         JSR AD1BCD        04D7 20 E8 04
0181         STA NBLKS         04DA 85 80
0182         BCC ENDBL2        04DC 90 07
0183         LDA NBLKS+1       04DE A5 81
0184         JSR AD1BCD        04E0 20 E8 04
0185         STA NBLKS+1       04E3 85 81
0186 ENDBL2 JMP NXTBLK         04E5 4C 64 04
0187 ;                         04E8
0188 ;Add 1 to BCD number in   04E8
0189 ;accumulator.             04E8
0190 ;                         04E8
0191 AD1BCD CLC                04E8 18
0192         SED               04E9 F8
0193         ADC #1            04EA 69 01
0194         CLD               04EC D8
0195         RTS               04ED 60
0196 ;                         04EE
0197 ;Check if block is a file 04EE
0198 ;header and extract file- 04EE
0199 ;name if it is.           04EE
0200 ;Returns Carry set if the 04EE
0201 ;block is a file header    04EE
0202 ;                         04EE
0203 CHKHED LDX #$1F           04EE A2 1F
0204         LDA #'*           04F0 A9 2A
0205 CHKH2   CMP $200,X        04F2 DD 00 02
0206         BNE CHKH1         04F5 D0 10
0207         DEX               04F7 CA
0208         CPX #5            04F8 E0 05
0209         BNE CHKH2         04FA D0 F6
0210 ;Copy filename            04FC
0211 ;                         04FC
0212 CHKH3   LDA $200,X        04FC BD 00 02
0213         STA FNAME,X       04FF 9D 77 04
0214         DEX               0502 CA
0215         BPL CHKH3         0503 10 F7
0216         SEC               0505 38
0217         RTS               0506 60
0218 ;Not header block         0507
0219 CHKH1   CLC               0507 18
0220         RTS               0508 60
0221 ;                         0509
0222 ;String print routine     0509
0223 ;                         0509
0224 ;This routine prints the  0509
0225 ;string following the     0509
0226 ;instruction that called  0509
0227 ;it.                      0509
0228 ;If a byte has its most   0509
0229 ;significant bit set, then0509
0230 ;the contents of that page0509
0231 ;zero location is printed 0509
0232 ;as a hex number.         0509
0233 ;The string is terminated 0509
0234 ;by a zero byte.          0509
0235 ;The routine returns to   0509
0236 ;the instruction following0509
0237 ;the zero byte.           0509
0238 MESAGE PLA                0509 68
0239         STA COPL          050A 85 1E
0240         PLA               050C 68
0241         STA COPH          050D 85 1F
0242         LDY #1            050F A0 01
0243 MES1    LDA (COPL),Y      0511 B1 1E
0244         BEQ EOM           0513 F0 11
0245         BMI HEXOUT        0515 30 06
0246         JSR OUTALL        0517 20 0E F8
0247 MES2    INY               051A C8
0248         BNE MES1          051B D0 F4
0249 ;                         051D
0250 ;Print contents of page   051D
0251 ;zero location as hex     051D
0252 ;                         051D
0253 HEXOUT TAX                051D AA
0254         LDA $0,X          051E B5 00
0255         JSR JHXPN         0520 20 1A F8
0256         JMP MES2          0523 4C 1A 05
0257 ;End of message           0526
0258 EOM     TYA               0526 98
0259         CLC               0527 18
0260         ADC COPL          0528 65 1E
0261         STA COPL          052A 85 1E
0262         BCC EOM1          052C 90 02
0263         INC COPH          052E E6 1F
0264 EOM1    LDA COPH          0530 A5 1F
0265         PHA               0532 48
0266         LDA COPL          0533 A5 1E
0267         PHA               0535 48
0268         RTS               0536 60
```

```
NBLKS = 0080 NLINES= 0082    FNAME = 0477 SPDTXT= 0494
NBYTES= 0084 ERRS  = 0086    RDERR = 049C NOTEOF= 04A6
BYTCNT= 001D COPL  = 001E    NOTEF2= 04A8 NOTEF1= 04B8
COPH  = 001F SPEED = 0050    ENDBLK= 04CA ADDBLK= 04D5
OUTALL= F80E JHXPN = F81A    ENDBL2= 04E5 AD1BCD= 04E8
INIT  = F4AA RDBLK = C103    CHKHED= 04EE CHKH2 = 04F2
CATLOG= 0400 SPED1 = 0408    CHKH3 = 04FC CHKH1 = 0507
SPDMSG= 0426 NXTFIL= 0451    MESAGE= 0509 MES1  = 0511
CAT1  = 0455 CAT2  = 045A    MES2  = 051A HEXOUT= 051D
NXTBLK= 0464 EOF1  = 046C    EOM   = 0526 EOM1  = 0530
```

```
F
6400
Tape Catalog - Fast

File  No.Lines Bytes Blocks Errs
 PLTLN  0283   0E8C  0015   00
 GRPRT  0205   0C32  0013   00
 MEMDP  0095   0401  0005   00
 CATLG  0268   0ECC  0016   00
 ASLST  0148   08D3  0010   00
 BANGN  0866   2F98  0050   00
 BANER  0288   1258  0020   00
```

Character Width 1 ,Height 1 Gap 1

Character Width 2 ,Height 1 Gap 1

Banner Printer

The routine described here is a utility for creating very bold print for headings etc, as used by 'spoolers' on larger computer systems for the identification of print jobs.

This routine is somewhat more versatile as it is able to produce characters of many different sizes. It used the graphics capability of an Epson MX80 printer to achieve this.

The number of characters that you can fit on a line depends upon the width of the characters to be produced, and the space between each character.

How to Use the Routine

The routine itself is not stand-alone, and needs some form of driver to make it produce anything. The simplist you could get away with is something like that shown in the sample driver listing. This will produce characters with the default or last settings of width, height and inter-character gap.

To use the routine STRPTR (A3/A4) must contain the address of the string you wish printed, which must be terminated with a zero byte.

If the string you ask for contains any character the routine cannot print, it will substitute a space in its place. The list of valid characters may be found in the listing at lines 81-96. The routine makes no attempt to ensure that the line doesn't overflow - that's up to you. Nor does it establish any print modes or output any blank lines after completion of the print task.

The routine has three variables to control the character size and aspect ratio (Ratio of height to width). These are:-

Width, Height and Inter-Character gap.

These variable should be set up by altering the required locations before calling the routine. These may be found on lines 97-102 of the listing.

The following table shows the number of characters that will fit on one line for various values of width.

Character Width	Max Chars.
1	13
2	6
3	4
4	3
5	2
6	2
7 - 13	1

This table assumes that the inter-character gap is set at 1. Use of the Epson's emphasised print mode greatly enhances the solidity of the characters.

How it Works

The data for each character is held as five consecutive bytes of the table CHRTAB. Each byte contains the information for one column of the character.

e.g. 'G' is held as

	3	4	4	5	3
	E	1	1	1	2
Bit					
0		X	X	X	
1	X				X
2	X				
3	X				
4	X			X	X
5	X				X
6		X	X	X	
7	Not Used				

Each character is output as seven rows, each row being made up of each of the five columns of all the characters in the string and their inter-character gaps.

To add extra characters of your own, you must modify the data tables to append the new character data. Alternatively you could just replace an existing one.

The steps required to add new characters are as follows:

a) Relocate the routines surrounding the tables to make room for the extra data.

b) Add the extra symbol that will cause your character to be printed to the end of CHRLST, and increase the value in line 56 by I for each new character.

c) Finally add the five bytes describing your character to the end of CHRTAB.

It should be possible to use this routine with an OK1 Microline 80 if the value DF in line 129 is changed to BF.

A couple of examples of the 'Smaller' characters are given below.

```
0001 ;                                  0400
0002 ;**************************0400
0003 ;      B A N N E R         0400
0004 ;      P R I N T E R       0400
0005 ;**************************0400
0006 ;                                  0400
0007 ; (C) A.S. Murphy 1983.    0400
0008 ;                                  0400
0009 ;This package is capable   0400
0010 ;of producing banner       0400
0011 ;characters the size of    0400
0012 ;which are governed by the0400
0013 ;following variables:      0400
0014 ;INCHGP - Inter-char gap   0400
0015 ;CHWDTH - Character width 0400
0016 ;CHRHGT - Character height0400
0017 ;                                  0400
0018 ;The text to be printed in0400
0019 ;this way should be        0400
0020 ;pointed to by STRPTR & be0400
0021 ;terminated by a zero byte0400
0022 ;The maximum length of the0400
0023 ;string is determined by   0400
0024 ;the character size. It is0400
0025 ;up to the user to ensure 0400
0026 ;that its not too long.    0400
0027 ;                                  0400
0028 ;The routines output the   0400
0029 ;characters to the printer0400
0030 ;by using the TANBUG print0400
0031 ;routines.  The printer is0400
0032 ;assumed to be an EPSON.   0400
0033 ;                                  0400
0034 ROW     EPZ $A0                    0400
0035 COLCNT  EPZ $A1                    0400
0036 TEMP    EPZ $A2                    0400
0037 STRPTR  EPZ $A3 ;FOR 2             0400
0038 ROWCNT  EPZ $A5                    0400
0039 OCHAR   EPZ $2                     0400
0040 ;                                  0400
0041 OUTPAR  EQU $F803                  0400
0042 ;                                  0400
0043 ;Print expanded text               0400
0044 ;string.  Takes the string0400
0045 ;pointed to by STRPTR &    0400
0046 ;causes it to be printed   0400
0047 ;as a banner heading.      0400
0048 ;                                  0400
0049 BANNER LDA #1                      0400 A9 01
0050         STA ROW                    0402 85 A0
0051 BAN1    LDA CHRHGT;Height          0404 AD 67 04
0052         STA ROWCNT                 0407 85 A5
0053 BAN6    LDY #0                     0409 A0 00
0054 BAN3    LDA (STRPTR),Y             040B B1 A3
0055         BEQ BAN2;End string040D F0 11
0056         LDX #49;Len let tab040F A2 31
0057 BAN5    CMP CHRLST,X               0411 DD 33 04
0058         BEQ BAN4                   0414 F0 03
0059         DEX                        0416 CA
0060         BNE BAN5                   0417 D0 F8
0061 ;Outputs space if cant     0419
0062 ;find character in table   0419
0063 BAN4    TXA                        0419 8A
0064         JSR PRTROW                 041A 20 68 04
0065         INY                        041D C8
0066         BNE BAN3;Next char         041E D0 EB
0067 ;                                  0420
0068 ;End of a line                     0420
0069 ;                                  0420
0070 BAN2    LDA #13 ;<CR>              0420 A9 0D
0071         JSR PRTCH                  0422 20 9B 05
0072         LDA #10 ;<LF>              0425 A9 0A
0073         JSR PRTCH                  0427 20 9B 05
0074 ;Repeat for row height     042A
0075         DEC ROWCNT                 042A C6 A5
0076         BNE BAN6                   042C D0 DB
0077 ;Next row of string        042E
0078         ASL ROW                    042E 06 A0
0079         BPL BAN1                   0430 10 D2
0080         RTS                        0432 60
0081 ;                                  0433
0082 ;List of valid characters  0433
0083 ;                                  0433
0084 CHRLST  BYT ' ,'A,'B,'C            0433 20 41 42 43
0085         BYT 'D,'E,'F,'G            0437 44 45 46 47
0086         BYT 'H,'I,'J,'K            043B 48 49 4A 4B
0087         BYT 'L,'M,'N,'O            043F 4C 4D 4E 4F
0088         BYT 'P,'Q,'R,'S            0443 50 51 52 53
0089         BYT 'T,'U,'V,'W            0447 54 55 56 57
0090         BYT 'X,'Y,'Z,'$            044B 58 59 5A 24
0091         BYT '.,'?,'0,'1            044F 2E 3F 30 31
0092         BYT '2,'3,'4,'5            0453 32 33 34 35
0093         BYT '6,'7,'8,'9            0457 36 37 38 39
0094         BYT ',,'',',!,'(           045B 2C 27 21 28
0095         BYT '),'#,'/,':            045F 29 2A 2F 3A
0096         BYT '-,'+                  0463 2D 2B
0097 ;                                  0465
0098 ;Character statistics      0465
0099 ;                                  0465
0100 INCHGP  BYT 1 ;Inter ch gap0465 01
0101 CHWDTH  BYT 1 ;char width  0466 01
0102 CHRHGT  BYT 1;Char height  0467 01
0103 ;                                  0468
0104 ;This routine outputs a    0468
0105 ;row of the character      0468
0106 ;specified by the offset   0468
0107 ;passed in the accumulator0468
0108 ;The row to output is      0468
0109 ;specified by ROW          0468
0110 ;                                  0468
0111 PRTROW  TAX                        0468 AA
0112         TYA                        0469 98
0113         PHA ;Save Y reg            046A 48
0114         TXA                        046B 8A
0115 ;Calculate table position  046C
0116         STA TEMP                   046C 85 A2
0117         ASL @                      046E 0A
0118         ASL @                      046F 0A
0119         CLC                        0470 18
0120         ADC TEMP                   0471 65 A2
0121         TAX                        0473 AA
0122 ;Setup column count        0474
0123         LDA #5                     0474 A9 05
0124         STA COLCNT                 0476 85 A1
0125 ;Column output loop        0478
0126 PRTRW3  LDA CHRTAB,X               0478 BD A1 04
0127         AND ROW                    047B 25 A0
0128         BEQ PRTRW1                 047D F0 04
0129         LDA #$DF;Block             047F A9 DF
0130         BNE PRTRW2                 0481 D0 02
0131 PRTRW1  LDA #32                    0483 A9 20
0132 PRTRW2  LDY CHWDTH                 0485 AC 66 04
0133 PRTRW4  JSR PRTCH                  0488 20 9B 05
0134         DEY                        048B 88
0135         BNE PRTRW4                 048C D0 FA
0136 ;                                  048E
0137 ;Next column               048E
0138 ;                                  048E
0139         INX                        048E E8
0140         DEC COLCNT                 048F C6 A1
0141         BNE PRTRW3                 0491 D0 E5
0142 ;Output inter-char gap     0493
0143         LDA #32                    0493 A9 20
0144         LDY INCHGP                 0495 AC 65 04
0145 PRTRW5  JSR PRTCH                  0498 20 9B 05
0146         DEY                        049B 88
0147         BNE PRTRW5                 049C D0 FA
0148 ;Restore Y reg and exit    049E
0149         PLA                        049E 68
0150         TAY                        049F A8
0151         RTS                        04A0 60
0152 ;                                  04A1
0153 ;Character data table      04A1
0154 ;                                  04A1
0155 ;This table has five bytes04A1
0156 ;to describe each banner   04A1
0157 ;character. Each byte      04A1
0158 ;holds the data for one    04A1
0159 ;column of the character.  04A1
0160 ;The least significant bit04A1
0161 ;representing the top row  04A1
0162 ;of the character.         04A1
0163 ;The most significant bit  04A1
0164 ;is unused.                04A1
0165 ;                                  04A1
0166 CHRTAB  BYT $00,$00,$00;SP 04A1 00 00 00
0167         BYT $00,$00                04A4 00 00
0168         BYT $7E,$09,$09;A          04A6 7E 09 09
0169         BYT $09,$7E                04A9 09 7E
0170         BYT $7F,$49,$49;B          04AB 7F 49 49
```

```
0171          BYT $49,$36          04AE 49 36
0172          BYT $3E,$41,$41;C    04B0 3E 41 41
0173          BYT $41,$22          04B3 41 22
0174          BYT $7F,$41,$41;D    04B5 7F 41 41
0175          BYT $41,$3E          04B8 41 3E
0176          BYT $7F,$49,$49;E    04BA 7F 49 49
0177          BYT $49,$41          04BD 49 41
0178          BYT $7F,$09,$09;F    04BF 7F 09 09
0179          BYT $09,$01          04C2 09 01
0180          BYT $3E,$41,$41;G    04C4 3E 41 41
0181          BYT $51,$32          04C7 51 32
0182          BYT $7F,$08,$08;H    04C9 7F 08 08
0183          BYT $08,$7F          04CC 08 7F
0184          BYT $00,$41,$7F;I    04CE 00 41 7F
0185          BYT $41,$00          04D1 41 00
0186          BYT $20,$40,$40;J    04D3 20 40 40
0187          BYT $40,$3F          04D6 40 3F
0188          BYT $7F,$08,$08;K    04D8 7F 08 08
0189          BYT $14,$63          04DB 14 63
0190          BYT $7F,$40,$40;L    04DD 7F 40 40
0191          BYT $40,$40          04E0 40 40
0192          BYT $7F,$02,$04;M    04E2 7F 02 04
0193          BYT $02,$7F          04E5 02 7F
0194          BYT $7F,$06,$08;N    04E7 7F 06 08
0195          BYT $30,$7F          04EA 30 7F
0196          BYT $3E,$41,$41;O    04EC 3E 41 41
0197          BYT $41,$3E          04EF 41 3E
0198          BYT $7F,$09,$09;P    04F1 7F 09 09
0199          BYT $09,$06          04F4 09 06
0200          BYT $3E,$41,$51;Q    04F6 3E 41 51
0201          BYT $61,$3E          04F9 61 3E
0202          BYT $7F,$09,$19;R    04FB 7F 09 19
0203          BYT $29,$46          04FE 29 46
0204          BYT $26,$49,$49;S    0500 26 49 49
0205          BYT $49,$32          0503 49 32
0206          BYT $01,$01,$7F;T    0505 01 01 7F
0207          BYT $01,$01          0508 01 01
0208          BYT $3F,$40,$40;U    050A 3F 40 40
0209          BYT $40,$3F          050D 40 3F
0210          BYT $07,$38,$40;V    050F 07 38 40
0211          BYT $38,$07          0512 38 07
0212          BYT $7F,$20,$10;W    0514 7F 20 10
0213          BYT $20,$7F          0517 20 7F
0214          BYT $63,$14,$08;X    0519 63 14 08
0215          BYT $14,$63          051C 14 63
0216          BYT $03,$04,$78;Y    051E 03 04 78
0217          BYT $04,$03          0521 04 03
0218          BYT $61,$51,$49;Z    0523 61 51 49
0219          BYT $45,$43          0526 45 43
0220          BYT $26,$49,$7F;$    0528 26 49 7F
0221          BYT $49,$32          052B 49 32
0222          BYT $00,$00,$40;.    052D 00 00 40
0223          BYT $00,$00          0530 00 00
0224          BYT $02,$01,$51;?    0532 02 01 51
0225          BYT $09,$06          0535 09 06
0226          BYT $3E,$71,$49;0    0537 3E 71 49
0227          BYT $47,$3E          053A 47 3E
0228          BYT $00,$42,$7F;1    053C 00 42 7F
0229          BYT $40,$00          053F 40 00
0230          BYT $46,$61,$51;2    0541 46 61 51
0231          BYT $49,$46          0544 49 46
0232          BYT $22,$41,$49;3    0546 22 41 49
0233          BYT $49,$36          0549 49 36
0234          BYT $0F,$08,$08;4    054B 0F 08 08
0235          BYT $08,$7F          054E 08 7F
0236          BYT $37,$45,$45;5    0550 37 45 45
0237          BYT $45,$39          0553 45 39
0238          BYT $3E,$49,$49;6    0555 3E 49 49
0239          BYT $49,$32          0558 49 32
0240          BYT $01,$71,$09;7    055A 01 71 09
0241          BYT $05,$03          055D 05 03
0242          BYT $36,$49,$49;8    055F 36 49 49
0243          BYT $49,$36          0562 49 36
0244          BYT $26,$49,$49;9    0564 26 49 49
0245          BYT $49,$3E          0567 49 3E
0246          BYT $00,$40,$30;,    0569 00 40 30
0247          BYT $00,$00          056C 00 00
0248          BYT $00,$04,$02;'    056E 00 04 02
0249          BYT $01,$00          0571 01 00
0250          BYT $00,$00,$5F;!    0573 00 00 5F
0251          BYT $00,$00          0576 00 00
0252          BYT $1C,$22,$41;(    0578 1C 22 41
0253          BYT $00,$00          057B 00 00
0254          BYT $00,$00,$41;)    057D 00 00 41
0255          BYT $22,$1C          0580 22 1C
0256          BYT $14,$08,$3E;*    0582 14 08 3E
0257          BYT $08,$14          0585 08 14
0258          BYT $60,$10,$08;/    0587 60 10 08
0259          BYT $04,$03          058A 04 03
0260          BYT $00,$00,$44;:    058C 00 00 44
0261          BYT $00,$00          058F 00 00
0262          BYT $08,$08,$08;-    0591 08 08 08
0263          BYT $08,$08          0594 08 08
0264          BYT $08,$08,$3E;+    0596 08 08 3E
0265          BYT $08,$08          0599 08 08
0266 ;                             059B
0267 ;                             059B
0268 ;Print character routine      059B
0269 ;This routine uses the        059B
0270 ;TANBUG parallel printer      059B
0271 ;routine to output the        059B
0272 ;character in ACC. All        059B
0273 ;registers are preserved.     059B
0274 ;                             059B
0275 PRTCH   STA TEMP              059B 85 A2
0276         TYA                   059D 98
0277         PHA                   059E 48
0278         TXA                   059F 8A
0279         PHA                   05A0 48
0280         LDA TEMP              05A1 A5 A2
0281         STA OCHAR             05A3 85 02
0282         JSR OUTPAR            05A5 20 03 F8
0283         PLA                   05A8 68
0284         TAX                   05A9 AA
0285         PLA                   05AA 68
0286         TAY                   05AB A8
0287         LDA TEMP              05AC A5 A2
0288         RTS                   05AE 60

ROW    = 00A0 COLCNT= 00A1
TEMP   = 00A2 STRPTR= 00A3
ROWCNT= 00A5 OCHAR = 0002
OUTPAR= F803 BANNER= 0400
BAN1   = 0404 BAN6  = 0409
BAN3   = 040B BAN5  = 0411
BAN4   = 0419 BAN2  = 0420
CHRLST= 0433 INCHGP= 0465
CHWDTH= 0466 CHRHGT= 0467
PRTROW= 0468 PRTRW3= 0478
PRTRW1= 0483 PRTRW2= 0485
PRTRW4= 0488 PRTRW5= 0498
CHRTAB= 04A1 PRTCH = 059B


0001 ;                             1500
0002 ;Minimum Drive routine        1500
0003 ;                             1500
0004 PRPUP   EQU $F800             1500
0005 STRPTR EPZ $A3                1500
0006 BANNER EQU $400               1500
0007 ;                             1500
0008 START   LDA STRADR            1500 AD 0E 15
0009         STA STRPTR            1503 85 A3
0010         LDA STRADR+1          1505 AD 0F 15
0011         STA STRPTR+1          1508 85 A4
0012         JSR BANNER            150A 20 00 04
0013         BRK                   150D 00
0014 ;                             150E
0015 STRADR WOR STRING             150E 10 15
0016 STRING BYT 'M,'I,'C,'R        1510 4D 49 43 52
0017         BYT 'O,'T,'A,'N       1514 4F 54 41 4E
0018         BYT ' ,'6,'5          1518 20 36 35
0019         BYT 0                 151B 00
```

Adding RAM to Tanex

One of the more interesting possibilities of the Microtan system is that of substituting RAM memory for some of the EPROM on TANEX. For example, if RAM is substituted for addresses SE000 to SEFFF (Sockets D3 and E2), then the BASIC TOOL-KIT, the SPACE INVADERS program, the EPROM PROGRAMM-ER BOARD Programs, and others, can all be changed from cassette or disc, instead of physically changing eproms. System Rack users can, of course, use the eprom switching board for this, but even so this involves extra expense and does not have the add0 itional advantage of providing more memory for users programs.

The modification described here allows the substitution of 6116 CMOS RAM memories in the eprom positions mentioned above. These memories have two advantages: - their pin-out is essentially identical to the 2716 eprom's which normally occupy these sockets, and also they are low power devices, which allows the possibility of battery backup to retain the memory contents while the Microtan is switched off, or in case of mains failure.

On the TANEX board, the eproms in positions D3, E2 and G2 (X-BUG) are selected by address decoder I.C.'s K1 and G1. K1 produces a signal on its pin 7 whenever a read or a write operation is done to an address in the range SE000 to SFFFF.

This signal is gated with the 6502's Read/Write signal in such a way that when a READ operation is performed within this address range, decoder G1 is enabled. G1 splits the signal up further, producing four signals which indicate read operations in address ranges SE000 to SE7FF, SE800 to SEFFF, SF000 to SF7FF, and SF800 to SFFFF. This last range is, of course, TANBUG, on the MICROTAN 65 board, while the other three correspond to the three 2716 eproms on TANEX.

On TANEX, these decoded signals then go to both the Chip Select (CS) input and the Output Enable (OE) input of the corresponding eprom, pins 18 and 20.

The first problem is that these chip select/enable signals are only produced on read operations. To substitute RAM memory, we need these signals on both read and write operations. This can be achieved simply enough by using the decoded signal on pin 7 of I.C. K1 to enable decoder G1 directly, instead of first gating it with the Read/Write line.

This introduces a new problem, however. Unless we also want to replace the X-BUG eprom with RAM, we run the risk that if at any time a program tries to write to an address in X-BUG we will have both this eprom and the data line buffers trying to put data on the data bus at the same time, which is liable to damage one or both of the components involved. The solution to this is to connect the Output Enable Pin of X-BUG's eprom to the Read/Write control line (actually to an inverted copy of this signal) so that the eprom's outputs are only active when the CPU is performing a read operation.

Now all that remains is to connect the Read/Write signal to the RAM ships, on Pin 21.

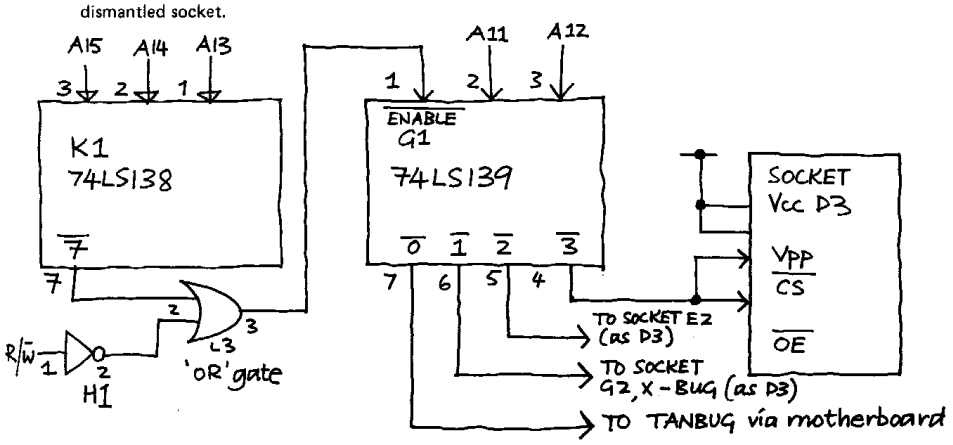
The best method I have found to do this modification is to take an I.C. socket of the type with turned pins rather than the folded sheet variety, (for example, R.S. Components stock number 402-298, 8-pin IC Socket). These are much more expensive than the normal type, but the Pins can be removed by gripping them beneath the socket with a pair of pliers, preferably thin-nosed, and forcing them out one by one through the top of the socket. Since the contact surfaces are inside the cylindrical pin they are still under tension, and the individual pins can be pushed onto the pins of an I.C. and will make a perfect connection.

To use them, measure out the length of wire necessary, using the thinnest wire possible to avoid unnecessary weight or tension, solder the wire to the spike of the socket pin, then bend up the appropriate leg of the I.C. in question, not too close to the body of the I.C., and not too small a radius bend, otherwise the leg of the I.C. might break, and then push the socket pin onto the leg. This avoids soldering directly onto the legs of the memory I.C., which could easily destroy them, and means that the memory I.C.'s can still be interchanged at a later date if desired. The only place where wires need to be soldered to an I.C. is to pick up the Read/Write signal from I.C. H1, and since this is a TTL circuit it is much less likely to suffer damage.

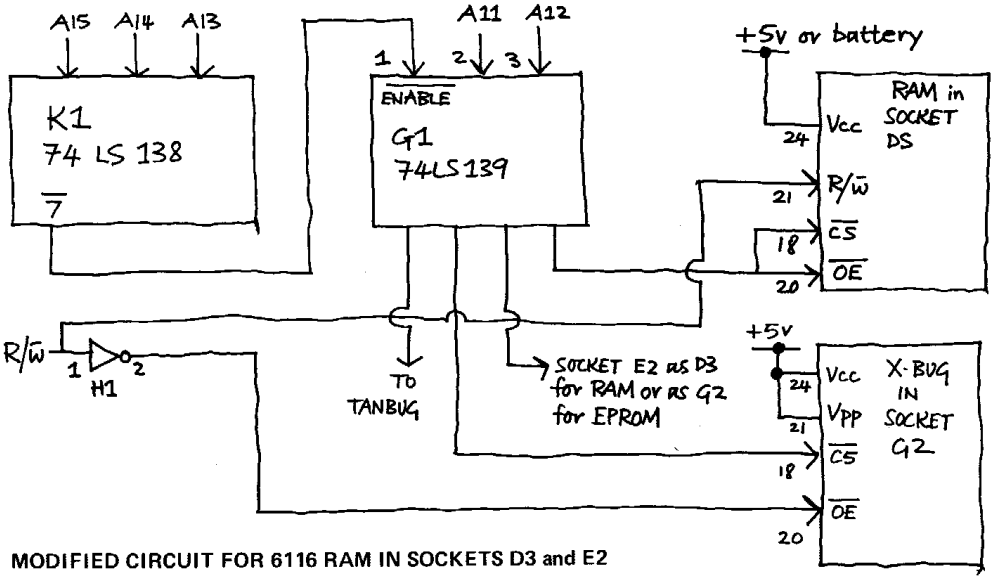
To carry out the modification, proceed as follows:

1. VERY IMPORTANT. Before starting the modification, make sure you have a copy on cassette or disc, of every eprom you have for the sockets that you are going to modify for RAM.
2. Remove I.C. K1 (74LS138) from its socket, carefully bend up pin 7 to a near-vertical position, and replace it in its socket.
3. Remove I.C. G1 (74LS139) from its socket, bend up pin 1, and replace it in its socket.
4. Take sufficient wire to run between these two bent-up pins, (bear in mind that you will want to tuck the wire down out of the way close to the board), solder a pin from the dismantled socket onto each end of the wire, and push these pins onto the two bent-up legs of these I.C.'s.
5. Remove X-BUG, very carefully bend pin 20 up, and replace X-BUG in its socket. If you are going to leave one of the other two eproms in place (The third BASIC eprom in socket D3, for example), do the same to it.
6. Take enough wire to run from pin 2 of I.C. H1 (74LS04) to the bent-up pin of X-BUG, and on from there to the other eprom if you are going to leave it too. Solder one end of this wire directly on to pin 2 of I.C. H1. You may prefer to remove this I.C. from its socket to do this, but be careful that the solder does not run down the leg of the I.C. or you won't be able to get it back into the socket. Solder the wire onto a pin or pins from the dismantled socket and push these onto the bent-up leg(s) of the eprom(s).
7. Take the RAM chip(s), bend up pin 21, and if you are going to include battery backup, bend up pin 24 as well. Insert the chip(s) in the corresponding socket(s).
8. Take enough wire to run from pin 1 of I.C. H1 to the raised pin 20 of the RAM chip(s). Solder one end of the wire directly to pin 1 of H1, and the other end to one or two pins from the dismantled socket. Push these pins onto the legs of the RAM chips.
9. If you are including battery back up, take a rechargeable battery (for example, R.S. Components part number 591-477) and fix it in a suitable position, preferably on the TANEX board itself. Silicon sealing compound can be used to glue it in place. Ensure that it will not touch anything when the TANEX board is replaced in your system.

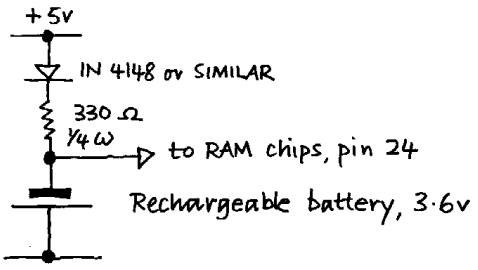
10. Wire up the battery to Ov and to 5v through a diode and a resistor of 220 or 330 ohms. Finally, connect the battery to pins 24 of the RAM chips with a wire and pins from the dismantled socket.



NORMAL EPROM (2716) SELECTION CIRCUIT ON TANEX



MODIFIED CIRCUIT FOR 6116 RAM IN SOCKETS D3 and E2



BACKUP BATTERY CIRCUIT

Ovar and Hultan updated

Graham Davies

Having used 'O' and 'H' with a number of software packages, I am now able to report on problems experienced with some of the algorithms used in these utilities. Suggested alternative algorithms are given at the end of this communication. To avoid confusion between old and new I refer to the routine being discussed only by its label. In fact these notes did contain line numbers but for the reason given, I have removed them.

The first problem relates to end-of-file marking. Within the context og 'OVAR' and 'HULTAN' as learning and development programs the use of $1A on its own is probably adequate. However, when extending the use of 'O' and 'H' to existing packages where '$1A' may already reside within the file, a more sophisticated sequence of end of text markers may be needed. This was brought home to me when I was developing TPAUG; a disc file handling patch for the Microtanic Two Pass Assembler. The problem came to light as follows:

A file had been created from 'TPAUG' but could not be recovered. In fact all efforts failed including the use of 'HULTAN' and 'LIST'. On disassembling the source listing with 'XBUG' I discovered that the very first byte of the source code for 'TPAUG' was 'CNTRL Z' alias '$1A'.

Perusal of routine LASREC in 'O', in conjunction with routine 'RECRDA' in 'H', should provide the answer to the problem. On retrieving the file from disc, 'HULTAN' found '$1A' in exactly the correct position in the first record for it to be an end-of-file marker.

Clearly, on its own, '$1A' can be inadequate for end-of-file recognition. Following Doug Deedman's recent article and discussions with Andy Michael, I have opted for two nulls followed by '$1A' for this purpose. I am still open to other suggestions particularly in relation to possible commong ground, or otherwise, between disc filing and modem use. Knowing of the efforts of these two gentlemen I am sure they are going to take us into an exciting new era of low cost communications between users. This is going to be so much more convenient thar sending discs across country.

The second problem turned up just as I was completing my article on 'TPAUG'. On recovering a file from disc, I discovered that the last ten lines, or so, of source code were missing. The missing bytes turned out to be equal to exactly ONE record; an old problem dating from 'OVAR' coming home to roost. Another maxim seems appropriate here . . . 'Let not thy apparent solution to an algorithmic problem go unchecked for surely it must return to haunt thee and diminish thy efforts and achievements in the sight of thy peers!'

Creating a file with 'OVAR' is based upon successive subtraction of #$FF (one full record) from the two byte 'DATALEFT' value. The algorithm for doing this, 'TRNSFW', is decicient in that if the low byte 'DTLFTL' contains $FF, subtracting $FF from the low byte produces 'S00' and the high byte 'DTLFTH' is decremented. This decrementing of the high byte should occur only when the low byte goes negative. A minimum of hexadecimal arithmetic will show that if the original low byte of 'DATALEFT' contains say '$FD' then the file has to be some 256D records long before the last record is lost! Statistically therefore, the routine would need considerable use before this bug turned up. In my case it took from August until late November. It should also be noted that in 'PENREC', 'LDA DTLFTL' should precede 'ADC #$FF'.

Suggested corrections to both 'O' and 'H' are given below:

1) Modifications to 'OVAR'

a) Fig.(1) shows the end-of-text address algorithm modified to recognise the alternative marker sequence; two nulls followed by '$1A$', which must be inserted by the user. Because the software packages so far encountered have provided the end-of-text address, often in the first two bytes of the file, routine 'ENDAT1' is likely to become redundant in most cases. It is not used with 'TPAUG', for example. Also listed in Fig.(1) is the algorithm for computing the number of bytes of file data. The additional lines are needed to ensure that the last byte of text data is in fact included in the penultimate record. Frankly, I am still a little unhappy about the limiting addresses and the number of inclusive bytes; particularly in relation to counting through ram and the corresponding count through the record as in 'TRNSFW' in Fig.(3).

b) Fig.(2) illustrates the updated version of 'LASREC' which outputs the new marker sequence in the last record. This is necessary in order that end-of-file may be recognised on file recovery with 'H'.

c) Fig. (3) refers to the lost record problem. It can be seen that the routine 'TRNSFW' has been modified to ensure that the high byte of 'DATALEFT' is not decremented if 'S00' is left in the low byte following a subtraction of '$FF'.

2) Modifications to 'HULTAN'

Only the routine 'RECRDA' needs to be modified and thi up-date is given in Fig.(4). The routine 'EXIT' is retained with the line 'STA (CRNTLO),Y' which places '$1A' at the end of the file in ram. Provided the file does not have a '$1A' resident elsewhere the printer routines should still operate. For greatest effect, the 'EXIT' routine should output the agreed marker sequence and the printer routine should be modified to suit.

I am now modestly confident that both 'O' and 'H' when updated, will be bugfree. My experience of applying these programs shows that 'H' comes into its own as a debugging aid whilst 'O', once its algorithms have been written into the new package, quietly recedes into the background although it could be used as test bed.

Department of Mechanical Engineering
Bolton Institute of Higher Education

```
0176   TRNSFW  LDY  YCOUNT     048E  A4  5B
0177   RCSTRT  LDX  #0         0490  A2  00
0178           SEC             0492  38
0179           LDA  DTLFTL     0493  A5  58
0180           SBC  #£FF       0495  E9  FF
0181           STA  DTLFTL     0497  85  58
0182           BNE  DATHI      0499  D0  06
0183           LDA  DTLFTH     049B  A5  59
0184           BEQ  PENREC     049D  F0  09
0185           BNE  FULREC     049F  D0  16
0186   DATH1   DEC  DTLFTH     04A1  C6  59
0187           BMI  PENREC     04A3  30  03
```

```
0188            JMP FULREC      04A5  4C  B7  04
0189  PENREC    LDA #1          04A8  A9  01
0190            STA PENFLG      04AA  85  5A
0191            CLC             04AC  18
0192            LDA DTLFTL      04AD  A5  58
0193            ADC #£FF        04AF  69  FF
0194            STA WRCBUF      04B1  8D  00  0B
0195            JMP FILL1       04B4  4C  BC  04
0196  FULREC    LDA #£FF        04B7  A9  FF
0197            STA WRCBUF      04B9  8D  00  0B
0198  FILL 1    1NX             04BC  E8
0199            LDA (CRNTLO) ,Y 04BD B1  52
0200            STA WRCBUF,X    04BF  9D  00  0B
0201            CPY #£FF        04C2  C0  FF
0202            BNE  FILL2      04C4  D0  02
0203            INC  CRNTH1     04C6  E6  53
0204  FILL2     INY             04C8  C8
0205            CPX WRCBUF      04C9  EC  00  0B
0206            BNE  FILL1      04CC  D0  EE
0207            STY  YCOUNT     04CE  84  5B
0208            JSR  RECOUT     04D0  20  18 A8
0209            LDA  PENFLG     04D3  A5  5A
0210            BNE  LASREC     04D5  D0  03
0211            JMP  TRNSFW     04D7  4C  8E  04
```

Fig; 3 Proposed TRNSFW algorithm to prevent loss of record

```
0100  RECRDA  JSR  RECIN      042B  20  15  A8
0101          LDA  RECBUF+1   042E  AD  01  08
0102          CMP  #£00       0431  C9  00
0103          BNE CARYON      0433  D0  0E
0104          LDA RECBUF+2    0435  AD  02  08
0105          CMP  #£00       0438  C9  00
0106          BNE CARYON      043A  D0  07
0107          LDA RECBUF+3    043C  AD  03  08
0108          CMP #FILEND     043F  C9  1A
0109          BEQ EXIT        0441  F0  1B
0110                          0443
0111 CARRYON  LDY  YCOUNT     0443  A4  5B
```

Fig; 4 New RECRDA algorithm

```
0063   ENDAT1  LDY #0          0417  A0  00
0064   ENDAT2  LDA (ENDATL), Y 0419  B1  54
0065           CMP #£00        041B  C9  00
0066           BNE CONTO       041D  D0  14
0067           INY             041F  C8
0068           LDA (ENDATL),Y  0420  B1  54
0069           CMP #£00        0422  C9  00
0070           BEQ TRY1A       0424  F0  04
0071           DEY             0426  88
0072           JMP CONTO       0427  4C  33 04
0073  TRY1A    INY             042A  C8
0074           LDA (ENDATL) ,Y 042B  B1  54
0075           CMP #£1A        042D  C9  1A
0076           BEQ RECDRD      042F  F0  0C
0077           DEY             0431  88
0078           DEY             0432  88
0079  CONTO    CPY #£FF        0433  C0  FF
0080           BNE CONT1       0435  D0  02
0081           INC ENDATH      0437  E6  55
0082  CONT1    INY             0439  C8
0083           JUMP ENDAT2     043A  4C  19 04
0084 RECORD    DEY             043D  88
```

```
0085           DEY             043E  88
0086           STY  ENDATL     043F  84  54
0087                           0441
0088   NOW COMPUTE THE NUMBER  0441
0089   OF DATA BYTES IN TEXT   0441
0090                           0441
0091                           0441
0092  LENDT1 CLD               0441  D8
0093         SEC               0442  38
0094         LDA   ENDATL      0443  A5  54
0095         SBC   STRTLO      0445  E5  50
0096         STA   LNDATL      0447  85  56
0097         STA   DTLFTL      0449  85  58
0098         LDA   ENDATH      044B  A5  55
0099         SBC   STRTH1      044D  E5  51
0100         STA   LNDATH      044F  85  57
0101         STA   DTLFTH      0451  85  59
0102         INC   DTLFTL      0453  E6  58
0103         BNE   CRCTLN      0455  D0  02
0104         INC   DTLFTH      0457  E6  59
0105 CRCTLN  NOP               0459  EA
```

Fig; 1 New END-OF-TEXT address - location algorithm

```
0219 LASREC LDA  #£03       04DA  A9  03
0220        STA  WRCBUF     04DC  8D  00  0B
0221        LDA  #£00       04DF  A9  00
0222        STA  WRCBUF + 1 04E1  8D  01  0B
0223        STA  WRCBUF + 2 04E4  8D  02  0B
0224        LDA  #£1A       04E7  A9  1A
0225        STA WRCBUF + 3  04E9  8D  03  0B
0226        JSR RECOUT      04EC  20  18  A8
```

Fig; 2 Updated LASREC Algorithm

Advertisements

MICROTANIC COMPUTER SYSTEMS LTD.

Computer Manufacturer
Software - Hardware - Books

16 Upland Road
Dulwich, London SE 22

Registration No: 1668843 V.A.T. No. 237794718 Telephone: 01-693 1137

TANSOFT 2 PASS ASSEMBLER

A Two Pass Assembler is now available for Tangerine Users. No more messy patches, just amend the Source Code and re-assemble to whatever location you desire. This Assembler is available in a 2732 Eprom and simply plugs into J2 on the Tanex Board. If you are a Basic User, there is also an Extension Eprom Board available which allows Software or Manual Switching between banks of Eprom space located between C000 and DFFF(hex). So 8k of Basic goes in one bank and the Assembler goes into the second bank leaving 8k Free for either our Forth Language or our Asimov Word Processor or 8k of your own Firmware.

What is a 2 Pass Assembler?

For those unfamiliar with 2 Pass Assemblers, they greatly simplify the task of writing Machine Code Programs. The main differences between a 2 Pass Assembler and the Line by Line Assembler (as in XBUG) are:

a) A copy of the Source Code is always retained, that is, a copy of what YOU actually type in (eg. LDA).
b) Any Instruction or Data Location can be given a Label (a name).
c) These Labels may be referenced by Instructions, so, for example, rather than JSR to a specific location you may JSR FRED, where Fred is perhaps the beginning of a Subroutine.

The great advantage of this system is that Instructions can be inserted or deleted from the Source Code, Lables swopped around, Data Areas altered in length etc, without worrying about the effect on addresses.
This system implies that you can assemble object code anywhere in memory.

Facilities:

The facilities offered by this particular Assembler are:

* Full source code editing including validation of instructions. *
* Cassette Handling Routines for maintenance of Source Code on Tape. *
* All standard 6502 Instructions are supported in the same format as used by the XBUG Line Assembler (except for ASL A which is replaced by ASL @ to avoid confusion with Labels). *
* Extra Instructions are supported - ORG (relative or absolute) EPZ(equate page zero), BYT(for single byte constants) and WOR(for 2 byte addresses). *
* Hexadecimal, character and decimal constants are supported. *
* Reference to Labels may include the format Label+constant(eg. FRED+2). *
* Various Print, List and Display options, including the facility to list the Labels and their Addresses. *
* Options to Dump or Assemble parts of the Source Code and to Load or Assemble several parts into one program. *
* Option to Assemble at an Eprom Location but to Store the Object in another RAM Address ready for Eprom Burning. *

*** AND ALL THIS IN 4K !!! ***

SUPPLIED IN EPROM 34.95

SPECIAL OFFER

H2 EXTENSION EPROM BOARD PLUS 2 PASS ASSEMBLER EPROM ... 49.45

SUBSCRIPTION

Microtanic Computer Systems Ltd. 16 Upland Road, Dulwich, London. SE 22. 01 693 1137.

Please send me the next six copies of Microtan World.

Name .....

Address .....

.....

I enclose £10:00 in full payment.

Cheques should be made payable to Microtanic Computer Systems Ltd.

NEXT ISSUE

A lot of the content of the next issue will depend on you the readers.

We are particularly interested in articles for single board owners, useful routines, hints and tips you may have found, etc. etc.

Don't forget to let us have your letters telling us what you think about this issue and of course any comments on the system or any matter of general interest.

If you know anyone with a Microtan Computer who is not yet subscribing to Microtan World why not let him have the subscription form and see if you can persuade him to join us. The more subscribers we can get the better we can make the magazine.

Printed by Christian Graphic Arts Ltd., 29 Freemantle Road, Gosport, Hants P012 4RD