TANDOC

ISSUE 3

A MICROTAN OWNER USER GROUP
PUBLICATION

EDITORS PAGE

Tandoc 3! Alas the last. By now many of you are aware that the user group (MOUG) has formerly ceased to exist, this issue being its finale.

Why you may ask? Well, the past eighteen months have been most successful and productive with the group going from strength to strength. Unfortunately it was in early June that the relationship between MCS and myself became intolerable, culminating with the threat of legal action being taken against me. In the interests of all parties concerned I am unable to give further explanation as to why. I am sure you will all understand. After due consideration it is with regret that I have decided the right and proper action is to exit from the Microtan scene. My main problem was how to do it without too much disappointment to you and the other users. As many of you had ordered and paid for Tandoc 3, I decided to carry out a steady run down of operations finishing with this issue. It would also give someone the opportunity to come forward and takeover from me. As yet this has not happened.

May I add, there is still a Microtan and will be for some time to come, its position today is mainly due to the efforts and interest shown in it by people like yourself, so please do not give up hope yet. I believe with a little effort the Microtan system can remain around for many more years, it still has a lot going for it. What a shame that ignorance and greed could be its downfall.

This being the last issue, I have endeavoured to make it a bumper one. Its size has been increased to permit the printing of program listings.

May I wish you all success with your continued adventures in the world of the Microtan. Finally, my thanks for all those orders, articles and programs. It was this that made the group successful.

Brian Gibbs.

14th July, 1985

****

Printed by:-

QUAYSIDE COPY, Salmon Parade, Bridgwater, Somerset.

****

CASSETTE BEEP

The XBUG eprom has a few unused locations at the end of it. By loading the following program into to them it allows those people with a Bulldog or similar sound generator board addressed by two locations (BC00 - BC01) to get a beep on completion of LOAD, SAVE and EXAMINE commands whilst using tape storage in both MACHINE CODE and BASIC.

```
F7CA    PHA             ;save acc.
F7CB    TXA
F7CC    PHA             ;save X reg.
F7CD    LDX #$0E        ;load date into sound generator
F7CF    STX $BC00
F7D2    LDA $F7DF,X
F7D5    STA $BC01
F7D8    DEX
F7D9    BNE $F7CF
F7DB    PLA             ;restore X reg. and acc.
F7DC    TAX
F7DD    PLA
F7DE    RTS             ;return
F7DF    05 01 00        ;data for sound generator
F7E2    00 00 00
F7E5    00 3E 10
F7E8    00 00 00
F7EB    A0 09
F7ED    JSR $F7CA       ;BASIC entry point
F7F0    STA $BFCB
F7F3    RTS
F7F4    NOP
F7F5    NOP
F7F6    NOP
F7F7    JMP $F65B       ;DON'T CHANGE THIS!
F7FA    JSR $F7CA       ;M/C entry point
F7FD    JMP $FC4B
```

Now change the cassette routines so that they jump to the new patch. First change the X-BUG eprom.

```
F6A2    JMP $F7FA    ;was JMP $FC4B
```

Now change the BASIC eprom.

```
E2DC    JSR $F7ED    ;was STA $BFCB
E7D0    JSR $F7ED    ;was STA $BFCB
E7DC    JSR $F7ED    ;was STA $BFCB
```

Now when you use your cassette the sound generator will beep with decay on completion of the task.

Terry Royle

PCB'S versus WIRE WRAP

Throughout the life of the the Microtan one major problem has been the availibility of printed circuit boards, it appears that many users are unaware that from an accurate circuit diagram all the boards can be wire wrapped. Wire wrapping means that one becomes independent of the manufacturer, especially when supplies of boards are either un-available or dry up, which has so often happened during the last five years, although during this period most if not all the circuit diagrams for the system have been made readily available from various sources. This is a definite case for wire wrapping your own boards, it requires very little skill, a little patience and a few hours of your time.

I propose to outline a method of approach and the tools required. From experience you will require a SQUARE PAD prototype board, a 6U board will have to be cut down as no square pad version is manufactured for the 19 inch card system, the square pad enables wire wrap sockets to be positioned anyware on the board.

1.Fit 64 way "ab" plug to end of board, this will be a solder type, thus you will have to solder all wires terminating on this bus connector.

2.Position wire wrap IC holders on board, leaving sufficient room for the hard wiring of any discete components required ie. pull up resistors and decoupling capacitors etc.

3.Hard wire using a fairly heavy guage wire the power rails to each IC holder, this stops any problems with volt drop which would most certainly occur if the power rails were wire wrapped. Also at this stage hard wire in the de-coupling capacitors.

4.Having obtained a simple wire wrap tool and some suitable single core wire, you are now ready to commence wrapping. This should be carried out in a planned logical manner, thus avoiding mistakes.

5.On completion of the wrapping, check the board, if you are fortunate to have an oscilloscope, then populate the board systematically checking waveforms as you go. With a little bit of luck you will now have a working board.

Please remember most production boards start out life as wire wrapped prototype boards, PCB's are for the benefit of mass production.

Wire wrap tools can be obtained from suppliers such as Maplin, RS Components etc.
Prototype boards from RS Components, Verospeed and others.

Ed

============================

FLEX and all that

Doug Deedman

The 6502 microprocessor, used by so many microcomputers over the years and at the heart of the Microtan 65, is beginning to show its age. It was never the world's most powerful processor but, for all its limitations, has been made to perform with considerable agility and has gained affection from those who have programmed it. However, the 6502 has suffered throughout its life from one quite appalling disability: it has no standard operating system.

It is only recently that users of microcomputers have woken up to the importance of operating systems. In fact, there are still certain computer manufacturers who are content to produce systems which require machine specific software and which, as a result, are provided with minimal support, usually in the form of alien-zapping games and barely usable "business" software. With this in mind, I suppose it is not surprising that users have been slow to demand grown up computer systems with real, machine independent software environments, but experience has been the teacher and the recent crash of the home computer market and the giant manufacturers with it is, in my view, one of the signs of the learning.

The preceding ramblings have been to bring me to the main point of this article which is the FLEX operating system. FLEX is an industry standard operating system which can be run on computers using 6800 or 6809 microprocessors. Of these two processors, the younger and unquestionably more powerful is the 6809. This device can, with much justification, claim to be the most powerful eight bit microprocessor available and it would not be unreasonable to refer to it as a sixteen bit, since it can perform many tasks in that role. Programmers used to the 6502 are certain to be impressed by the power of the 6809 and the consequent ease with which programs can be encoded. One immensely useful feature of the instruction set is the ability to produce position independent code; others are the simplicity with which 16 bit addresses can be handled and the extension of the short, zero page instructions, familiar to 6502 users, to every page in the 64k set of the 6809 via its Direct Page register. However, the virtues of the processor are not really under examination. Let us turn to FLEX itself.

FLEX consists of three parts: the File Management System (FMS), the Disc Operating System (DOS), and the Utility Command Set (UCS). The first two of these effectively convert your particular piece of hardware into a standard FLEX computer and allow you to run any of the enormous number of off-the-shelf software packages currently available. An example is the extremely powerful word processor, Screditor III, which I am using to write this article! The Utility Command Set is a suite of programs, each of which is loaded into system RAM from disc when needed. These programs perform such tasks as the saving, loading, copying, renaming, deleting, appending and listing of disc files.

Included in the UCS is the facility to arrange the terminal and disc set-up to the user's convenience. I have set up my terminal (Intelgraph running Intel-T) so that lines of 80 characters are displayed by FLEX which, when writing to the VDU, waits after every 25 lines of output until ESC is entered from my keyboard. Backspace is used to delete characters and the Delete key cancels an entire command line. I could just as easily have set up Delete instead of Backspace to erase characters and arranged a screen format of 64 character lines with no display pause. True FLEXibility!

The Disc Operating System forms the link between the user and the File Management System. Essentially, it processes all commands to FLEX. These commands normally come from the user via the keyboard but it is possible to arrange command input to come from a text file loaded from disc. This is done by using two routines from the UCS called BUILD and EXEC. BUILD is used to create a small text file, which consists of the commands to be given to FLEX, and then EXEC calls this file. EXEC replaces the FLEX keyboard input routine with one which reads the text in exactly the same way as if it had come from the keyboard. Those who are familiar with the Acorn Disc Filing System will recognise this facility and will, I am sure, conclude that the author of the Acorn DFS must be an admirer of FLEX.

The File Management System is the soft interface between the DOS and the user's disc hardware. The FMS looks after all file allocation and removal on the disc and, assuming it knows where to find certain low level disc handling routines, does not really care what machinery you hook up to your computer. FLEX is quite happy to work with either eight inch or five and a quarter inch floppies, three and a half inch encased floppies or even hard discs.

Those of you who are confirmed fiddlers or merely wish to write your own utilities for FLEX will need to obtain a copy of the Flex Programmer's Manual. This is also known as the Advanced Programmers Manual so, clearly, it is not written for beginners. This book provides all the inside information about FLEX that is needed to write programs which will run with it. In case there is still anyone who has not taken the point, I mean programs which will run on ANY computer running FLEX.

I feel I ought to point out that when you buy FLEX you get two extras: an extremely clever assembler and a neat little text editor. The assembler is really good. Those of you who have never used a professional assembler will be impressed by its range of facilities which include conditional assembly and the provision of macros. The assembler produces object code directly to disc from source code which is stored as a text file. In order to write such source code you need a text editor, which accounts for the its inclusion in the package. Actually, you can write source code with any program which produces standard text files, which means just about every FLEX word processor except Stylograph!

By now you must have realised that I am a FLEX fan. I know of nobody who has used it that isn't impressed by this very sensible operating system. Try it and see.

TANEX SERIAL OPTION

First lets clear up a minor understanding, the serial option on Tanex is NOT a true RS 232C as the manual leads you to believe, as the control signals DTR, CTS, RTS, DCD and DSR are active LOW, on a true RS 232C these lines would be active HIGH.

Also the serial port has be set up to 110 baud by TUGBUG location $F861, to change the baud rate means reblowing the main operating system with modification of this location, or write your own program to communicate with the 6551 UART.

Note: If serial option installed a header must be fitted to socket E1 on Tanex connecting pins 8, 10 and 11 to +5v via a 10k resistor, this will prevent the system hanging up.

Further through Tandoc you will find a circuit diagram of an Industrial standard RS 232C suitable for use with the Microtan. This might be a good exercise for wire wrapping.

Ed

==============================

TANDOS ROM

Due to a hardware design address fault on Tandos board, when blowing the Tandos eprom it is necessary to blow the $B000 - $B7FF block into the lower 2k of the eprom, ikewise the $A800 - $AFFF block into the high block of the eprom.

This will only apply if the eprom is used on board Tandos card.

Dan Shaw

==============================

PROGRAM LISTINGS

The following two listings were both produced using MDASM. This latest version of EPACPN, DASM.

1.The first listing is to improve the printer output speed of CWORD and CWORDT.

2.The second listing is for the removal of a bug which in some eariler versions of DASM. This bug would not allow patch work to be carried out, except when using JMP or JSR instructions.

For those of you that have neither of these programs, use could be made of some of the routines in the development of your own programs.

David Cawthorne

==============================

```
CWORD MODS                          PAGE :  1      DATED : 01.07.85
                 10      MODIFICATIONS TO CWORDT
                 20      USING MDASM
                 30
                 40      improving printer output
                 50      speed
                 60
                 70               MAXLEN   =   $43
                 80               TEMP     =   $39
                 90               BUFF     =   $80
                 100              PARPNT   =   $F803
                 110              OUTPUT   =   $12D1
                 120              MEM      =   $4D
                 130              MAINLP   =   $B18
                 140              PATCH    =   $1285
                 150
                 160              *        =   $F7B
                 170     this should have been done
                 180     with the 127 character mod
                 190
0F7B   7F        200              DEFB $7F                  ;max length
                 210
                 220              *        =   $10BE
                 230
10BE   A643      240              LDX   MAXLEN
10C0   CA        250      LOOP1:  DEX
10C1   3006      260              BMI   JPATCH
10C3   B580      270              LDA   BUFF,X
10C5   C920      280              CMP   #$20
10C7   F0F7      290              BEQ   LOOP1
10C9   208512    300     JPATCH:  JSR   PATCH
                 310
                 320              *        =   $1279
                 330
1279   2003F8    340              JSR   PARPNT
127C   F001      350              BEQ   ERROR
127E   60        360              RTS
127F   A2FF      370      ERROR:  LDX   #$FF
1281   9A        380              TXS
1282   4C180B    390              JMP   MAINLP
1285   E8        400      PATCH:  INX
1286   8639      410              STX   TEMP
1288   A200      420              LDX   #$0
128A   B580      430      LOOP2:  LDA   BUFF,X
128C   20D112    440              JSR   OUTPUT
128F   E8        450              INX
1290   E439      460              CPX   TEMP
1292   90F6      470              BCC   LOOP2
1294   E64D      480              INC   MEM
1296   60        490              RTS
                 500
                 510
                 520     $1297 to $12CE now free
                 530     due to using printer

No errors.
End of object = $1296

    MAXLEN = $0043   TEMP   = $0039   BUFF   = $0080   PARPNT = $F803
    OUTPUT = $12D1   MEM    = $004D   MAINLP = $0B18   PATCH  = $1285
    LOOP1  = $10C0   JPATCH = $10C9   ERROR  = $127F   PATCH  = $1285
    LOOP2  = $128A
```

```
                  10      DASMNC MODS TO ALLOW PATCH
                  20      WORK AND XBUG COMPATABILITY
                  30
                  40
                  50                PATCH    =    $E116
                  60                CONT     =    $CD6F
                  70                CODE     =    $C0B2
                  80                ICURS    =    $A
                  90                ICURSH   =    $B
                  100               JHXPK    =    $F817
                  110               JRERD    =    $F01B
                  120               JRCDBL   =    $F01E
                  130               *        =    $CD5F
                  140
CD5F   2016E1     150               JSR   PATCH
CD62   EA         160               NOP
CD63   EA         170               NOP
                  180
                  190               *        =    $CD45
                  200
CD45   2016E1     210               JSR   PATCH
CD48   4C6FCD     220               JMP   CONT
                  230
                  240               *        =    $DDF7
                  250
DDF7   201EF0     260               JSR   JRCDBL
                  270
                  280               *        =    $DE16
                  290
DE16   201BF0     300               JSR   JRERD
                  310
                  320               *        =    $DE25
                  330
DE25   C60B       340               DEC   ICURSH
DE27   A0E9       350               LDY   #$E9
DE29   2017F8     360               JSR   JHXPK
DE2C   E60B       370               INC   ICURSH
DE2E   EA         380               NOP
DE2F   EA         390               NOP
DE30   EA         400               NOP
                  410
                  420               *        =    $DE48
                  430
DE48   201BF0     440               JSR   JRERD
                  450
                  460               *        =    $E116
                  470
E116   20B2C0     480               JSR   CODE
E119   A009       490               LDY   #$9
E11B   B10A       500     LOOP1:    LDA   (ICURS),Y
E11D   C920       510               CMP   #$20
E11F   D004       520               BNE   COUNT
E121   88         530               DEY
E122   88         540               DEY
E123   D0F6       550               BNE   LOOP1
```

```
DASM MODIFICATIONS                     PAGE :  2      DATED : 01.07.85

E125  98       560      COUNT:  TYA
E126  38       570              SEC
E127  E905     580              SBC    #$5
E129  4A       590              LSRA
E12A  A8       600              TAY                        ;IY=No. of bytes for
E12B  60       610              RTS                        ;opcode

No errors.
End of object = $E12B
```

==============================

FLEX

Why not update your system to 6809 and Flex?

The least expensive method is without doubt the direct replacement of the Microtan 65 board with a 6809 board. This means you will be able to carry on using all your existing boards.

Why not give Ralph Allen a ring, for his latest details.

Ralph is endeavouring to be able to supply a complete range of 6809 products which are Microtan compatible. This means the user will no longer be totally dependent of MCS.

For all your 6809, FLEX and Microtan <u>COMPATIBLE</u> additional boards please contact:-

Ralph Allen Engineering Co.

FORNCETT END NORWICH
NORFOLK ENGLAND

Telephone: 0953 89420

Ed

==============================

DOS UPDATE

Anyone who has purchased the new Bacic compatible Disk Utilities, will have no doubt discovered that after using KILL or DEL a number of times on the same disk, when trying to copy to that disk, receives the reply DISK FULL, although there appears to be more than enough sectors free for the operation. Well, this is due to a error in both KILL and DEL. Please correct as follows:-

DEL/KILL disc error
****

```
DEL:
BBA6 = AD 7E B9, not AD 7F B9

KILL:
BB78 = AD 6E B9, not AD 6F B9
```

David Cawthorne

==============================

YET MORE OF DAVID'S JOTTINGS

THE VERY LAST P.G.M. TOOLKIT RENUMBER MODIFICATION

****

After the TANDOC 1 mods I was looking into discs and now back with basic still notice the very occasional renumber error ??
This was traced down to two things:
1) lack of detection of a carry at EBA1 means that the renumber step should be kept less than 1536 decimal which is not much of a hardship so I did not alter it.
2) the renumber routine uses a JSR $00E2 it increment the text pointer E9/EA. The trouble is that it treats E9/EA as a text pointer and increments it past any spaces, which is fine if you are looking along text for GOTOs,GOSUBs etc. BUT when you pass over either a line number or next line pointer with $20 in one of the two bytes it skips this and misses the line and gets the line number wrong.
ALTER:

```
EC68 20 17 EF 20 17 EF
EC72 20 17 EF 20 17 EF

EF17 E6 E9 D0 02 E6 EA A0 00 B1 E9 60
```

PBASIC

******

If you load basic in from E.S.C. and have the P.G.M. toolkit with discs, a simple mod to get PBASIC is to alter DBASIC:

```
B9B2 20 39 BA
BA39 20 06 A8 A9 00 8D 90 E8 8D 87 E8 A9 A8 8D 91 E8 4C 00 E8
```

CWORD/DWORD
****

In the manual it said that the mailing facility (!an.) could have n = 1 to 99 this was not so thus FFB (CWORD) and CFFB (DWORD) were altered to $64

On some printers you may observe the head going all the way across its travel even if the line only contains a few characters.
Here are the mods to stop this:

```
VVVV A6 43 CA 30 06 B5 80 C9 20 F0 F7 20 WW XX
YYYY E8 86 39 A2 00 B5 80 20 D1 ZZ E8 E4 39 90 F6 E6 4D 60
```

and for CWORD only:

```
1279 20 03 F8 ;see TANDOC 2 for serial patch
127C F0 01 60 A2 FF 9A 4C 18 0B
```

There is also a word wrap error which can occure (e.g. printer width = 75, and on a new line you have 38 continuous letters,a space and another 38 continuous letters, when using Print command it gets stuck in a loop)

TRY WORD WRAPPING THIS LETTER TO 31 CHARACTERS AND YOU WILL SEE WHAT I MEAN.
ALTER:

```
RRRR A6 46 B5 80 C9 20 F0 07 E8 E4 43 D0 F5 F0 0B B5 80
     C9 20 D0 0A E8 E4 43 D0 F5 68 68 4C BE SS A5 4D 29 01 60
UUUU 20 QQ TT EA
```

```
      CWORD/DWORD
VVVV = 10BE/D0BE
    WW = 99/F1
    XX = 12/DA
YYYY = 1299/DAF1
    ZZ = 12/D2
    SS = 10/D0
    TT = 12/DB
    QQ = AB/03
RRRR = 12AB/DB03
UUUU = 1106/D106
```

DASMNC/DASM MODS
****

It was observed that you could not use the assembler for patch work as the following few bytes would always be corrupted. This is because the EPA always outputs three bytes regardless of the length of the opcode.
ALTER:

```
CD5F 20 MM NN EA EA
CD45 20 MM NN 4C 6F CD
```

```
NNMM  20 B2 C0 A0 09 B1 0A C9 20 D0 04 88 88 D0 F6 98 38 E9 05
4A A8 60
```

MMNN is the end of the EPA, this may vary,- E116 for DASMNC, DBF4 for DASM.

DASMNC with its own mnemonics table and cassette load/verify/save options does not use the XBUG vectored jumps making it incompatible with some XBUGs thus alter:

```
DDF7 20 1E F0
DE16 20 1B F0
DE48 20 1B F0
DE25 C6 0B A0 E9 20 17 F8 E6 0B EA EA EA
```

David Cawthorne

VIDEO TOOLKIT

Problems have been experienced in getting the Video Toolkit to run. Normally most Basic's have an $E800 Auto Run routine in them, if yours does not then once in Basic enter the following:-

```
POKE34,0:POKE35,232:X=USR(X) [CR]
```

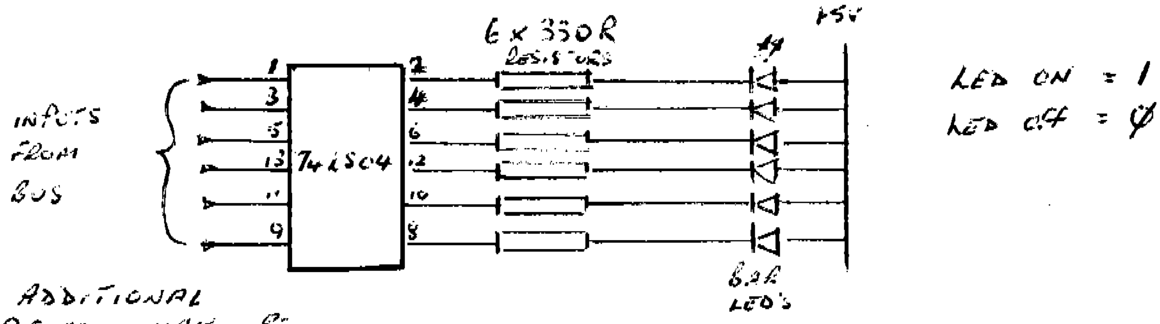
This should do the trick as with all other toolkits.

Ed

SEEING IS BELIEVING

There are many occasions when it is desirable to see what is going on within the machine, especially when fault finding. Easy, you might say, use an oscilloscope or logic probe. Well, that's alright if you are lucky enough to have one, most have not. Even so there are limitations, ie a scope normally has only two channels, so what does one do if for instance you want to look at the sixteen address lines simultaneously? Yet again you might say use a Logic analyser, which no doubt is out of the question, so at this juncture one normally gives up. Why, there is a cheap and simple answer, LED's.

You can purchase a various Bar Arrays, 5, 10 and 30 segment. The 30 segment being the most useful, as you can construct a test box to look at the 16 address, 8 data and up to 6 additional lines simultaneously. See following diagram, this device can be connected to the bus and left there whilst running with no adverse effect.

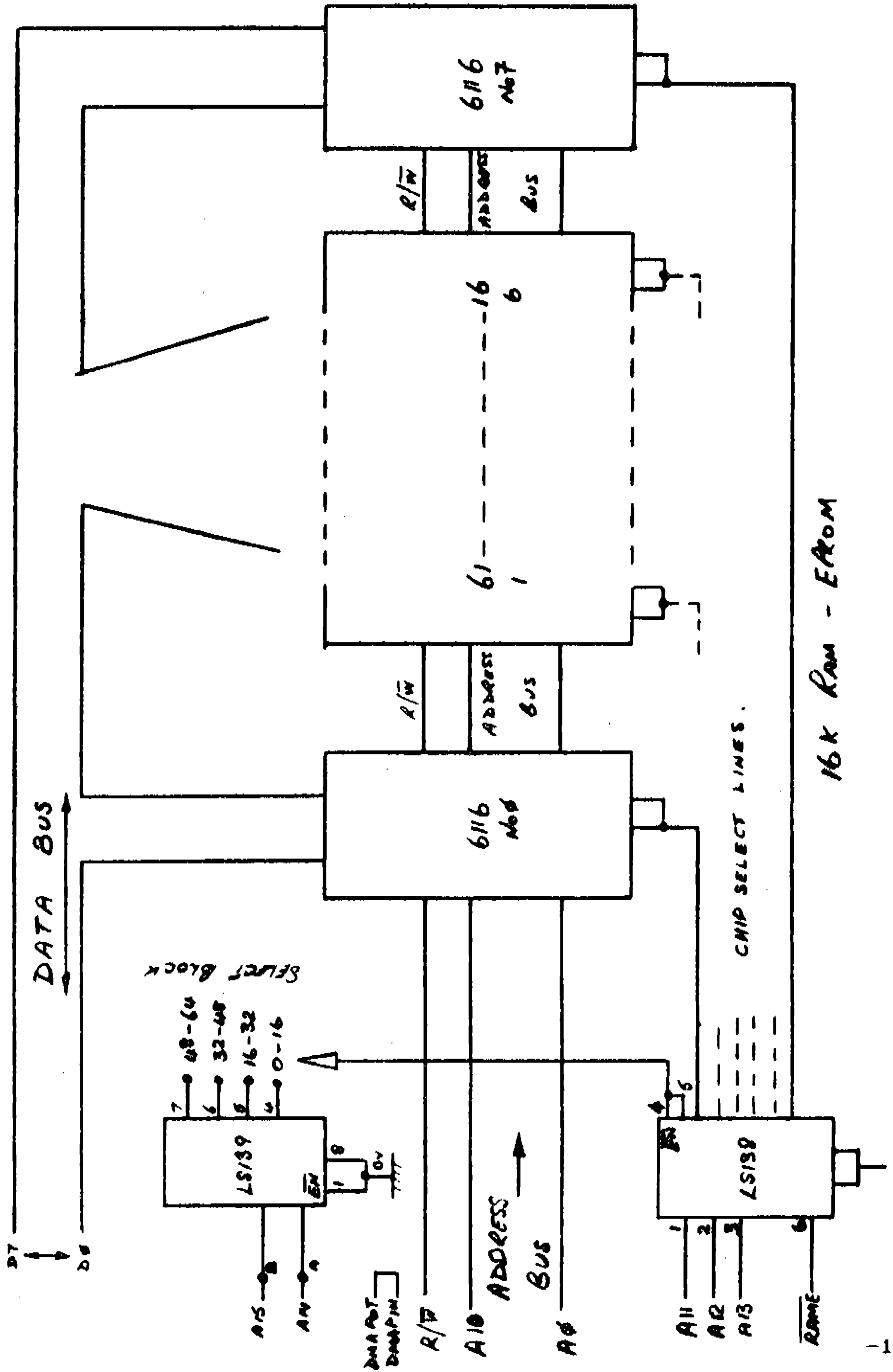


NOTE ADDITIONAL +5V P.SUPPLY MAY BE REQUIRED IF ALL 30 SEGMENTS ARE USED - TYPICAL I = 700mA

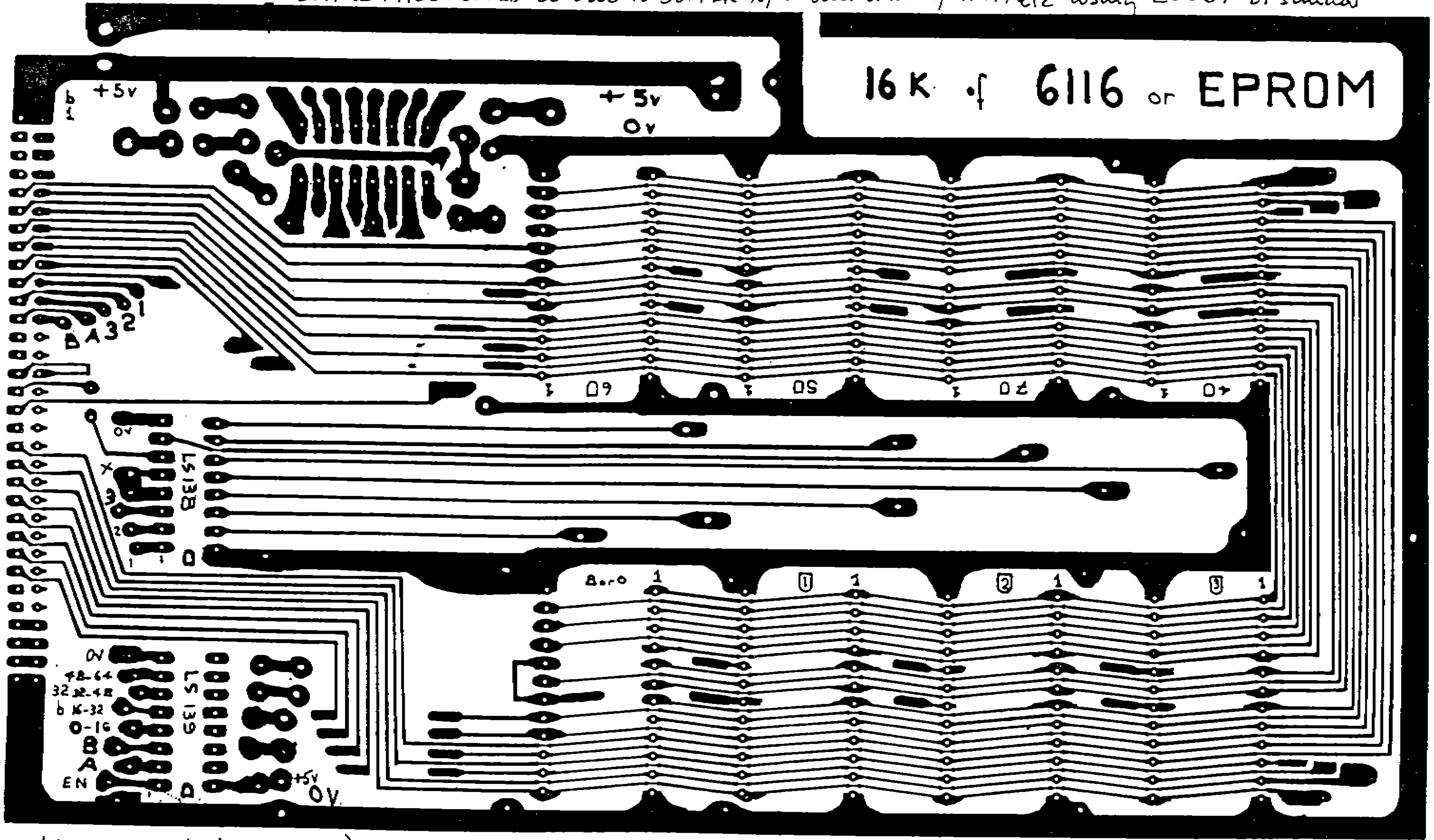
Ed

6116 OR 2732 RAM/EPROM EXTENSION BOARD

The following board was developed to enable a cheap alternative to increase available memory ram/eprom. It should be noted, now that 6264 ram has become so cheap, with a little thought the 6116's could be replaced with 8k ram and likewise 2732's with 2764's, thus a very competitively priced and powerfull addition to the system. These boards can be used in multiples by simply changing the addressing.



SPARE PADS COULD BE USED TO BUFFER R/$\overline{W}$ line and A$\emptyset$ to A4 etc using LS367 or similar



39 determines Block of Memory used
ect block required to 'X' on LS138) NOTE

UNDERSIDE!

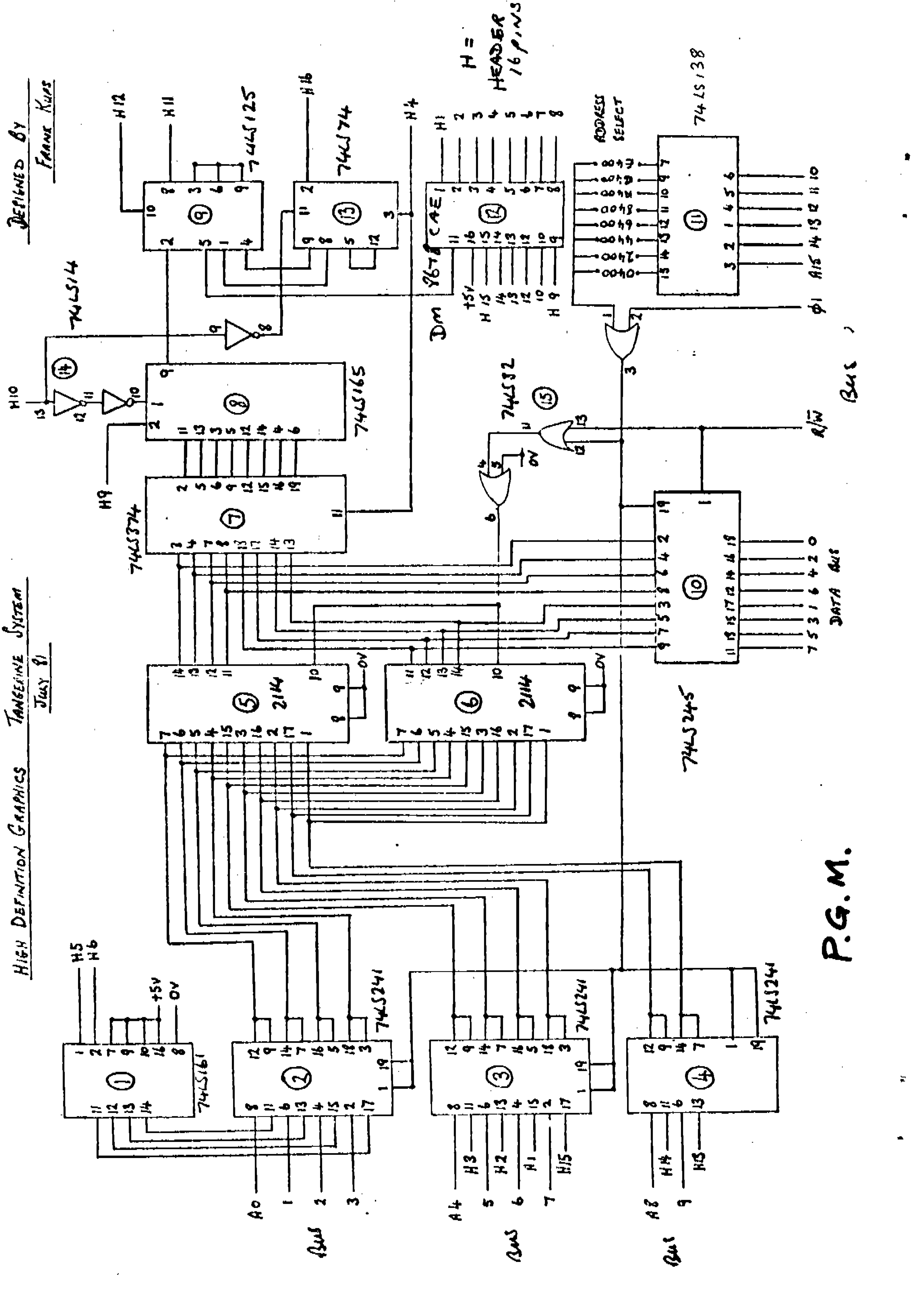
Use 0·1µF's to decouple supply lines every 2 chips.

JOIN 17 & 19 on all 6116
to $\overline{CS}$ LINES

4 JUMPERS for data LINES

Tony Kersey

PGM CIRCUIT DIAGRAM



NEW DISK CREATION

The following routine generates a new disk in one fast operation. It is a great improvement on SYSGEN, INIT and FORMAT. A copy of INIT must be retained for those rare occasions when only initialisation of a disk is required, ie Disk FORTH.

```
CREATE                          PAGE :  1       DATED : 01.07.85

          10      ***************************
          20      *                         *
          30      *       C R E A T E       *
          40      *                         *
          50      ***************************
          60
          70       This version of CREATE by
          80       C.P.Nowell
          90       (c) 29th March 1985
          100     (CREATE does the lot!!! )
          110
          120             ICHAR     =    1           ;0-page RAM
          130             VDUIND    =    3           ;  "      "
          140             ICURS     =    $A          ;  "      "
          150             HXPK      =    $13         ;  "      "
          160             MEMSEG    =    $40         ;  "      "
          170             ITRACK    =    $42         ;  "      "
          180             TNUM      =    $44         ;  "      "
          190             SCNT      =    $45         ;  "      "
          200
          210             DIRSO     =    $13BF       ;TRACK buf locs
          220             LASTSC    =    $1886       ;  "     "   "
          230             LASTTR    =    $1887       ;  "     "   "
          240
          250             JRESET    =    $B00C       ;DOS ROM calls
          260             JFDMAI    =    $B00F       ; "    "    "
          270             JSETIN    =    $B015       ; "    "    "
          280             JESCAP    =    $B7B2       ; "    "    "
          290             JZSAVE    =    $B7B5       ; "    "    "
          300             JGETLI    =    $B7C4       ; "    "    "
          310             JERRRE    =    $B7C7       ; "    "    "
          320             JQUERY    =    $B7CA (B94F); "    "    "
          330             JWRTSC    =    $B7CD       ; "    "    "
          340             JGETSS    =    $B7D3       ; "    "    "
          350             JBADDE    =    $B7D6       ; "    "    "
          360             JOUTST    =    $B7D9       ; "    "    "
          370             JALPNU    =    $B7FA       ; "    "    "
          380
          390             UNIT      =    $B800       ; "   RAM locs.
          400             TRACK     =    $B801       ; "    "    "
          410             SECT      =    $B802       ; "    "    "
          420             DENS      =    $B803       ; "    "    "
          430             DMAA      =    $B804       ; "    "    "
          440             ERROR     =    $B808       ; "    "    "
          450             DSCDEF    =    $B815       ; "    "    "
          460             SBUFO     =    $B825       ; "    "    "
          470             FSCT      =    $B835       ; "    "    "
          480             FTRK      =    $B836       ; "    "    "
          490             DSCT      =    $B837       ; "    "    "
          500             DTRK      =    $B838       ; "    "    "
          510             FCNT      =    $B839       ; "    "    "
          520             USED      =    $B83B       ; "    "    "
          530             TITLE     =    $B83D       ; "    "    "
          540             COUNT     =    $B943       ; "    "    "
          550             LINLEN    =    $B943       ; "    "    "
```

```
                   560             DOSVER    =    $B865     ; "     "     "
                   570
                   580             VDUSTA    =    $BE00     ;VDU PORT locs
                   590             VDUCTL    =    $BE01     ; "     "     "
                   600
                   610             DATA      =    $BF93     ;DOS PORT locs
                   620             CNTROL    =    $BF94     ; "     "     "
                   630
                   640             OUTRET    =    $F80C     ;MON ROM calls
                   650             OUTALL    =    $F80E     ; "     "     "
                   660             JHXPK     =    $F817     ; "     "     "
                   670             JHXPN     =    $F81A     ; "     "     "
                   680             JPLKB     =    $F81D     ; "     "     "
                   690             RETMON    =    $F823     ; "     "     "
                   700             JCURSF    =    $F829     ; "     "     "
                   710
                   720             *         =    $400
                   730
0400   10
0401   0A          740         A1: DEFB16,10
0402   00          750     TRKTOT: DEFB0
0403   00          760     SCTTOT: DEFB0
0404   04
0405   05
0406   06
0407   07
0408   08
0409   09
040A   0A          770     SECSEQ: DEFB4,5,6,7,8,9,10
040B   01
040C   02
040D   03          780             DEFB1,2,3
040E   0D          790       MSG1: DEFB$D
040F   44
0410   69
0411   73
0412   63
0413   20
0414   6E
0415   61
0416   6D
0417   65
0418   3A
0419   20          800             DEFM 'Disc name: '
041A   00          810             DEFB 0
041B   0D          820       MSG2: DEFB$D
041C   44
041D   69
041E   73
041F   63
0420   20
0421   6E
0422   6F
0423   77
0424   20
```

```
0425  72
0426  65
0427  61
0428  64
0429  79
042A  20
042B  66
042C  6F
042D  72
042E  20     830             DEFM'Disc now ready for '
042F  75
0430  73
0431  65
0432  2E     840             DEFM'use.'
0433  0D
0434  00     850             DEFB$D,0
0435  54
0436  41
0437  4E
0438  44
0439  4F
043A  53
043B  20
043C  36
043D  35
043E  20
043F  56
0440  31
0441  2E
0442  31
0443  20     860     MSG3:   DEFM'TANDOS 65 V1.1 '
0444  28
0445  43
0446  50
0447  4E
0448  29
0449  20
044A  31
044B  39
044C  38
044D  35
044E  2E     870             DEFM'(CPN) 1985.'
044F  00     880             DEFBO
0450  0D     890     MSG4:   DEFB$D
0451  4C
0452  6F
0453  61
0454  64
0455  20
0456  64
0457  69
0458  73
0459  63
045A  20     900             DEFM'Load disc '
045B  30     910     DRV:    DEFM '0'        Ascii Drv No , ie 32
```

```
045C  20
045D  61
045E  6E
045F  64
0460  20
0461  70
0462  72
0463  65
0464  73
0465  73
0466  20
0467  52
0468  45
0469  54
046A  55
046B  52
046C  4E        920                  DEFM' and press RETURN'
046D  00        930                  DEFB 0
                940
                950          This is 1 whole track!!
                960
046E  FF
046F  FF
0470  FF
0471  FF
0472  FF
0473  FF        970          WTRK:   DEFW$FFFF,$FFFF,$FFFF
0474  FF
0475  FF
0476  FF
0477  FF
0478  00
0479  00
047A  00
047B  00
047C  00
047D  00        980                  DEFW$FFFF,$FFFF,0,0,0
047E  FE        990                  DEFB$FE                    ;CRC byte
047F  00        1000     TRKNUM:     DEFB 0                     ;(initially)
0480  00        1010     SIDNUM:     DEFB 0                     ;(1=rev side)
0481  01
0482  01
0483  F7
0484  FF
0485  FF
0486  FF
0487  FF
0488  FF        1020                 DEFW$101,$FFF7,$FFFF,$FFFF
0489  FF
048A  FF
048B  FF
048C  FF
048D  FF
048E  FF
048F  00
```

```
0490   00
0491   00
0492   00
0493   00
0494   00      1030                DEFW$FFFF,$FFFF,$FFFF,0,0,0
0495   FB      1040                DEFB$FB                     ;CRC byte
               1050
0496   00
0497   00
0498   00
0499   00
049A   00
049B   00
049C   00
049D   00      1060      STRKO:    DEFB0,0,0,0,0,0,0,0
049E   00
049F   00
04A0   00
04A1   00
04A2   00
04A3   00
04A4   00
04A5   00
04A6   00
04A7   00
04A8   00
04A9   00
04AA   00
04AB   00
04AC   00
04AD   00
04AE   00
04AF   00
04B0   00
04B1   00
04B2   00
04B3   00      1070                DEFW0,0,0,0,0,0,0,0,0,0,0
04B4   00
04B5   00
04B6   00
04B7   00
04B8   00
04B9   00
04BA   00
04BB   00
04BC   00
04BD   00
04BE   00
04BF   00
04C0   00
04C1   00
04C2   00
04C3   00
04C4   00
04C5   00
```

```
04C6   00
04C7   00
04C8   00
04C9   00          1080            DEFW0,0,0,0,0,0,0,0,0,0,0
04CA   00
04CB   00
04CC   00
04CD   00
04CE   00
04CF   00
04D0   00
04D1   00
04D2   00
04D3   00
04D4   00
04D5   00
04D6   00
04D7   00
04D8   00
04D9   00
04DA   00
04DB   00
04DC   00
04DD   00
04DE   00
04DF   00          1090            DEFW0,0,0,0,0,0,0,0,0,0,0
04E0   00
04E1   00
04E2   00
04E3   00
04E4   00
04E5   00
04E6   00
04E7   00
04E8   00
04E9   00
04EA   00
04EB   00
04EC   00
04ED   00
04EE   00
04EF   00
04F0   00
04F1   00
04F2   00
04F3   00
04F4   00
04F5   00          1100            DEFW0,0,0,0,0,0,0,0,0,0,0
04F6   00
04F7   00
04F8   00
04F9   00
04FA   00
04FB   00
04FC   00
```

```
04FD  00
04FE  00
04FF  00
0500  00
0501  00
0502  00
0503  00
0504  00
0505  00
0506  00
0507  00
0508  00
0509  00
050A  00
050B  00        1110                DEFW0,0,0,0,0,0,0,0,0,0,0
050C  00
050D  00
050E  00
050F  00
0510  00
0511  00
0512  00
0513  00
0514  00
0515  00
0516  00
0517  00
0518  00
0519  00
051A  00
051B  00
051C  00
051D  00
051E  00
051F  00
0520  00
0521  00        1120                DEFW0,0,0,0,0,0,0,0,0,0,0
0522  00
0523  00
0524  00
0525  00
0526  00
0527  00
0528  00
0529  00
052A  00
052B  00
052C  00
052D  00
052E  00
052F  00
0530  00
0531  00
0532  00
0533  00
```

```
0534   00
0535   00
0536   00
0537   00        1130          DEFW0,0,0,0,0,0,0,0,0,0
0538   00
0539   00
053A   00
053B   00
053C   00
053D   00
053E   00
053F   00
0540   00
0541   00
0542   00
0543   00
0544   00
0545   00
0546   00
0547   00
0548   00
0549   00
054A   00
054B   00
054C   00
054D   00        1140          DEFW0,0,0,0,0,0,0,0,0,0
054E   00
054F   00
0550   00
0551   00
0552   00
0553   00
0554   00
0555   00
0556   00
0557   00
0558   00
0559   00
055A   00
055B   00
055C   00
055D   00
055E   00
055F   00
0560   00
0561   00
0562   00
0563   00        1150          DEFW0,0,0,0,0,0,0,0,0,0
0564   00
0565   00
0566   00
0567   00
0568   00
0569   00
056A   00
```

```
056B  00
056C  00
056D  00
056E  00
056F  00
0570  00
0571  00
0572  00
0573  00
0574  00
0575  00
0576  00
0577  00
0578  00
0579  00       1160               DEFW0,0,0,0,0,0,0,0,0,0,0
057A  00
057B  00
057C  00
057D  00
057E  00
057F  00
0580  00
0581  00
0582  00
0583  00
0584  00
0585  00
0586  00
0587  00
0588  00
0589  00
058A  00
058B  00
058C  00
058D  00
058E  00
058F  00       1170               DEFW0,0,0,0,0,0,0,0,0,0,0
0590  00
0591  00
0592  00
0593  00
0594  00
0595  00
0596  F7
0597  FF
0598  FF
0599  FF       1180               DEFW0,0,0,$FFF7,$FFFF
059A  FF
059B  FF
059C  FF
059D  FF
059E  FF
059F  FF       1190               DEFW$FFFF,$FFFF,$FFFF
               1200
               1210        START is main entry for
```

```
                1220    ALOAD of CREATE.
                1230
05A0   B10A     1240    START:   LDA    (ICURS),Y         ;first
05A2   C920     1250             CMP    #32               ;find the drive no
05A4   D003     1260             BNE    GETHX             ;on the cursor
05A6   C8       1270             INY                      ;line.
05A7   10F7     1280             BPL    START             ;till found...
                1290
05A9   88       1300    GETHX:   DEY                      ;adjust IY
05AA   2017F8   1310             JSR    JHXPK             ;pack DRV no.
05AD   7003     1320             BVS    OK1               ;check valid
                1330
05AF   4CC7B7   1340    ERREX:   JMP    JERRRE            ;mainexit
                1350
05B2   F0FB (BA4F)1360    OK1:   BEQ    ERREX             ;chck syntax
05B4   B10A     1370             LDA    (ICURS),Y         ;for (:)
05B6   C93A     1380             CMP    #$3A              ;terminator
05B8   D0F5     1390             BNE    ERREX
05BA   C8       1400             INY                      ;now make sure cursor
05BB   B10A     1410             LDA    (ICURS),Y         ;finished
05BD   10EA     1420             BPL    GETHX             ;else try again
05BF   A514     1430             LDA    HXPK+1            ;(must be 0)
05C1   D0EC     1440             BNE    ERREX             ;else error.
05C3   A513     1450             LDA    HXPK              ;now check value
05C5   30E8     1460             BMI    ERREX             ;is between 0-7
05C7   C908     1470             CMP    #8                ;for UNIT.
05C9   10E4     1480             BPL    ERREX             ;else error.
05CB   8D00B8   1490             STA    UNIT              ;OK so store it
05CE   20D6B7   1500             JSR    JBADDE            ;chk its there!
                1510
05D1   AD00B8   1520             LDA    UNIT              ;now get it back
05D4   48       1530             PHA                      ;and check if reverse
05D5   2901     1540             AND    #$1               ;side of disc is
05D7   8D8004   1550             STA    SIDNUM            ;required and
05DA   68       1560             PLA                      ;save it in sector
05DB   18       1570             CLC                      ;mask (WTRK).
05DC   6930     1580             ADC    #$30              ;make it ASCII
05DE   8D5B04   1590             STA    DRV               ;store in string
05E1   A950     1600             LDA    #$50              ;M4VEC
05E3   8540     1610             STA    MEMSEG            ;MSG4
05E5   A904     1620             LDA    #4                ;M4VEC+1
05E7   8541     1630             STA    MEMSEG+1          ;MSG4
05E9   20D9B7   1640             JSR    JOUTST            ;output MSG4
05EC   201DF8   1650             JSR    JPLKB             ;wait for key
05EF   A501     1660             LDA    ICHAR             ;get it
05F1   C91B     1670             CMP    #$1B              ;?esc
05F3   D003     1680             BNE    FORMAT            ;no so start
05F5   4C6F06   1690             JMP    FIN2              ;via VCURSN
                1700
05F8   2029F8   1710    FORMAT:  JSR    JCURSF
                1720    N.B. change this value in
                1730    bottom three bits of
                1740    IX for slower drives.
                1750    0=6mS,1=12mS,2=20mS and
                1760    3=30mS.
```

```
05FB  A200     1770               LDX    #0              ;1793 restore commd
05FD  200FB0   1780               JSR    JFDMAI          ;get to TRK 0
0600  201307   1790               JSR    INIT            ;go prepare SYS
0603  20C507   1800               JSR    VCURSF          ;sector & cursf
0606  CE0204   1810               DEC    TRKTOT          ;adjust
               1820      (INIT takes it one beyond
               1830      (proper total for calcs.
0609  A9FF     1840               LDA    #$FF            ;ready for bump
060B  8544     1850               STA    TNUM            ;to 0 by SETUP
060D  200CF8   1860               JSR    OUTRET          ;CR
               1870
0610  20A206   1880      WRTTRK:  JSR    SETUP           ;setup 1
0613  A900     1890               LDA    #0              ;complete track in
0615  8540     1900               STA    MEMSEG          ;memory @ $1000
0617  A910     1910               LDA    #$10            ;then set pointer
0619  8541     1920               STA    MEMSEG+1        ;bytes.
061B  A975     1930               LDA    #$75            ;OUTTRK LO
061D  8D04B8   1940               STA    DMAA            ;start of INT
0620  A906     1950               LDA    #6              ;OUTTRK HI
0622  8D05B8   1960               STA    DMAA+1          ;driven routine
               1970      same comments for this                 $177
               1980      byte as lines 1720-1750                $145
0625  A2F4     1990               LDX    #$F4            ;1793 wrt-track
0627  200FB0   2000               JSR    JFDMAI          ;do a track
062A  AD08B8   2010               LDA    ERROR   40      ;all ok?
062D  F003     2020               BEQ    OK2             ;0=yes
062F  4CAF05   2030               JMP    ERREX           ;else exit
               2040
0632  A900     2050         OK2:  LDA    #0              ;zero cursor
0634  8503     2060               STA    VDUIND          ;index
0636  A544     2070               LDA    TNUM            ;get track number
0638  48       2080               PHA                    ;save it temporarily
0639  201AF8   2090               JSR    JHXPN           ;print it to
063C  2029F8   2100               JSR    JCURSF          ;scrn,no cursor
063F  68       2110               PLA                    ;get it back
0640  CD0204   2120               CMP    TRKTOT          ;got to end ?
0643  F008     2130               BEQ    FINISH          ;yes so skip
               2140      (same comments on speed)
0645  A258     2150               LDX    #$58            ;1793 step-in
               2160      command
0647  200FB0   2170               JSR    JFDMAI          ;in 1 track
064A  4C1006   2180               JMP    WRTTRK          ;&back for more
               2190
064D  A900     2200      FINISH:  LDA    #0              ;set up all
064F  8D01B8   2210               STA    TRACK           ;pointers for
0652  A901     2220               LDA    #1              ;SYS sector to be
0654  8D02B8   2230               STA    SECT            ;written on 0,1
0657  A925     2240               LDA    #$25            ;which was prep'd
0659  8D04B8   2250               STA    DMAA            ;by the INIT subr
065C  A9B8     2260               LDA    #$B8            ;& has been wait-
065E  8D05B8   2270               STA    DMAA+1          ;ing patiently!
0661  20CDB7   2280               JSR    JWRTSC          ;catch it on
               2290      the way back to track 0
0664  A91B     2300               LDA    #$1B            ;M2VEC
0666  8540     2310               STA    MEMSEG          ;MSG2
```

```
0668   A904     2320             LDA    #4                  ;M2VEC+1
066A   8541     2330             STA    MEMSEG+1            ;MSG2
066C   20D9B7   2340             JSR    JOUTST              ;output MSG2
066F   20C807   2350      FIN2:  JSR    VCURSN              ;Vcursor o
0672   4CB2B7   2360             JMP    JESCAP              ;and finish.
                2370
                2380    This routine is used by
                2390    TANDOS routine FDMAIN @
                2400    $B00F under DRQ interrupt
                2410    control.
                2420
0675   2015B0   2430    OUTTRK:  JSR    JSETIN              ;prep
0678   58       2440             CLI                        ;all pointers & turnon
                2450
0679   AD94BF   2460     WAIT1:  LDA    CNTROL              ;get 1793
067C   0A       2470             ASL    A                   ;DRQ bit and wait
067D   90FA     2480             BCC    WAIT1               ;till it goes HI
067F   B140     2490             LDA    (MEMSEG),Y          ;get data
0681   C980     2500             CMP    #$80                ;end of track ?
0683   F00B     2510             BEQ    DONE                ;yes so go end
0685   8D93BF   2520             STA    DATA                ;else pass it
0688   C8       2530             INY                        ;to 1793 and bump IY
0689   D0EE     2540             BNE    WAIT1               ;till end page
068B   E641     2550             INC    MEMSEG+1            ;bump HIBYTE
068D   4C7906   2560             JMP    WAIT1               ;back for more
                2570
0690   A9FF     2580      DONE:  LDA    #$FF                ;replace 80
0692   8D93BF   2590             STA    DATA                ;with FF & sendit
0695   AD94BF   2600    DRQLOP:  LDA    CNTROL              ;now
0698   0A       2610             ASL    A                   ;go into endless
0699   90FA     2620             BCC    DRQLOP              ;loop and keep
069B   A9FF     2630             LDA    #$FF                ;writing FF's
069D   8D93BF   2640             STA    DATA                ;till 1793 finds
06A0   D0F3     2650             BNE    DRQLOP              ;index hole.
                2660
                2670    This routine sets up a
                2680    complete track in memory
                2690    from $1000 to $1BF3.
                2700    Each sector is fully
                2710    prepared and the Free-
                2720    space link pointers @ the
                2730    start of each sector are
                2740    are all pre-linked with
                2750    the sector sequence as
                2760    would have been produced
                2770    by the old INIT program.
                2780    Note that the SYS sector
                2790    on track 0 can't be fully
                2800    prepared because of the
                2810    possibility of bytes >$F0
                2820    being in the PAGDEF part
                2830    which would cause CRC
                2840    bytes to be written to
                2850    the disc instead.
                2860
```

```
06A2   E644     2870    SETUP:   INC    TNUM              ;bump TRK
06A4   A900     2880             LDA    #0                ;set up pointers
06A6   8542     2890             STA    ITRACK            ;to the prep'd
06A8   8545     2900             STA    SCNT              ;track setting
06AA   A910     2910             LDA    #$10              ;sect count also
06AC   8543     2920             STA    ITRACK+1          ;index=$1000
                2930
06AE   E645     2940   NXTSCT:   INC    SCNT              ;bump SECT
06B0   A96E     2950             LDA    #$6E              ;WTRKLO
06B2   8540     2960             STA    MEMSEG            ;set up for
06B4   A904     2970             LDA    #4                ;start of sector
06B6   8541     2980             STA    MEMSEG+1          ;image
06B8   A544     2990             LDA    TNUM              ;get & store in
06BA   8D7F04   3000             STA    TRKNUM            ;image & user
06BD   A545     3010             LDA    SCNT              ;store true
06BF   8D8104   3020             STA    TRKNUM+2          ;sector no.
06C2   AA       3030             TAX                      ;now get link pointer
06C3   BD0304   3040             LDA    SECSEQ-1,X        ;for user
06C6   A644     3050             LDX    TNUM              ;then track
06C8   8D9704   3060             STA    STRKO+1           ;store sct ptr
06CB   C901     3070             CMP    #1                ;if 1 then bump trk
06CD   D001     3080             BNE    INLINK            ;if not skip
06CF   E8       3090             INX                      ;point last sct to new
06D0   8E9604   3100   INLINK:   STX    STRKO             ;track
06D3   A000     3110             LDY    #0                ;zero IY
06D5   B140     3120   TFRSCT:   LDA    (MEMSEG),Y        ;get
06D7   9142     3130             STA    (ITRACK),Y        ;store it
06D9   E640     3140             INC    MEMSEG            ;adjust all
06DB   D002     3150             BNE    NXT1              ;pointers looking
06DD   E641     3160             INC    MEMSEG+1
06DF   E642     3170     NXT1:   INC    ITRACK
06E1   D002     3180             BNE    NXT2
06E3   E643     3190             INC    ITRACK+1          ;for $5A0 as
06E5   A541     3200     NXT2:   LDA    MEMSEG+1
06E7   C905     3210             CMP    #5                ;that means we've
06E9   D0EA     3220             BNE    TFRSCT            ;reached the
06EB   A540     3230             LDA    MEMSEG            ;end of a
06ED   C9A0     3240             CMP    #$A0              ;completed
06EF   D0E4     3250             BNE    TFRSCT            ;sector
06F1   A544     3260             LDA    TNUM              ;if trkO then O
06F3   D003     3270             BNE    NTRKO             ;the 2nd link
06F5   8DBF13   3280             STA    DIRSO             ;ptr on DIR sct
06F8   A543     3290    NTRKO:   LDA    ITRACK+1          ;look
06FA   C91B     3300             CMP    #$1B              ;for end of track
06FC   D0B0     3310             BNE    NXTSCT            ;else keepgoing
06FE   A980     3320             LDA    #$80              ;mark it!
0700   88       3330             DEY                      ;adjust IY
0701   9142     3340             STA    (ITRACK),Y        ;store it
0703   A544     3350             LDA    TNUM              ;look for end of
0705   CD0204   3360             CMP    TRKTOT            ;disc...
0708   D008     3370             BNE    NOTEND            ;else return
070A   A900     3380             LDA    #0                ;here we zero the
070C   8D8618   3390             STA    LASTSC            ;last free
070F   8D8718   3400             STA    LASTTR            ;space pntrs
0712   60       3410   NOTEND:   RTS
```

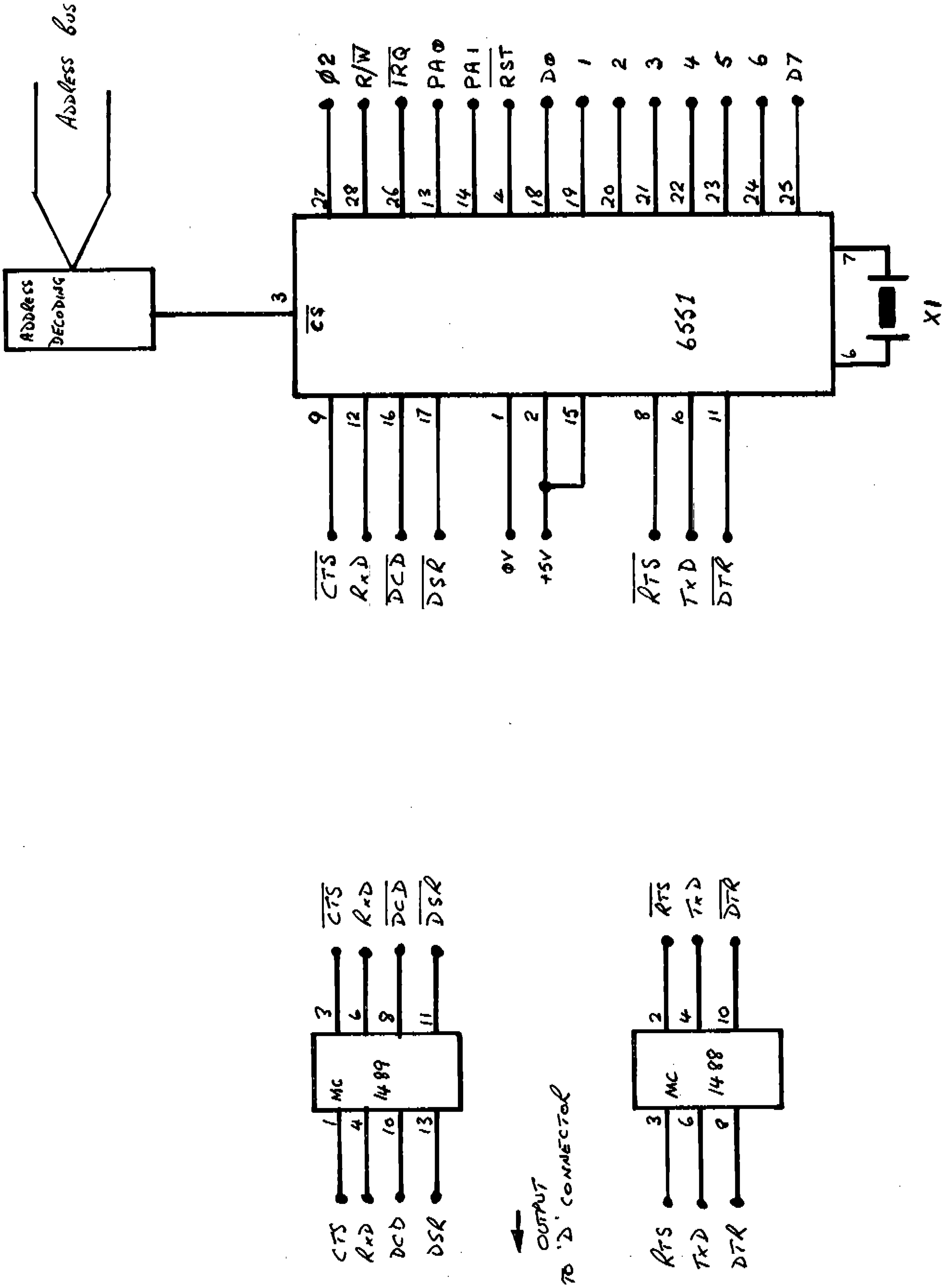
```
                3420
                3430   This routine prepares the
                3440   system sector in the SECT
                3450   buffer @ $B825 ready for
                3460   when FORMAT has finished
                3470   it's work.
                3480
0713  A90E      3490     INIT:  LDA    #$E             ;M1VEC
0715  8540      3500            STA    MEMSEG          ;ask for name
0717  A904      3510            LDA    #4              ;M1VEC+1
0719  8541      3520            STA    MEMSEG+1
071B  20D9B7    3530            JSR    JOUTST          ;here !!
071E  20C4B7    3540            JSR    JGETLI          ;get it
0721  8D43B9    3550            STA    LINLEN          ;savelen
0724  C90A      3560            CMP    #10             ;?<10
0726  3006      3570            BMI    OK3             ;ok if <=9
0728  20CAB7    3580   OUTQRY:  JSR    JQUERY          ;o/p ?
072B  4C1307    3590            JMP    INIT            ;& again
072E  2029F8    3600      OK3:  JSR    JCURSF          ;rid cursor
0731  A920      3610            LDA    #32             ;then clear out
0733  A200      3620            LDX    #0              ;buffer to spaces
0735  9D25B8    3630    LOOP1:  STA    SBUFO,X
0738  E8        3640            INX
0739  D0FA      3650            BNE    LOOP1
073B  A20F      3660            LDX    #15             ;get current def.
073D  BD15B8    3670    LOOP2:  LDA    DSCDEF,X        ;& sav
0740  9D25B8    3680            STA    SBUFO,X         ;in
0743  CA        3690            DEX                    ;buffer for
0744  10F7      3700            BPL    LOOP2           ;modding
0746  A900      3710            LDA    #0              ;prepare
0748  8D36B8    3720            STA    FTRK            ;for a
074B  8D38B8    3730            STA    DTRK            ;fresh
074E  8D3BB8    3740            STA    USED            ;disc
0751  8D3CB8    3750            STA    USED+1          ;1st dir
0754  A904      3760            LDA    #4              ;sect= 0,4
0756  8D37B8    3770            STA    DSCT            ;1st free
0759  A907      3780            LDA    #7              ;= 0,7
075B  8D35B8    3790            STA    FSCT            ;prepare
075E  A2FF      3800            LDX    #$FF            ;for calc
0760  8E3AB8    3810            STX    FCNT+1          ;of free
0763  CA        3820            DEX                    ;space cnt
0764  8E39B8    3830            STX    FCNT
0767  AE00B8    3840            LDX    UNIT            ;get no.of
076A  BD15B8    3850            LDA    DSCDEF,X        ;trks
076D  AA        3860            TAX                    ;on this
076E  8D0204    3870            STA    TRKTOT          ;drive
0771  AC03B8    3880            LDY    DENS            ;get no.of
0774  B90004    3890            LDA    A1,Y            ;sectors
0777  8D0304    3900            STA    SCTTOT          ;save it
077A  EE0304    3910            INC    SCTTOT          ;bump it
077D  AD39B8    3920    LOOP3:  LDA    FCNT            ;now calc
0780  18        3930            CLC                    ;total no
0781  790004    3940            ADC    A1,Y            ;of sects
0784  8D39B8    3950            STA    FCNT            ;for this
0787  AD3AB8    3960            LDA    FCNT+1          ;drive
```

```
078A  6900    3970            ADC   #0
078C  8D3AB8  3980            STA   FCNT+1          ;keep on
078F  CA      3990            DEX                   ;till
0790  D0EB    4000            BNE   LOOP3           ;finished
0792  A200    4010            LDX   #0
0794  BD3504  4020    LOOP4:  LDA   MSG3,X          ;put
0797  F006    4030            BEQ   DOTITL          ;TANDOS
0799  9D65B8  4040            STA   DOSVER,X        ;V1.1
079C  E8      4050            INX                   ;in buffer
079D  D0F5    4060            BNE   LOOP4
079F  A208    4070   DOTITL:  LDX   #8              ;now do disc
07A1  A920    4080            LDA   #32             ;title
07A3  9D3DB8  4090    LOOP5:  STA   TITLE,X
07A6  CA      4100            DEX
07A7  10FA    4110            BPL   LOOP5
07A9  A403    4120            LDY   VDUIND          ;check
07AB  AE43B9  4130            LDX   LINLEN          ;for all
07AE  88      4140   NXTCHR:  DEY                   ;valid chrs
07AF  CA      4150            DEX
07B0  3012    4160            BMI   SAVSYS
07B2  B10A    4170            LDA   (ICURS),Y
07B4  20FAB7  4180            JSR   JALPNU
07B7  C900    4190            CMP   #0
07B9  D003    4200            BNE   SAVIT
07BB  4C2807  4210            JMP   OUTQRY
07BE  9D3DB8  4220    SAVIT:  STA   TITLE,X
07C1  4CAE07  4230            JMP   NXTCHR
07C4  60      4240   SAVSYS:  RTS
              4250
              4260   These are a bit of
              4270   cosmetics for VDU board
              4280   users. Note that OUTVDU
              4290   is NOT used to prevent
              4300   a crash on non-TUGBUG
              4310   system.
              4320
07C5  A932    4330   VCURSF:  LDA   #'2
07C7  2C      4340            DEFB$2C
07C8  A931    4350   VCURSN:  LDA   #'1
07CA  48      4360            PHA
07CB  A91B    4370            LDA   #$1B
07CD  20DE07  4380            JSR   SENCHR
07D0  A95B    4390            LDA   #'[
07D2  20DE07  4400            JSR   SENCHR
07D5  68      4410            PLA
07D6  20DE07  4420            JSR   SENCHR
07D9  A962    4430            LDA   #'b
07DB  4CDE07  4440            JMP   SENCHR
              4450
07DE  2C00BE  4460   SENCHR:  BIT   VDUSTA
07E1  10FB    4470            BPL   SENCHR
07E3  8D01BE  4480            STA   VDUCTL

No errors.
End of object = $07E6
```

Colin Nowell
==============================

RS 232C OPTION



THE TANDOS ALTERNATIVE

Why not construct your own DOS controller system?

The following design was constructed by David Cawthorne and has proved to be totally successful. Because of limited space in Tandoc only the circuit diagram, board layout and component listing have been included. Those of you which require the full details as submitted by David, please send a cheque for 1.00 the cover postage and handling to Mandarin Micro Systems Ltd., for your copy.

The circuit is based on the WD2797 Floppy Disk Controller, which will run with the Tandos Operating System by changing only two bytes within the Eprom. NOTE:- This operating system and system disk must be obtained from MCS.

The software modification to Tandos:-

$B1D7 = AC $B1DB = 8C

This corrects the READ and WRITE commands for use with 2797, will still run the 1793.

Components Required

WD2797 FDC I.C.
13 10nF decoupling capacitors
1 5.5 to 65 pF variable capacitor
1 1N4148 signal diode

IC's

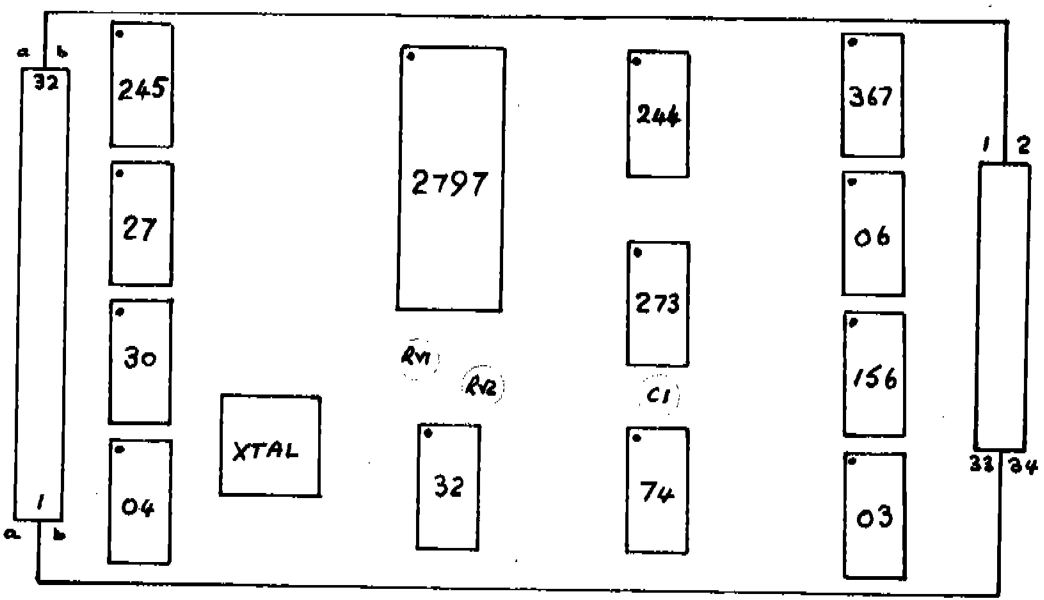
74LS245
74LS367
74LS32
74LS244
74LS30
74LS03 or 38
74LS273
74LS27
74LS04
74LS156
74LS74
7406

RESISTORS

1 47K Min Pot
2 470R
1 10K Min Pot
5 150R
1 1K
1 2K2
1 3K3

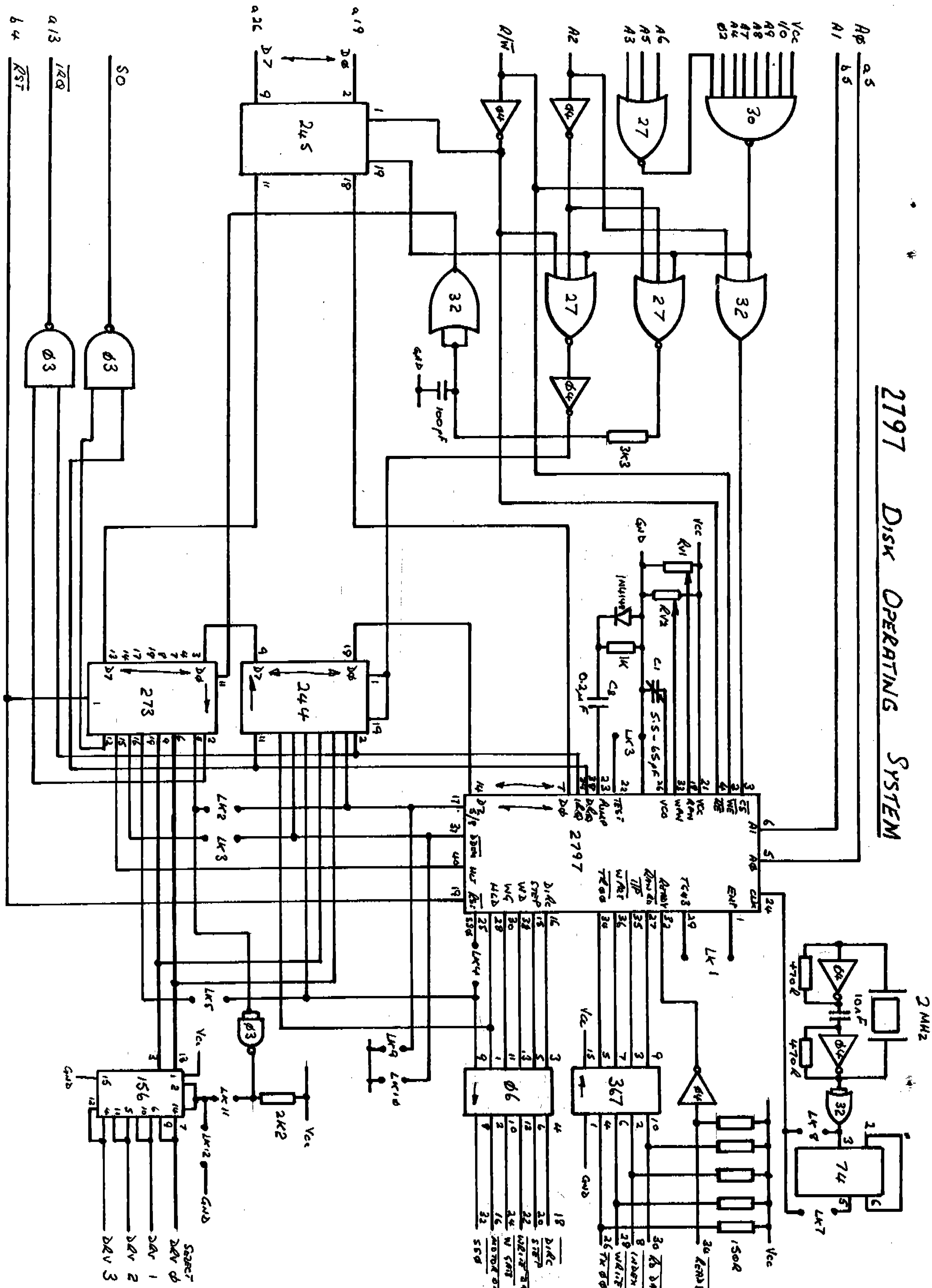
MISC

1 0.2uF capacitor
1 2MHz Crystal
1 100pF capacitor
1 10nF capacitor



COMPONENT SIDE

2797 DISK OPERATING SYSTEM



READ A 40 TRACK DISK
with a
80 TRACK DISK DRIVE

```
R40W80                                  PAGE :  1      DATED : 01.07.85

                     10      READ A 40 TRACK DISC WITH A
                     20      80 TRACK DISC DRIVE
                     30      FOR 2797 FDC SEE LINE 760
                     40
                     50      START ADDRESS = $9043
                     60
                     70      TANDOS NEEDS TO BE IN RAM
                     80
                     90      the program exits into the
                     100     monitor $FC00 so that the
                     110     SYS ram is zeroed, thus SYS
                     120     is then of the net (40
                     130     track) disc
                     140
                     150
                     160     SYS NOTES: if track size
                     170     is under 41 it treats that
                     180     DRVn as a 40 track disc
                     190     drive-----if track size
                     200     is over 40 it treats that
                     210     DRVn as normal (80 track).
                     220
                     230
                     240              UNIT      =    $B800
                     250              TRACK     =    $B801
                     260              DSCDEF    =    $B815
                     270              TRKREG    =    $BF91
                     280              EXIT1     =    $B207
                     290              FDMAIN    =    $B1DC
                     300              SSOSET    =    $AFE8
                     310              MONITR    =    $FC00
                     320              CWORD     =    $B807
                     330
                     340              *         =    $9000
                     350
9000   48            360     DRVSEL:  PHA
9001   98            370              TYA
9002   48            380              PHA
9003   AC00B8        390              LDY   UNIT
9006   B915B8        400              LDA   DSCDEF,Y
9009   C929          410              CMP   #$29
900B   B033          420              BCS   EXIT                ;80 TRACK DISC
900D   8A            430              TXA                       ;FDMAIN COMMAND
900E   3030          440              BMI   EXIT                ;no track move
9010   29F0          450              AND   #$F0
9012   F02C          460              BEQ   EXIT                ;restore
9014   C910          470              CMP   #$10
9016   F007          480              BEQ   SEEK
9018   203D90        490              JSR   JFDMAN
901B   A248          500              LDX   #$48
901D   D021          510              BNE   EXIT                ;step,no update
901F   AD01B8        520        SEEK: LDA   TRACK
9022   0A            530              ASLA                      ;X2
9023   8D01B8        540              STA   TRACK
9026   AD91BF        550              LDA   TRKREG
```

```
R40W80                                         PAGE :  2      DATED : 01.07.85

9029  0A        560               ASLA                       ;X2
902A  8D91BF    570               STA   TRKREG
902D  203D90    580               JSR   JFDMAN
9030  AD01B8    590               LDA   TRACK
9033  4A        600               LSRA                       ;/2
9034  8D01B8    610               STA   TRACK
9037  8D91BF    620               STA   TRKREG
903A  4C07B2    630               JMP   EXIT1
903D  48        640      JFDMAN:  PHA
903E  98        650               TYA
903F  48        660               PHA
9040  4CDFB1    670        EXIT:  JMP   FDMAIN+3
9043  A205      680       START:  LDX   #$5
                690
9045  BD5190    700               LDA   PATCH,X
9048  9DDCB1    710               STA   FDMAIN,X
904B  CA        720               DEX
904C  10F7      730               BPL   START+2
904E  4C00FC    740               JMP   MONITR
                750
                760      1797 PATCH
9051  4C0090    770       PATCH:  JMP   DRVSEL
9054  8E07B8    780               STX   CWORD
                790
                800      2797 PATCH
                810      PATCH:JMP DRVSEL
                820      JSR SSOSET
                830      see DOSMOD ,jump to $AFE8
                840      to set 2797 FDC SSO pin.

No errors.
End of object = $9054
```

David Cawthorne

WANTED. NEW OWNER

As I shall be no longer running the User Group, it needs a new owner or it will cease to exist. I believe this would be very damaging to the Microtan system and most detrimental to existing owners. So, is there anybody with a lot of time to spare who would like to take over?

The assets of MOUG are:-

1. Approximately 5 MBYTES of software in source and object code.
2. Documentation for the above plus an enormous amount of documentation on the system in general
3. Current mailing list in access of 180 persons.
4. Potential yearly turnover in excess of 5k.

If you are interested, then please contact:-

Brian Gibbs
48 Brightstowe Road
Burnham on Sea
Somerset, TA8 2HP.

Tel: 0278 786701

****

Mandarin Micro Systems Ltd.

48 Brightstowe Road, Burnham on Sea, Somerset TA8 2HP.

****

Mandarin Micros has taken on the marketing of Intelgraph. It is also able to supply all your computing needs from stationery to components at very competitive prices. Send SAE for list.

Available late 85, a complete Industrial Standard 6809 Micro Processor, comprising of :-
2Mhz 6809 processor.
Twin industrial standard RS232C interfaces.
Industrial standard Disk Operating System supporting WD2797 enabling use of 3.5, 5.25 or 8 inch disk drives running double density.
Eprom Blower for 2764's , 27128's and 27256's.

Intelligent Terminal Intelgraph or users own.
96 way ac Bus.

The above system will run FLEX.

<u>Current Products</u>

Intelgraph PCB and Manual	38.50
Intel-T Terminal Operating System..........	15.00
DIN 41612 64 way ab Plug	2.50
DIN 41612 64 way ab Socket	2.95

All R.S. Components available, why not drop us a line for a quotation. SAE please.

Also high quality PCB materials and prototype boards.

To sum up - "You name it, Mandarin will supply it"

STOP PRESS

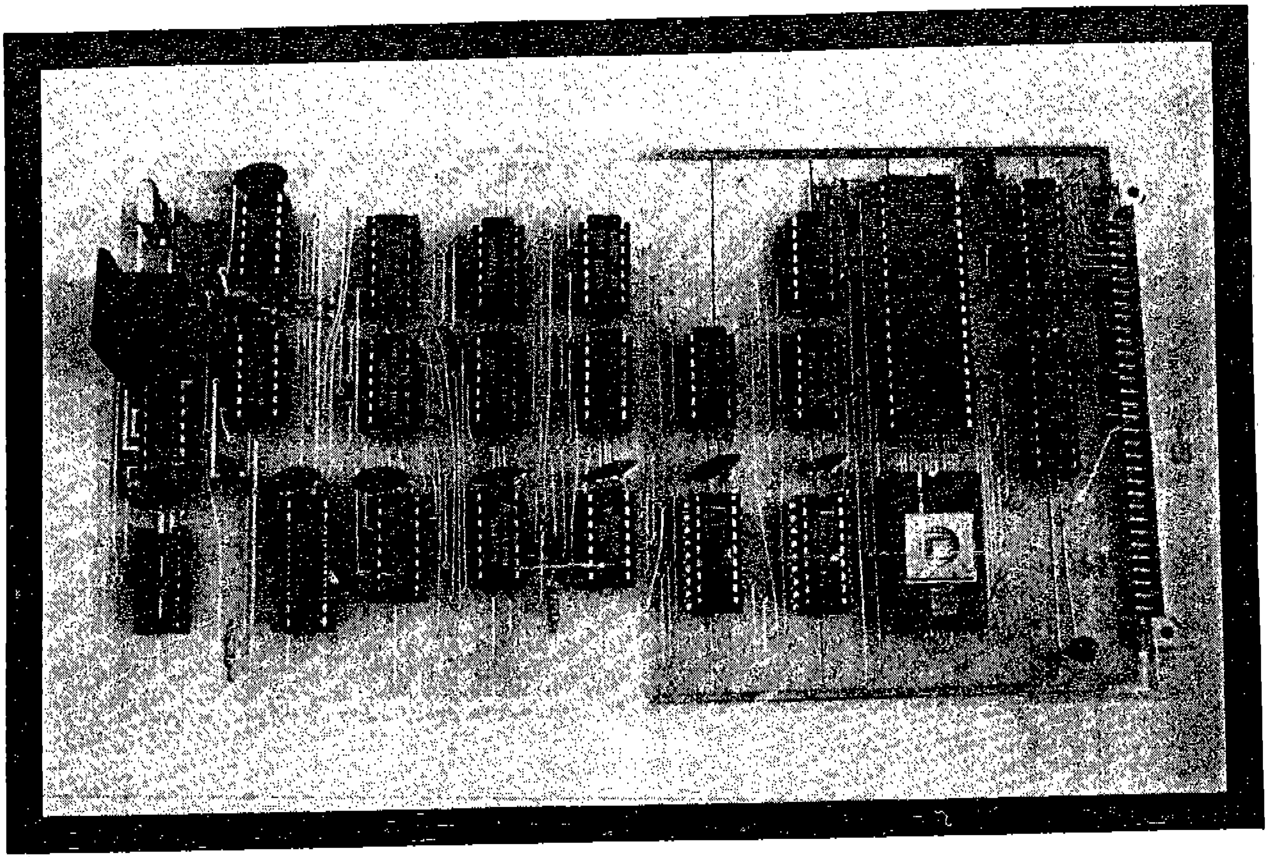
6264LP - 15 CMOS 8K RAMS.........4.50 each inclusive P&P

Ralph Allen Engineering Co.

FORNCETT END NORWICH
NORFOLK ENGLAND
TEL: BUNWELL (095389) 420

Reg. No. 1711588 V.A.T. No. 353 2120 00

* * * 6809 C.P.U. CARD * * *



This 6809 C.P.U. Issue 2 card will plug into the 65 slot on a standard Tangerine system rack replacing the 6502 card, this then turns your system into a full stand-alone 6809 Flex computer unlike other cards on the market that turn your 6502 into a terminal talking to a 6809 card.

All your original cards can still be used as well as a range of cards that we are developing to run with this system. The main advantage of changing to the 6809 apart from its 4 16 bit registers and 2 accumulators is a disc operating system called FLEX this is a very good operating system that opens the way to hundreds of professional programs that will run under it.

The only addition that is needed to a standard system is the need to have Ram where Tanex now has Eproms, we have a 14K Ram card that can be pluged into an additional slot or you can use our 32K Ram card or convert your Tanex to Ram, to make full use of the 4 Eprom monitors we supply for either the 80/82 or MPD colour Video cards you will need either the 32K Tangerine Rom card or our 32K Eprom ram card, although neither of these are essential also one or other of the above mentioned video cards is needed.

PRICE 6809 £24.50p BARE P.C.B. + VAT and £1.50p postage.
FLEX OPERATING SYSTEM £85.00p + VAT
PLEASE PHONE FOR BUILT AND TEST PRICES.