TOSIE Toronto Ohio Scientific Idea Exchange

T. O. S. I. E.
P.O. Box 29
Streetsville, Ontario
Canada L5M 2B7

The TOSIE PRINTOUT / April 1, 1984

T.O.S.I.E. is a non-profit user's group for Ohio Scientific home computer users. The TOSIE Printout is published by TOSIE approx. ten times a year. For more information please write to us at the above address or call one of our executive members.

Club Moderator...Paul Chidley...705-292-8004
C1 Editor........John Horemans..416-826-5362
C4 Editor........Ed Maste.......416-839-9493
C8 Editor........Paul Vail......416-622-0599
Treasurer/Secr...David Cho......416-494-3567

Contents:

Club News: by Paul C.

- Please make note of my new address and phone number. Despite the many miles between me and Toronto I do intend to continue attending our regular meetings. The past few months have been very busy for me so I have made little contribution to our club's activities, I hope this can change in time for our next newsletter. It so happens that this has also been a busy time for the rest of our executive, so if anyone has been disappointed with the recent slump I can only say we're sorry and ask how many articles you contributed lately.

Paul Chidley
R.R. #2
Ennismore, Ontario
KOL 1TO

- Gemini 10X, this printer has been causing quite a stir lately with its low prices, however if you bought one you may already have a problem. I recently learned, from a friend and by experience, that a number of Gemini were shipped with defective print heads. The heads work fine when new but the print quality rapidly deteriorates with use. If your Gemini prints light, misses the first character, or doesn't always print the descenders on small letters then you have one of the bad heads. Now for the bad news, Star Electronics doesn't officially admit there was a problem with the heads, so if you don't catch it in the 90 day warranty you'll have to buy a new one at about $70.00 Can. I am told this problem was limited and that the new heads don't have a problem, this front page was printed on my Gemini with a new head and as you can see it has worked fine so far.

- NOTE: Our club's meetings are still the last Sunday of the month. Today's meeting (Apr 1/84) is on the first Sunday only because we got bounced from our normal spot.

- Our annual 'elections' should be coming up soon so if anyone would like to help out in the executive let me know and I'll make sure you are elected for something.

A Common 5.25" drive Interface Problem by Paul C.

In the past I have been asked to repair many different OSI computers. My last adventure reminded me of a problem that has bitten me more than once so I would like to share it with you.

The symptom is a common one, "My system won't boot up". If you examine further you will find that the disk is indeed accessed and that the first track (track 0) is put into memory but it doesn't seem to execute. You can determine this by using the 65V monitor to record the memory contents from $2200 to $2210 and from $29F0 to $2A10. Once recorded you can then hit the break key and try to boot from the disk, if you have our problem the machine then appears to go to never-never-land. You can then use the 65V monitor to re-examine the same memory contents where you should find that $2200 to $29FF equals the contents of track zero as shown in table 1. The table is taken from a 5.25" 65D V3.2 disk, differences may of course be present with different versions. The memory greater than $2A00 however has not changed.

Now that we know track 0 is being loaded the question is whether or not it is executing, i.e. does the CPU jump to location $2200 for its next instruction? This can be tested with a simple program such as the one in listing 1. This program was intended to be put on data and other such disks that did not have an operating system on them, then when you try to boot it you get the message on your screen. If such a program will boot on your system you have just proved that track 0 does get loaded and that the CPU does jump to $2200 and execute the machine code found there.

The next step is to determine why the drive does not step to track 1. The program in listing 2 can be merged into a 65D V3.2 disk on track 0. When this disk is then booted it allows you three commands, H to home the head to track 0, O to step the head out and I to step the head in. The command is reflected when entered followed by the track number in decimal followed by the disk's PIA status in hex. If your drive does not behave as expected with this program you have a different problem than the one I'm building up to. Assuming that the program does behave we now know that the drive does step properly so let's look at the status word. Broken into binary the meaning of the bits is listed in table 2. A healthy drive will display a status of $EE or $EC if on track 0, but lets look at bit number zero. This bit is a left over from the OSI 8" disk interface, with the exception of some very new models, 5.25" drives don't have a drive ready line. If bit 0 is equal to 1 then we have just found our problem.

When you hit "D" to boot the disk your system loads track 0 into memory at $2200 and then does a jump to that address. If you examine the code at $2200 you would find that one of the very first things it tries to do is load track 1. It does this by loading the accumulator equal to one (the target track number) and then jumping to the subroutine at $26BC. This subroutine is the standard one used by the operating system. When this routine executes it checks for drive ready, which in this case we don't have, so it then jumps to the error entry point at $2A4B to report ERROR #6 drive not ready. The problem is that the error reporting routines are in memory greater than $2A00, i.e. they are on track 1 which hasn't been loaded yet. The result is that the CPU jumped to a location in memory still full of garbage.

The solution to the problem is therefor quite simple. Just make sure that the drive 0 ready line (pin 2 of the interface's PIA) is grounded. "So why did we do all those steps above if the answer was so easy?" Simple, now that you know WHY the drive is doing what it is doing you don't have to do all those steps, just make sure the line is grounded.

I hope this helps people further understand but I especially hope it saves someone a day (or days) of trouble shooting an easy problem. If you have any problems or questions make sure you ask, that's what user's groups are supposed to be for.

Table 1
=======

addr	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
2200	A9	01	20	BC	22	20	BC	26	A9	2A	85	FF	20	54	27	86
29F0	DF	00	DF	AA	F0	F1	48	20	BC	26	20	73	2D	0D	0A	54
2A00	52	41	43	4B	20	00	68	20	92	2D	BA	86	FC	20	54	27

Table 2
=======

Bit	Function
0	Drive 0 Ready (0 if ready)
1	Track 0 (0 if at track 0)
2	Fault (0 if fault, 8" drives only)
3	Not Used (usually = 1)
4	Drive 1 Ready (0 if ready)
5	Write Protect (0 if write protected)
6	Drive Select (1 = A or C, 2 = B or D)
7	Index (0 if at index hole)

Listing 1

```
.P

  10; TRACK ZERO PROGRAM FOR DATA DISKETTES
  20;
  30; PLACE ON TRACK ZERO OF DISKETTES WHICH
  35; DO NOT HAVE A FULL OPERATING
  40; SYSTEM ON THEM
  50;
  60; By Leroy Erickson, 1981.  *OSMOSUS **
  70;
  80          *=$2200
  90          CLD          ;CLEAR THE DECIMAL FLAG
 100          LDA #$D0     ;CLEAR THE SCREEN
 110          STA $FF      ;
 120          LDA #0       ;
 130          STY $FE      ; $FE,$FF = $D000
 140          LDA #$20     ; GET A BLANK
 150 LOOP1    STA ($FE),Y  ; STORE IT
 160          INY          ; INCR INDEX
 170          BNE LOOP1    ; LOOP FOR EACH PAGE
 180          INC $FF      ; INCR PAGE PTR
 190          LDX $FF      ;
 200          CPX #$D8     ; DONE? ; $D4 FOR C1P
 210          BNE LOOP1    ; NO, KEEP GOING
 220          LDA #$D4     ; SCREEN MIDDLE $D2 FOR C1
 230          STA $FF      ;
 240          LDA #$40-MSGLEN/2 ; LEFT MARGIN
 250          STA $FE      ; CENTERED ON LINE
 260          LDY #0       ; ZERO THE INDEX
 270 LOOP2    LDA MESSAG,Y       ; GET CHR
 280          BEQ DONE     ; ZERO IS END OF MESSAGE
 290          STA ($FE),Y  ; STORE IT
 300          INY          ; BUMP
 310          BNE LOOP2    ; LOOP TILL END
 320 DONE     JMP DONE     ; STAY HERE FOREVER
 330 MESSAG .BYTE'*** THIS DISK IS NOT BOOTABLE! ***',0
 340 MSGLEN=*-MESSAG
 350          .END         ; THAT'S ALL FOLKS!!!
```

Listing 2
=========

```
.A

 10                    ;*****************************************
 20                    ;*                                       *
 30                    ;* DSTTRO - Disk Stepper Tester on Track 0 *
 40                    ;*                                       *
 50                    ;* by Paul C. - March 10,1984            *
 60                    ;*                                       *
 70                    ;*****************************************
 80                    ;
 90                    ;
100 2200                       * = $2200
110                    ;
120 2683=                      STEPIN = $2683
130 268A=                      STEPOT = $268A
140 2663=                      HOME   = $2663
150 265D=                      TRKNUM = $265D
160 FD00=                      KEYPOL = $FD00
170 2343=                      PRINT  = $2343
180 DE00=                      VIDSIZ = $DE00
190 2321=                      INDST  = $2321
200 2322=                      OUTDST = $2322
210 C000=                      FLOPIN = $C000
220 29C6=                      SETDRV = $29C6
230 00E0=                      TS1    = $00E0
240                    ;
250 2200 A000                  LDY #$00
260 2202 8C01C0                STY FLOPIN+1
270 2205 C8                    INY
280 2206 8C00DE                STY VIDSIZ
290 2209 C8                    INY
300 220A 8C2123                STY INDST
310 220D 8C2223                STY OUTDST
320 2210 A040                  LDY #$40
330 2212 8C00C0                STY FLOPIN
340 2215 A004                  LDY #4
350 2217 8C01C0                STY FLOPIN+1
360 221A A901                  LDA #1
370 221C 20C629                JSR SETDRV
380 221F 20D122                JSR SCLEAR
390 2222 A000                  LDY #$00
400 2224 B9F522   FP1          LDA MESSAG,Y
410 2227 F00F                  BEQ S2
420 2229 204323                JSR PRINT
430 222C C8                    INY
440 222D D0F5                  BNE FP1
450 222F 207822   START        JSR CONVRT
460 2232 20DC22                JSR STATUS
470 2235 20AF22                JSR CRLF
480 2238 2000FD   S2           JSR KEYPOL
490 223B C949                  CMP #$49
500 223D D013                  BNE S1
510 223F 20A422                JSR CPRINT
520 2242 AE5D26                LDX TRKNUM
530 2245 8A                    TXA
540 2246 F0E7                  BEQ START
550 2248 CA                    DEX
```

```
2249 8E5D26          STX TRKNUM
224C 208326          JSR STEPIN
224F 4C2F22          JMP START
2252 C94F     S1     CMP #$4F
2254 D015            BNE S3
2256 20A422          JSR CPRINT
2259 AE5D26          LDX TRKNUM
225C E8              INX
225D 8A              TXA
225E C928            CMP #40
2260 B0CD            BCS START
2262 8E5D26          STX TRKNUM
2265 208A26          JSR STEPOT
2268 4C2F22          JMP START
226B C948     S3     CMP #$48
226D D0C9            BNE S2
226F 20A422          JSR CPRINT
2272 206326          JSR HOME
2275 4C2F22          JMP START
              ;
2278 AD5D26   CONVRT LDA TRKNUM
227B 38              SEC
227C A2FF            LDX #$FF
227E E8              INX
227F E90A            SBC #10
2281 B0FB            BCS *-3
2283 690A            ADC #10
2285 85E0            STA TS1
2287 8A              TXA
2288 0A              ASL A
2289 0A              ASL A
228A 0A              ASL A
228B 0A              ASL A
228C 05E0            ORA TS1
228E 85E0            STA TS1
2290 48       PRT2HX PHA
2291 4A              LSR A
2292 4A              LSR A
2293 4A              LSR A
2294 4A              LSR A
2295 209922          JSR PRTHEX
2298 68              PLA
2299 290F     PRTHEX AND #$0F
229B C90A            CMP #$0A
229D F8              SED
229E 6930            ADC #$30
22A0 D8              CLD
22A1 4C4323          JMP PRINT
              ;
22A4 204323   CPRINT JSR PRINT
22A7 48              PHA
22A8 A920            LDA #$20
22AA 204323          JSR PRINT
22AD 68              PLA
22AE 60              RTS
              ;
22AF A90D     CRLF   LDA #$0D
22B1 204323          JSR PRINT
22B4 A90A            LDA #$0A
```

```
1150 22B6 4C4323            JMP PRINT
1160                 ;
1170 22B9 A920       SCLSUB LDA #$20
1180 22BB A008              LDY #$08
1190 22BD A200              LDX #$00
1200 22BF 9D00D0     SCL1   STA $D000,X
1210 22C2 E8                INX
1220 22C3 D0FA              BNE SCL1
1230 22C5 EEC122            INC SCL1+2
1240 22C8 88                DEY
1250 22C9 D0F4              BNE SCL1
1260 22CB A9D0              LDA #$D0
1270 22CD 8DC122            STA SCL1+2
1280 22D0 60                RTS
1290                 ;
1300 22D1 A9E0       SCLEAR LDA #$E0
1310 22D3 8DC122            STA SCL1+2
1320 22D6 20B922            JSR SCLSUB
1330 22D9 4CB922            JMP SCLSUB
1340                 ;
1350 22DC A920       STATUS LDA #$20
1360 22DE 204323            JSR PRINT
1370 22E1 AD00C0            LDA FLOPIN
1380 22E4 209022            JSR PRT2HX
1390 22E7 60                RTS
1400                 ;
1410 22E8 208326     PATCH  JSR $2683
1420 22EB E6FD              INC $FD
1430 22ED D005              BNE P1
1440 22EF A906              LDA #$06
1450 22F1 204323            JSR PRINT
1460 22F4 60         P1     RTS
1470                 ;
1480 22F5 48         MESSAG .BYTE 'H/I/O ?',$A,$A,$D,0
1480 22F6 2F
1480 22F7 49
1480 22F8 2F
1480 22F9 4F
1480 22FA 20
1480 22FB 3F
1480 22FC 0A
1480 22FD 0A
1480 22FE 0D
1480 22FF 00
1490                 ;
1500 2673                   *=$2673
1510 2673 20E822            JSR PATCH
1520 267A                   *=$267A
1530 267A A062              LDY #$62
```

RAM AT $C800 AND $8000

John Horemans TOSIE

NOTE: All these changes were made on a RevD Superboard. I believe the Rev A is identical for this project, but do check it.
NOTE: The diagrams are on page 10 of this issue.

Why would you put a 2K block of memory up there? Mine gets used for the Extended monitor, relocated to there. It can be placed there without fear of conflicting with the programs that you may want to modify. A Basic utility, that includes a machine code renumber as well as Search and Surchange (Micro, August 82) also fit there.

A further small hardware change allows this 2k block to be switched into $F800, the monitor rom space, and allows you to load up a new monitor rom into $C800, and then switch it into $F800. You can thus modify a monitor rom to suit, and keep it on cassette or disk.

Since the 74LS139 decoder has two separate sections, and one is still unused, we can also decode $8000 to $9FFF, the 8k block just below BASIC, and extend a 32K machine to 40K. Of course, the same thing can be done for any 8k block, it only needs the proper 8K select from U23.

Where to put the 2 extra chips and the 24 pin rams is a problem. You could make a separate board, or in the hacker style, piggyback a few chips. They can be piggybacked onto the BASIC roms, except for pins 18, 20 and 21. Alternately, a cable can be run from one of these sockets to a board with the new ram chips and the removed rom.

If the 4 Basic roms are put into 2732's, and a few changes made to the decoding, one can have two empty sockets to work with. This is where mine are. the $C800 chip occupies the one socket, the four 2K ram chips are in the other. (Yes-four on top of each other!) The soldering does not seem to be a problem. All the pins can be soldered together except for pin 18.

The figures should be sufficient to allow you to go ahead. Fig. 1 shows how to obtain the WRITE and OUTPUT ENABLE signals for pins 21 and 20 of the ram chips.It also shows a convenient place to get the required signals.

Fig. 2 show the hookup of the LS139 decoder. Section one is used to decode the 8k block into 4 2k blocks for the 6116's. Of course, any 8K block can be chosen. Just pick up the right line from U23, which decodes all the 8k blocks available.

Section two of the LS139 decodes $C800 . As shown the other enables from block two are not usefull. If however you wanted to decode $E000 and E800, just get the appropriate 8K block from U23 pin 7. Now use pin 9 and 10 of the LS139 as the chip select (pin 18) of those two rams. In case you haven't checked the memory map, those locations will be unused memory in nearly all OSI's.

Note that it is not necessary to do both projects. Do whichever half you desire. You could even do 2 8k blocks.

Now for the last possibility. Check figure 4 for the memory switch. Put the switch in the normal position, and the RAM is at $C800. Flip it to the * side, and the RAM appears instead of your monitor rom. Read or write it at $C800, it can read only at $F800. To load a new monitor, load it from tape or disk to $C800. Then flip the switch, and press break to reset. You can also use the extended monitor to move the monitor to $C800, make your changes, then flip the switch and reboot to see the changes.

This does seem a little complicated, but if you have some experience with modifications, this is an uncomplicated mod.

SIGNED INTEGERS

John Horemans TOSIE

Signed integers are used by both the FORTH language, and in the process of saving ML routines to disk with HEXDOS. Many subroutines used by BASIC also use signed two byte integers.

These numbers range from -32768 to +32766. Why is this and how do you access numbers above the range, say video memory?

The first thing to realize is that there is no minus sign anywhere inside the computer. What is done instead is that a chosen bit is set to one to indicate this. With integers, the high bit of high byte is the indicator. Thus numbers from 0 to 32766 are represented as 0000 to 7FFF. In binary 7FFF is 0111111 11111111. Note that if 1 is added the most significant bit will turn to a 1 and because of the convention for the minus sign, the numbers now appear negative.

If you want to access $D000 you cannot use 53248. You can however use -12288. This is in range and is in fact 53248. The monitor would show D000, binary would be 11010000 0000. These are correct for the location we want, but are interpreted as negative numbers by the integer routines. (remember-highest bit set).

Now I want to save my new ML routine from $C800 to my HEXDOS disk. I create a file. I know the format to save to track 5 is: SAVE#5,nnnn with nnnn being the decimal value of the start of memory. Try SAVE#5,51200. An error is indicated, as the number is out of range. What now?

Simply take the 51200, subtract 65536. You should get -14336. Now try SAVE#5,-14336. SUCESS! By subtracting 65536 we have set the high bit to a 1 and can now access the upper half of the memory.To summarize, to save to HEXDOS disk, from the upper half of memory, the integer has to be negative. The calculation goes as follows:

Change to decimal $C800 = 51200

Subtract 65536 to get -14336. Use this negative number to save.

With FORTH the situation is similar, one must be aware that negative numbers have a high bit set. Do however check your version of FORTH as it may well have a command to use unsigned numbers, often it is U. With this 51200 would be acceptable, and in fact would be represented identically in the computer memory. It would give 51200 in the unsigned mode, and -14336 in the signed integer mode.

If you have an Apple with integer Basic, you would have had to learn this early in the game!

CLASSIFIED ADVERTISEMENTS

DATA SEPARATOR- Original equipment MPI data separator. Plugs into the drive to make it OSI compatible. With schematics, $20.00 Call John Horemans 826-5362 or see me at a meeting.

16K RAM board from Progressive. Has been used for experimenting, but works. Includes everything but the 2114 RAM chips. $25.00 Also 2114 chips, some L450, some L300 and L200. $25 for all 2 1 chips. Will sell separately. Call John 826-5362 or at the meeting.

FOR SALE: 1 WORKING SUPERBOARD 2 REV. D WITH 8K RAM AND RS-232 PORT ON BOARD RUNNING AT 1 MHZ. ASKING $100.00. 1 SEB-1 EXPANSION BOARD (16K RAM, HIRES COLOR GRAPHICS,AND PARALLEL PORT) ALL DISCRETES AND FULLY SOCKETED, WITH MANUALS, ASKING $60.00. FOR FURTHER INFORMATION, CALL RON AT 519 886 0363, OR WRITE TO: RON SINGH, 594 HIGHPOINT AVENUE, WATERLOO, ONT., N2L 4N1

LATE NEWS.....From the March Issue of OSMOSUS NEWS

- One of the members reported that the DTACK Grounded 68000 board has been adapted to the C1P by a user in Belgium.

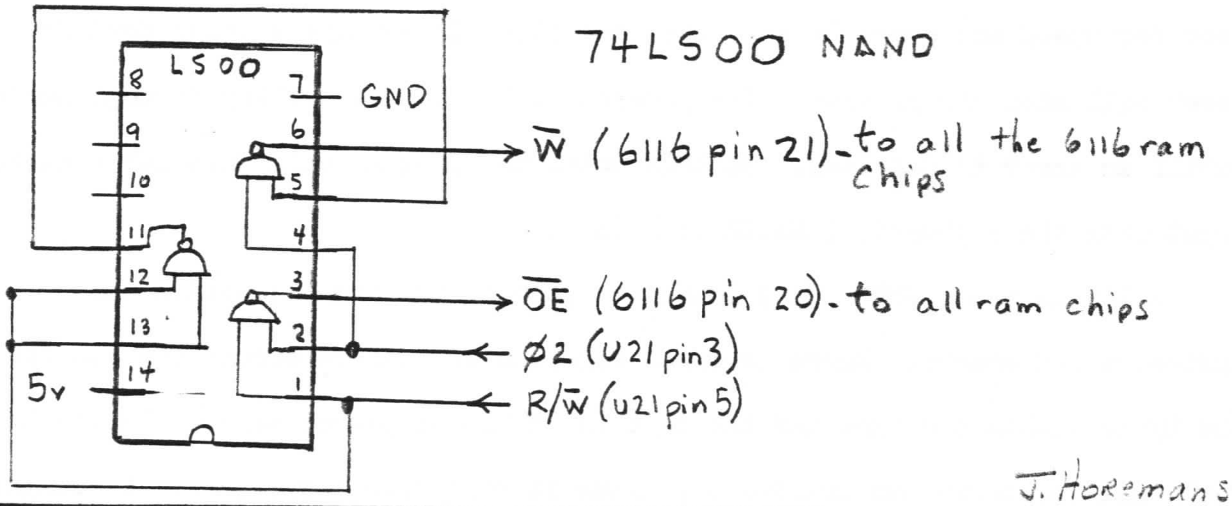
- Someone who checks into the CompuServ OSI SIG has adapted an 80 column apple board to the C1P.

- A company in Europe is making OSI compatible boards on the EURO-BUS cards. Bare boards are available. Apparently the boards may be ordered from California. They are getting more details.

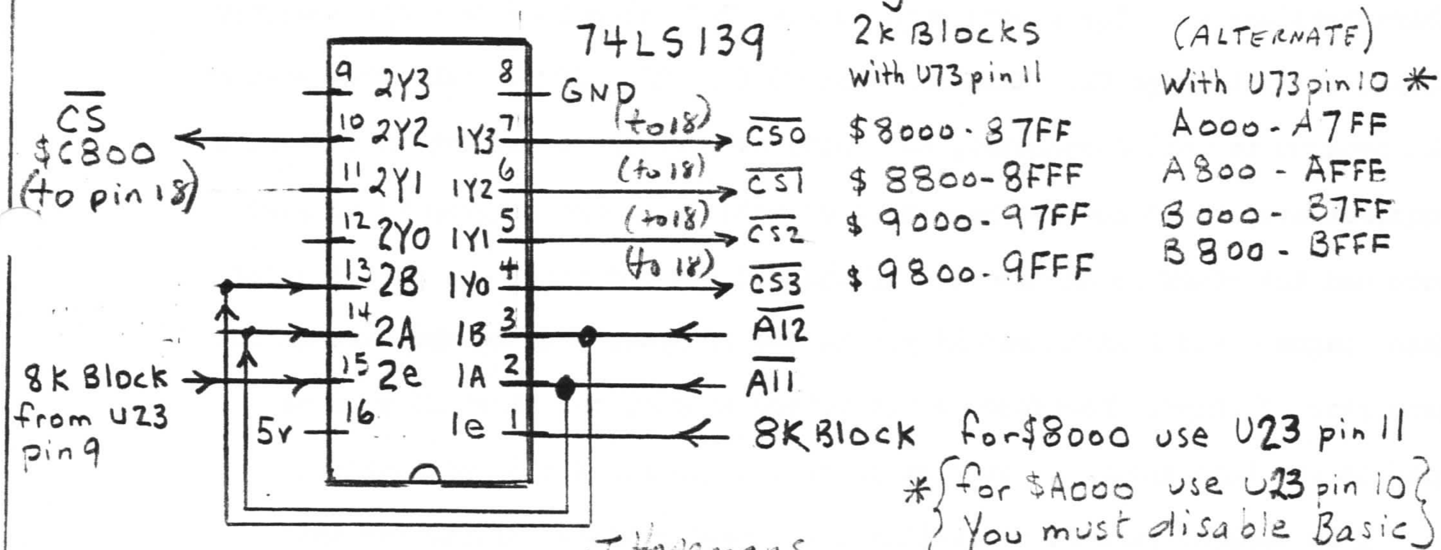
- OSMOSUS will try to log on to COMPUSERVE at 7:00 pm on the THIRD THURSDAY of each month. There is also a weekly 'meeting' on THURSDAY evenings at 10 pm.

More information on the Conference area, and the special commands available are contained in the March OSMOSUS newletter.

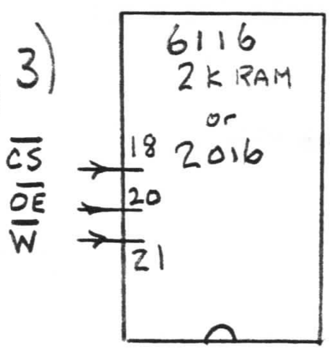
$\overline{W}$ and $\overline{OE}$ (fig 1) for 6116 2K RAM



$C800 DECODE $8000 (fig 2)



(fig 3)

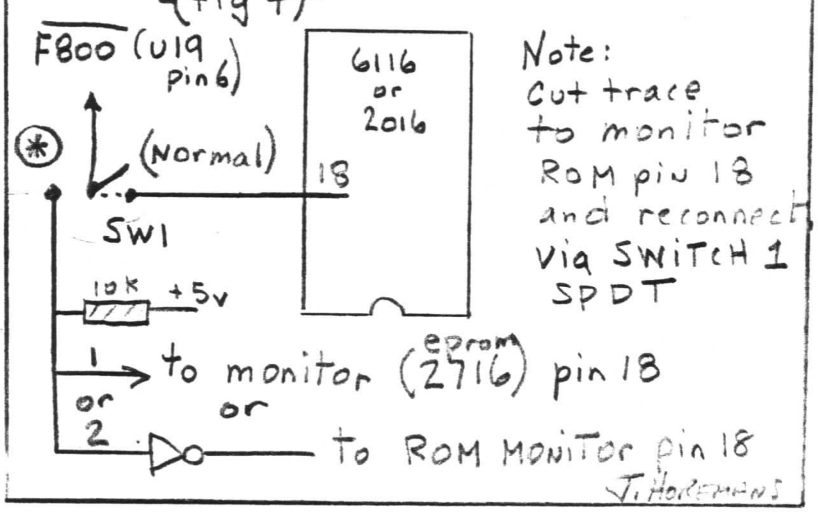


24 pin package can be piggy backed to Basic Roms, except for pins 18, 20, 21

J. Horemans

$C800 Switched to $C800/F800

(fig 4)



The following is a very short memory test routine which is in machine code for speed and compactness. Once you start it running a power down or reset will stop the program. The program will continue cycling through memory untill an error is detected. At this point the program will wait for a character input from the keyboard, (ENTER will do).

A little man (CHR - F1) indicates good memory and a question mark indicates bad memory. There is cross referencing numbers across the top for the bytes within the page and the page in HEX is displayed below. Thanks to Paul Vail for these two additions to make it more user compatible. I designed the program on a C8P SF but can be modified to work on screens other than 64 characters across. The second byte on line 720 determines were the computor should reset to page 01. Example , use C0 for 32K , BF for 48K. The rest of the program is well documented, so I have been told. A comment on the space requirements, the 0 and 1st page must be good since the program is on page zero and the stack is on page 1. If the 2nd and 3rd pages are out of order then pages 0 and 1 will most likely be out of order because they are in the same pair of chips. Therefore to put things simply, the first 1K must be good to run this program, but just in case pages 1,2 and 3 are checked.

I am looking forward to getting a bit test routine working for this program when I have time. This addition will hopefully still leave the program under a page in length.

By Bob Wickson

```
                         ;*******************************************
                         ;
                         ;      MEMORY TEST ROUTINE FOR PAGE ZERO
                         ;
                         ;      created by R. G. Wickson    84/01/20
                         ;      added to by Paul T. Vail    84/01/22
                         ;
                         ;*******************************************
                         ;
                         ;** THE FIRST 1K MUST BE GOOD TO RUN THIS **
                         ;
                         ;
0000                            *=$0000         ;ROUTINE RESIDES IN 1st PAGE
FD00=                           KEBORD=$FD00
                                ;E0 LOCATION - DATA BEING SENT TO LOC.
                                ;E1 LOCATION - A "1" INDICATES ERROR FOUND
0000 A900                       LDA #$00        ;SET FLAGS TO 00
0002 85E0                       STA $E0
0004 85E1                       STA $E1
0006 207D00                     JSR ENTIRE      ;CLEAR ENTIRE SCREEN
0009 A000                       LDY #$00        ;SET REGISTERS TO ZERO
000B 4C5100                     JMP SETUP       ;SORRY !!
000E A900         BEGIN         LDA #$00        ;CHARACTER USED FOR CHECK
0010 A200         PAGE          LDX #$00
0012 9D0001       START         STA $0100,X     ;SEND DATA TO LOCATION
0015 85E0                       STA $E0         ;SAFE GUARD DATA
0017 BD0001       CHECK         LDA $0100,X     ;RETURN DATA FROM LOCATION
001A C5E0                       CMP $E0         ;IS DATA STILL THE SAME?
001C D00F                       BNE WRONG       ;NO GOTO WRONG
001E BD00D2                     LDA $D200,X     ;IF ERROR WAS FOUND DON'T
0021 C93F                       CMP #$3F        ;CHANGE INDICATOR ON SCREEN
0023 F008                       BEQ WRONG
0025 A9F1                       LDA #$F1        ;YES  LOAD LITTLE MAN
0027 9D00D2                     STA $D200,X     ;PUT MAN ON DISPLAY
002A 4C3600                     JMP END
002D A93F         WRONG         LDA #$3F        ;NO LOAD ?
002F 9D00D2                     STA $D200,X     ;SHOW ? FOR THIS LOCATION
0032 A901                       LDA #01         ;SET ERROR FLAG
0034 85E1                       STA $E1
0036 A5E0         END           LDA $E0         ;GET LAST CHECK CHARACTER
0038 E8                         INX
0039 D0D7                       BNE START       ;DO ENTIRE PAGE
003B E6E0                       INC $E0         ;CREATE NEXT CHECK CHARACTER
003D A5E0                       LDA $E0
003F D0CF                       BNE PAGE        ;DO A PAGE OF THESE CHAR.
0041 EA                         NOP             ;TO INCREASE PAGE CHECK (DEY)
0042 EA                         NOP             ;BNE(BEGIN)
0043 EA                         NOP             ;BEGIN
0044 A900                       LDA #$00
0046 C5E1                       CMP $E1         ;CHECK ERROR FLAG
0048 F007                       BEQ SETUP       ;NO ERROR SETUP SCREEN
004A 2000FD                     JSR KEBORD      ;WAIT WHEN PAGE IS CHECKED
004D A900                       LDA #$00        ;    -BECAUSE OF ERROR
004F 85E1                       STA $E1
0051 20AE00       SETUP         JSR PAGE3       ;SETUP SCREEN FOR NEXT CHECK
0054 E614                       INC START+2     ;GO ON TO NEXT PAGE
0056 A514                       LDA START+2     ;GET # OF PAGE BEING CHECKED
0058 8519                       STA CHECK+2
                         ;
005A 4A                         LSR A           ;SEPARATE HIGH NIBBLE
005B 4A                         LSR A
```

```
620 005C 4A               LSR A
630 005D 4A               LSR A
640 005E 20B900           JSR HEXCON
650 0061 8D20D4           STA $D420       ;PUT HIGH NIBBLE ON SCR IN HEX
660 0064 A514             LDA START+2
670 0066 290F             AND #$0F        ;SEPARATE THE LOW NIBBLE
680 0068 20B900           JSR HEXCON
690 006B 8D21D4           STA $D421
700                ;
710 006E A514             LDA START+2
720 0070 C9C0             CMP #$C0        ;TEST FOR LAST PG OF SYSTEM
730 0072 D09A             BNE BEGIN       ;IF NOT THEN CONT. MAIN PROG.
740 0074 A900             LDA #00         ;IF LAST PG SET PG TO FIRST PG
750 0076 8514             STA START+2
760 0078 8519             STA CHECK+2
770                ;
780 007A 4C5100           JMP SETUP
790                ;
800                       ;CLEAR SCREEN ROUTINES
810 007D A920      ENTIRE LDA #$20
820 007F A200             LDX #$00
830 0081 9D00D0    CLEAN  STA $D000,X     ;CLEAR 1st PAGE
840 0084 9D00D1           STA $D100,X     ;CLEAR 2nd PAGE
850 0087 9D00D3           STA $D300,X     ;CLEAR 4th PAGE
860 008A 9D00D4           STA $D400,X     ;CLEAR 5th PAGE
870 008D 9D00D5           STA $D500,X     ;CLEAR 6th PAGE
880 0090 9D00D6           STA $D600,X     ;CLEAR 7th PAGE
890 0093 9D00D7           STA $D700,X     ;CLEAR 8th PAGE
900 0096 E8               INX
910 0097 D0E8             BNE CLEAN
920                ;
930 0099 A20F             LDX #$F
940 009B 8A        REF    TXA             ;PUT REFERENCE NO. ON SCREEN
950 009C 20B900           JSR HEXCON
960 009F 9DF0D1           STA $D1F0,X
970 00A2 9DE0D1           STA $D1E0,X
980 00A5 9DD0D1           STA $D1D0,X
990 00A8 9DC0D1           STA $D1C0,X
1000 00AB CA              DEX
1010 00AC 10ED            BPL REF
1020               ;
1030 00AE A920     PAGE3  LDA #$20
1040 00B0 A200            LDX #$00
1050 00B2 9D00D2   CLEAN2 STA $D200,X     ;CLEAR 3rd PAGE
1060 00B5 E8              INX
1070 00B6 D0FA            BNE CLEAN2
1080 00B8 60              RTS
1090               ;
1100 00B9 0930     HEXCON ORA #$30        ;CONVERT No. IN 'A' TO HEX
1110 00BB C93A            CMP #$3A
1120 00BD 3003            BMI H.1
1130 00BF 38              SEC
1140 00C0 6906            ADC #6
1150 00C2 60       H.1    RTS
```