PEEK (65)

The Unofficial OSI Users Journal

P.O. Box 347
Owings Mills, Md. 21117
(301) 363-3268

$1.75

FEBRUARY 1984
VOL. 5, NO. 2

INSIDE

Column One

This column has become an OSI/ISOTRON observation post. as we all watch them put the company back together again after the recent near disaster. We are constantly amazed (and pleased) with the tenacity of OSI. They refuse to succumb to difficulties!

The long range good news from ISOTRON is that they are determined to be back at the leading edge of this industry with a new, you should excuse the expression, third generation, true 16-bit machine supporting intelligent work stations. It should be ready in 1984. Equally important is that they firmly believe that there is a definite place for the current machines and thus nothing will be dropped in favor of the new machine. In fact. a new wrinkle for the 200 series might be unveiled very shortly.

PEEKers will remember that we had called for a machine with the architecture of the current multiprocessor machines for some time before they were announced. We still believe in this architecture. so we are delighted to hear that ISOTRON plans to continue making them.

We don't know what the new 16-bit machine will be like. However, if it isn't IBM PC compatible in disk format and operating system, they will be missing the fastest moving bus in town. In fact. we believe the ideal machine would feature intelligent workstations which can run IBM PC programs, all networked into a larger system in much the same way the present 300 series machines work with CP/M. This would produce something like the present PC-Net systems. but with better performance and lower cost.

Did I say lower cost? Now I am going out on a limb, but if the publisher leaves this paragraph in. you will know how I personally feel: one of OSI's major problems with the 300 machines was price. Dealers I know were ecstatic to see a CP/M compatible, multiprocessor machine with OSI's big disks. in a nice looking cabinet -- and dismayed by its cost. They reported that Molecular. Micromation. ACE and a half dozen others had the same architecture. the same operating system compatibility. at maybe $2,000 less per system. Hard to compete under those conditions unless your initials are I.B.M.!

So let's all look for a resumption of delivery of the existing excellent machines. with a great increase in quality and quantity of advertising to the business community. and hope for a shiny new machine which will be compatible with the existing machines but run IBM software as a Christmas present.

Speaking of advertising, ISOTRON's national ad campaign is undergoing final revisions. and we should be seeing those new ads Real Soon Now. We should see ads in both major national publications and publications directed toward specific markets in line with the 0vertical market software packages which they will also advertise.

To go along with all this. first steps have already been taken to improve the availability of peripherals. DataSouth's DS-180 and DS-220 printers will be included at nice prices; we may even see Alloy back in the fold with cartridge and 9-track tape units, - all to be available through ISOTRON and your local dealer.

Unfortunately, the Source Book is not out yet. It is at the printers. though. so maybe soon... Mean while. the vertical market packages we discussed earlier are coming along nicely. By the time you read this. contracts should be signed. and we should see those packages announced in sets of three shortly.

If there is anything more important than price in selling a computer. it is the availability of quality software. Dealers have long known that most business users don't buy machines... they buy solutions to problems. Show them the programs which will do the job they need done. and they will buy the machine which runs the programs. So this last item about vertical software packages may be the most important of all.

If you want later information than we can give you. what with the inevitable delays of publication and mailing, see your dealer. He should be receiving new dealer kits and information momentarily. al.

WHAT ELSE FOR OSI BASIC?

Leo Jankowski
Otaio RD 1
Timaru, New Zealand

So you wanted structured OSI BASIC but were afraid to ask? Well here is a step in the right direction, :ELSE.

This is the kind of BASIC now made possible:

```
10 IF A=1 THEN PRINT "1" :ELSE
IF A=2 THEN PRINT "2" :ELSE IF
A=3 THEN PRINT "3"
```

And,

```
10 IF D$="MONDAY" THEN PRINT
"PRUNES" :ELSE PRINT "RHUBARB"

20 PRINT "AND CUSTARD"
```

:ELSE is implemented by making it an initial keyword; replace WAIT with ELSE in the table in ROM 1. The code for :ELSE goes in at $BEF5. See the Steve Hendrix article in the August '82 issue, on how to find over 200 free bytes in the BASIC 4 ROM. There is so much space that I put 4 routines in; CLS, OLD, CALL and :ELSE! And there are two bytes spare!

The :ELSE code is completely relocatable, but you will have to make address changes as indicated in Table 1.

Three changes in ROM are required:

1) At $A0C9 change the word WAIT to ELSE. From 57 41 49 D4 to 45 4C 53 C5.

2) At $A024 change the address from 31 B4 to F4 BE. Notice that the first byte, $F4, is one less than the actual address.

3) The final change is in BASIC 1, at $A74F. Change 20 1D A7 to 20 FB BE.

This is how it works. At $A74F, if the expression following an IF is false, then the jump is to the REM routine at $A71D. This is the point at which the code is intercepted and forced to jump to $BEFB. If ELSE is not the next statement then the jump to REM is taken as usual. If an ELSE token is found then the branch to $BF16 is made, where everything following the token is parsed. The token for ELSE is $92.

A few guidelines on how to use :ELSE. Precede ELSE with a colon. It is an initial keyword by definition. If :ELSE is not one of an IF..THEN statement, then it will be ignored: the first 6 bytes, at $BEF5, do that. But any BASIC that follows will be parsed. If you must use NEW and CLEAR with ELSE then write :ELSE :NEW and :ELSE :CLEAR. The second colon is necessary only with these two keywords.

BASIC overlaps the IF..THEN code with the REM code. This has implications for using REM with the new ELSE routine. REM on its own is now illegal, e.g.

```
20 REM
```

There must be at least one non-blank character following the REM, e.g.

```
20 REM X
```

When BASIC sees a REM token it jumps to $A74F and then JSRs to the REM routine. Exactly the same JSR is taken if the expression following IF is evaluated as false. This is also the jumping off point for ELSE, to $BEFB. At $BEFB the code is parsed until either an ELSE token or a null is found. If the line

```
20 REM
```

is being parsed then the next null found is part of a line number. BASIC would then interpret the next four bytes as a pointer and a line number. Code following would not make sense giving SN ERROR. Placing at least one non-blank character after REM gets round this problem.

The code for WAIT can still be used. For example, this line will wait until the CTRL key is pressed (C1P)

```
10 POKE 11,50: POKE 12,180:
X=USR(X) 57100, 191, 254
```

Using IF..THEN..ELSE logic simplifies BASIC programming, but there are traps for the unwary. Experimenting with :ELSE FOR .. NEXT and :ELSE :FOR .. NEXT would be a good example.

:ELSE program - OSI BASIC.

```
BEF5 201AA7 JSR $A71A      Search for next BASIC statement/line.
BEF8 4C52A7 JMP $A752      Do statement or next line of BASIC.

BEFB 20BC00 JSR $00BC      Get next character of BASIC.
BEFE C992   CMP #$92       Token for ELSE?
BF00 F014   BEQ $BF16      Branch if ELSE token found, else
BF02 C900   CMP #$00       look for end of this BASIC line.
BF04 D0F5   BNE $BEFB      Keep looking for ELSE or a null.
BF06 4C1DA7 JMP $A71D      Jump to REM, then return to $A752.
BF09 68     PLA            Remove 2 unwanted return addresses.
BF0A 68     PLA
BF0B 68     PLA
BF0C 68     PLA
BF0D 2054A7 JSR $A754      Do a line of BASIC.
BF10 4C1AA7 JMP $A71A      Find end of statement.
BF13 4CC3A7 JMP $A7C3      Nested IF .... THEN to do.

BF16 20BC00 JSR $00BC      Get a character.
BF19 C99A   CMP #$9A       Check for IF token.
BF1B F0F6   BEQ $BF13
BF1D C98D   CMP #$8D       Check for RETURN token.
BF1F F0E8   BEQ $BF09
BF21 C982   CMP #$82       Check for NEXT token.
BF23 F0E4   BEQ $BF09
BF25 C981   CMP #$81       Check for FOR token.
BF27 F0E0   BEQ $BF09
BF29 D0E2   BNE $BF0D      Must be some other token.
```

TABLE 1

ROM1	From	To
A024	31 B4	F4 BE
A0C9	57 41 49 D4	45 4C 53 C5
A74F	20 1D A7	20 FB BE

★

published monthly

Editor - Al Peabody
Technical Editor - Brian Hartson
Circulation & Advertising Mgr. - Karin Q. Gieske
Production Dept. - A. Fusselbaugh, Ginny Mays

Subscription Rates
US (surface) $15
Canada & Mexico (1st class) $23
So. & Cen. America (Air) $35
Europe (Air) $35
Other Foreign (Air) $40

All subscriptions are for 1 year and are payable in advance in US Dollars.

For back issues, subscriptions, change of address or other information, write to:

PEEK (65)
P.O. Box 347
Owings Mills, MD 21117

Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes endorsements of the product or products by this magazine or the publisher.

XREF: BASIC Program Cross Reference Generator

By: Rick Trethewey
8 Duran Court
Pacifica, CA 94044

How many times have you looked at a BASIC program and said to yourself "I can't make heads or tails out of this thing!"? No matter if the program is your own or someone else's, some BASIC programs are just indecipherable. Rather than harp on programming technique, I'll show you a utility I wrote that can help you debug and optimize any BASIC program. I call this utility XREF because it displays all line numbers that are referenced by GOTOs, GOSUBs, or IF.. THENs along with the line numbers on which the reference occurs. XREF also displays all variables and defined functions and the line numbers on which each variable is referenced.

The source code was written for the A/65 assembler which allows linking multiple files. For the OSI Assembler/Editor, all that need be done is to add the reference for "XRQ" to the first file, duplicate the externals table and add the references for "WARM" and "WARMNS" to the second file. With the OSI Assembler/Editor, if you have less than 48K of RAM you should delete most of the comments. Since XREF is written to reside in the transient language processor area of OS-65D, you will need to assemble it with an offset to the top of your available memory. The object code is almost exactly eleven pages in length.

Once assembled, using XREF is pretty straightforward. To get to OS-65D's "A*" prompt, call the object code into memory at $0200 and enter "GO 0200". XREF will then display its menu. Option number 1 will display the directory of the currently selected drive. Option number 2 will execute XREF and provide your cross-references. Two other options are available from XREF's menu. Entering "E" will exit to OS-65D. You can also issue a command to OS-65D by preceding it with an exclamation point.

When you select option 2 to cross-reference a file, you are asked for the name of the file and the drive the file is on. You are then asked for the output device number. If you respond with just a <RETURN> here, all output will

```
 10; XREF : BASIC FILE CROSS REFERENCE GENERATOR
 20; WRITTEN BY RICHARD L. TRETHEWEY
 30; COPYRIGHT 1983
 40; ALL RIGHTS RESERVED
 50;
 60; OS-65D EXTERNALS
 70;
 80        PNTL   =$E1
 90        PNTH   =$E2
100        MAXMEM =$2300
110        INFLAG =$2321
120        OUFLAG =$2322
130        INCH   =$2340
140        OUTCH  =$2343
150        DISC   =$265C
160        SECT   =$265E
170        PAGES  =$265F
180        ADRLX  =$2660
190        ADRHX  =$2661
200        TRAKX  =$2662
210        HOME0  =$2663
220        SEEKX  =$26A6
230        LOAD   =$2754
240        UNLOAD =$2761
250        CALLX  =$295D
260        SELECT =$29C6
270        ERROR  =$2A4B
280        OS65D3 =$2A51
290        ERRSU  =$2A7D
300        CSI    =$2A84
310        ERR7   =$2AC0
320        DEFAUL =$2AC5
330        LOADER =$2BA7
340        SRCSIZ =$2BE9
350        REASM  =$2C04
360        REBAS  =$2C0B
370        REEM   =$2C12
380        TIN0   =$2CEC
390        SWAP   =$2CF7
400        CRLF   =$2D6A
410        STROUT =$2D73
420        PRBYTE =$2D92
430        FNDNUM =$2DA6
440        DIRTRK =$2DC4
450        TXTBUF =$2E1E
460        DIRBUF =$2E79
470        CASECK =$3A5F
480        SRCSTR =$3A79
490;
500; XREF EXTERNALS
510;
520        PUTPTR =$01   Z-PAGE POINTER TO WORKSPACE
530        NLAL   =$03   NEXT LINE ADDRESS LSB
540        NLAH   =$04   NEXT LINE ADDRESS MSB
550        LC     =$05   LINE COUNT
560        T1     =$07   TEMPORARY REGISTER
570        T2     =$09   ALTERNATE TEMPORARY REGISTER
580        ONFLAG =$0B   PROCESSING "ON" FLAG
590        TMPPTR =$0C   TEMPORARY HOLDER FOR LINE INDEX
600        GETPTR =$0D   TABLE MOVEMENT POINTER
610        REFBOT =$0F   BOTTOM OF REFERENCE TABLE
620        SOF    =$11   STARTING INDEX OF FILE
630        NR     =$13   # OF REFERENCES HOLDER
640        INDEX  =$14   INDEX TO BUFFER
650        FETPTR =$15   FETCH POINTER
660        TFLAG  =$17   MATCH FLAG
670        OLDPTR =$18   OLD POINTER TO ENTRY
680        LNLO   =$1A   LINE # LSB
690        LNHI   =$1B   LINE # MSB
700        VARNAM =$1C   VARIABLE NAME
710        TYPE   =$1E   VARIABLE TYPE
720        CHRGET =$C0   ROUTINE TO FETCH A CHAR. FROM TEXT
730        CHRGOT =$C6   ROUTINE TO RE-FETCH CHARACTER
740        TXTPTR =$C7   ADDRESS STORAGE TO TEXT
750        BUFFER =$1500 TRACK BUFFER
760        INBUF  =$2000 GENERAL INPUT BUFFER
770        TXBUFF =$2100 PROGRAM TEXT LINE BUFFER
780;
790; ASSEMBLY CONSTANTS
800;
810        LF     =$0A
```

Continued on page 4

be directed to the console. You can also enter the OS-65D device number of your printer.

XREF will tell you if your program refers to a non-existent line number. All line number references are documented. That is, if a reference is made to a single line number more than once on the same line, each reference is documented. Duplicate references to a single variable on the same line are suppressed.

One thing that surprised me about XREF was the sheer volume of the output even with small files. Part of that is because I chose to double space everything for clarity. Just be forewarned that XREF will eat a lot of paper. I know XREF will save you a lot of time and effort.

```
 820       BS     =$08
 830       CR     =$0D
 840       SP     =$20
 850       SKIP2  =$2C
 860       ESC    =$1B
 870       DEL    =$5F
 880       GOTOTK =$88
 890       GOSBTK =$8C
 900       REMTK  =$8E
 910       ONTK   =$90
 920       FNTK   =$9E
 930       THENTK =$A0
 940;
 950;
 960       *=$0200       TRANSIENT LANGUAGE AREA
 970;
 980       JMP START     JUMP TO COLD START CODE
 990;
1000ESCBYT .BYTE ESC             ESCAPE CODE
1010CLSBYT .BYTE $1C             SCREEN CLEAR CODE
1020CURFIL .BYTE 'XXXXXX'
1030 RESLO .BYTE $00             RESULT LSB
1040 RESHI .BYTE $00             RESULT MLSB
1050 FIFTH .BYTE $00             RESULT MSB
1060 STTK  .BYTE $00             FILE START TRACK
1070 ENDTK .BYTE $00             FILE END TRACK
1080BFENPG .BYTE $00             PAGE# AFTER BUFFER
1090 COUNT .BYTE $00             SECTOR NUMBER FOR DIRIN
1100 POWER .BYTE $00             NUMBER CONVERSION EXP.
1110NUMBER .BYTE $00,$00,$00,$00,$00,$00,$00
1120 TOTAL .BYTE $00,$00,$00,$00
1130 TBL1  .BYTE $01,$16,$56,$96,$36,$76
1140 TBL2  .BYTE $00,$00,$02,$40,$55,$85
1150 TBL3  .BYTE $00,$00,$00,$00,$06,$04
1160 TBL4  .BYTE $00,$00,$00,$00,$00,$01
1170;
1180HEXADD PHA        SAVE ORIGINAL BYTE
1190       JSR SHIFT  SHIFT 4 BITS
1200       JSR HADD   ADD FOR THIS #
1210       PLA        RETRIEVE ORIGINAL BYTE
1220       AND #$0F   MASK TO LOW NYBBLE
1230       DEC POWER  ADJUST INDEX
1240       JSR HADD   ADD FOR THIS #
1250       DEC POWER  ADJUST INDEX
1260       RTS        QUIT
1270 HADD  LDY POWER  GET INDEX TO TABLE
1280       TAX        SET COUNTER
1290       BEQ HADD2  IF 0, NO ACTION NEEDED
1300 HADD1 SED
1310       CLC
1320       LDA TBL1,Y
1330       ADC TOTAL
1340       STA TOTAL
1350       LDA TBL2,Y
1360       ADC TOTAL+1
1370       STA TOTAL+1
1380       LDA TBL3,Y
1390       ADC TOTAL+2
1400       STA TOTAL+2
1410       LDA TBL4,Y
1420       ADC TOTAL+3
1430       STA TOTAL+3
1440       CLD
1450       DEX
1460       BNE HADD1   LOOP 'TIL COUNTER ZEROES
1470 HADD2 RTS         QUIT
1480;
1490HEXDEC LDA #$00    ROUTINE TO CONVERT HEX NUMBER
1500       TAY         IN RESULT REGISTER TO
1510 HEXD1 STA POWER,Y DECIMAL ASCII
```

```
1520       INY
1530       CPY #$C
1540       BNE HEXD1
1550       LDA #$05
1560       STA POWER
1570       LDA FIFTH
1580       JSR HEXADD
1590       LDA RESHI
1600       JSR HEXADD
1610       LDA RESLO
1620       JSR HEXADD
1630       LDX #$03
1640       LDY #$00
1650 HEXD2 LDA TOTAL,X
1660       PHA
1670       JSR SHIFT
1680       STA NUMBER,Y
1690       INY
1700       PLA
1710       AND #$0F
1720       STA NUMBER,Y
1730       INY
1740       DEX
1750       BPL HEXD2
1760       RTS                 STOP! NO PRINT!
1770;
1780NUMOUT JSR HEXDEC          CONVERT RESULT TO DECIMAL
1790       LDX #$FF                     AND PRINT IT
1800       INX
1810       LDA NUMBER,X
1820       BNE HEXD4
1830       CPX #$07
1840       BNE NUMOUT+5
1850 HEXD4 LDA NUMBER,X
1860       ORA #'0
1870       JSR OUTCH
1880       CPX #$07
1890       BEQ HEXD5
1900       INX
1910       BNE HEXD4
1920 HEXD5 RTS
1930;
1940 SHIFT LSR A
1950       LSR A
1960       LSR A
1970       LSR A
1980       RTS
1990;
2000 DECHEX LDX #$00           ROUTINE TO CONVERT
2010        STX RESLO          ASCII INPUTS TO HEX
2020        STX RESHI
2030 DECH1  LDA INBUF,X
2040        CMP #'0            CHECK FOR LEGAL CHARACTER
2050        BCC DECDUN
2060        CMP #'9+1
2070        BCS DECDUN
2080        AND #$F            MASK OFF ASCII
2090        STA T2
2100        LDA RESHI
2110        STA T2+1
2120        LDA RESLO
2130        ASL A              X 2
2140        ROL T2+1             SHIFT CARRY'S INTO T2+1
2150        ASL A              X 4
2160        ROL T2+1
2170        ADC RESLO          + 1
2180        STA RESLO
2190        LDA T2+1
2200        ADC RESHI
2210        STA RESHI
```

Listing cont. on page 6

High Resolution Color Graphics

Our new Color Plus board provides 256 x 192 high-resolution graphics with 15 colors. Two 8-bit resolution joystick interfaces are included. Software extensions to OS65-D BASIC provide a superset of APPLE II® graphics instructions.

Color Plus connects to the standard 48-pin bus or the 16 pin bus.

Pricing:

CP-8 for C8 or C3 computers:	$195
CP-4 for C4 computers (5V only):	$245
CP-bare Bare board with software:	$ 75

Generos Operating System

Generos is a new operating system for OSI computers. Features include descriptive error messages, optimized disk usage, device independence, and:

- ASSEM — Fast disk based assembler generates relocatable code.
- BASIC — Basic Interpreter
- DDT — Machine language debugger with single step, trace, more.
- TECO — Most powerful and widely used text editor ever.

Currently available for 48K 8-inch systems. Call or write about availability for other configurations. Cost: $85

Low Power Memory Board

Our popular MEM + board is ideal for:

- Partitions for multi-user systems
- 64K CP/M systems when combined with the D&N-80 CPU board
- Upgrading systems where backplane space, low power consumption, and/or low heat dissipation is required

Options include:

- OSI compatible floppy disk controller — protects against disk crashes caused by power failures
- Real time clock/calendar — Date and time with battery backup
- Centronics parallel printer interface — Supported by software that automatically patches OS65D and OS65U
- One year warranty

VISA, MasterCard, personal checks and C.O.D.s all accepted. Add $5 per board for shipping and handling.

To order, or for more information, contact:

Fial Computer
5221 S.W. Corbett
Portland, Oregon 97201

(503) 227-7083

MEM + includes the following features:

- Memory chips in quality production sockets; high reliability machine screw sockets optional
- Low power consumption
- Uses 2K x 8-bit memory chips — will accept 2716-type EPROMS.
- Versatile addressing

NEW LOW PRICES!!

Bare — Solder masked and silkscreened $75
Software and documentation provided

16K	$200		
24K	$250	Disk controller	$85
32K	$300		
40K	$350	Real time clock	$65
48K	$390		
52K	$415	Centronics interface	$45
56K	$440		
64K	$490	High-rel sockets add 15%	

07514 51893

Generic Computer Products

5740 S.E. 18th Ave. Portland, OR 97202

```
2220         ASL RESLO        X 2  OR 10 TIMES TOTAL
2230         ROL RESHI
2240         LDA RESLO
2250         ADC T2
2260         STA RESLO
2270         BCC DECH2
2280         INC RESHI
2290 DECH2   INX
2300         BNE DECH1
2310 DECDUN RTS
2320;
2330 DRSEL JSR STROUT
2340        .BYTE CR,LF,'Drive (A/B/C/D) ? ',0
2350        JSR GETSTR
2360        LDA INBUF
2370        JSR CASECK
2380        CMP #'A
2390        BCC DRSEL
2400        CMP #'E
2410        BCS DRSEL
2420 DRS1   AND #$F
2430        CMP DISC
2440        BEQ DRS2-3
2450        STA TOTAL      SAVE VALUE
2460        JSR SWAP       * DOS CONTEXT *
2470        LDA TOTAL      RETRIEVE VALUE
2480        JSR SELECT
2490        BCS DRS2
2500        JSR HOME0      HOME DRIVE !!
2510        JSR SWAP       * LANGUAGE CONTEXT *
2520        JMP CRLF
2530 DRS2   LDA #$06
2540        JMP ERROR
2550;
2560SCRCLR LDA ESCBYT
2570        JSR OUTCH
2580        LDA CLSBYT
2590        JMP OUTCH
2600;
2610 START LDA #ERR7
2620        STA REBAS+1     DISABLE "RE BA"
2630        STA REASM+1     DISABLE "RE AS"
2640        STA REEM +1     DISABLE "RE EM"
2650        LDA #ERR7/256
2660        STA REBAS+2
2670        STA REASM+2
2680        STA REEM +2
2690        LDA #WARM
2700        LDY #WARM/256
2710        JSR ERRSU        RETURN ON ERRORS HERE
2720        LDA DEFAUL+1
2730        STA INFLAG
2740        STA OUFLAG
2750        JSR SCRCLR       CLEAR SCREEN
2760        LDA #$05
2770        STA $DE00        COLOR ON, SOUND OFF
2780        LDA #$11
2790        STA TINO+1       SET UP DOS CMD LENGTH
2800        ;
2810        ; WARM START RE-ENTRY POINT
2820        ;
2830 WARM   JSR SWAP         SET LANGUAGE CONTEXT
2840WARMNS LDX #$FE         RESET STACK
2850        TXS
2860        LDA #BUFFER/256
2870        CLC
2880        ADC SRCSIZ
2890        STA BFENPG
2900 MENU   JSR STROUT       SAY "HELLO"
2910        .BYTE CR,LF
2920        .BYTE '    XREF'
2930        .BYTE CR,LF,LF
2940        .BYTE '(1) Directory',CR,LF
2950        .BYTE '(2) XREF a File',CR,LF,LF
2960        .BYTE '    Your Selection ? ',0
2970        JSR GETSTR
2980        JSR SCRCLR
2990        LDY #$00
3000        LDA INBUF,Y
3010        JSR CASECK
3020        CMP #'1
```

Continued on page 7

```
3030        BEQ DIR
3040        CMP #'2
3050        BEQ XRE
3060        CMP #'E
3070        BEQ EXIT
3080        CMP #'!
3090        BEQ DOS
3100 INERR JSR STROUT
3110        .BYTE 'INVALID ENTRY'
3120        .BYTE CR,LF,LF,$00
3130        JMP MENU
3140;
3150 DIR   JSR SWAP
3160        JSR D
3170        JSR SWAP
3180        JMP MENU
3190;
3200 XRE   JSR XR
3210        JMP MENU
3220;
3230 EXIT  LDA #OS65D3      RESET ERROR RETURN TO 65D
3240        LDY #OS65D3/256
3250        JSR ERRSU
3260        LDA #$01
3270        JSR DRS1        MAKE SURE OF DRIVE A
3280        JSR SWAP        DOS CONTEXT
3290        JMP OS65D3      AND GO TO 65D KERNEL
3300;
3310 DOS   JSR SWAP        DOS CONTEXT
3320        LDA #INBUF+1     SET PNTL/PNTH TO INBUF+1
3330        STA PNTL
3340        LDA #INBUF+1/256
3350        STA PNTH
3360        JSR CSI         COMMAND STRING INTERPRETER
3370        JSR SWAP        LANGUAGE CONTEXT
3380        JMP MENU        AND QUIT
3390;
3400; STRING INPUT ROUTINE
3410;
3420GETSTR LDY #$00
3430 GETS1 JSR INCH
3440        STA INBUF,Y
3450        CMP #CR
3460        BEQ GETS2
3470        CMP #DEL
3480        BEQ BKSPC
3490        CMP #DEL+$20
3500        BEQ BKSPC
3510        INY
3520        BNE GETS1
3530 GETS2 JMP CRLF     CLEAN UP AND QUIT
3540;
3550 BKSPC TYA
3560        BEQ GETS1
3570        PHA
3580        JSR STROUT
3590        .BYTE BS,BS,SP,SP,BS,BS,0
3600        PLA
3610        TAY
3620        DEY
3630        JMP GETS1
3640;
3650GETANS JSR GETSTR       GET "YES" OR "NO" FROM USER
3660        LDA INBUF
3670        JSR CASECK
3680        CMP #'Y
3690        RTS
3700;
3710; INPUT FILE NAME AND FIND IT
3720;       IN THE DIRECTORY
3730;
3740FNDFIL JSR STROUT
3750        .BYTE 'File Name ? ',0
3760        LDY #$00
3770        LDA #SP
3780 FNDF0 STA CURFIL,Y    FILL CURFIL
3790        INY
3800        CPY #$06
3810        BNE FNDF0
3820        JSR GETSTR       GET NAME FROM USER
3830        LDY #$00
3840 FNDF1 LDA INBUF,Y      X-FER NAME TO CURFIL
```

Continued on page 8

```
3850       CMP #CR
3860       BEQ FNDF2
3870       STA CURFIL,Y
3880       INY
3890       CPY #$07
3900       BNE FNDF1
3910       JSR STROUT
3920       .BYTE CR,LF
3930       .BYTE 'TOO LONG',CR,LF,LF,0
3940       JMP FNDFIL
3950 FNDF2 TYA
3960       BEQ FNDFIL    FORCE USER RESPONSE
3970 FNDF3 LDA #$01
3980       STA COUNT
3990 FNDF4 JSR SWAP
4000       JSR DIRIN
4010       JSR SWAP
4020       LDY #$00
4030       LDX #$00
4040 FNDF5 LDA CURFIL,X
4050       JSR CASECK
4060       STA STTK
4070       LDA DIRBUF,Y
4080       JSR CASECK
4090       CMP STTK
4100       BNE FNDF6
4110       INY
4120       INX
4130       CPX #$06
4140       BNE FNDF5
4150       BEQ FNDF8
4160 FNDF6 INY
4170       BEQ FNDF7
4180       INX
4190       CPX #$08
4200       BNE FNDF6
4210       LDX #$00
4220       BEQ FNDF5
4230 FNDF7 INC COUNT
4240       LDA COUNT
4250       CMP #$03
4260       BNE FNDF4
4270       SEC           SHOW NO MATCH!
4280       RTS
4290 FNDF8 LDA DIRBUF,Y
4300       JSR BCDH
4310       STA STTK
4320       INY
4330       LDA DIRBUF,Y
4340       JSR BCDH
4350       STA ENDTK
4360       CLC           SHOW MATCH!
```

```
4370            RTS
4380;
4390; READ A SECTOR OF THE DIRECTORY
4400;           TRACK INTO "DIRBUF"
4410;
4420 DIRIN      LDA #DIRBUF       LOAD LSB OF DIRECTORY BUF
4430            STA ADRLX         GIVE IT TO 65D
4440            LDA #DIRBUF/256   LOAD MSB
4450            STA ADRHX         SEND IT TOO
4460            LDA COUNT         GET REQUESTED SECTOR #
4470            STA SECT          GIVE TO 65D
4480            LDA DIRTRK        FETCH DIRECTORY TRACK #
4490            JSR BCDH          MAKE IT HEX FOR TERM-PLUS
4500            STA TRAKX         GIVE IT TO 65D
4510            JSR SEEKX         MOVE HEAD TO TRACK
4520            JSR LOAD          LOAD HEAD
4530            JSR CALLX         CALL CODE INTO RAM
4540            JMP UNLOAD
4550;
4560; DIRECTORY PRINTER
4570;
4580 D          JSR STROUT        SAY "HELLO"
4590            .BYTE '* Directory *',CR,LF,LF,$00
4600            LDY #$01          INIZ
4610            STY COUNT         START WITH SECTOR #1
4620            DEY               (Y=0)
4630            STY FIFTH         INIZ ROW PRINT COUNT
4640            JSR DIRIN         READ IN DIRECTORY SECTOR
4650            JSR D1            PRINT IT
4660            INC COUNT         BUMP SECTOR REQUEST #
4670            JSR DIRIN         READ SECTOR 2 AND FALL TH
4680 D1         LDY #$00          INIZ BUFFER INDEX
4690            LDX #$00          INIZ ENTRY INDEX
4700 D2         LDA DIRBUF,Y      FETCH CHARACTER FROM BUFF
4710            CPX #$06          AT START TRACK #?
4720            BEQ TK1           YES! => TK1
4730            CPX #$07          AT END TRACK #?
4740            BEQ TK2           YES! => TK2
4750            CMP #'#           NO, NULL ENTRY?
4760            BNE D2.5          NO => D2.5
4770            CPX #$00          MAYBE. 1ST CHAR OF ENTRY?
4780            BEQ D0            YES! SKIP THIS ENTRY => D
4790 D2.5       JSR OUTCH         PART OF FILE NAME. PRINT
4800 D3         INX               BUMP ENTRY INDEX
4810 D4         INY               BUMP BUFFER INDEX
4820            BNE D2            LOOP 'TIL DONE
4830            DEY               BACK UP ONE
4840 D0         TYA               PUT BUFFER INDEX IN ACC.
4850            CLC
4860            ADC #$08          ADD 8 TO IT
4870            TAY               PUT RESULT IN BUFFER INDE
4880            BCS QUIT          QUIT ON PAGING
```

Continued on page 9

```
4890        BCC D2-2         OR LOOP 'TIL DONE
4900 TK1    LDA #SP          GET A <SPACE>
4910        JSR OUTCH        PRINT IT
4920        JSR TKOUT        PRINT TRACK #
4930        JMP D3           AND LOOP
4940 TK2    LDA #'-          LOAD "-"
4950        JSR OUTCH        PRINT IT
4960        JSR TKOUT        PRINT END TRACK #
4970        INC FIFTH        BUMP ROW PRINT COUNT
4980        LDA FIFTH        FETCH IT
4990        CMP #$04         DONE 4 ON THIS LINE?
5000        BNE TK3          NO => TK3
5010        LDX #$00         YES! INIZ
5020        STX FIFTH        RESET ROW COUNT
5030        JSR CRLF         DO A <CR><LF>
5040        JMP D4           AND LOOP
5050 TK3    TYA              PUT BUFFER INDEX IN ACC.
5060        PHA              SAVE ON STACK
5070        JSR STROUT       PRINT THIS:
5080        .BYTE SP,SP,$00
5090        LDX #$00         INIZ
5100        PLA              RETRIEVE BUFFER INDEX
5110        TAY              PUT BACK IN Y
5120        JMP D4           AND LOOP
5130 TKOUT  LDA DIRBUF,Y     FETCH TRACK #
5140        JMP PRBYTE       AND PRINT IT
5150;
5160 QUIT   LDA COUNT        GET SECTOR COUNT
5170        CMP #$02         DONE BOTH?
5180        BEQ QUIT1        YES!
5190        RTS              NO, CONTINUE
5200 QUIT1  JMP CRLF         DO CLEAN-UP & QUIT
5210;
5220; BCD TO HEX CONVERSION ROUTINE
5230;
5240 BCDH   PHA
5250        AND #$F0
5260        LSR A
5270        LSR A
5280        LSR A
5290        LSR A
5300        TAX
5310        LDA #$00
5320 BCDH1  CLC
5330        ADC #$A
5340        DEX
5350        BNE BCDH1
5360        STA BCDH2+1
5370        PLA
5380        AND #$F
5390        CLC
5400 BCDH2  ADC #$FF
5410        RTS
5420;
```

```
5430; HEX TO BCD CONVERSION ROUTINE
5440;
5450 HBCD   LDX #$FF
5460        SEC
5470 HBCD1  INX
5480        SBC #$A
5490        BCS HBCD1
5500        ADC #$A
5510        STA HBCD2+1
5520        TXA
5530        ASL A
5540        ASL A
5550        ASL A
5560        ASL A
5570        CLC
5580 HBCD2  ADC #$FF
5590        RTS
5600;
5610; FETCH A BYTE FROM TEXT.
5620; THIS CODE IS MOVED DOWN TO
5630; PAGE ZERO AT $00C0 AND IS
5640; CALLED "CHRGET" THERE.
5650;
5660 CHR    INC TXTPTR       BUMP TEXT POINTER
5670        BNE CHR$         WATCH PAGING
5680        INC TXTPTR+1     BUMP MSB ON PAGING
5690 CHR$   LDA $FFFF        FETCH CHARCTER FROM TEXT
5700        RTS              AND QUIT
5710;
5720; LOAD SOURCE FILE FOR OUTPUT
5730;
5740 LODFIL LDY #LODFIL-CHR  INIZ COUNTER & INDEX
5750 LODF1  LDA CHR,Y        FETCH BYTE OF CHRGET
5760        STA CHRGET,Y     MOVE IT TO Z-PAGE
5770        DEY              DECREMENT COUNTER
5780        BPL LODF1        LOOP 'TIL DONE
5790        JSR SWAP         * DOS CONTEXT *
5800        LDA STTK         GET 1ST TRACK OF FILE
5810        JSR HBCD         MAKE IT BCD
5820        JSR LOADER+3     LOAD FILE INTO WORKSPACE
5830        JSR SWAP         * LANGUAGE CONTEXT *
5840        LDA SRCSTR       GET START OF FILE ADDR
5850        STA NLAL         SAVE AS NEXT LINE AD. LO
5860        LDA SRCSTR+1     HANDLE MSB TOO
5870        STA NLAH
5880        LDA #SRCSTR      SET UP PUTPTR
5890        STA PUTPTR
5900        LDA #SRCSTR/256
5910        STA PUTPTR+1
5920        RTS              AND QUIT
5930;
5940 XR     JSR DRSEL        SELECT DRIVE
5950        JSR FNDFIL       GET FILE NAME & FIND IT
```

Continued on page 10

```
5960          BCC XR0          FILE FOUND? ==> XR0
5970          JSR STROUT       NO GOOD! TELL USER
5980          .BYTE CR,LF
5990          .BYTE 'FILE NOT FOUND',CR,LF,$00
6000          RTS              QUIT
6010 XR0      JSR STROUT
6020          .BYTE CR,LF,'Output Device # ? ',$00
6030          JSR GETSTR
6040          JSR DECHEX
6050          LDA RESLO
6060          BEQ XR1
6070          ORA OUFLAG
6080          STA OUFLAG
6090 XR1      JSR LODFIL       LOAD FILE INTO RAM
6100          LDA #$00         INIZ
6110          STA FIFTH
6120          STA LC
6130          STA LC+1
6140 XR2      LDA NLAL         GET NEXT LINE ADDRESS
6150          STA TXTPTR       SAVE IT
6160          LDA NLAH
6170          STA TXTPTR+1
6180          BNE XR3
6190          RTS              NULL FILE!
6200 XR3      LDY #$00         INIZ
6210          JSR CHRGOT       GET NEXT LINE ADR. LSB
6220          STA NLAL         SAVE IT
6230          JSR CHRGET       GET NEXT LINE ADR. MSB
6240          STA NLAH         SAVE IT TOO
6250          BNE XR31         E.O.F. CHECK
6260          LDA NLAL         RECHECK NEXT LINE LSB
6270          BEQ XR7
6280 XR31     JSR CHRGET       GET LINE # LSB
6290          STA (PUTPTR),Y   SAVE IT IN TABLE
6300          JSR CHRGET       GET LINE # MSB
6310          INY              BUMP PUT POINTER
6320          STA (PUTPTR),Y   SAVE IT TOO
6330          INY              BUMP PUT POINTER
6340          LDA #$00         INIZ
6350          STA (PUTPTR),Y   SHOW NO REFERENCES YET
6360          LDA #$03         INIZ
6370          CLC
6380          ADC PUTPTR       BUMP PUT POINTER ADDRESS
6390          STA PUTPTR       SAVE IT
6400          BCC XR4          HANDLE PAGING
6410          INC PUTPTR+1
6420 XR4      INC LC           BUMP LINE COUNTER LSB
6430          BNE XR5          HANDLE PAGING
6440          INC LC+1         BUMP MSB
6450 XR5      LDA NLAH         REFETCH NLAH
6460          BEQ XR6          IF $00, ==> XR6
6470          CLC
6480          ADC #$09         NO, ADD 3.3 OFFSET
6490          STA NLAH         SAVE RESULT
6500 XR6      JMP XR2          JUMP TO LOOP TOP
6510 XR7      JMP XRQ          JUMP TO LOOK-UP
6520;
6530          .FILE XREF2
```

XREF: BASIC FILE CROSS REFERENCE GENERATOR
PART TWO NEXT MONTH

PARALLEL PRINTER INTERFACE FOR SBII OR C1P

by: Guy Vanderwaeren
Wilgenstraat 73
B-2800 Mechelen, Belgium

Printed listings can be a great help in debugging. Being able to print opens a whole new world of art printings, text editors and printed reports. The trouble begins with the SBII. It has two unused serial ports, but you have to populate them and they will need a complicated switch to select only one port at a time or the cassette port. Most printers come with a parallel port, which let you spend more money for a special, optional serial port. Why shouldn't we consider designing a parallel interface, which can be easily made to our needs? The interface here described is going to be it. It has a parallel output port for data, another for some control signals and an input port to read the status of the printer. It has also its own EPROM to keep the printer software permanently and a few bytes of RAM to remember some parameters. These memories are not strictly needed, but can be helpful sophistication. The idea came from an article in MICRO (October 1982), but was changed a bit on the decodification of the locations and I added the memories. Because this circuit is not yet tested, I would appreciate any comments on eventual errors very much.

The interface is designed here especially for an EPSON MX80 printer, which has a CENTRONICS type interface. Most printers have this type of interface, but I will mention the signals that possibly could be different.

The whole interface occupies different memory locations. The I/O ports are located at $F3EA and $F3EB, the EPROM at $F300 to $F37F and the RAM at $F3D0 to $F3DF.

Let's have a look at the decodification of all those locations. The decoder consists of U1, U2, N1 to N5, N8, U4 and U5. A9 to A15 are decoded, together with N1 and N2, in U1. A2 to A7 in U2, together with N3 and N4. U4 and U5 do the final decoding of the outputs of U1 and U2, the R/W line and the clock 02. Finally, N8 and N5, together with the wired OR, formed by DD1 and DD2, decode the chip select for the RAM.

N11, N12 and N13 make the data direction signal DD. N12 and N13 are added to get an open collector output with enough power.

The input port is formed by N7, T1, N9 and N10. T1 is used as inverter. N7 and several printer status signals: BUSY, SELECT, PAPER END and ERROR. Any of these signals can be different on another printer. Some printers have more or less or may be inverted You will have to consult your printer manual to know how your particular printer tells its status.

The output port for data is U6. The output port for two control signals is U7. This last one is a double monostable multivibrator, which gives an output pulse of 1.5 microsec (strobe) and about 75 microsec (init). This could also be different for another printer. Consult your manual to see if the pulse widths are correct. The times in your manual will be minimal, so don't bother if the circuit gives a bit more. Another thing to check is if your printer needs these two signals as active low, otherwise you will have to use the inverted output(s) of the multivibrator

How does it work? This is a job for the software, which will be resident in the EPROM as a subroutine, called from Basic by a USR instruction. The software should first send a pulse to $F3EB, which sends an initialize signal. Some printers need this to be able to do anything. Then you have to send the data to $F3EA. At the same time, the circuit will generate the strobe pulse to tell the printer it can read the data. Next you have to read $F3EB to get the printer status which tells you if the printer is still busy, has no paper, detected an error, or is not selected. It would be wise to check this also after sending the init-pulse and before starting to print. This status is read from bit 7, which allows you to check it with a BMI command easily. After that, continue with the next data byte and another status check, etc.

Why did I add some RAM? The SBII has a normal number of 24 characters per line. If you are going to print, the computer will send, after each 24 characters, a carriage return and line feed. To be able to print longer lines, you can put the maximum length in a RAM location and use another one to count the characters already printed on the line. With the software, you can detect a CR code and not send it to the printer, but send one after it has printed the max. number of characters per line, controlled by a counter. It will make your program a bit more complex, but enlarges the possibilities.

There are only 128 bytes used and decoded by the EPROM. IF you should need more, you will have to change the decoding a bit and add an 8th address line to the EPROM. In RAM you have 16 bytes to use.

All connections to the left of the circuit are made to the 40 pin extension connector on the 600 board. Don't forget to put the two data buffers 8T28 in U6 and U7 on the 600 board. It would also be a good idea to buffer the address lines, the R/W line and the clock line.

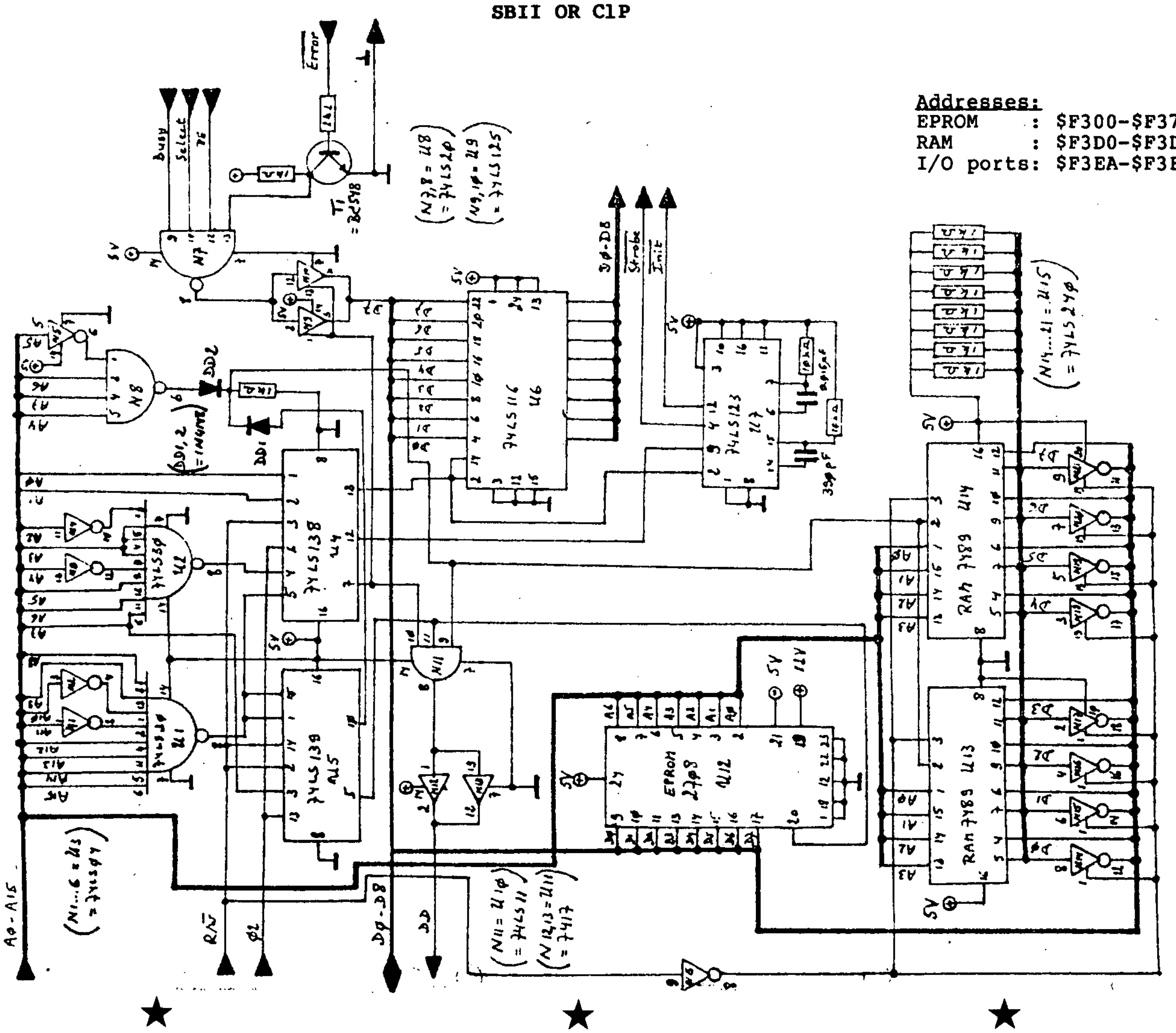
The connections to the right are going to the printer. It is up to you to find out at which pin on the printer connector must be connected. Your manual will tell you that.

Good luck.

Ed' Note:

Okidata, for one, supplies a line of printers called the Microline series which comes with both serial and parallel interfaces at no additional cost.

FIG.6.
PARALLEL PRINTER INTERFACE
SBII OR C1P



Addresses:
EPROM : $F300-$F37F
RAM : $F3D0-$F3DF
I/O ports: $F3EA-$F3EB

D&N MICRO PRODUCTS, INC.

3702 N. Wells St.
Fort Wayne, Ind. 46808
(219) 484-6414

TERMS $3.00 shipping, Foreign orders add 15%, Indiana residents add 5% sales tax.

COMPUTER

MICRO-80 COMPUTER
Z-80A CPU with 4Mhz clock and CP/M 2.2 operating system. 64K low power static memory. Centronics parallel printer port. 3 serial ports. 4" cooling fan. Two 8" single or double sided floppy disk drives. IBM single density 3740 format for 243K or storage, double density format for 604K of storage. Double sided drives allow 1.2 meg on each drive. Satin finish extruded aluminum with vinyl woodgrain decorative finish. 8 slot backplane, 48 pin buss compatible with OSI boards.

MODEL 80-1200 $2995
2 8" Single sided drives
MODEL 80-2400 $3495
2 8" Double sided drives

MICRO-65 COMPUTER
6502 CPU with 2Mhz clock and DOS-65 operating system. 48K of low power static memory. 2 serial ports and 1 Centronics parallel port. 2 8" single or double sided drives. Satin finish extruded aluminum with vinyl woodgrain finish. 8 slot backplane, 48 pin buss compatible with OSI. Will run OSI 65D and 65U software.

MODEL 65-1 $2995
2 8" Single sided drives
MODEL 65-2 $3495
2 8" Double sided drives

BP-580 8 Slot Backplane ..... $ 47
OSI 48 pin Buss compatible

MEM-CM9 MEMORY/ FLOPPY CONTROLLER
24K memory/floppy controller card uses 2114 memory chips, 1 8K and 1 16K partition. Supports OSI type disk interface

24MEM-CM9 .... $325
16MEM-CM9 .... $260
8MEM-CM9 .... $180
BARE MEM-CM9 .... $ 50
Controller on assembled unit add .... $ 90

BIO-1600 Bare IO card .... $ 50
Supports 8K of memory, 2 16 bit parallel ports, 5 serial ports, with manual and Molex connectors.

PRINTERS

Okidata
ML82A, 120 cps, 10" $409
ML83A, 120 cps, 15" $895
ML84 Parallel, 200 caps, 15" . $1150

C. Ioth
8510AP Prowriter, parallel ... $419
120 cps, correspondence quality
8510APD Prowriter, serial $585
F10-40PU Starwriter, parallel $1319
Letter quality daisy wheel
F10-40RU Starwriter, serial . . $1319
F10-55PU Printmaster $1610
parallel, Letter quality daisy wheel
F10-55RU Printmaster, serial $1610

DISK DRIVES AND CABLES
8" Shugart SA801 .... $385
single sided
8" Shugart SA851 $585
double sided
FLC-6 6 ft cable from D&N $69
or OSI disk controller to 8" drive
5¼" MPI B51 disk drive with . . $450
cable, power supply and cabinet. Specify computer type.
FLC-5¼ cable for connection . $75
to 5¼ drive and D&N or OSI controller, with data separator and disk switch. Specify computer type

HARDWARE
OSI COMPATIBLE

IO-CA10X Serial Printer Port . . $125
Specify Device #3 or #8
IO-CA9 Parallel Printer Port . . $150

CMOS-MEM
64K CMOS static memory board, uses 6116 chips, 3 16K, 1 8K and 2 4K blocks, Partitionable for multi-user, OSI type disk controller, 2 IO mapped serial ports for use with D&N-80 CPU. Ideal way to upgrade from cassette to disk.

64K CMOS-MEM $490
48K CMOS-MEM $390
24K CMOS-MEM $250
16K CMOS-MEM $200
Controller add . $ 90
2 IO mapped serial ports add. $125
on assembled memory board
Z80-IO 2 IO mapped serial $160
ports for use with D&N-80 CPU card
FL470 Disk Controller $155
Specify 5¼ or 8" drive

STANDARD CP/M FOR OSI

D&N-80 CPU CARD
The D&N-80 CPU allows the owner of an OSI static memory computer to convert to Industrial Standard IBM 3740 single density disk format and CP/M operating system. Double density disk operation is also supported for 608K of storage on an 8" diskette. When used with a 5¼" disk system 200K of storage is provided. Includes parallel printer and real time clock. Also available for polled keyboard and video systems. Compatible with C2, C3, C4 and 200 series OSI computers.

D&N-80-P **$349**

CP/M 2.2 .... **$150**

64K CMOS-MEM with D&N-80 CPU card **$450**

HARD DISK DRIVER $140
Allows D&N-80 CPU board to control OSI 40 or 80 meg hard disk unit. Will not destroy OSI files. Will also allow for a true 56K CP/M system. Specify 40 or 80 meg drive.

BUSS TRANSFER $135
Allows for D&N-80 and OSI CPU to be in the computer at the same time. Toggle switch provides for alternate CPU operation.

DISK TRANSFER $100
Utility program to transfer OSI CP/M format disk to IBM 3740 single density format. Will also transfer IBM to OSI format.

SYSTEM HARDWARE REQUIREMENTS
D&N-80 CPU, D&N FL470 or OSI 470 controller, 48K memory at 0000-BFFF, 4K memory at D000--DFFF, two disk drive cables.

FORMAT TRANSFER $15
You supply software on 8" diskette D&N will transfer OSI CP/M format to IBM 3740 CP/M format. Can also transfer IBM 3740 CP/M format to OSI CP/M format. Original diskette returned.

C1P CORNER

By: David A. Jones
9226 N.W. 17th Street
Coral Springs, FL 33065

Since publishing my article entitled EPROM BASIC in MICRO (Feb 1983), I have received more correspondence asking about my 32/64 character video modification to the C1P than requests for clarification on the subject of the article itself.

Suspecting there are other C1P owners who are interested in upgrading their system, I decided to submit the entire modification package to PEEK(65) for release to the public domain rather than try to answer any more individual queries myself. Bear in mind though, the modification was not designed to be a kit for hobbyists but rather to fulfill my personal needs.

It is definitely not for the inexperienced or casual tinkerer and certain parts may not be readily available, namely the crystal oscillator shown. I used a hybrid oscillator for the master clock as I had one at the right frequency on hand. Others may have to duplicate the original circuitry around U58 on the daughter board and use a crystal instead of an oscillator chip. The exact frequency is not critical, but 11.79648 MHz would be ideal. I used 11.750 MHz with good results.

It is possible to accomplish the task without the use of sophisticated test equipment. Both boards I modified were done at home with only a VOM and a flip-flop driving a LED (to check for presence of transitions) but I don't recommend this shoestring approach.

It would be nice if an OSI User's group would pick up on this. As a group effort, maybe some more improvements could be made and fedback to PEEK(65). I think there is still some life left in C1P's, but it requires more than someone working alone to get it. If anyone is interested, I'm offering my Superboard II for sale for $75.00. Again, maybe a group would be interested in it to use as a model for modifications. It has all the mods installed (these and the Micro article's). I'll be left with my C1PMF.

Naturally, the new video display requires new firmware to drive it. This will be the subject of the next article. For cassette users, a new monitor ROM is really the way to go. OS65D users can put the new firmware on disk. HEXDOS users can go either way. To save OS65D users the tedium of typing in the code, I'll install the patches and code on your bootable disk for $10.00, source code included. Let me know if your system is other than 32K. I'd offer the same to HEXDOS users, but I chose to put the code in EPROM and run it from there when using HEXDOS.

A MONITOR

A 64 character per line display requires direct video input to a TV or a monitor. I had been using a converted black and white 12" TV and was quite happy with the resolution but just recently bought a Zenith Data Sytems 123 green monitor. Locally, they are discounted to around $95.00. There is no comparison with the TV. The linearity is better, the contrast and brightness have greater ranges and the general design of the case is more pleasing. One minor complaint though. The screen is not as flat as the TV and some of the more expensive monitors, so a little optical distortion is present. I didn't notice this point until after I got it home. I would recommend this model for anyone looking for an under $100 monitor.

64/32 CHARACTER MODIFICATION

9 IC's are mounted on a daughter board interfaced to the 600 board by a 16 pin ribbon cable through U26. Additionally, the remaining 3 proto locations are used on the main board.

Not shown on the schematic are the 2 additional 2114 rams required to support the extra video locations. MCS1 goes to the original pair and MCS2 to the new ones. I piggy backed them on top of the first two. Also, section E of inverter U16 is shown on the schematic as driving the input of U44B. This inverter is not available unless you have accomplished the EPROM BASIC modification mentioned in the MICRO article. If you have replaced your monitor ROM with an EPROM, then U18 section B should be available for this function. Replacing the monitor ROM is inevitable for cassette users if you are going to 64 characters per line as the screen update routines access a constant there to determine the width of the video display. More on this next month.

Selection between the 64 mode and 32 mode is made by writing to $F300 (64) and $F700 (32) (POKE 62208 and 63232 respectively). Also, observe the changes made to U20, U23, and U56. Compare with your original schematic.

All IC's with U numbers are original locations. IC's with Z numbers are additions. Z4 and Z12 form a divider chain to generate horizontal and vertical sync pulses in place.

of the original chain U30-U61 which is now used solely to generate video ram addresses. By making the video ram addresses independent of the horizontal and vertical sync, the address counter can be stopped during the retrace interval thus allowing use of the 8 video locations on each line which were formerly blanked.

Flip-flop U44B halts the address counter after a count of 64 (or 32) and waits for a pulse from counter Z5, which is acting as a precision one shot, indicating a horizontal sync. The exact timing of this clear pulse and thus the positioning of the horizontal line is controlled by selecting the appropriate output of Z5. This may vary depending upon the amount of overscan in the particular monitor used.

Counter Z13, which is also acting as a precision one shot, clears the address counter during vertical retrace and controls the vertical positioning of the display. Fine positioning is controlled by selecting the output of Z13 and coarse positioning by selecting the source of the input to Z13.

Instead of using CMOS 4520 counters, TTL 74LS163 counters could be used, but the package count would be increased by 2 IC's and the power requirements would go up. IC Z2 generates the 5.875 MHz clock required for the 32 character mode and Z10B is necessary because of the limited frequency capabilities of CMOS when operated at 5V. Since the other half of Z10 is used in another circuit, no extra packages are used.

The CPU clock must come from Z10B as U30 now runs at a higher frequency and is halted during retrace, and the ACIA clock comes from Z4A for the same reason. On the schematic, arrows pointing up go to +5v and those pointing down go to 0v.

LOCA.	IC	VCC	Ov
Z1	Crystal Oscill	14	7
Z2	7492	5	10
Z3	74LS157	16	8
Z4	4520 CMOS	16	8
Z5	4017 CMOS	16	8
Z6	74LS04	14	7
Z7	not used		
Z8	not used		
Z9	7492	5	10
Z10	74LS74	14	7
Z11	not used		
Z12	4520 CMOS	16	8
Z13	4017 CMOS	16	8
U27	74LS139	16	8
U28	74LS157	16	8
U44	74LS74	16	8

64 CHAR CUTS AND JUMPERS

	FROM/TO	WHERE	SIGNAL
CUT U30-7	5V BUS & REST OF CHAIN	1 PLACE BOTTOM	
ADD U30-7	U44B-8		ENABLE
CUT U30-1,9	5V BUS	2 PLACES BOTTOM	
CUT U59,60, 61-1 & 9	5V BUS	1 PLACE BOTTOM	
ADD U30-1,9	U59-1		CLEAR
ADD U59,60, 61-1 & 9	U26-13		
CUT U20-1	A10 BAR	1 PLACE BOTTOM	
ADD U20-1	+5V		
CUT U20-15	U4-1,4,10,13 AND U5-1,4,10, 13	1 PLACE TOP	
ADD U20-14	U4-1,4,10,13 AND U5-1,4,10,13		RKB BAR
CUT U20-11	U21-11	1 PLACE BOTTOM	
ADD U20-10	U21-11		WKB BAR
CUT U56-2	A10 BAR	1 PLACE TOP	
ADD U56-2	+5V		
CUT U60-14	U41-8	1 PLACE TOP	
ADD U60-14	U28-3,5		C8
ADD U60-14	U26-6		
ADD U28-7	U41-8		C9/C8
CUT U60-13	U41-7	1 PLACE TOP	
ADD U60-13	U28-6,11		C9
ADD U28-9	U41-7		C10/C9
CUT U60-12	U41-6	1 PLACE TOP	
ADD U60-12	U28-10,14		C10
ADD U28-12	U41-6		C11/C10
ADD U59-11	U28-2		C7
ADD U60-11	U28-13		C11
ADD U60-11	U26-7		
ADD U44-10, 12	+5V		
CUT U17-5	U16-11	1 PLACE BOTTOM	* NOTE 1
ADD U28-4	U16-11		C8/C7
ADD U16-10	U44-11		C8/C7 BAR
ADD U26-12	U44-13		
CUT U18-1	U15-13 & W4 CENTER	1 PLACE BOTTOM	
ADD U61-11	U18-1		C15
ADD U18-2	U44-3		C15 BAR
ADD U44-6	U44-2		C16 BAR
ADD U44-6	U55-13		

At this time the 600 board should still function normally. Test to ensure that no mistakes have been made and that the board does in fact still work. The following steps must be completed in their entirety before additional testing can be accomplished.

CUT U54-6	U60-11		
ADD U26-9	U54-6		C8/C11
CUT U55-10	0V	1 PLACE BOTTOM	
CUT U55-11	02	1 PLACE TOP	
ADD U55-11	U21-9		A10
ADD U55-14	U21-8		A10 BAR
CUT U8-37	U30-13	1 PLACE BOTTOM	
ADD U26-3	U8-37		PHASE 0
CUT W9	U58-3	1 PLACE TOP	
ADD U26-16	W9 & U30 ETC		CLK
CUT U65-1	U61-11	1 PLACE TOP	
ADD U26-11	U65-1		HORIZ
CUT U65-9	U59-11	1 PLACE TOP	
ADD U26-10	U65-9		VERTICAL
ADD U26-2	U28-1		64/32
ADD U26-4	U27-11		32 BAR
ADD U26-5	U27-9		64 BAR
ADD U27-15	U17-9		
ADD U27-14	U21-2		A11
ADD U27-13	U21-8		A10 BAR
CUT U59-14	U57-2	1 PLACE TOP	
ADD U26-14	U57-2		ACIA CLK
ADD U55-12	PIGGY BACK U39 & U40-8		MCS2 BAR

NOTE 1: ROMS MUST HAVE BEEN REPLACED WITH EPROMS IN ORDER TO FREE THIS INVERTER.

Schematic on page 16

3 USERS-80 Mega Bytes —$9990^00*

INTRODUCTORY SPECIAL

WITH DUAL FLOPPIES
BRAND NEW —
1 YEAR WARRANTY ON HARD DISK!
REGULAR $12,990.00

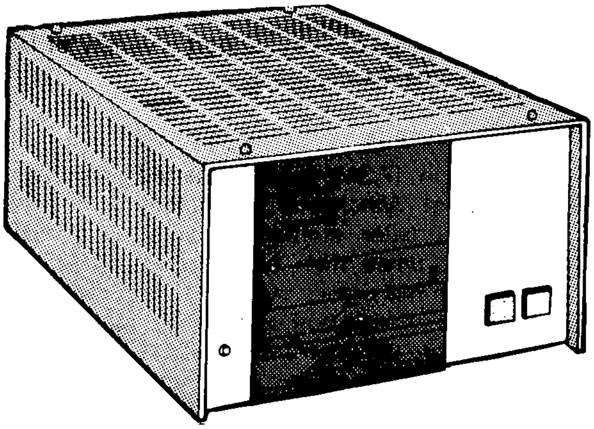
- 90 Days on Power Supply, Floppy Drives — Circuit Boards.
- Configured for Time-Share @ 2 MHZ
- Includes: 2 Serial Printer Ports with Handshake, Improved Cooling, and Ball Bearing Roller Chassis Rails

ALSO AVAILABLE WITH 3 MULTI-PROCESSOR

Denver Boards with 64K each user and Centronics Parallel Printer Port at
$10,990.00

*DEALER DISCOUNTS AVAILABLE

8" HARD DISK SYSTEMS



SINGLE BOX TABLE TOP WITH IMPROVED COOLING 10 M/B HARD DISK AND 8" FLOPPY DISK **2 USERS AND 2 SERIAL PRINTER PORTS**
$5990.00

AS ABOVE WITH 2 MULTI-PROCESSOR 64K DENVER BOARDS PLUS CENTRONIC PARALLEL INTERFACE **$6990.00**

OR INSTALLED IN CABINET AS ABOVE
WITH DUAL FLOPPIES PLUS 10 M/B.

STD. BOARD TYPE	1 USER w/ Centronics Printer Port	$6490.00
	2 USER w/ 2 Serial Printer Ports	$6990.00
DBI MULTI PROC.	2 USER w/Centronics Printer Port	$7790.00
	3 USER w/Centronics & Serial Printer Ports	$8990.00

MULTI-PROCESSOR DEVELOPMENT SYSTEM SPECIAL

SPECIAL

- 5 M/B Hard Disk-1 8" Floppy
- 1 Centronics Parallel Printer Port
- 1 Serial Printer Port, 1 Modem Port
- 2 DB-1 Multi-Processors
- Complete Programmer Manual and Software Overlays

ONLY **$5990.00**

DEALERS — We have lots of OSI machines and can build virtually any combination you need. Appropriate dealer discounts.

Please Give Us a Call!

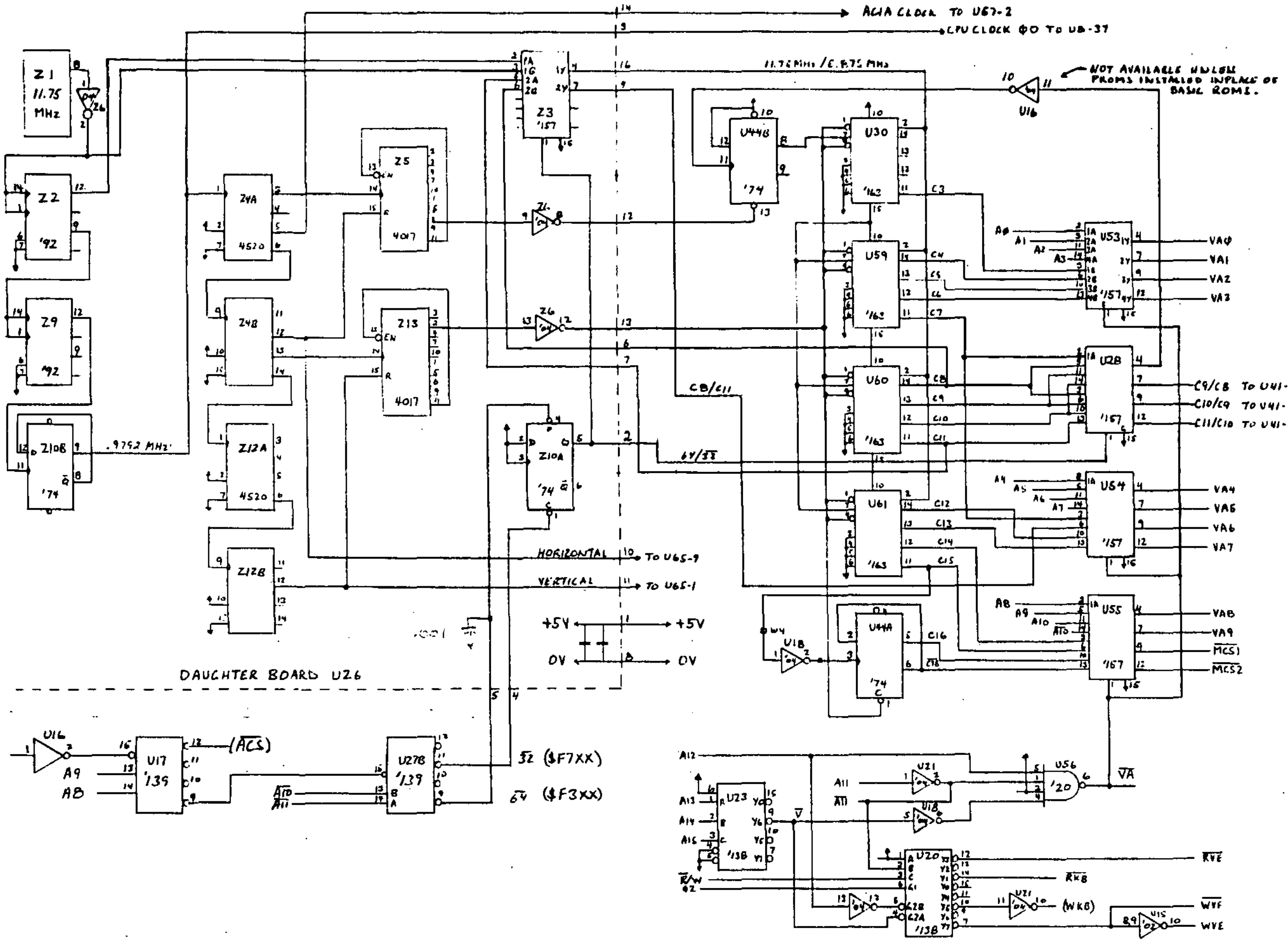
WHERE WE STILL LOVE OS-65U — AND SUPPORT IT!

Space-Com International

22991 LA CADENA DRIVE, LAGUNA HILLS, CALIFORNIA 92653

ORDER TODAY (714) **951-4648** SOME QUANTITIES LIMITED

Continued from page 14



D JONES 1-25-82, REV 4-30-83

★ ★ ★

LETTERS

ED:

The odd's and ends below are my way of trying to pay back for all the help I have received from PEEK(65) in the past.

I am using a C3 with a CD-36 under 65U version 1.43, so this information should work on versions 1.40 and later.

Here is a map of the FLAG's under 65U V1.43 and the program used to create it. Thanks to the great article by Greg Stevens in the Dec. 1980 issue of PEEK(65), I have been able to add some of my own FLAG's. As you may have noticed, OSI has cleaned up this table as well as leaving about 30 bytes at the end of the table.

I came across the syntax for an input statement I have not seen in any manual:

```
INPUT#A,"STRING:";#B,ST$
```

The string is printed to device A and input comes from device B. Are there some more odd syntax forms?

In response to the question by Daniel J. McDonald in the June issue about making a VALPTR command, a good place to start would be Tech. newsletter 21.

The article by Colin Law in the June issue prompted me to put my two cents worth in about DIRectory programs. Since the more you add to the DIRectory, the more time it takes, I have three DIRectory programs. A fast one to print just the names, the normal one that comes with the system and an extended, listing one that has some extra information.

If you are changing over to V1.43, you will have to change any THEN's with a variable (IF X THEN G1) to GOTO (IF X GOTO G1).

For those who have a CD-72/36 and are thinking of converting to an S-100 computer, you can get a disk controller from Gifford Computer Systems, 1922 Republic Avenue, San Leandro, CA 94577.

If you are like me, you spend a lot of time working on your computer and every little annoyance becomes maddening after a while, so here are a few ways I got rid of some that were annoying me.

I hate having to use a control Q to continue an output so I removed the test in the OS by putting NOP's in 16095 to 16098, so any key will continue output. Now if I could only find out how to get control C to work on an input!

Sometimes I want to do something special. If after a set time, the terminal has not been used and I don't want to leave the program, I change the RUN"RTMON" in the OS to a GOTO 999 at the start of the program so that when the count down timer hits zero it goes to line 999 where I do whatever it is I want to do on a time out. Then before any input, I set the timer and turn it on. Then after the input I turn off the timer. The RUN"RTMON" is at 55905 to 55913, just poke in whatever commands you want and space allows.

Here are a few questions for anyone who knows and cares to answer:

Why did OSI find it necessary to require extended input and print variables to be a string even if a number is to be entered, and why are multiple variable inputs disallowed?

With extended input on, you can input up to 255 characters, so why does the EDITOR still limit you to 71 characters?

Does anyone know of any free space under 65U V1.43?

How do you get Input/Output and handshaking on the CA-10 board?

How do you get a UF error? If I call a function that has not been defined, I get a NF error!!

What is the 16 byte file header used for?

I haven't had a chance to try this yet, but I was wondering if arranging the BASIC's tables, the dispatch and reserved word tables, into a different order would speed up a program? I sometimes change the reserved word table when I get tired of seeing the same commands (e.g. PRINT to WRITE, etc.).

Here are a few things that I think would be nice to have.

Hint! Hint!!

How about date stamping for files, maybe using the same setup as CP/M plus where the date info is kept as a DIR entry, or on the file header? I consider date stamping a must for a good automatic back-up program.

A second high level language that works under level III and is file compatible.

FLAGS

```
10 dv=6: ad=18919
1000 Print#dv,"Address";Tab(10);"Flag No.";Tab(19);"Value";Tab(25);
1010 Print#dv,"Low Add";Tab(34);"High Add";Tab(45);"Basic": Gosub 2010
1020 If Peek(ad)=255 Then Gosub 2010: Print#dv: End
1030 If Peek(ad)=254 Then Gosub 2000: Goto 1020
1040 Print#dv,Tab(11);[4,"R"]Str$(Peek(ad)And127);Tab(19);
1050 Print#dv,[4,"R"]Str$(Peek(ad+1));Tab(25);[4,"R"]Str$(Peek(ad+2));
1060 Print#dv,Tab(35);[4,"R"]Str$(Peek(ad+3));Tab(45);"POKE";
1070 Print#dv,Str$(Peek(ad+2)+Peek(ad+3)*256);",";
1080 Print#dv,Mid$(Str$(Peek(ad+1)),2): ad=ad+4: Goto 1020
1090 ad=ad+4: Goto 1020
2000 Print#dv,[7,"R"]Str$(ad): ad=Peek(ad+2)+Peek(ad+3)*256
2010 Gosub 3000: Print#dv,[7,"R"]Str$(ad);: Return
3000 For loop=1 To 60: Print#dv,"-";: Next loop: Print#dv: Return
```

```
1 REM ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
2 REM ::                          Mike Fowler                        ::
3 REM ::      P.O. Box 5971  /  San Bernardino  /  CA  /  92412      ::
4 REM ::    Name: BIGDIR   /   Password: ANAN   /   Date: 07-22-83   ::
5 REM ::    Desc: Prints an extended directory listing.              ::
6 REM ::          Slow due to the fact that each file header is read. ::
7 REM ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
9 :
10 Goto 1000 : REM SAVE SPACE FOR SUBROUTINES
20 :
30 REM          P A G E   H E A D I N G
40 :
50 Print#dv,Tab(25);"<*<*<* OS-65U EXTENDED FILE DIRECTORY *>*>*>";
55 pg=pg+i:Print#dv,Tab(84);"Page";pg: Gosub 360
60 Print#dv: Print#dv,"Source Device -- ";df$;Tab(70);"Date ------ ";dt$
70 Print#dv,"Source Type ---- ";st$;Tab(70);"Time ------ ";ti$
80 Print#dv,"Source Title --- ";t$;
90 If st$<>"Floppy" Then Print#dv,Tab(70);"Disk Cyl. -";cy
100 Print#dv: Print#dv: Print#dv,"Name";Tab(9)"Type";Tab(15);"Access";
110 Print#dv,Tab(23);"P/W";Tab(30);"Lenth";Tab(37);"Pack";Tab(43);
120 Print#dv,"Offset";Tab(53);"Used";Tab(62);"Ba-Addr";Tab(72);
130 Print#dv,"Sec Len";: If st$="Floppy" Tnen 150
140 Print#dv,Tab(82);"H/D Addr";
150 If dv=i Then Print#dv: Goto 170
160 tm=Pos(dv): Print#dv,Chr$(13);
170 For loop=i To tm: Print#dv,"_";: Next loop: Print#dv: Return
200 :
210 REM         L O W E R   C A S E   T O   U P P E R   C A S E
220 :
230 Ifaw$=""ThenReturn
240 xx$=aw$:aw$="":Forxx=iToLen(xx$):zz=Asc(Mid$(xx$,xx,i))
250 aw$=aw$+Chr$(zz+(32*(zz>96Andzz<123))):Nextxx:Return
260 :
270 REM         G E T   S Y S T E M   D A T E
280 :
290 xx=55922: If lv<2 Then xx=24569
300 dt$=Right$("0"+Mid$(Str$(Peek(xx+i)),2),2)+"-"
310 dt$=dt$+Rignt$("0"+Mid$(Str$(Peek(xx)),2),2)+"-"
320 dt$=dt$+Mid$(Str$(Peek(xx+2)),2): Return
330 :
340 REM         G E T   S Y S T E M   T I M E
350 :
360 ti$="": If lv<2 Then Return
370 ti$=Rignt$("0"+Mid$(Str$(Peek(55921)),2),2)+":"
380 ti$=ti$+Right$("0"+Mid$(Str$(Peek(55920)),2),2)+":"
390 ti$=ti$+Right$("0"+Mid$(Str$(Peek(55919)),2),2): Return
600 :
610 REM        X F R   S E T U P
620 :
630 Poke 8778,192: Poke 8779,36: REM pnt USR tp interf subr
640 Poke 9435,232: Poke 9436,40: REM interf subr GET$
650 ra=9970: REM scrbuf
660 cb=9889: REM xfr control block
670 Poke cb+5,0: Poke cb+6,i: REM xfr 256 bytes
680 Poke cb+7,ra-Int(ra/256)*256: Poke cb+8,ra/256: REM ram adr
690 a=9899: REM loc of dir da
700 ea=256*(Peek(a)+256*(Peek(a+i)+256*Peek(a+2)))
710 s=9902: REM loc if dir size
720 es=256*(Peek(s)+256*(Peek(s+i)+256*Peek(s+2)))
730 en=ea+es: REM end of dir da
740 ha=en: REM highest file adr found
750 Return
800 :
810 REM        S E T U P   F O R  R E A D
820 :
830 dh=Int(ea/16777216): rm=ea-dh*16777216
840 dm=Int(rm/65536): rm=rm-dm*65536
850 dl=Int(rm/256): rm=rm-dl*256: db=rm
860 Poke cb+i,db: Poke cb+2,dl: Poke cb+3,dm: Poke cb+4,db
870 el=Peek(134)*Peek(135): er=Usr(0): If er<>0 Then 50130
880 Return
900 :
1000 Gosub 10030: REM init
1010 Gosub 2030:  REM get setup data
1020 Gosub 5030:  REM do it
1100 Goto 30030:  REM exit
2000 :
2010 REM       S C R E E N   D I S P L A Y
2020 :
2030 Print cs$;Tab(20);fg$;" E X T E N D E D   D I R E C T O R Y ";bg$: Print
2040 aw$=dd$: Input "Device: ";[i,"A"]aw$: If aw$="" Then aw$=dd$
2050 Gosub 230: If aw$="ABORT" Then 30030
```

Continued on page 19

How about a "RAM disk" to speed things up! The RAM should be a self contained external unit to save space and to prevent power supply problems.

Hark, what error breaks upon yonder terminal? I hear bells and it isn't Avon, so I guess I had better quit for now.

Mike Fowler
San Bernardino, CA 92412

Mike:

Re your question. Does anyone know of any free space under 65U V1.43? There are reported to be a few bytes, but sparce few AND as many users may confess, it is mighty disheartening to upgrade to a new version and find the space used. For that reason it's not recommended.

You ask, how do you get Input/Output and handshaking on the CA-10 board? It can, and is done regularly by shop people, but a bit much to describe here. We hope to have an article on this soon.

PEEK(65) Staff.

* * * * *

```
2060 If aw$<"A" Or aw$>"F" Then Print bl$;Chr$(13);: Goto 2040
2070 df$=aw$: Print: Print
2080 dv=5: REM print device
2110 Print: Print
2120 t$="": Input"Title for Printout: ";[40,"A"]t$
2130 Dev df$: Open"DIREC*","PASS",1: Close i
2140 :
2150 REM   determine current system size
2160 :
2170 If df$<"E" Then cd=74: hs=275967: cs=3584: Goto 2190
2180 Open"BEXEC*","PASS",1: Index<i>=16: Input%i,cd,hs,cs: Close i
2190 sa=0: no$=""
2200 If lv=2 Or lv=4 Then no$=Chr$(Peek(57199)+75)
2210 If df$=no$ Or df$="E" Or (lv=4 And df$="F") Then sa=-i
2212 REM check for level I, TS or TSNET local access
2220 If lv=i Or lv=3 Or (lv=4 And sa) Then Goto 2280
2222 REM TSNET remote or workstation remote
2230 If lv=4 Or (lv=2 And Not sa) Then Goto 2420
2235 REM workstation local
2240 Goto 2480
2250 :
2260 REM level I, TS or TSNET local
2270 :
2280 If cd<>36 And cd<>74 Then 2320
2290 hs=Peek(13651)+Peek(13655)*cs
2300 hs=hs-(Peek(13314)+Peek(13315)/128)*cs: Goto 2520
2310 :
2320 If cd<>23 Then 2360
2330 hs=Peek(13634))*cs: hs=hs-((Peek(13314)+(Peek(13315)/128))*cs
2340 Goto 2520
2350 :
2360 If cd<>28 And cd<>7 Then 2380
2370 hs=(Peek(13637)+(Peek(13639)*256))*cs
2380 hs=hs-((peel(13314)+(Peek(13315)/128))*cs: Goto 2520
2390 :
2400 REM TSNET remote or workstation remote
2410 :
2420 hs=Peek(18152)*256+Peek(18153)*65536
2430 hs=hs+Peek(18154)*16777216: t=Peek(18146)*256
2440 t=t+Peek(18147)*65536+Peek(18148)*16777216: hs=hs-t: Goto 2520
2450 :
2460 REM workstation local access
2470 :
2480 hs=Peek(18149)*256+Peek(18150)*65536+Peek(18151)*16777216
2490 t=Peek(18143)*256+Peek(18144)*65536+Peek(18145)*16777216
2500 hs=hs-t
2510 :
2520 st$="Floppy": If df$>"D" Then st$="CD-"+Mid$(Str$(cd),2)
2530 Gosub 290: REM get date
2540 cy=(Peek(13314)*cs)/cs+Peek(13315)*2*cs: Return
3000 :
3010 REM      R E A D   D I R
```

Listing continued

```
3020 :
3030 of=16: REM skip header
3040 :
3045 Gosub 830: REM read p of dir
3050 rt=ra+of: REM ram adr cur ent
3060 ec=ec+i:  REM entry no.
3070 If Peek(rt)=0 Tnen Return: REM empty de
3075 If f% Then er=8: el=3500 :Goto 50130
3080 :
3090 REM        read name
3100 :
3110 n$="": For l=0 To 5: n$=n$+Chr$(Peek(rt+i)): Next l
3120 REM      read dir bit
3140 :
3150 REM       read type
3160 :
3170 tm=Int((Peek(rt+8) And 28)/4): ty$="Other"
3180 If tm=0 Then ty$="Data"
3190 If tm=i Tnen ty$="Basic"
3200 If (tm And 128)<>0 Then ty$="Direc"
3210 :
3220 REM     read access
3230 :
3240 tm=Peek(rt+8) And 3: ar$="None"
3260 If tm=i Then ar$="Read"
3270 If tm=2 Then ar$="Write"
3280 If tm=3 Tnen ar$="R/W"
3290 :
3300 REM      read adr
3310 :
3320 da=256*(Peek(rt+9)+256*(Peek(rt+10)+256*Peek(rt+11)))
3330 :
3340 REM      read size
3350 :
3360 sz=256*(Peek(rt+12)+256*(Peek(rt+13)+256*Peek(rt+14)))
3370 :
3380 REM       password
3390 :
3400 pl=(Peek(rt+6) And 240)/16: p2=Peek(rt+6) And 15
3410 p3=(Peek(rt+7) And 240)/16: p4=Peek(rt+7) And 15
3420 pl=(pl+(65*Abs(pl<16))): p2=93+(p2=15)
3430 p3=(p3+(65*Aos(p3<16))): p4=93+(p4=15)
3440 pw$=Chr$(pl)+Chr$(p2)+Cnr$(p3)+Cnr$(p4)
3450 If pw$="A]A]" Then pw$="ANAN"
3460 If pw$="P\P]" Then pw$="PASS"
3470 :
3472 REM          HD addr & sec len
3474 :
3480 dn=256*(Peek(rt+25)+256*(Peek(rt+26)+256*Peek(rt+27)))
3490 dx=da: If df$>"D" Tnen dx=da+Peek(13314)*cs+Peek(13315)*2*cs
3500 sn=dn-da: If sn>=1e9 Or sn<sz Tnen sn=sz: f%=-i
3510 :
3512 REM         pack check
3514 :
3520 pk$="Yes": If (sz/3584)>Int(sz/3584)Then pk$="No"
3522 :
3530 If n$="DIREC*" Then ty$="Data": Goto 3570
3532 :
3534 REM        offset and bytes used
3536 :
3540 sa=ea: ea=da: Gosub 830: ea=sa
3550 os=Peek(9986)+Peek(9987)*256: bu=Peek(9988)+Peek(9989)*256
3560 Gosub 830: If Peek(rt)=i Tnen n$="[----]": pw$="": ty$="": ar$=""
3562 :
3564 REM         print data
3566 :
3570 If Peek(14457)=Peek(15908) Then Gosub 50
3572 Print#dv,n$;Tab(9);ty$;Tab(16);ar$;Tab(23);pw$;Tab(27);
3580 Print#dv,[8,"R"]Str$(sz);Tab(38);pk$;: If ty$="Data" Then 3590
3582 Print#dv,Tab(42);[6,"R"]Str$(os);Tab(50);[7,"R"]Str$(bu);
3590 Print#dv,Tab(60);[9,"R"]Str$(da);Tab(71);[8,"R"]Str$(sn);
3600 If st$<>"Floppy" Then Print#dv,Tab(80);[10,"R"]Str$(dx);
3620 Print#dv: If Peek(15006)=0 Then 3660
3630 aw$="N": Input "Do you wish to continue? ";[i,"A"]aw$: Gosub 230
3640 If Left$(aw$,i)="N" Then Return
3650 Print Chr$(13);Spc(70);Chr$(13);: Poke 15006,0
3652 :
3654 REM         repeat until done
3656 :
3660 If da+sz>ha Then ha=da+sz: REM update nignest adr
3670 If Peek(rt)=i Then re=re+sz: dc=dc+i
3680 of=of+16: REM pnt to nxt
3690 If of<240 Then 3050
3700 ea=ea+240: REM da of next page of dir
3710 of=0: REM offset in scrbuf
3720 If ea<en Tnen 3045
3730 :
3740 Return
5000 :
5010 REM          M A I N
5020 :
5030 Gosub 630: REM xfr setup subr
5040 Print#dv,Chr$(27)+Cnr$(56)+Chr$(13);
5180 Gosub 3030: REM prt data
5190 Print#dv: Print#dv,"System Size";Str$(hs);", Bytes Free";
5200 Print#dv,Str$(hs-ha);", Bytes Used";na
5210 If dc>0 Then Print#dv,"File(s) Deleted";Str$(dc);", ";
5220 If re>0 Then Print#dv,"Bytes Recoveraole";re
5230 Print#dv,"Files In Use";Str$(ec-i-dc);", ";
5232 Print#dv,"Total Files Definded";ec-i;"of";es/16-i;"Possible."
5240 Print#dv,Chr$(27)+Chr$(54);
5250 Flag 101: Print#dv!: Return
```

Listing cont. on page 21

ED:

The OS65D3.2 version of WP6502 that I have (circa 4/82) has manifested several annoying problems: 1) When using the View mode with material containing referenced blocks of text, if a block extends across a page break garbage is displayed (or printed) following the end of the block; 2) Backwards scrolling through the text in L/Edit mode occasionally locks up, particularly when using the 24 character video display on the C1P. Wider screen formats rarely encounter this problem; 3) The polled keyboard routine alters lower case characters that are entered immediately following any depression of the CTRL key. Interestingly this occurs even if a CTRL character is not entered, i.e. the CTRL key is merely pressed and released. This problem does not occur in systems where a serial keyboard is used. This problem is particularly annoying when one is trying to edit in 'Move this' markers.

I have developed the following solution to these problems:

1) Boot up the system using a standard OS65D3.2 disk.

2) EXIT to the DOS kernal and load the Extended Monitor by typing EM.

3) Exit the EM by typing EX.

4) Remove the OS65D3.2 disk and insert a copy of the WP 6502 disk.

5) Type LO WP6502.

6) Type RE EM to re-enter the Extended Monitor.

7) Enter the following commands to correct problem #1:

```
     @3769<cr>
     3769/20 will be displayed
     enter 4C<cr>
```

8) Enter the following commands to correct problem #2:

```
     @3D65<cr>
     3D65/20 will be displayed
     enter 2F<cr>
```

9) Enter the following command to correct problem #3:

```
     M40B0=40B3,4109<cr>
     and then the following:
```

Command	Display	Enter
@4106<cr>	4106/AD	8D<lf>
	4107/17	00<lf>
	4108/02	10<cr>
@40A8<cr>	40A8/3D	3A<cr>
@40AB<cr>	40AB/11	0E<cr>
@40B5<cr>	40B5/E1	E4<cr>
@40E2<CR>	40E2/9D	A0<CR>

Continued on page 22

```
10000 :
10010 REM       I N I T
10020 :
10030 Flag 25: Flag 23: Flag 27: i=1: Poke 23721,255
10040 If Peek(18959)>2 Then 10070
10050 Print: Print"Extended input required for this program!!";Chr$(7)
10060 Print: er=-2 : Goto 30030
10070 Gosub 63010: ulsr=Peek(8778): u2sr=Peek(8779)
10080 dd=Peek(9832): If dd>127 Then dd=dd-124: If dd>63 Then dd=dd-58
10090 d d$=Chr$(dd+65): lv=Peek(16317)
10100 Return
30000 :
30010 REM       E X I T
30020 :
30030 Flag 24: Poke 23721,0: If er=-2 Then 30060
30040 Print#dv!: Poke 8778,ulsr: Poke 8779,u2sr: Dev dd$
30050 If er=0 Then Flag 28: Flag 26: Run"BEXEC*"
30060 Print: Input"Hit <cr> to continue ";aw$: Gosub 230
30070 If aw$="A" Then Run
30080 Flag 28: If aw$="STOP" Then Flag 26: End
30090 If er And er<>-i Then Flag 26: End
30100 Flag 26: End
50000 :
50010 REM       User Programmable Error Recovery
50020 :
50030 el=Peek(11774)+256*Peek(11775): REM Get Error Line
50040 er=Peek(18176): If er=23 Goto 50120: REM BASIC or Disk
50050 :
50060 REM      Decode BASIC Error
50070 z$=Chr$(Peek(er+867)And127)+Chr$(Peek(868+er)And127)
50080 er$="BASIC "+z$+" Error in line"+Str$(el)+"."
50090 Goto 50170
50100 :
50110 REM      Decode Disk Error
50120 er=Peek(10226)
50130 z=Peek(9832): If z>127 Then z=z-124: If z>63 Then z=z-58
50140 er$="Device "+Chr$(65+z)+" Disk Error"+Str$(er)
50150 er$=er$+" in line"+Str$(el)
50160 :
50170 Print: Print er$+Chr$(7) : Print : Goto 30030
63000 :
63010 z=6345
63020 ad$="":dl$="":de$="":ar=1:xf=0:yf=0
63030 cs$="":ce$="":cl$="":fg$="":bg$="":bl$=Chr$(7)
63040 REM Fetch Cursor Codes from System
63050 zl=Peek(z): If zl>127 Then ar=2:zl=zl-128
63060 ad$=ad$+Chr$(zl) : REM - cursor address
63070 z=z+i: zl=Peek(z): If zl<128 And zl Goto 63060
63080 If zl=0 Goto 63120
63090 zl=zl-128
63100 dl$=dl$+Chr$(zl):z=z+i: zl=Peek(z): If zl<128 And zl Goto 63100
63110 If zl=0 Goto 63120
```

Listing continued

Address	Flag No.	Value	Low Add	High Add	Basic
18919	1	44	156	68	POKE 17564,44
	2	32	156	68	POKE 17564,32
	5	140	64	74	POKE 19008,140
	5	60	65	74	POKE 19009,60
	6	144	64	74	POKE 19008,144
	6	41	65	74	POKE 19009,41
	7	0	23	61	POKE 15639,0
	8	32	23	61	POKE 15639,32
	9	3	126	62	POKE 15998,3
18955					
16285	9	249	133	62	POKE 16005,249
	10	0	126	62	POKE 15998,0
	11	255	63	56	POKE 14399,255
	12	0	63	56	POKE 14399,0
	13	104	48	18	POKE 4656,104
	13	104	49	18	POKE 4657,104
	13	96	50	18	POKE 4658,96
16313					
8576	14	162	48	18	POKE 4656,162
	14	2	49	18	POKE 4657,2
	14	76	50	18	POKE 4658,76
	15	13	156	11	POKE 2972,13
	15	13	160	11	POKE 2976,13
	16	58	156	11	POKE 2972,58
	16	44	160	11	POKE 2976,44
	17	0	116	10	POKE 2676,0
	17	0	123	10	POKE 2683,0
	18	13	116	10	POKE 2676,13
8616					
8463	18	10	123	10	POKE 2683,10
	21	28	106	36	POKE 9322,28
	21	11	107	36	POKE 9323,11
	22	123	106	36	POKE 9322,123
	22	37	107	36	POKE 9323,37
	23	0	133	62	POKE 16005,0
	23	3	126	62	POKE 15998,3
8491					
17578	24	249	133	62	POKE 16005,249
	24	0	126	62	POKE 15998,0
	25	96	25	8	POKE 2073,96
	26	76	25	8	POKE 2073,76
	27	0	72	11	POKE 2888,0
	28	27	72	11	POKE 2888,27
	29	96	184	64	POKE 16568,96
	30	234	184	64	POKE 16568,234
	31	36	202	77	POKE 19914,36
	32	73	202	77	POKE 19914,73
17618					
6070	33	36	236	77	POKE 19948,36
	34	73	236	77	POKE 19948,73
	100	0	103	25	POKE 6503,0
	101	255	103	25	POKE 6503,255
	0	0	2	52	POKE 13314,0
	0	0	3	52	POKE 13315,0
	26	0	158	58	POKE 15006,0
6098					MINE

```
63120 de$="":Goto 63140
63130 z=z+i:zl=Peek(z)
63140 If zl Then de$=de$+Cnr$(zl):Goto 63130
63150 xf=Peek(z+i):yf=Peek(z+2): If xz>127 Tnen xf=xf-128: ar=ar+2
63160 z=z+3:cs$=""    : REM - clear screen
63170 zl=Peek(z):z=z+i:If zl Then cs$=cs$+Cnr$(zl):Goto 63170
63180 cs$=cs$+Cnr$(13): ce$=""    : REM - clear to end of screen
63190 zl=Peek(z):z=z+i:If zl Then ce$=ce$+Cnr$(zl):Goto 63190
63200 cl$=""    : REM - clear to end of line
63210 zl=Peek(z):z=z+i:If zl Then cl$=cl$+Cnr$(zl):Goto 63210
63220 fg$=""     : REM - foreground
63230 zl=Peek(z):z=z+i:If zl Then fg$=fg$+Cnr$(zl):Goto 63230
63240 bg$=""     : REM - background
63250 zl=Peek(z):z=z+i:If zl Then bg$=bg$+Cnr$(zl):Goto 63250
63260 Return
63270 :
63999 Save"BIGDIR","ANAN"
```

* *

CONTINUED FROM PAGE 20

```
@40F1<cr>    40F1/8E      91<cr>
```

10) Exit the EM by typing EX.

11) Type PU WP6502.

You now have a corrected version of WP6502.

Jim Hays
Seattle, WA 98116

* * * * *

ED:

Re: Larry Horst's letter: November '83 issue PEEK(65). Several possible things come to mind.

The first is that the IO-1600 board is addressed at $CE00, not $CF00. The address given in the letter, 52992, is for $CF00. BUT, the IO-1600 board comes wired for $CE00 (52736). This is OK generally, because the IO-1600 usually supports OS65U, which requires the $CE00 address. The board isn't normally used with a C1P!

To check how the board is wired, place it in front of you, with the 48-pin connector to the right, and components up. Along the edge next to you, just to the right of center, is a 24-pin chip (a 74154). Between that chip and the one to the left is a pad pattern like this:

```
              o (W14)
  (W18) o
              o (W13)
              o (W17)
  (W19) o     o (W16)
              o (W15)
```

(The numbers aren't marked!) If, on the back of the board, W18 and W14 are connected with a trace, the address is $CE00 (52736). To change the address to $CF00 (52992), cut the trace and put a jumper between W18 and W13.

Another possible problem is that the address listed (52992) is for the CONTROL register of the ACIA of the serial port... not the DATA register. It may be that the only change needed is to POKE 52993 instead of 52992!

About software: OS65D3.2 supports the cluster ports as device #8. BUT... since a cluster port can't be put on a C1P, the I/O vectors are arranged to point to a 'null' routine and not the actual I/O code. It is still there, however, and can be used if the I/O vectors are reset. To reset the vectors, change $230F to $AF; $2310 to $24 for the input, and change $231F to $BC; $2320 to $24 for output. (POKE 8975,175: POKE 8976,36: POKE 8981,188: POKE 8982,36)

The ACIA's are initialized by cold-boot code, so it shouldn't be a problem.

One last point regarding device #8 w/OS65D. The cluster port software always operates on a "base address" + "offset". The base address is $CF00 (52992). The offset is contained in address $2323 (8995). EVEN offsets point to CONTROL registers, while ODD offsets point to DATA registers. OS65D initializes this offset to $00. You have to change it to use the device #8, such as PRINT#8,"xxx". With two ports installed, the value of the offset must be between 0 and 3.

If Mr. Horst isn't using OS65D, its harder. I recommend using OS65D!

To use the cluster port without OS65D, you have to initialize the cluster ports, and do all the work yourself for input or output. The BASIC program shows a way to do output, but a word of warning. My IO-1600 IS NOT on my C1P. Therefore, I don't KNOW that this will work. It should.

```
ACIA = 52992
RST  = 3:REM RESET ACIA VALUE
SET  = 17:REM /16, 8 BIT,
       2STOP, EVEN
POKE ACIA,RST:REM PORT 0
POKE ACIA,SET
POKE ACIA+2,RST:REM PORT 1
POKE ACIA+2,SET
.......
.......
PROGRAM
.......
REM TO OUTPUT, SOMETHING LIKE
       THIS WILL WORK
OUT$  = "ANSWER"
FOR I = 1 TO LEN(OUT$)

POKE
ACIA+1,ASC(MID$(OUT$,I,1))
WAIT ACIA,1
NEXT I
```

This routine assumes that the device at the output can accept the data as fast as it is sent. If it can't, you're in trouble. It's a LOT harder!!!

Oh, one other thing. Make sure that SW3 is set correctly. It provides clock to the ACIAs. No clock, no work. The D&N instructions explain how to set the switches (it's just like that for the serial port).

I have a fully populated D&N IO-1600 board on my C2. Hardware has been no problem. I'm only using the Centronics port, and the serial port, so I don't know from experience of any problems with the cluster port. D&N has always been good stuff, and the people very responsive and helpful.

If Mr. Horst wants to talk to me, my 'phone number is (612)-781-1359. After 10 PM, or before 10 AM on Saturdays and

Sundays I'm a bit surly! Any other time, I'm just my usual nasty self. I'm not sure that I could help, but I'm willing to try if he thinks that he could benefit from a discussion.

Good luck.

Donn Burke Baker
Minneapolis, MN 55418

* * * * *

ED:

Enclosed is a note from an Australian newsletter (KAOS) pointing out a bug in OSI ROM Basic. I happened to run across both the bug and this article in the same week!

A CORRECTION TO ROM BASIC
By: Rodney Eisfelder

After many hours of hard thinking, the solution to another bug in ROM BASIC can now be revealed. The problem occurs in systems with more than 8K of RAM. When an INPUT statement is between $2000 and $20FF then the first character typed in response to the INPUT is ignored as well as the first non-space character. The problem is described in the 'Dear Paul' column in KAOS 3.6.

The solution is to change two bytes of the second BASIC ROM. The two bytes are $A969 and $A9CD (or in English, 43369 and 43469) which currently have the value $12. This is the address used to save the high byte of the BASIC program counter and is also immediately before the BASIC line input buffer.

The problem occurs when the high byte of the BASIC program counter is the same as an ASCII space i.e. $20. When BASIC starts processing an INPUT line, the buffer pointer points one byte before the start of the buffer. that is it points to $12. The get-current-character routine ($00C2) is called to detect end of line. For the first INPUT variable this is not meant to do anything because a special test is made for zero length INPUT lines. However, if the byte before the buffer is a space, then the pointer will be moved on and the first character skipped. BASIC even goes to the trouble of writing $2C (a comma) into $12 before overwriting it with the program counter.

The solution is therefore to change the two bytes mentioned above so that a harmless location is used to save the BASIC Program Counter. Any location not used elsewhere by BASIC is obviously 'safe' to use and I would suggest $D8 as a contender. People who suffer from this problem should note that my solution is not tested and is therefore as reliable, complete and correct as any untested program can be.

Earl D. Morris
Midland. MI 48640

* * * * *

ED:

You may be interested to note that the TAB(x) function on the C1 sends actual spaces to the ACIA. That means my Selectric <u>will</u> respond to TABs in PRINT statements. Sure saves a lot of SPC(x) calculations!

Bruce Showalter
Abilene, TX 79601

* * * * *

ED:

Our group purchase of the Grafix SEB-3 80 column video board/floppy controller has been a success (see PEEK July 1983). Because of the number of OSI users getting boards, the price was lowered from $59 to $35. There are a few more bare boards left at this price. Please contact me if you are interested.

Earl Morris
3200 Washington St.
Midland. MI 48640

BUG FIX FOR ALLOY/OSBU BACK-UP OF PLANNER PLUS
(older versions)

Problem:

If Planner needed a larger data file, it created a new file with the same name but a different password. Fine until you want to back-up. These utilities will ignore the new file as it assumes it is a duplicate (no password check is made).

Newer versions of Planner add a "%" or "&" to the file name, thus eliminating the problem.

Fix:

Program <u>OSBU</u>
```
1430 Poke RT,1:Poke 9467,141:
Poke 9468,42:GOTO 1340
```

* * * * *

AD$

32K C1PMF - Includes disk copy program and several programs - OSI 65 Operating System. Also has PICO DOS operating system. $500 or best offer. Will consider any offer. 801-544-9826. M. G. White, 1322 E. 1225 N., Layton, UT 84041.

* * * * *

Send for free catalog, Aurora Software, 37 South Mitchell, Arlington Heights, IL 60005. Phone (312) 259-3150.

* * * * *

Ohio Scientific C3D (6502, Z80A), 10M hard & 8" floppy drives, CP/M 2.2, Basic, 56K RAM. $1800 or best offer. 607-273-5011 or Nan Stalnaker. 217 West Ave., Ithaca. NY 14850.

* * * * *

FOR SALE: OSI C3C' running at 2MHz with 36MB hard disk. Three user system (152K RAM), 3 extra serial ports plus 1 parallel port. Three terminals also available (2 Televideo 920B's and one ADDS Viewpoint). Lots of software, OSU V1.43, OSD, CP/M, MBasic, FORTRAN, COBOL, WP-6502, OS-DMS, Brown/Collinson EDITV4, FIND and MONTIR. Software Consultants Fig-FORTH and REF. Complete set of documentation. Will help get started if needed. Will accept bids for all or part of system. Write or ask for Mike Fowler at Ctr for Behavioral Medicine, 155 W. Hospitality Lane, Suite 123, San Bernardino, CA 92408. (714) 889-0526.

* * * * *

OHIO SCIENTIFIC C8P-DF, $3000 or offer. 6502a cpu (2.5mhz), 48k static memory, 16 color video and Centronics printer boards. Two 8" ss sd floppies (IBM 3740 format). RS-232, printer, game and I/O ports. OS-65-D v3.2, v3.3, Microsoft BASIC, WP6502, MDMS and games. Documentation, Technical notes, diskettes, magazines. Separately, four years of PEEK(65), OSIO, MICRO 6502, Okidata 82A ($450), Novation D-CAT ($150). Contact Max Munger, 8639 Reseca Lane, Springfield, VA 22152, (703) 451-7627.

* * * * *

C2-OEM (two cases) with 48K RAM, dual 8" floppies, includes Centronics interface, RS-232C board, OS-65D, OS-65U, plus miscellaneous software. Almost brand new. No documentation. Must sell. $1500 (includes shipping) or offer. Rick Brown, 316 California #712, Reno, Nevada 89509, 702-322-9936.

* * * * *

PEEK (65)
The Unofficial OSI Users Journal

P.O. Box 347
Owings Mills, Md. 21117

DELIVER TO:

[illegible]

GOODIES for OSI Users!

PEEK (65)
The Unofficial OSI Users Journal

P.O. Box 347 • Owings Mills, Md. 21117 • (301) 363-3268

( )	**C1P Sams Photo-Facts Manual.** Complete schematics, scope waveforms and board photos. All you need to be a C1P or SII Wizard, just	$7.95	$ ______
( )	**C4P Sams Photo-Facts Manual.** Includes pinouts, photos, schematics for the 502, 505, 527, 540 and 542 boards. A bargain at	$15.00	$ ______
( )	**C2/C3 Sams Photo-Facts Manual.** The facts you need to repair the larger OSI computers. Fat with useful information, but just	$30.00	$ ______
( )	**OSI's Small Systems Journals.** The complete set, July 1977 through April 1978, bound and reproduced by PEEK (65). Full set only	$15.00	$ ______
( )	**Terminal Extensions Package** - lets you program like the mini-users do, with direct cursor positioning, mnemonics and a number formatting function much more powerful than a mere "print using." Requires 65U.	$50.00	$ ______
( )	**RESEQ** - BASIC program resequencer plus much more. Global changes, tables of bad references, **GOSUBs** & GOTOs, variables by line number, resequences parts of programs or entire programs, handles line 50000 trap. Best debug tool I've seen. MACHINE LANGUAGE - VERY FAST! Requires 65U. Manual & samples only, $5.00 Everything for	$50.00	$ ______
( )	**Sanders Machine Language Sort/Merge** for OS-65U. Complete disk sort and merge, documentation shows you how to call from any BASIC program on any disk and return it or any other BASIC program on any disk, floppy or hard. Most versatile disk sort yet. Will run under LEVEL I, II, or III. It should cost more but Sanders says, "...sell it for just..."	$89.00	$ ______
( )	**KYUTIL** - The ultimate OS-DMS keyfile utility package. This implementation of Sander's SORT/MERGE creates, loads and sorts multiple-field, conditionally loaded keyfiles. KYUTIL will load and sort a keyfile of over 15000 ZIP codes in under three hours. Never sort another Master File.	$100.00	$ ______
	BOOKS AND MANUALS (while quantities last)		
( )	**65V Primer.** Introduces machine language programming.	$4.95	$ ______
( )	**C4P Introductory Manual**	$5.95	$ ______
( )	**Basic Reference Manual** — (ROM, 65D and 65U)	$5.95	$ ______
( )	**C1P, C4P, C8P Users Manuals** — ($7.95 each, please specify)	$7.95	$ ______
( )	**How to program Microcomputers.** The C-3 Series	$7.95	$ ______
( )	**Professional Computers Set Up & Operations Manual** — C2-OEM/C2-D/C3-OEM/C3-D/C3-A/C3-B/C3-C/C3-C'	$8.95	$ ______

( ) Cash enclosed ( ) Master Charge ( ) VISA

Account No. ______________ Expiration Date ______________

Signature ______________

Name ______________

Street ______________

City ______________ State ______________ Zip ______________

TOTAL	$______
MD Residents add 5% Tax	$______
C.O.D. orders add $1.65	$______
Postage & Handling	$ 3.50
TOTAL DUE	$______

POSTAGE MAY VARY FOR OVERSEAS