PEEK (65)

The Unofficial OSI Users Journal

★★ $1.75 ★★

MAY 1982

Vol. 3, No. 5

P.O. Box 347
Owings Mills, Md. 21117
(301) 363-3267

INSIDE:

Column One

Last month, after we had put the entire issue to bed, exciting word trickled in that M/A-COM OSI was to launch a new upgraded line of computers. Voila, last month's cover, and no Column One.

The cover gave you a sample. This month M/A-COM OSI's ad on page 22 will whet your appetite still further.

The problem is that PEEK (65) goes to the printers a month before you read it. Our schedule was bent out of shape (hence the late arrival of the April issue) to coordinate with OSI's release of the new machines. Some marketing legalities delayed the release, but the April issue had already gone to print. Still nothing in hand and again we must go to print.

But alas, by the time you read this, all OSI dealers should be well informed and of course all the details to satisfy that patient appetite will appear next month.

With that out of the way we can get back to last month's column. Truer than ever.

Until recently, guys with Apples and TRS-80's and S-100 machines were in a different class from us OSI guys. Partly because we had better machines; but partly because they had "standard" machines, with configurations supported by alternate manufacturers. We didn't. If we wanted something OSI didn't make, we had to make it ourselves, or do without.

No more. Just have a look at the ads in PEEK (65) and the other computer magazines. The 48-line bus has arrived.

Of course, Aardvark and D&N have been around, and are still there to help us when we need a fix for a garbage collection bug or a slightly different RAM selection; but look at the profusion of hardware and software now available to us!

Micro-Interface offers a board with up to 56K of high-speed static RAM plus a parallel port, not to mention IEEE-488 interface boards for scientific types.

Modular Systems will sell you a mod to double your disk capacity without changing drives or controller.

Progressive Computing offers alternate character generators sound generator boards and ROMS to suit your fancy.

The Software Federation now offers the "proxy-80" board to change your OSI computer into a standard-format CP/M computer with up to 64K of RAM and double-density 8" floppies.

By the time you read this, you will be able to buy, through your regular dealer, a computer which will read all those 78 disks full of software available essentially free through the CP/M user's group. If your dealer doesn't know this, point him to D&N -- they will sell it to him wholesale, for less than a C3-OEM.

A few months ago, I wrote in this space that the appearance of hard disks and Lifeboat's CP/M was important, even if you own a C4P and never intend to use either one. These recent developments bear me out. Anything which makes OSI's 48-line bus more versatile and powerful makes us, the people who already own OSI computers, a more attractive market for them, the alternate suppliers of hardware and software.

Did I say software? I remember not so long ago when every OSI owner was a software developer. Had to be. Wasn't much of anything out there to buy!

Again, not so any more. Just look within these pages and recent back issues and you will find many more ads for OSI software. If you are into CP/M, it comes in 3 versions from OSI, Lifeboat and D&N.

I am no longer envious when I read the ads for S-100 products. If I can't get it, or something better, for my OSI, just wait till next month.

ae

OS 65U VER 1.42

Dickinson H. McGuire,
Technical Editor

As the wag said, "There's Good News and there's Bad News..." about this new release of 65U. The good news is that it has several features which make considering it worthwhile when one decides to write a new end-user application. Some of the enhancements which are useful (to a varying extent) to the programmer are

Common Variables
Extended Input/Print
Printer Map/Set
New Dev 5 Driver
Trapping of BASIC Errors
New FLAGs
Terminal Independence
New Manual

Some of these, such as common variables, trapping of BASIC errors, the new manual and the new Dev 5 driver are just great! The others leave something to be desired.

The new manual from M/A-COM OSI is not up to the standards used by IBM, neither is it written to the standards previously used by OSI. It is a clear, concise reference manual designed for experienced programmers. There is a very complete Table of Contents, a section about the enhancements in 1.42, it includes a brief discussion of each of the System Utilities (DIR, CREATE, etc.) and an extensive discussion of the Transient Utilities (EDITOR, RSEQ, COMMON and INP$), and there is a discussion of each of the BASIC commands as implemented by OSI and Microsoft. But the best section is the programmer's reference guide which gives all sorts of handy hints, PEEKs and POKEs and sample code for a lot of the neat ways to fake the operating system !

The transient utilities COMKIL and INP$ are very nice. Common Variables (COMKIL) allows the programmer to write programs of nearly infinite length carrying the variables from one to another without the operator's knowledge. This is not a true common because this scheme carries all variables from one program to the next rather than only specified ones. A KILL command is supplied to get rid of unwanted variables, and it may be used to kill specific variables, arrays, all simple variables or all array variables to prevent variable buildup. Of course KILL may be used at any time to kill unwanted variables and help prevent garbage collection.

INP$ is a utility to allow the operating system to perform certain checks on data supplied from the keyboard and not allow incorrect data to be entered. One may specify the type of input string (Alpha, Integer, Cash or Floating Point) and the length of the input string in bytes as though it were alpha-numeric data. 65U checks the data as it is entered and, if incorrect, echos a BELL to the screen. This allows data to be checked for accuracy as to type and length before the RETURN is hit. Another very nice feature is that the string may be pre-loaded before being presented to the operator and then may be edited as with the basic editor. One no longer has to re-enter the entire entry to change the 'i' to an 'e'. One may also suggest entries such as dates, invoice numbers, etc. which the operator may accept with a RETURN. This utility has a new form of the PRINT command. PRINT [10,"R"] QA$ is the functional equivalent of PRINT RIGHT$(SP$+QA$,10) without the attendant buildup of garbage strings.

Another very good feature is the ability to trap BASIC errors. FLAG 23 will cause a trap to line 50000 on the occurance of any error except the overflow error which is handled by FLAG 30. Sample code is provided to decode the error. Programs can now be written which absolutely prohibit the operator from entering the immediate mode. Code can also be written which will store error information in a disk file to avoid the problem "....I had an error last week, but all I remember is that it said something about error and line."In this connection there are several new features available for printers. There are two new FLAGs. FLAG 100 causes a conditional top-of-form and FLAG 101 causes an unconditional top-of-form. If your printer will respond to CHR$(12) then there is a utility which modifies the operating system to use this instead of a number of Line Feeds. Only Dev 5 has line counting software, but there is a utility to map physical device 5 to logical device 3,5,6 or 8. There is another utility to define physical device 3 or 8 to any ACIA anywhere in the machine.

The bad news is that in order to get all these good things one has to give up something. Frequently what one has to give up makes life a lot more difficult than it once was. For instance, COMKIL prevents use of the DEF, FN and EXP operators and INP$ prevents use of a whole raft of arithmetic operators. The loss of DEF and EXP are, I feel, major problems which the authors must address and correct if possible.

Terminal Independence is achieved by storing the parameters of the VDT is a data file which will then be accessed by the EDITOR and INP$. These utilities read the file and insert in the operating system the various codes to forward and backspace the cursor, clear the screen, erase to the end of the screen, erase to the end of the line, foreground follows, background follows and position the cursor. The programmer must add code (supplied) to each program which needs these functions. This code interrogates the operating system and assembles the parameters in basic variables. A much better way of doing things would be to modify the PRINT driver to accept something like the PRINT[R,C] from Projects, Inc. or the PRINT @ R,C from Software Consultants. There still is no PRINT/USING or anything like it.

You can get ver 1.42 in any of several ways. It will be

PEEK (65) is published monthly by DBMS, Inc.
Owings Mills, MD 21117.
Editor - Al Peabody
Technical Editor - Dickinson H. McGuire
Asst. 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

included in all new serial terminal computers shipped from the factory. If your computer was from dealer stock, he can get you a free update if he reported the computer as part of his inventory. There is an update price of $100 for CD36/74 or floppy only and $50 for CD7/23. If none of the above apply, you must ask your dealer to obtain a return authorization from the factory and be prepared to return your ORIGINAL OS 65U diskettes. Otherwise the price is $1,250.00

★

MOVING MACHINE CODE INTO DOS MEMORY SPACE

by: Gary E. Wolf
227 Grove Street
Clifton, NJ 07013

Before getting a disk for my 32K C1P, I had established a small collection of machine code tapes. One was a chess program that ran almost 12K and started at just above $0300.

With the help of a modification to load m/c programs at 1200 baud and the CEGMON ROM save feature, loading and saving these programs was relatively easy. The CEGMON ROM is very useful for m/c work, and its many other features make it well worth the price.

When I added a disk drive, the first job was converting my BASIC tape programs over to disk. A lengthy task, but straightforward, with the exception of a few changed POKE's here and there. Then I came to the problem that forms the basis of this article.

You have a m/c program on tape running from $0300 to $2CFF. You want to put it on disk for faster loading, but alas, the program resides in the same spot as either BASIC or Assembler/Ex. Mon. under 65D!

What to do?

First, I moved the program from $0300-2CFF to $5300-7CFF. When I ran the program from the higher block of memory, it worked. Hooray! But wait...

Let's assume there are JMP's and tables in the program (I knew there were). These would still jump to the old table or location in the lower (original) portion of memory. The code was still there now, but if I reloaded the program in high memory when the old code was gone, the program would probably crash when run.

I tried it. It did.

Next step. Disassemble the program and change all the jump locations by hand to make it stand alone at $5300-up. Wow! Just a short run of the disassembled code made it clear that there must be a better solution. There were so many jumps that changes would be as time consuming as a complete re-write. Then the simple solution hit me like a safeload of software.

Why not run the program in the original location? If you simply saved on disk the relocated program ($5300-7CFF), you could write a BASIC program to load it into high memory and then jump to a m/c routine to relocate it back to the original block. You now have stand alone machine code, and the DOS is no longer needed.

Here's the procedure. You must first create a file for your m/c program of 6 tracks and one for the loader program of 1 track. Then...

1) Enter your m/c program.
2) Relocate it using the CEGMON move feature or listing #1
3) Enter listing #2
4) Break and boot up 65D. Your code in high memory should still be there.
5) Save the m/c from $5000 to 7CFF as follows. (This assumes a m/c file on tracks 21 to 26.)

```
DISK!''SA 21,1=5000/8
DISK!''SA 22,1=5800/8
DISK!''SA 23,1=6000/8
DISK!''SA 24,1=6800/8
DISK!''SA 25,1=7000/8
DISK!''SA 26,1=7800/7
```

6) Enter listing #3 and save in your loader file.
7) Break and reboot. RUN the loader. The m/c program should load and RUN in seconds.

I chose $5300 as a temporary location because I found it convenient. Also, it left a few pages unused at the top. This prevents variable storage from garbaging code.

An obvious variation on this scheme would be to make BEXEC* the loader. This way booting up would LOAD and RUN the program, wiping out everything else. Add a disk copy prevent feature, and you have a lot of program security. But that would be the basis for another article.

Listing #1

Move code to high memory block

```
5000 A200   LDX #$00
5002 A02A   LDY #$2A
5004 BD0003 LDA $0300,X get
            code
5007 9D0053 STA $5300,X shift
            code
500A E8     INX
500B D0F7   BNE $5004 loop
            back
500D EE0650 INC $5006 inc.
            hi-byte
5010 EE0950 INC $5009   ''
5013 88     DEY
5014 D0EE   BNE $5004 not
            done, loop back
5016 4C00FE JMP $FE00 return
            to monitor
```

Listing #2

Move code to low memory block

```
5000 A200   LDX #$00
5002 A02A   LDY #$2A
5004 BD0053 LDA $5300,X get
            code
5007 9D0003 STA $0300,X shift
            code
500A E8     INX
500B D0F7   BNE $5004 loop
            back
500D EE0650 INC $5006 inc.
            hi-byte
5010 EE0950 INC $5009   ''
5013 88     DEY
5014 D0EE   BNE $5004 not
            done, loop back
5016 4C5004 JMP $0450 jump to
            start of chess
            program
```

Listing #3

```
 5 REM LOADER
10 POKE133,79:PRINT"*****
   LOADING *****":PRINT:PRINT
20 DISK!"CA 5000=23,1
30 DISK!"CA 5800=24,1
40 DISK!"CA 6000=25,1
50 DISK!"CA 6800=26,1
60 DISK!"CA 7000=27,1
70 DISK!"CA 7800=28,1
80 DISK!"GO 5000
```

★

OS-65U DISASSEMBLY AID

By: Carl Eidbo
1509 12th St. So.
Fargo, ND 58102

I am frustrated by the unavailablity of an OS-65U disassembly manual. There are many functions that could be improved and/or modified to better serve the user. So I have begun to disassemble specific areas of 65U myself. I have written a Disassembler in BASIC as well as the program listed below, to aid in my quest.

I have a feeling (with very little basis) that a large part of 65U/BASIC could be removed, different parts left in for different applications, and allow a substantially larger programming area. I realize this would cause a problem concerning compatibility between systems.

About the program:

After disassembling a few very small subroutines with my BASIC disassembler, it became apparent that the areas that called a subroutine would be just as important as the subroutine itself.

I first wrote a BASIC version of a program that would search, by means of many PEEKs, a specified area in memory, for a given two byte string (usually the beginning address of a subroutine). Depending on the area searched (usually Dec. 1000 - 25000), and on the search string, the run time varied, but was always at least two minutes (at 2MHZ). This may not seem like too much time, but if you need to search for many different addresses, it really adds up.

Partly for speed, because I work on the company computer mainly on coffee breaks and at lunch time, and partly for fun, I decided to write a machine code version, using the Assembler supplied with OS-65D.

A Brief Description:

1) The search string is input to the BASIC program.

2) The hexadecimal string (address) is converted to two decimal bytes, low and high, and is POKEed to locations $6000 and $6001.

3) The "last searched address" is initialized to Zero.

4) The machine code is called from BASIC.

In the Machine Code Subroutine:

1) The last searched address is moved to Page Zero.

2) The last searched address is incremented. (The program will not find a string starting at 0000.)

3) The first byte of the string is compared successively to memory locations, until it matches or runs out of memory.

4) If it is not found, the value 'Zero' is returned to BASIC.

5) If found, the second byte of the string is compared to the next memory location.

6) If this is not a match, the program branches to step 2.

7) If it is a match, the address is returned to BASIC, the low byte in the Y register, and the high byte in A. To return this value to BASIC as the result of the USR(X) function, an Indirect JMP through PAGE ZERO location $08 is executed.

Miraculously, this program ran the first time I tried it! The run time is now about seven seconds.

This program may be used by anyone reading it, PROVIDED a report of any successful disassemblies is submitted to PEEK (65)!

'Find a String' BASIC Only Version

```
10 DV=1
20 AD$="$2D96"
25 REM AD$ Must be in $----
   form.
30 IFDV=1THENPRINTCHR$(126);
   CHR$(28)
40 FORL1=2TO5:X$=MID$(AD$,L1,
   1):X(L1)=ASC(X$):X(L1)=X(L
   1)-48
50 IFX(L1)>9THENX(L1)=X(L1)-7
60 IFX(L1)>15THENSTOP
```

```
70 NEXTL1:FB=X(4)*16+X(5):
   SB=X(2)*16+X(3)
80 PRINT#DV,TAB(4);AD$;" =
   (";FB+SB*256;")"
90 PRINT#DV,"----------------
   ---------"
100 PRINT#DV," LOW   HIGH"
110 PRINT#DV,"----------------
   ---------":PRINT#DV
120 FOR L1=1000TO25000
130 IFPEEK(L1)<>SBTHEN160
140 IFPEEK(L1-1)<>FBTHEN160
150 PRINT#DV,L1-1,L1
160 NEXTL1
170 IFDV=5ANDPEEK(15908)<>PEEK
   (14457)THENPRINT#5:GOTO170
180 END
```

'Find a String'
BASIC/MACHINE Version

```
10 DV=1
20 PRINT:PRINT
21 INPUT"4-place Hex Address
   for Search";AD$
22 IFAD$="/"THEN170
23 AD$="$"+AD$
25 REM AD$ Must be in $----
   form.
30 IFDV=1THENPRINTCHR$(126);
   CHR$(28): REM Screen clr
   for Haz. 1420
40 FORL1=2TO5:X$=MID$(AD$,
   L1,1):X(L1)=ASC(X$):X(L1)=X
   (L1)-48
50 IFX(L1)>9THENX(L1)=X(L1)-7
60 IFX(L1)>15THENSTOP
70 NEXTL1:FB=X(4)*16+X(5):SB=X
   (2)*16+X(3)
75 POKE24576,FB :POKE24577,SB
76 POKE 24578,0 :POKE24579,0
80 PRINT#DV,TAB(4);AD$;"
   =(";FB+SB*256;")"
90 PRINT#DV,"----------------
   ---------"
100 PRINT#DV,"  LOW  HIGH"
110 PRINT#DV,"----------------
   ---------":PRINT#DV
120 POKE8778,4 :POKE8779,96
130 Y=USR(X)
140 IF Y=0 THEN 20
144 IFY=24576THEN130
145 IFY<0THENY=256^2+Y
150 PRINT#DV,Y,Y+1:GOTO130
170 IFDV=5ANDPEEK(15908)<>PEEK
   (14457)THENPRINT#5:GOTO170
180 END
```

'Find a String' ASSEMBLY Listing

```
 10                ;     This program will search for the two byte
 20                ;  string poked at $6000 (lo,hi). It will search
 30                ;  from $0000 to $FFFF.  If the string is not
 40                ;  found, the returned value will be zero.
 50                ;
 60                ;
 70 6000=                    FIND = $6000
 80 6002=                    LAST = $6002
 90 00B2=                   LASTZ = $B2
100 0008=                   REVAR = $08
110               ;
120 6004                    * = $6004
130               ;
140 6004 AC0260   START      LDY LAST
150 6007 84B2                STY LASTZ
160 6009 AC0360              LDY LAST+1
170 600C 84B3                STY LASTZ+1
180               ;
190 600E A000      NEXT      LDY #0
200 6010 E6B2                INC LASTZ
210 6012 D004                BNE CHECK
220 6014 E6B3                INC LASTZ+1
230 6016 F01D                BEQ OUTNO
240               ;
250 6018 B1B2     CHECK      LDA (LASTZ),Y
260 601A CD0060              CMP FIND
270 601D D0EF                BNE NEXT
280 601F A001                LDY #01
290 6021 B1B2                LDA (LASTZ),Y
300 6023 CD0160              CMP FIND+1
310 6026 D0E6                BNE NEXT
320               ;
330 6028 A4B2     OUTYES     LDY LASTZ
340 602A 8C0260              STY LAST
350 602D A5B3                LDA LASTZ+1
360 602F 8D0360              STA LAST+1
370 6032 4C3960              JMP REBAS
380               ;
390 6035 A000     OUTNO      LDY #00
400 6037 A900                LDA #00
410 6039 6C0800   REBAS      JMP (8)
420               ;
```

★ ★ ★

CORRECT DISPLAY OF LONG SCREEN LINES FOR ClP UNDER 65D 3.3

By: Eugene E. Baldwin
10650 North 75th Street
Longmont, CO 80501

This is a revised program to fit 65D 3.3 on older ClPs. Not original, just cleaned up. Was sent to me by Cleveland Consumer Computers and probably originated at some sharp Users Group. All mods worked for me except part of Disk 1. Probably cockpit error.

Mod necessary for older Superboard II or ClP disk based systems which are limited to a 24 by 24 display. OS65D version 3.3 assumes that a Series 2 system with the option of a 48 char. by 12 line display will be used. Non Series 2 systems will display the data with offset and wrap-around. If these suggested changes are implemented the display will appear correctly centered on the screen.

Tutorial Disk 1

Step 1 - Boot the system as explained in the 65D Tutorial and Reference Manual. Ignore (as best you can) the screen appearance. In response to the question: "Depress the number of your selection?" type: PASS (then hit the RETURN key.)

Step 2 - Type: 15 (then hit the RETURN key.)

Step 3 - Save the corrected program by typing: DISK! "PU BEXEC* (then hit the RETURN key.)

Step 4 - Boot the system to view the corrected version. You may at this time wish to write-protect the disk by placing a piece of opaque tape over the write-protect notch.

Tutorial Disk 2

Step 1 - Boot the system.

Step 2 - Type: DISK!"LO BEXEC* (then hit the RETURN key.)

Step 3 - Type: 22 (then hit the RETURN key.)

Step 4 - Boot the system to

OSI COMPATIBLE PRODUCTS

56K 2-MHz Ultra Low Power CMOS Static Memory Board **MEM-56K $850**

Partially Populated Boards (Specify address locations required) ... **MEM-48K $750**
MEM Board uses the new 2K-Byte Wide Static RAM chips which are 2716 EPROM compatible. Any 2K byte memory segment can be populated with RAM or EPROM (or left empty for use of Address Space by another board). Fully expandable to any memory size you will ever need. No special addressing requirements, just solder in extra sockets
MEM-32K $550
MEM-24K $450
MEM-16K $350
MEM- 8K $250
MEM- 4K $200
Extra 2K RAM Memory Chip **$24**
Optional Parallel Printer Port **-P $120**
Optional Calendar/Clock Software available in EPROM) **-T $ 25**
Both options (Disk software mods provided for use of 6522 VIA on printer). **-PT $125**

EXAMPLE USES:

C4P & C8P: Expansion to **4K RAM** of Basic workspace.
Parallel Printer Port — Reserve Serial Port for MODEM
Calendar/Clock Displaying on unused portion of screen.
Space for 5.75K of **Enhanced System Monitor EPROMS.**

All of this on 1 Board, using only one of your precious slots. Software for Enhanced System Monitor capabilities is continuously being developed and improved. As new EPROM Monitors are available, you may upgrade to them for any price differential plus a nominal $10 exchange fee. Another possibility is to fill any portion of the memory with Basic Programs in EPROM for **Power-on Instant Action.** This custom EPROM programming service is available at $25 per 2716 (Includes EPROM). Extra copies at $15 for each EPROM.

C4P-MF & C8P-DF:
Memory expansion to **48K.**
Add 6K **Memory** above BASIC for special software requirements.
Parallel Printer Interface and/or **Displaying Calendar/Clock.**
Add 1.75 K **Enhanced System Monitor ROM.**

C3: Up to **56K of Memory** Expansion — can be addressed for **Multiuser.** (Optionally, each user can have his own **Dedicated Printer Port).**

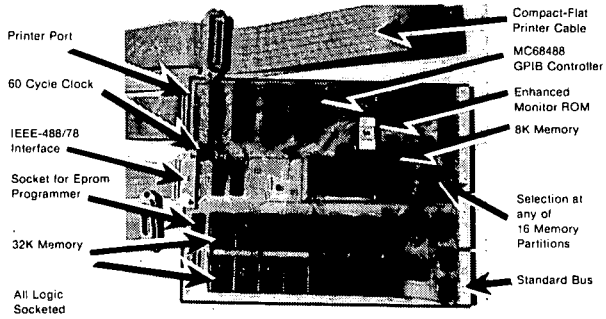
C1P,C4P & C8P FLOPPY DISC CONVERSIONS:
Memory/Floppy Board (Includes MI48P1 ROM) ..... **MEM F-16K $450**
C1P-600 Board Adapter & Cable ..... **A600/48 $ 50**
Additional Memory/Printer/Times ..... **(See MEM Board Prices)**
5¼" Drive/Case/Power Supply & Cable to MEMF Board **FD5 $399**

IEE-488 INTERFACES AND SOFTWARE:
The General Purpose Instrumentation Bus (GPIB Controller interface is available for all OSI Computers. Machine code GPIB Drivers are linked to Basic to provide easy control of IEEE-488 instruments which is equal to the best of Hewlett-Packard Controllers and far superior to most others. Basic Commands for Serial Poll, Parallel Poll, IFC Clear, full Local/Remote Control, Respond to SRQ Interrupts, Send Trigger, do Formatted Input/Output, Direct Memory Input/Output and MORE. Interface includes IEEE-488 Ribbon Cable/Connector.

GPIB Controller Interface for C2, C3, C4 and C8 Systems ..... **GPIB 4-488 $395**
GPIB Software for **OS-65D** (Add -8 for 8" or -5 for 5") ..... **GPIB 488-D $ 70**
GPIB Software for **OS-65U** ..... **GPIB 488-U $100**
GPIB Software on two 2716 EPROMS for **ROM** based systems ... **GPIB 488-R $100**
Add Optional **Parallel Printer Interface** to GPIB 4-488 ..... **-P $120**
Add Optional **Calendar/Clock** to GPIB 4-488 ..... **-T $ 25**
Add **2K RAM** to GPIB 4-488 (Specify location, $4000-$BFFF & $D000-$EFFF available) **-M $25**
GPIB Controller for C1P, Includes Software, Clock, All Features of ROMTERMS, & space for 6K EPROM ..... **GPIB 6-488R $395**
Add Optional Parallel Printer Interface to GPIB 6-488R ..... **-P $120**

EPROMS:
C1P ROM with 24/48 Col Display for Series II, Smart Terminal, Line Editing, Corrected Keyboard Screen Clear and More ..... **ROM-TERM II $59.95**
C1P ROM with 24 Col Display. Other ROM-TERM II Features, Disk Boot, and ROM/Disc Basic Interchange ..... **ROM-TERM $59.95**
C4P-MF/C8P-DF Disk warm start, changed IRQ Vector and just flip switch for Serial or Video System with Corrected Keyboard ..... **SYNKEY $39.95**
ENHANCED MONITOR ROOMS FOR USE ON GPIB 4-488 & MEM BOARDS:
Expanded Support for **C4P & C8P** Featuring Calendar/Clock, Line Edit, Smart Terminal, Memory Files, Parallel Printer Control, Corrected Keyboard. All Features of ROMTERMS, **Disk Support** with Warm Start and More **MI48P1 $59.95**
Expanded C1 **Monitor** with Calendar/Clock Software, Hard Disk Boot, Warm Start and Multi-User Control for C2 Systems **MIC2-1 $59.95**

IEEE-488 CONTROLLER INTERFACE



THE GPIB 4-448 INTERFACE BOARD CONVERTS ANY OSI COMPUTER INTO AN IEEE-488 INSTRUMENT BUS CONTROLLER!

BENEFITS — Provides a Sophisticated Instrumentation Controller at very low cost (often saving thousands of Dollars). The combination of IEEE-488 Instrumentation Controller and High Capacity Hard Disk file storage available on OSI Computer systems is available at a fraction of the cost required by the nearest competitor. The IEEE-488 Bus, also known as the GPIB,HP-IB or IEC-625 is the most popular International Standard for connecting instrumentation systems. This 16-line bus is designed to interconnect and control up to 15 instruments at a time. Currently, over 2000 different instruments are available to work on this bus. They include: Plotters, Digitizers, Printers, Graphic Displays, Recorders and a multitude of specialized Test/MeasurementControl Equipment.

EPROM-ABLE — Can be used with a C4-P to create a dedicated IEEE-488 controller.

C2-D MULTIPLE USER SYSTEMS

SAVE — 2 and 3 user Time Sharing Systems are available on the C2-D Winchester Disk Computer at a considerable cost savings from C3 Multiple User Systems. The 3 user C2-D System can be expanded to include a word processing printer, 4 other parallel printers and 3 serial printer interfaces.

COMPATABLE — The special C2-D Multi-User Executive Program is 100% compatable with OS-65U V1.4. The Multi-User Real Time Clock, Memory Partition Control and IRQ Interrupt Management are done on the Micro Interface Memory Board. Thus, the CPU board is not modified and remains in factory condition.

CONVERSIONS — The Up-Grade of your existing C2-D Computer to Multiple User Configuration is also available. Call for details.

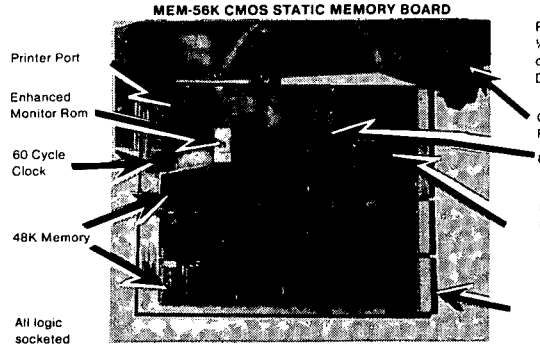
FLOPPY DISK UPGRADES FOR C1P, C4P & C8P

Our Memory/Floppy Board provides easy conversion of 502 and 600 CPU Computers to Floppy Disk Operation. The **MEMF** Board has a floppy disk interface which includes a data separator and the ability to automatically lift the disk drive heads — your floppy disk lifetime will be extended many times. You will retain the cassette interface for your existing software; which can easily be converted to Disk.

This **MEMF-16K** Board is populated with **16K RAM** (50K possible) and has features of the MEM CMOS Static Memory Board with an added floppy interface. The low power memory means extra power supply not required. **ROM Basic** is retained even when Board is populated for 48K Disc Basic. An optional Parallel Printer Port and Real Time Calendar/Clock is on board.

Complete Ready to Run conversion kits with 5¼" or 8" Disk Drives are available.

MEM-56K CMOS STATIC MEMORY BOARD



ULTRA-LOW POWER — By using CMOS Static RAM Memory, the total power consumption is about ½ Amp at 5 Volts when populated for 48K. In fact, most of power is used by the Address Line Buffers and the Data Transceivers.

MULTI-USER — Can be addressed for any of the 16 multi-user memory partitions. The low power and single memory board/partition simplify installation and provide a typical $1400 saving for a 3-user system.

MICRO-INTERFACE
3111 SO. VALLEY VIEW BLVD., SUITE I-101
LAS VEGAS, NEVADA 89102
Telephone: (702) 871-3263

Check with your local Dealer or Order Direct.
Phone orders accepted.
TERMS: Check/Money Order/Master Charge/VISA
Sent POSTPAID ON PREPAID ORDERS.
Foreign Orders: Prepaid only.
Add 5% for handling/shipping.

view the corrected version. You may at this time wish to write-protect the disk by placing a piece of opaque tape over the write-protect notch.

Tutorial Disk 3

Step 1 - Boot the system.

Step 2 - Type: 1 (then hit the RETURN key.)

Step 3 - While holding down the ESC key, type 1

Step 4 - Type: DISK!"LO MENU (then hit the RETURN key.)

Step 5 - Change line 215 to the following: 215 PRINT! (20) (then hit the RETURN key.)

Step 6 - Change line 230 to the following: 230 TB=0: FOR X=2 TO 9 (then hit the RETURN key.)

Step 7 - Save the program by typing: DISK!"PU MENU (then hit the RETURN key.)

Step 8 - Boot the system to view the corrected version. You may at this time wish to write-protect the disk by placing a piece of opaque tape over the write-protect notch.

Tutorial Disk 4

Step 1 - Boot the system.

Step 2 - Type: 1 (then hit the RETURN key.)

Step 3 - While holding down the ESC key, type: 1

Step 4 - Type: DISK! "LO MENU (then hit the RETURN key.)

Step 5 - Change line 215 to the following: 215 PRINT! (20) (then hit the RETURN key.)

Step 6 - Change line 230 to the following: 230 TB=0: FOR X=2 TO 4 (then hit the RETURN key.)

Step 7 - Save the program by typing: DISK! "PU MENU (then hit the RETURN key.)

Step 8 - Boot the system to view the corrected version. You may at this time wish to write-protect the disk by placing a piece of opaque paper over the write-protect notch.

Tutorial Disk 5

Step 1 - Boot the system.

Step 2 - Type: 9 (then hit the RETURN key.)

Step 3 - while holding down the ESC key, Type: 1

Step 4 - Change line 50000 to the following: 50000 PRINT! (20) :RETURN (Type the word RETURN as part of the line before hitting the RETURN key.)

Step 5 - Save the program by typing: DISK!"PU BEXEC* (then hit the RETURN key.)

Step 6 - Boot the system to view the corrected version. You may at this time wish to write-protect the disk by placing a piece of opaque tape over the write-protect notch.

CASSETTE CORNER

By: David A. Jones
8902 SW 17th Terrace
Miami, FL 33165

Three game reviews this month and then some information on the ROM Monitor (SYN600) which is/was used in the early SII and C1P models. First the reviews.

Universe - DMP Systems - Machine Language - $14.95. An instant hit in our house. Everyone liked it right off. Advertised as 'like Scramble and Cobra Copter', neither of which I've seen. This time you are the invader and your mission is to penetrate the defenses of the planet Arcton IV. SAM rockets are launched by the defenders and meteorites hinder your progress. Should you make it past these obstacles without running out of fuel you are rewarded with a chance to try to make it through the maze. It's all skill here and your success is directly commensurate with your ability. The longer you manage to keep your spacecraft flying, the more difficult it gets. The SAM's get smarter and the meteorites stronger. Being in machine language, the controls are smooth and predictable. I've yet to see an arcade type graphics game written in BASIC that performs well in this respect.

It has a unique pause control for when the telephone rings or your wife (husband) gets jealous (romantic). "What's that computer got that I don't have?" "Not now honey, I just got my first extra man!" All action stops until you're ready to resume. A nice touch. The only fault seems to be that every game starts from the same point over the terrain. A random location would be more interesting as you begin to memorize the locale. Not that it makes it any easier to play though. You need all the help you can get. A real challenge. I give this one a 9.

Interceptor - Aardvark - Machine Language - $15.95. One of the most expensive games I've seen for the C1P. Your job is to defend the city from the invaders approaching from the sky. As the aliens come in and crash into the city the buildings disappear. When all of the buildings are gone, the game is over. Bonus points are awarded as the game progresses and sometimes the city gets rebuilt extending the game. There is an automatic cannon on each side of the screen to assist your defense and you are the pilot of the interceptor. You can manuver the interceptor around the sky shooting at the aliens or you can remain on the launch pad and fire from there. You can even land on top of one of the buildings and defend that single building only. If you choose to do that then you will never lose and the game becomes very boring. A major flaw! Even without this flaw though, the game is not very exciting. I returned the tape to Aardvark with an explanation and received a different game in exchange. No rating.

Humanoid Defender - Pretzelland - BASIC - $9.95. The instructions seemed incomplete on this game. Possibly because Pretzelland continually revises their games and the instructions lag the updates. My copy was REV.16.

Little squares appear that aren't mentioned and are quite deadly if they get you but don't count for anything if you get them. ??? You must defend the colony from the invaders by shooting them before they land and kidnap the humanoids. If they do manage to kidnap one you can rescue him before he is mutated by shooting the invader and then catching the humanoid and returning him to the surface. Baiter ships and bonus points are included.

As with many BASIC graphics games, the controls are slow to respond and sometimes don't respond at all. This can be extremely frustrating when the action gets fast. I've seen this game in the commercial version and I guess it's a good replica in theory but I wouldn't say REV.16 is the final revision of this one. I'll give this a 5 hoping it gets revised.

ROM Monitor

The monitor occupies address locations $F800-$FFFF, roughly divided into 2 parts. The lower portion contains the disk support so I won't go into detail on that here as I still don't have a disk and am not very knowledgeable about the code. $FCA6 through $FFFF contain the monitor, system initialization, polled keyboard and BASIC I/O. There are also 43 unused bytes from $FCD5 to $FCFF.

Table 1 is a list of the different routines contained in the ROM and a brief description of the function of each. The labels used in the list were compiled from various sources over the years and are the OSI labels I believe. Since I don't know all of the labels I have omitted the ones I'm not too sure of. All are subroutines and thus can be called by your own programs except VM, ADDRES, IN, INNER and ENTRY. I try to be consistent and reference these labels when interfacing them to my own programs.

If you have access to an EPROM programmer, some of the code can be rewritten to make it more efficient while at the same time enhancing its performance. (See my article in the January 1981 issue of PEEK (65).) You should be very careful if you choose to do this and make sure you remain completely compatible with the original, lest you discover that you can no longer run software from other sources. Occasionally you'll see a reference to a program for sale that is not compatible with such and such a PROM monitor.

If anyone has dissected the monitor ROM supplied with the S11/C1P series II and found it different, I for one would be interested in knowing what was changed and if possible why.

MONITOR SUPPORT

ADDRESS	LABEL	FUNCTION
FCA6		Initializes ACIA, 8 bits, no parity, 2 stop bits.
FCB1		Sends 1 byte to the ACIA
FCBE		Checks for key down on the polled keyboard.
FCD5		43 unused locations
FD00		Services the polled keyboard and loops until a key is depressed. Returns with character in accumulator and also in $0213.
FE00	VM	Usually here from basic entry point. Stack, flags, vectors and ACIA already set. Stack reinitialized, monitor load flag reset, screen cleared, address mode initialized and address set to zero.
FE2A	ADDRES	Gets one address character via $FEE9, converts it to hex and rolls it into $00FE,00FF, the address register checks for /, G, or L and branches accordingly.
FE43	IN	Address mode entry point without reinitialization.
FE77	INNER	Gets character via getch. Converts character to hex via legal and rolls character into $00FC via rola. Displays contents of address and data registers via disply. Checks for '.' and branches if found.
FE80	OTHER	Inputs character from ACIA. Strips parity bit.
FE93	LEGAL	Converts an ASCII character to a hex nibble. Flags a non-hex character with negative sign. (bit 7=1)
FEAC	DISPLY	Splits 8 bit word into 2 nibbles, converts them to ASCII (via DISNYB) and displays the ASCII characters at $D0C6-D0CD on your screen.
FECA	DISNYB	Converts LSB's of byte to ASCII hex.
FEDA	ROLA	Rolls MSB's of accum into LSB's of $00FC,X Rolls MSB's of $00FC into LSB's of $00FD,X Rolls MSB's of $00FD into LSB's of accum. Used to assemble 2 ASCII characters into 1 8 bit word.
FEE9	GETCH	Checks load flag, then services keyboard via $FD00 or ACIA via other. BASIC has its own ACIA at $FFCB.
FEED	INPUT	Services keyboard via $FD00. Does not check load flag. Also refered to as 'INKEY' sometimes.
		BASIC SUPPORT
FEF0		Basic vectors (8 addresses)
FF00	ENTRY	Presets stack, flags, vectors and ACIA. Clears screen. Displays D/C/W/M?
FF69	OUTPUT	Displays character on CRT, processes CR and LF, checks save flag and sends character to ACIA via $FCB1 if set. Sends 10 nulls if CR.
FF8B		Sets load flag and clears save flag.
FF96		Sets save flag
FF9B		Checks keyboard for control C down and branches accordingly.
FFBA	INPUT	Checks load flag, then services ACIA locally or keyboard via $FD00
FFEB	BASIN	Jumps indirect to address stored in $0218 to input a character. Normally this address will be $FFBA.
FFEE	BASOUT	Jumps indirect to address stored in $021A to output a character. Normally this address will be $FF69.
FFF1	CNTRLC	Jumps indirect to address stored in $021C to check for a control C. ($FF9B)
FFF4	BLOAD	Jumps indirect to address stored in $021E to service the load command. ($FF8B)
FFF7	BSAVE	Jumps indirect to address stored in $0220 to service the save command. ($FF96)
FFFA		NMI vector ($0130)
FFFC		Reset vector ($FF00)
FFFE		IRQ vector ($01C0)

★

HEXDOS REVIEW

COLUMN 2

By: Kenneth B. Shacter
& Norman McMullen
113 Dixie Circle
Slidell, LA 70458

Here we are, back again to give you more tasty information on Stephen P. Hendrix' HEXDOS operating system for the C1P-MF. This month's column will start our comparison of HEXDOS to OS65D, version 3.1 (sorry guys, that's all we

have!). From comments published in PEEK (65), there appears to be little difference between version 3.1 and 3.2, so the comments should remain valid for current owners of OS65D. From what we understand, version 3.3 has a few more bells and whistles, and is supposedly better documented. We have no idea how 3.3 compares to HEXDOS, except we know that 3.3 uses even more of our precious RAM! Speaking of documentation, let's compare what you get with each of the operating systems.

DOCUMENTATION

OSI provides the owner of OS65D with reams of paper to help explain the use of the computer and its associated software. Documents covering the hardware, the BASIC language processor, graphics, assembler/extended monitor, and of course, the operating system itself. Since our manuals are older, our crucial manuals are stamped "PRELIMINARY", but enough of that! Since OSI has been kind enough to supply a good deal of information, (even if a lot of it is either put together poorly or references the 8 inch system), the documentation needed for HEXDOS is minimized. Our comments on the HEXDOS User's Manual were presented in the last column. To briefly summarize, the information contained within the HEXDOS manual is well organized and clearly written. The DOS has been designed with the user in mind, and most tasks can be accomplished simply.

The use of the three basic utilities supplied with HEXDOS (FORMAT, CREATE, and DELETE) is presented concisely in the manual. Further elaboration is presented during execution of the program. Although no examples are provided as in the OS65D manual for utility use, we did not have any problems in using the programs. A comparison of the supplied applications software and utilities will be covered in a later column.

As stated in the previous column, HEXDOS exploits most of BASIC-in-ROM, while OS65D uses a separate RAM-based BASIC language processor. HEXDOS relies on OSI's documentation for the user to familiarize himself with the ROM system. Variances in commands and special features provided by HEXDOS are covered in the HEXDOS manual. The OS65D manual also references the user to another OSI volume for information on BASIC, as well as the Assembler/Extended Monitor. A summary sheet of all system commands, error messages, supplied software, important memory locations, and data on disk formats is provided at the rear of the OS65D manual. No such summary is provided by HEXDOS.

FEATURES

HEXDOS presently supports programs executing under BASIC-in-ROM or machine language (assemble them yourself!). No Extended Monitor or other language processors are currently available, although a CEGMON-compatible version of HEXDOS should handle any EM concerns. A disassembler (written in BASIC) is provided, and an assembler/editor (also written in BASIC) is coming down the pike in the near future. Meanwhile, if you don't have a DOS at all and are looking for one with an Extended Monitor and/or Assembler, HEXDOS does not have it (caveat emptor!)

OS65D claims that it supports a wide array of peripheral devices, and up to four (count 'em folks, four!) floppy drives, even though the lowly C1P only supports two drives. HEXDOS supports a dual-drive C1P, and provisions are made to allow the computer to talk to a printer via a parallel port, or the standard serial interface. OS65D states several drivers have been stripped from the C1P version of the DOS, and an address or two has been juggled to adapt itself to the C1P environment. HEXDOS has not deleted anything nor moved any pointers, as the DOS was designed for the C1P, and not a machine like the C3. Multiple I/O can be handled by both DOS packages, but not without additional programming.

One other note before moving on; OS65D works in an Operating System kernel mode. If you are not in the kernel (A* prompter), special commands must preceed the DOS command. These special commands vary from BASIC to the Assembler/Extended Monitor. HEXDOS has no kernel mode. Everything enters from BASIC in the immediate mode, or can be executed from a BASIC program, without the special front-end commands. The average owner, we feel, would prefer the simpler approach used in HEXDOS.

continued

SYSTEM COMMANDS

We could probably spend many pages writing on the various commands, but the easiest way to give you a good idea about how the systems compare is in the following table:

OS65D		HEXDOS
ASM	*	N/A
BASIC	*	N/A
CALL	+	LOAD #
DIR		N/A
EM	*	N/A
EXAM		LOAD #
GO nnnn16		?USR(-5) nnnn10
HOME		N/A
INIT		N/A (see FORMAT)
IO		N/A
LOAD		LOAD
MEM		N/A
PUT		SAVE
RET	*	N/A
SAVE	+	SAVE #
SELECT		LOAD !
XQT fn		LOAD $fn:?USR(-7)

Again, we point out the fact that HEXDOS has no DOS kernel, so the commands are entered and recognized as is. OS65D, on the other hand requires DISK!"command" from the BASIC - immediate, or program mode.

HEXDOS reduces the essential number of commands to two, LOAD and SAVE. Further, ASM, BASIC, EM and RET (marked above with a *) do not apply, since HEXDOS has no other language processors available. HEXDOS addresses only full tracks, no sectoring is provided. Since HEXDOS doesn't allow storage by sectors, the OS65D command DIR has no meaning. The OS65D commands CALL and SAVE (marked above wih a +) can control data/program load/save down to 256 byte sectors. HEXDOS is limited to 2K byte tracks. So far, we haven't felt like we are wasting too much disk space due to inefficient storage methods, but we've barely started to create machine language programs to date.

We are still at a loss why HOME exists in OS65D, and apparently so is Mr. Hendrix, since his system doesn't include this feature. If anyone has any comments here, send them in to PEEK (65). HEXDOS considers diskette initialization a utility rather than carrying the routine as DOS overhead like OS65D. After all, how often does one initialize diskettes, and how much time does it require? Another nice feature of the HEXDOS disk initializer (called FORMAT) is that track zero is automatically copied to the new disk. No fuss! With OS65D one needs to dig through the manual to determine the correct procedure, and then execute the track zero copying routine. Big fuss!

The IO command of OS65D is supposed to give the user multiple output device control, but fails to do the job correctly. Further BASIC errors automatically reset the distributor flag (and that is very irritating!). One must also remember to provide an embedded blank between the IO command and its arguments. HEXDOS has no equivalent to IO, but then again the average C1P owner feels privileged to own multiple output devices, not to mention the software to drive them simultaneously! On the other hand, HEXDOS provides a large number of I/O devices/files to be accessed, therefore it was impractical for Mr. Hendrix to assign each device a single status bit within a one word distributor flag. He offers solutions to the multiple output problem for more advanced users. Error messages are directed to the device being accessed by the current operation, thus the message may not always be seen on the video screen. To see the original error message you must temporarily change the INPUT or PRINT statement to access device zero (keyboard or screen) so the message will be displayed normally, This editing task is simple with HEXDOS' line editing function.

One more note on HEXDOS I/O; device numbers are reserved depending on whether the data is inbound or outbound, with a few exceptions. Devices 0, 1, 2, and 3 take I/O from the keyboard/video screen, expansion/parallel printer port, and 6850 ACIA. Devices from 4 through 25 are reserved simply for input disk files (odd) or output disk files (even). OS65D assigns each device an I/O channel number, but does not further differentiate as to whether the data can only be input or output from that device. (Question, are 25 devices in HEXDOS overkill?)

MEM is an interesting command in OS65D. The only use we've seen for it is as a "PRINT AT" function as pointed out in OSI's Graphics manual. Its true purpose in OS65D appears to be as a subroutine called within the DOS during the indirect file procedure for merging files. Since we have been unable to get our keyboard to produce the proper special characters needed to use indirect files, MEM has no

other use than first mentioned.

Since we are discussing I/O, the BASIC commands INPUT, PRINT, and LIST work the same under HEXDOS as under OS65D. The only difference is in the logical unit definitions as just described, and that HEXDOS allows for more devices /files.

The remaining commands have equivalents in each of the operating systems. We point out that HEXDOS uses the BASIC verbs LOAD and SAVE to OPEN and CLOSE data files as well as for the normal uses. This makes looking up the required command easy, but the application of the command requires a little forethought, and better in-code documentation via REM statements. On the other hand, OS65D requires just as much forethought, and sometimes more. Still, OS65D is using standard syntax for open and close statements in BASIC.

Let's look at an example on the use of the two DOS packages... Say you have just entered a new program over the last twenty minutes and now wish to save it on disk as a named file before testing for proper execution. We shall call the program SAMPLEBASIC (note the name is more than six characters, valid only under HEXDOS). In HEXDOS you would enter:

```
SAVE SAMPLEBASIC
```

If the program already existed on the disk you are accessing, the DOS would respond with a "D ERROR", otherwise it would save the program under the requested name. If you had just finished making some modifications to the existing program and wished to resave it under the same name, you would enter the following to avoid the "D ERROR":

```
DISK!"PUT SAMPLE"
      or
EXIT
PUT SAMPLE
RET BA
```

If the program is not in the directory of the OS65D disk, you have a BIG problem. First, you forgot to reserve space for the program in the directory using the BASIC utility CREATE. Second, you can't run CREATE now, since your program you wish to save is in the BASIC workspace. Since you are a "professional", you will have an up-to-date directory listing for this diskette, or you have the sacred TEMP file as recommended in the OS65D manual consuming as many tracks as available workspace needs. However, if you are a common hacker or a lazy computer services department manager, you will not have either of the previously mentioned items at your disposal. So, how do you get out of this pickle? You enter:

```
DISK!"PUT tt"
```

Where tt is a two digit number representing the track on which you wish to begin storing your program. With luck (or foresight) you will not have destroyed your favorite ADVENTURE program or whatever. You <u>still</u> must run CREATE for tracks not yet reserved or occupied by your program. If you attempt to run CREATE for the file you stored on track tt you will wipe it out. So you would have to run DIR (the BASIC program not the OS65D command) to find out where everything is, CREATE to reserve a new file on disk, then enter:

```
DISK!"LOAD tt"
DISK!"PUT SAMPLE"
```

Now, if SAMPLE had already existed, OS65D would have summarily replaced it with the current contents of your workspace, even it if <u>was</u> your only copy of SARGON.

Note that CREATE under HEXDOS is used to reserve space for machine language and/or data files. Files for BASIC programs, as just shown, are automatically created. HEXDOS CREATE automatically uses the next available free track(s); you cannot, in fact, specify the starting track.

QUICK COMMENTS ON CONVERTING TO HEXDOS

We will cover system conversion in more detail in a future column, although there really isn't much to it. In going from OS65D to HEXDOS, simply load your program in the BASIC workspace, and enter:

```
NULL 8:DISK!"IO,03":LIST
```

Be sure to have your cassette recorder fired up ready to take the source. Now, you boot up HEXDOS, rewind the tape, and enter:

Business System Users

Get Mainframe Software features with
"Computer Business Software"
"CBS"

the **INSTALLABLE** Accounting Package
for Floppy Disk and Hard Disk Systems -
Single or Multi User

Featuring: Accounts Receivable with Sales Analysis
Inventory with Product Analysis
Order Entry with Invoicing
Accounts Payable with Checkwriting
General Ledger with Financial Statements
Payroll with Labor Distribution

- Thoroughly Documented
- Already Installed
- Fully Supported
- Ready Now

Call **1-800-843-9838** for your free documentation kit, price schedule and sales brochure.

MICROSOFTWARE INTERNATIONAL

3300 South Madelyn, Sioux Falls, South Dakota 57106
"Mainframe Software for your Micro"

LOAD

and start your tape recorder. The program will be visible on the video screen as it is entered into BASIC workspace under HEXDOS. A brief understanding of how the program handles files is necessary to know, and the OPEN/CLOSE syntax must be corrected to HEXDOS' LOAD/SAVE. Generally, it is as simple as that. If you are moving over from BASIC-in-ROM tape-based system, simply load your programs in from tape as you would if you were still using BASIC-in-ROM, and SAVE them on disk. Be careful not to type SAVE to get a HEXDOS program out to tape, else it will store itself in the file of the last program loaded or saved! To place programs on tape from HEXDOS, simply enter:

LIST#2

and when the computer prompts you with its "OK", you're out to tape. However, just as with OS65D, no listing of the source is presented. There is a quick way around this too, but we'll cover that later.

Meanwhile, our fingers are getting tired! We'll finish the DOS comparison in the next column.

★

SOME NOTES ON OSI'S ASSEMBLER
-- CASSETTE VERSION (C4P)

By: Yasuo Morishita
9-22,4-chome,
Minami-machi,Minami-ku
Hiroshima, JAPAN 734

While in the U.S.A., I tried to understand the OSI Assembler (ASMBL for short) but I could not complete the job due to lack of time. I did my disassembly using a hand-operated printer (pencil), identified some special codes and relocated the whole program to $4640-$5790 (except the checksum loader).

The following information is what I got and may help very anxious PEEK (65) readers to complete the job.

I would like to thank Mr. Morris for his suggestions about ASMBL.

WHAT DOES IT LOOKS LIKE?

If you have tried to disassemble the ASMBL, you may have noticed the messy print-out with lots of "?"s. Yes, it is a combination of 6502 machine language and simulated CPU (hereafter I will use SCPU for short) codes, which occupy more than half of the entire memory space. Furthermore, the guy who programmed ASMBL scattered SCPU codes all over the place so that it is really hard to make a neat disassembly list without a hand-printer.

When you type in ".1300G" to start the ASMBL, it jumps to $1160 and sets up 0-page registers ($00-$FF) and get into SCPU operation. The 0-page initial values are located at $12C8-12FF and the addresses are at $1290-12C7. The SCPU is located at $1179-11E9 and its op-codes are all over the program. However, the main program starts at $0600. The initial entry point is $0601. $0780-07CF is the jump table for individual SCPU supporting (sub)routines, which are written in 6502 machine language.

Table 3 gives you the location of those SCPU op-codes and data. This is not perfectly accurate yet, but it should be almost okay. Table 4 gives you the identified locations, which you may use for further modification of ASMBL. Table 5 is a partial listing and explanation of the SCPU op-codes.

RELOCATION OF ASSEMBLER

The OSI/UK User Group Newsletter June 1981 issue showed a machine language program written by Mr. T. Parsons. My approach is mostly based on his idea and program, however, I have added 11 extra locations to be corrected. (They are marked with * in Table 2). Because I don't have permission to give you his program, I cannot show you the entire program listing. It requires the OSI Extended monitor (XMON), which is relocated to higher RAM/ROM location to do most of the relocating job.

The procedure is:

1. Relocate the entire ASMBL to new location by using XMON's relocate function "R".

2. Then correct the codes back to original ones, because they should not be changed. The locations of these codes are listed in Table 1.

3. The addresses shown in Table 2 must be changed. They are not corrected by XMON, because they are SCPU's codes hidden in the program. (They are jump destination operands of SCPU's JMP code (#$04)).

ADDRESSES NOT TO BE CHANGED

TABLE 1

03DE	0C08
047A	0C32
0486	0C35
04CD	0C68
05D7	0CC7
05FA	0CD3
0632	0CD9
0687	0CE7
068F	0CEB
0696	0CF1
069C	0D26
06A9	0D2F
06B3	0E00
06BF	0E33
06C3	0E79
06E2	0E88
06E7	0E93
06EF	0EA3
0702	0EB8
0716	0EE6
071E	0F1A
0726	0F20
072C	0F26
073A	0F5A
073F	0F8A
0754	0F8E
076D	0FD2
0775	0FDA
07D4	1002
07F3	102A
080F	103E
0824	107E
083F	10A0
0846	10AC
0852	10AF
085F	10B2
086F	1143
08AF	1155
0931	1158
094F	12DC

ADDRESSES SHOULD BE CHANGED TO NEW LOCATIONS

TABLE 2

03DD	07C8	10BE
05DC	07CA	10CE
0618	07CC	111D
0661	07CE	112E
0664	07D8	113A
0667	0823	1145
066A	0848	1148
066D	084B	1151*
068E	084E	1154
06DA	08A9	115D
06FE	08C1	120E
0779	090A*	1239
0780	094E	1242
0782	0958	1257
0784	095B	12CD
0786	098B	1347
0788	0A12	1358
078A	0A21	
078C	0A9B*	0CC6*
078E	0A9E*	
0790	0B75	
0792	0B7C	
0794	0B94	
0796	0C2E	
0798	0C56	
079A	0C89	
079C	0D25	
079E	0D85	
07A0	0DD2	
07A2	0E28	
07A4	0E92	
07A6	0E95	
07A8	0EBA	
07AA	0EC3	
07AC	0EDC	
07AE	0EE1*	
07B0	0EEF*	
07B2	0EF8	
07B4	0EFB	
07B6	0F01*	
07B8	100A*	
07BA	1015*	
07BC	1034	
07BE	1079	
07C0	1098	
07C2	10A2*	
07C4	10AE	
07C6	10B1	

ADDRESSES FOR SCPU OP-CODE & DATA ETC.

TABLE 3

0267-026F
02C8-02CF
03D3-03DF
0479-0492
04C0-04E2
05D3-05E8
05F8-05FF
0600-0964
098C-09C0
09F6-09FC
0A0D-0A1F
0A2A-0A2B
0A3F-0A40
0A97-0A9F
0B72-0B92
0BAB-0BAF
0C03-0C0B
0C16-0C6C
0CAD-0CE0
0D11-0D30
0D37-0D3D
0D6C-0D8B
0DA2-0DA5
0DC9-0DCD
0DD1-0ECB
0E80-0ECB
0ED6-1050
106F-107D
108D-10CF
111C-115F
11EA-11FF
120D-120F
1230-1257
1290-12FF
1336-1349
1350
1357-1359

4. The last thing you have to do is adjust the 2 address bytes at $1189 and $118F to the new address. Locations $1189 (Lobyte) and $118F (Hibyte) originally point to $0780 where SCPU's jump table starts. Therefore, if you relocate ASMBL to a new address, you have to adjust it. (Note: Table 1 and 2 show only first (Lo) byte address, so that you have to change H. byte which follows immediately too. Address with * mark are my additions to UK group listing.

Currently I have my ASMBL at $4640-$5790, and XMON at $5800-$5FFF, which is relocated as K. Lourash's suggests (September 1981).There is one more thing you have to do before you type in ".xxxxG" to start the new ASMBL. Enter the monitor mode and check the contents of the new addresses, which are equivalent to $12FC-$1310, if their op-codes are the same as the old ones. If not, you have to modify them to the original ones. The problem is due to the contents of $12FC-12FF. When you do a relocation with XMON, these codes may modify the contents of $1300 and other location. $1300 should always have #$4C. For example, if you had data #$00 in these 4 locations, you do not have to do this adjustment at all. If you had #$00,#$00,#$00,#$20, respectively, you sure have to modify them, because the code #$20 at $12FF will be interpreted as "JSR" (3-bytes long) instruction so that the contents of $1300, 1301 will be modified. I think those who have had problems relocating by using UK Group program may have been trapped by this

TABLE 4 MEMORY MAP (PARTIAL)

```
ADRS   DESCRIPTION
0240-  SCPU CODE #$01 SUBROUTINE = COMPARE DATA WITH ($00,0)
0244   SCPU CODE #$02 SUBROUTINE = COMPARE DATA WITH ($00,6)
0270   SCPU CODE #$03 SUBROUTINE = MOVE FUNCTION
0284   SCPU CODE #$05 SUBROUTINE = RELATIVE JMP
029C   SCPU CODE #$04 SUBROUTINE = ABSOLUTE JMP
028C   SCPU CODE #$11 SUBROUTINE "RTS"
02D0   SCPU CODE #$13 SUBROUTINE = INC
02DC   SCPU CODE #$14 SUBROUTINE = DEC
032C   PART OF SCPU ROUTINE WITH TEXT POINTER UPDATING
033C   CR/LF PRINT OUT
033E   CHARACTER + LF PRINT OUT
0343   CHARACTER PRINT OUT
0356   LF PRINT OUT
0474   INPUT A CHARACTER JMP
0590   RENUMBER FUNCTION ENTRY
0600-077F  SCPU PROGRAM MAIN ROUTINE
0780-07CF  SCPU PROGRAM JMP TABLE
0BB0       CONVERT HEX DATA TO 2-DIGIT ASCII & PRINT OUT
1160   ASSEMBLER INTERNAL INITIAL ENTRY POINT
1179   SCPU FUNCTION ENTRY
11EA   SCPU DECODING DATA TABLE
1290-12C7  Ø-PAGE INIZ. ADDRESS
12C8-12FF  Ø-PAGE INIZ. SET VALUES
1300   MAIN ENTRY POINT
1302   INPUT HANDLING ROUTINE FOR OSI SYNMON.
1329   CONVERT HEX NIBBLE TO ASCII DIGIT & PRINT OUT
135A   FILL THE INPUT LINE BUFFER ROUTINE
1391-  OSI CHECK SUM LOADER
```

TABLE 5 SIMULATED CPU (SCPU) OP-CODE (PARTIAL)

```
CODE/OPERAND  FUNCTION                        @=(25)(24)=PROGRAM COUNTER
ØØ            =RECEIVE INPUT WITH PRINT OUT:  @=@+1
Ø1,L,H        =CMP L WITH ($ØØ,Ø) BEQ THEN    @=@+2+H --RELATIVE JUMP
              =                  BNE THEN     @=@+3
Ø2,L,H        =CMP L WITH ($ØØ,6) BEQ THEN    @=@+2+H --RELATIVE JUMP
                                  BNE THEN    @=@+3
Ø3,L,H        =MOVE $L TO $H                  @=@+3   --Ø-PAGE MOVE
Ø4,L,H        =JMP $HL                        @=$HL   --ABSOLUTE JUMP
Ø5,L          =JMP $L RELATIVELY              @=@+L   --RELATIVE JUMP
Ø7,****,ØØ    =PRINT STRING "****"            @=@+2+STRING LENGTH
Ø7,****,ØD    =PRINT STRING "****",CR/LF      @=@+2+STRING LENGTH
11            =RTS (RETURN)                   @=@*+$2A--@*=JSR ORIGIN
                                                      2A=OFFSET
13,L          =INC $L                         @=@+2
14,L          =DEC $L                         @=@+2
1F            =NOP                            @=@+1
28-7F         =EXECUTE THROUGH "BRK" AT $1186
8Ø-A7         =JSR ($0780+#$FFAND(2*OP-CODE) -----RELATIVE JSR
               THE ORIGIN ADDRESS IS SAVED IN STACK IN ORDER OF
               $2A (LENGTH OFFSET)
               $25 (PROGRAM COUNTER HIGH BYTE)
               $24 (PROGRAM COUNTER LOW  BYTE)
               THESE DATA WILL BE PULLED OUT WHEN "RTS" IS EXECUTED
12,L           RTS
```

trick as well as some missing adjustment addresses.

HINTS FOR MODIFICATION

1. INPUT/OUTPUT modification - If you modify INPUT/OUTPUT destinations ($1311 & $1329 respectively), to your editing routine, you can add your own editing features.

2. SOURCE STORAGE START ADDRESS - In the original program the source code storage area starts at $1391. If you have relocated the assembler, you may want to change the source code storage area, such as $0301, $2000 etc. To do this, you have to modify 4 locations to your source code storage area.

$12C9 (Low byte), $12CA (High byte), $12FE (Low byte), $12FF (High byte)

3. SCPU CODE DISASSEMBLING ATTEMPT - The main program of SCPU starts at $0600, and I have tried to disassemble it. However, the very initial entry is made at $0601. Table 6 gives you part of SCPU program disassembly.

To my regret, I have not done much about the OSI ASMBL so that my information may contain errors. I hope you smart PEEK (65) readers may give us complete and accurate information in the future.

★

NOTES ON OS-65D V3.3

By: Richard L. Trethewey
5405 Cumberland Road
Minneapolis, MN 55410

A lot has been written lately about OS-65D V3.3. Unfortunately, much of what has been written has not been truly correct. There are no "bugs" in 3.3 and all of the new features that you have been told about run fine. In addition, 3.3 is fully upward and downward compatable with 3.2. There were some early releases of 3.3 that did have one problem with the video to printer dump routine. This was due to the fact that the code for this routine was directly transferred from MDMS Planner-Plus and needed some pointers reset. This problem has been fixed at the factory. Other than that, 3.3 runs without a hitch.

If one were to criticize 3.3 it would be in the area of the speed of output to the video. To implement the Hazeltine

TABLE 6, SCPU PROGRAM DISASSEMBLY (PARTIAL)

```
ADRS DATA     DISASSEMBLY          REMARKS

0600 00       INPUT
0601 96       JSR ($07AC)          ;=JSR $0626
0602 61       BRK
0603 14 1C    DEC $1C
0605 02 41    CMP #'A W/($00,6)    ;ASSEMBLE COMMAND?
          59  BEQ $0660
0608 02 49    CMP #'I W/($00,6)    ;
          69  BEQ $0673
060B 02 50    CMP #'P W/($00,6)    ;PRINT COMMAND?
          59  BEQ $0666
060E 02 44    CMP #'D W/($00,6)    ;DELETE COMMAND?
          59  BEQ $0669
0611 02 52    CMP #'R W/($00,6)    ;RENUMBER COMMAND?
          59  BEQ $066C
0614 02 45    CMP #'E W/($00,6)    ;EXIT COMMAND?
          59  BEQ $066F
0617 04 37 13 JMP $1337            ;INTERPRET OTHER CODE
061A 44 20 45                      ;"D ERR(CR)" MESSAGE
     52 52 0D
0620 07 2E 00 PRINT "."
0623 05 DC    BRANCH $0600         ;ALWAYS BRANCH
0625 1F       NOP
0626 03 00 04 MOVE $00→$04
0629 14 00    DEC $00
062B 13 00    INC $00
062D 01 20    CMP ($00,0) W/#$20   ;SPACE ?
          FC  BEQ $0629            ;IGNORE SPACE
0630 03 2C 08 MOVE $2C→$08
0633 03 00 06 MOVE $00→$06
0636 01 80 11 CMP ($00,0) W/#$41   ;CHECK FOR ALPHABET
              BMI $0639            ;INPUT IS NOT ALPHABET
              CMP ($00,0) W/#$5B
              BMI $0649            ;INPUT IS ALPHABET
0639 01 81 16 CMP ($00,0) W/#$30   ;CHECK FOR NUMERAL
              BMI $063C            ;INPUT IS NOT NUMERAL
              CMP ($00,0) W/#$3A
              BMI $0651            ;INPUT IS NUMERAL
063C 01 0D 13 CMP ($00,0) W/#$0D   ;"RETURN" ?
              BEQ $0651
063F 03 00 06 MOVE $00→$06
0642 03 F0 08 MOVE $F0→$08
0645 13 00    INC $00
0647 12 02    RTS
0649 13 00    INC $00
064B 13 08    INC $08
064D 01 80 FA CMP ($00,0) W/#$41
              BMI $0650
              CMP ($00,0) W/#$5B
              BMI $0649
0650 11       RTS
0651 13 00    INC $00
0653 13 08    INC $08
0655 01 81 FA CMP ($00,0) W/#$30
              BMI $0658
              CMP ($00,0) W/#$3A
              BMI $0651
0658 12 01    RTS
```

★ ★ ★

```
10
20
30
40
50 000A=        ; OS-65D V3.3 TRACK 1 SECTOR 3
60 000D=        ; SCREEN DUMP ROUTINE (CORRECTED)
70 0020=        ;
80 2343=        ;
90 235E=        LF     =$000A
100 2360=       CR     =$000D
110 2363=       SP     =$0020
120 25B3=       OUTCH  =$2343
130 25B6=       XSAVE  =$235E
140 25BF=       YSAVE  =$2360
150 2761=       DATA   =$2363
160 2D73=       CRTO10=$25B3
                VIDFLG=$25B6
                CRTO11=$25BF
                UNLOAD=$2761
                STROUT=$2D73
```

emulator, some 23 bytes on page zero were needed. These bytes get swapped in and out every time a character is printed to the screen and every time the keyboard on video systems is polled. In addition, every time a character is printed to the screen, the software has to compute a lot of addresses, compute the X,Y position of the cursor, and watch for special commands. All this takes time. When you see a review that says 3.3 is slow, you can bet it's a video system owner that is doing the review. The PRINT command has also been patched quite a bit, both in BASIC and the DOS which doesn't help since the adaptation of BASIC for disk was already a patch. So now you have a patch to a patch. The upper/lower case enhancements don't take enough time to execute to really consider. If you are careful with your output, you may actually notice an increase in speed with 3.3. OSI implemented the ROR instruction in BASIC which really speeds up the math package. This was done to make room for the patches for 3.3 but who cares why it was done, it was sorely needed.

There are some things that video system owners can do to speed things in the keyboard poll. This won't make any difference in program execution, but entering and editing your inputs will get a little zip. The new keyboard poll no longer needs the 4 bytes that are used by the monitor ROM poll to debounce the keyboard. Unfortunately, the DOS routine does the swapping of those 4 bytes anyway. In addition, the DOS routine actually does a JSR to a JMP. Obviously, as the software was being developed, rather than having to change the DOS everytime they made a change, OSI sent the routine to a fixed point that in turn was automatically changed when the code was reassembled. This is fine for development, but a little sloppy when left in the final product. To fix these, change locations $252B to $4C, $2532 to $90, $2533 to $35, and $2539 to $60. There is also a short delay loop at the end of the keyboard poll that can be sped up. Changing $363C will do this. The original setting is $10. I find a value of $8 is nice. This value has to be 1 or more. Disabling the cursor flashing may be desirable if you make this change since it gets a little distracting at this speed.

```
170 32FC=            PRSWAP=$32FC
180 3305=            SWBACK=$3305
190 331E=            FLON  =$331E
200 3321=            FLOFF =$3321
210                  ;
220 3180             *=$3180
230                  ;
240 3180 8CAA31  P0  STY T2
250 3183 68          PLA
260 3184 38          SEC
270 3185 EDAA31      SBC T2
280 3188 8DA931      STA T1
290 318B 8DAB31      STA T3
300 318E CEC025      DEC CRTO11+1
310 3191 100E        BPL P1
320 3193 EEC025  P18 INC CRTO11+1
330 3196 C050        CPY #$50        SCREEN DUMP ?
340 3198 F012        BEQ P3          YES! DO IT!
350 319A C043        CPY #$43        SET FORM LENGTH ?
360 319C D008        BNE P2          NO, SKIP THIS
370 319E EEC025      INC CRTO11+1
380 31A1 A900    P1  LDA #$00        CLEAR DATA
390 31A3 8D6323      STA DATA
400 31A6 4CB325  P2  JMP CRTO10      AND GO BACK
410                  ;
420 31A9 41      T1  .BYTE $41
430 31AA 01      T2  .BYTE $01
440 31AB 00      T3  .BYTE $00
450                  ;
460                  ; VIDEO TO MX-80 DUMP ROUTINE
470                  ;
480 31AC EEB625  P3  INC VIDFLG      PREVENT OUTPUT TO VIDEO
490 31AF AD5F23      LDA XSAVE+1     SAVE REGISTERS
500 31B2 48          PHA
510 31B3 AD6123      LDA YSAVE+1
520 31B6 48          PHA
530 31B7 A900        LDA #$00        SEND A NULL
540 31B9 204323      JSR OUTCH
550 31BC 20CD31      JSR DUMP        DO THE DUMP
560 31BF 68          PLA             RESTORE REGISTERS
570 31C0 8D6123      STA YSAVE+1
580 31C3 68          PLA
590 31C4 8D5F23      STA XSAVE+1
600 31C7 CEB625      DEC VIDFLG      CLEAR VIDEO OUTPUT FLAG
610 31CA 4CA131      JMP P1          AND GO BACK
620                  ;
630 31CD 20FC32  DUMP JSR PRSWAP     SWAP IN PRTABLE TO PG. 0
640 31D0 202133      JSR FLOFF       FLASH CURSOR OFF
650 31D3 203332      JSR LINE3       PRINT 3 LINES
660 31D6 A9AF        LDA #$AF        LOAD ACC WITH ROW CHAR.
670 31D8 A2DF        LDX #$DF        LOAD X WITH CORNER CHAR.
680 31DA 204C32      JSR BORDER      DO BORDER LINE
690 31DD 8406        STY $06         INIZ LINE COUNT (Y=$FF)
700 31DF 206832  P17 JSR PRBLK       PRINT BORDER CHARACTER
710 31E2 E606        INC $06         BUMP LINE COUNT
720 31E4 A606        LDX $06         LOAD FOR ADDRESS COMPUTING
730 31E6 203033      JSR $3330       COMPUTE LINE ADDRESS
740 31E9 A000        LDY #$00        INIZ TO START OF LINE
750 31EB B1E2    P16 LDA ($E2),Y     LOOK AT SCREEN CHARACTER
760 31ED A207        LDX #$07        INIZ POINTER TO LIST
770 31EF DD3C32  P5  CMP BLOX,X      IS IT THIS ONE ?
780 31F2 F00E        BEQ P15         YES! CHECK BKRND COLOR
790 31F4 CA          DEX             NO, DECREMENT POINTER
800 31F5 10F8        BPL P5          LOOP TO END OF LIST
810 31F7 C97F        CMP #$7F        IS IT A GRAPHICS CHAR ?
820 31F9 B004        BCS P6          YES! PRINT BLANK INSTEAD
830 31FB C920        CMP #SP         IS IT A <CTRL> CHAR ?
840 31FD B013        BCS P8          NO, PRINT IT AS IT IS
850 31FF A920    P6  LDA #SP         YES, LOAD A BLANK
860 3201 E8          INX             BUMP COUNTER
870 3202 B1E0    P15 LDA ($E0),Y     LOOK AT BKRND COLOR
880 3204 2901        AND #$01        CHECK BIT 0
890 3206 DD4432      CMP CLRS,X      CHECK IT AGAINST LIST
900 3209 F004        BEQ P7          MATCH ? THEN SKIP INVERT
910 320B 8A          TXA             YES!
920 320C 490F        EOR #$0F        INVERT LOW 4 BITS
930 320E AA          TAX
940 320F 8A      P7  TXA
950 3210 09A0        ORA #$A0        MAKE IT AN EPSON CODE
960 3212 204323  P8  JSR OUTCH       PRINT IT
970 3215 C4F5        CPY $F5         AT END OF LINE YET?
```

```
 980 3217 C8              INY             BUMP POINTER
 990 3218 90D1            BCC P16         NOPE, NOT YET
1000 321A 206832          JSR PRBLK       YES, PRINT BORDER CHARACT-
1010 321D 206332          JSR PRCR        DO A <CR>                ER
1020 3220 A506            LDA $06         LOAD LINE COUNTER
1030 3222 C5F4            CMP $F4         END OF WINDOW YET ?
1040 3224 90B9            BCC P17         NOPE, LOOP AGAIN
1050 3226 A9AF            LDA ##$AF       LOAD ACC WITH ROW CHAR.
1060 3228 A2AC            LDX ##$AC       LOAD X WITH CORNER CHAR.
1070 322A 204C32          JSR BORDER      PRINT BORDER LINE
1080 322D 201E33          JSR FLON        RESTORE CURSOR
1090 3230 200533          JSR SWBACK      RESTORE Z-PAGE, FALL THRU
1100 3233 20732D  LINE3   JSR STROUT      PRINT 3 EMPTY LINES
1110 3236 0D              .BYTE CR,LF,LF,LF,$00
1110 3237 0A
1110 3238 0A
1110 3239 0A
1110 323A 00
1120 323B 60              RTS
1130                      ;
1140                      ; TABLE OF BLOCK GRAPHICS CHARACTERS THE
1150                      ; SCREEN DUMP REPLACES FOR THE MX-80
1160                      ;
1170 323C 20       BLOX   .BYTE $20,$A8,$A6,$9A,$A7,$9D,$AA,$A5
1170 323D A8
1170 323E A6
1170 323F 9A
1170 3240 A7
1170 3241 9D
1170 3242 AA
1170 3243 A5
1180                      ;
1190 3244 00       CLRS   .BYTE $00,$00,$00,$01,$00,$01,$00,$01
1190 3245 00
1190 3246 00
1190 3247 01
1190 3248 00
1190 3249 01
1190 324A 00
1190 324B 01
1200                      ;
1210 324C 48      BORDER  PHA             SAVE ROW CHARACTER
1220 324D 206332          JSR PRCR        DO A <CR>
1230 3250 8A              TXA             FETCH CORNER CHARACTER
1240 3251 204323          JSR OUTCH       PRINT IT
1250 3254 68              PLA             RETRIEVE ROW CHARACTER
1260 3255 A4F5            LDY $F5         LOAD Y WITH WIDTH
1270 3257 0980    P11     ORA ##$80       SET BIT 7 FOR EPSON
1280 3259 204323          JSR OUTCH       PRINT IT
1290 325C 88              DEY             DECREMENT COUNTER
1300 325D 10F8            BPL P11         LOOP 'TIL DONE
1310 325F 8A              TXA             RETRIEVE CORNER CHARACTER
1320 3260 204323          JSR OUTCH       PRINT IT AND FALL THRU
1330 3263 A90D    PRCR    LDA #CR         LOAD A <CR>
1340 3265 4C4323          JMP OUTCH       AND PRINT IT
1350 3268 A9DC    PRBLK   LDA ##$DC       LOAD EDGE CHARACTER
1360 326A 4C4323          JMP OUTCH       AND PRINT IT
1370                      ;
1380 326D 43              .BYTE $43,$23,$20,$3B,$0D,$DA
1380 326E 23
1380 326F 20
1380 3270 3B
1380 3271 0D
1380 3272 DA
1390 3273 20              .BYTE $20
1400                      ;
1410                      ; TRACK 8 SECTOR 5 GETS OVERLAYED HERE
1420                      ;
1430 3274 20              .BYTE $20,$CB,$00
1430 3275 CB
1430 3276 00
1440                      ;
1450 3277 A2F7            LDX ##$F7
1460 3279 D005            BNE P14
1470 327B 206127          JSR UNLOAD
1480 327E A209            LDX ##$09
1490 3280=                P14=*
```

★ ★ ★

digital technology

BUS-II LEVEL I BOOKKEEPING & ACCOUNTING SYSTEM

The BUS-II turn-key multi-client accounting package is the leading OSI business software package. BUS-II Version 32 includes five principle modules:

	Inst. Price	List Price
GENERAL LEDGER	$1200	$599
ACCOUNTS RECEIVABLE (a)	1000	599
ACCOUNTS PAYABLE (a)	1000	599
ORDER ENTRY W/ INVENTORY (a) (b)	1000	599
PAYROLL (no extra charge for optional versions)	1200	799

01 - STANDARD MULTI-STATE OPERATION
02 - CPA FIRMS & SERVICE BUREAUS
03 - RESTAURANTS
04 - COMMISSION SALES
05 - CONTRACTOR'S JOB-COST ACCOUNTING

The Accounts Receivable, Accounts Payable, and Order Entry W/ Inventory are completely interactive with the BUS-II General Ledger. Two optional specialized packages (completely interactive) are also available.

CPA EXTENSIONS (see below)
POINT-OF-SALE TERMINAL W/ INVENTORY (see below)

The BUS-II CPA EXTENSIONS Package provides special features for accountants and bookkeepers. The POS-1 Point-of-Sale Terminal package enables the operator to use the computer system's video terminal as an on-line "electronic cash register."

Note: BUS-II operates on floppy-disk or hard disk-based systems running the OS-65U operating system (single-or multi-user). Multi-client use can accommodate any number of client companies on floppy disk systems or hard disk system with H/D/E (required for hard disk use). BUS-II LEVEL I files are limited in size for floppy disk back-up; floppy disk operation continues in case of hard disk failure.

BUS-II "SOFTWARE EXCHANGE" SPECIAL

Users of other business software packages who wish to upgrade to Digital Technology's full-function BUS-II BOOKKEEPING & ACCOUNTING SYSTEM can, in many cases, "trade in" their old, unuseable, or unsupported software for full rebate of the original purchase price (up to $750.00). This "SOFTWARE EXCHANGE" offer includes virtually all of the previously-available OSI business software packages. Contact Digital Technology or your dealer for more information.

BUS-II LEVEL II (EXPANSION TO BUS-II LEVEL I)

BUS-II LEVEL II is designed for much larger businesses. Expanded file size and special operations allow virtually unlimited numbers of accounts and transactions. BUS-II LEVEL II requires BUS-II LEVEL I. Minimal back-up is data cassette (tape) or floppies--although multiple Winchester disk operation is recommended (provides ability to continue computerized bookkeeping functions in case of hard disk failure.) H/D/E Hard Disk Executive is required.

	Inst. Price	List Price
GENERAL LEDGER (c) (d)	$	$399
ACCOUNTS RECEIVABLE (c) (d)	600	399
ACCOUNTS PAYABLE (c) (d)	600	399
ORDER ENTRY W/ INVENTORY (c) (d)	600	399

CPA EXTENSIONS PACKAGE

CPA EXTENSIONS is designed for public accounting firms. A number of special operations are provided: "bankers" Balance Sheet and Profit and Loss statement with summarization and consolidation options, Statement of Changes in Financial Position, Statement of Changes in Components of Working Capital, Cash Flow Analysis, Departmentalized Sales Analysis, Asset Depreciation Schedule (compatible with TAXMAN-1040), and Loan Amortization Schedule. In addition, a pre-processed or "after-the fact" payroll system is provided.

CPA EXTENSIONS is interactive with BUS-II 32 BOOKKEEPING & ACCOUNTING SYSTEM

CPA EXTENSIONS (a) Inst. Price $2400 List Price $1500

POINT-OF-SALE TERMINAL

POS-1 is an on-line multi-store point-of-sale terminal program with integrated inventory designed for cash register emulation. POS-1 controls cash drawer and ticket printer (or system printer). Automates taxable or nontaxable sales, cash transactions, and credit sales (with verification operations). POS-1 also allows the use of industry-standard bar code readers with the point-of-sale terminal system through a "Siamese port-- on the C2 or C3 CPU card. (Extra serial port NOT needed except in multi-user operation.) Configured for industry-standard RS232 bar code "wand" (INTERMEC) or "window" (SPECTRA-PHYSICS).

POS-1 is interactive with the BUS-II V 3.1 BOOKKEEPING & ACCOUNTING SYSTEM.

POS-1 POINT-OF-SALE TERMINAL (a)(b)
Inst. Price $1600 List Price $1199

TAXMAN-1040 PERSONAL INCOME TAX PREPARATION

TAXMAN-1040 is designed for tax practitioners and public accountants. TAXMAN-1040 is the leading tax package for OSI micorcomputers--the package has been installed on OSI, Hewlett-Packard, DEC and IBM systems. Designed and supported by CPA tax experts. This package automatically prepares FORM 1040 and 32 schedules. Support includes annual forms, tax tables, and computational revisions in accordance with Federal Tax Law changes.

TAXMAN-1040 Inst. Price $3600 List Price $2399

H/D/E HARD DISK EXECUTIVE

Digital Technology's implementation of H/D/E is the answer to AMCAP's HDM. Digital Technology's H/D/E provides user functions not found on HDM of similar products: ability to copy from any user "system" to another; automatic recovery in case of "back-up to floppy" or "restore from floppy" utility failures, allowing the user 3 options: (1) ignore error, (2) abort to menu, (3) try again; use of both "A" and "B" floppy drives to back-up hard disck files; and automatic back-up diskette initialization. H/D/E operates on any OSI Winchester disk system from 7 - 80 megabytes. Re-use of hard disk space is provided. Superior to AMCAP's hard disk manager in every respect (and Digital Technology software does not self-destruct).

NOTE: H/D/E is required when installing any Digital Technology business applications packages on OSI hard disk systems.

H/D/E HARD DISK EXECUTIVE List Price $399

OS-DMX DATABASE MANAGEMENT SYSTEM

Command-oriented OS-DMS compatible database management system. OS-DMX operates under the OS-65U V1.2 operating system (single- or multi-user). Features such as control files, extensive operating commands and the innovative HELP function, make this one of the most usable--as well as powerful--systems available for microcomputers. OS-DMX may be used instead of, or in addition to, OS-DMS Nucleus, Query, Sort; OS-DMX will replace virtually all of the specialized OS-DMS modules-- and in most applications will provide greatly improved performance.

OS-DMX Database Management System buyers will receive (no extra charge) a number of "extras" previously sold separately:

DMX-MAIL	Mailing List Management (FEB 82)
DMX-STAT	Comprehensive Statistical Analysis package (JULY 82)
DMX-COPY	Edit Database Structure after the fact (FEB 82)
DMX-MERGE	File Merge Operation (FEB 82)
DMX-TUTOR	450-Pg Tutorial w/ Demo Data Diskette (AVAILABLE)

In addition, DMX-SORT operations will be upgraded to machine-code sorting for faster operation. There will be no need to purchase high-speed sort programs separately.

OS-DMX DATABASE MANAGEMENT SYSTEM
Inst. Price $1600 List Price $1199

BISYNC-80/HASP

BISYNC-80/HASP is a full-function Multileaving Workstation package which allows communication with any remote CPU that supports a HASP Multileaving Workstation, and, as such, is ideally suited to Remote Job Entry applications.

OS-BISYNC-80/HASP (e)(f) List Price $1195

BISYNC-80/3270

BISYNC-80/3270 is a full-function IBM 3270 terminal emulator which allows the microcomputer to communicate over point-to-point telephone lines with any IBM S/360, S/370, or S30xx CPU that provides standard IBM support for one of the following:

IBM 3275 Model 2
IBM 3271 Model 2 or control unit w/ attached 3277 Model 2
IBM 3284 or 3286 printer

OS-BISYNC-80/3270 (e)(f) List Price $895

BISYNC-80/3780

BISYN-80/3780 is a full-function IBM 2780/3780 emulator allowing the microcomputer to communicate over point-to-point telephone lines with any CPU or device that provides standard IBM support for:

IBM 2780 Models 1, 2, 3 or 4
IBM 3780 w/ or w/o 3781 card punch
IBM CPU to CPU BSC communications

OS-BISYNC-80/3780 (e)(f) List Price $895

BISYNC-80/ASYNC

BISYNC-80/ASYNC is a full-function asynchronous communications package which allows microcomputers to communicate asynchronously with a mainframe or other microcomputers. This package is an ASYNC adaptation of BISYNC-80/3780 terminal emulation program, providing asynchronous communications at 75 to 9600 baud, using full IBM BISYNC protocol.

OS-BISYNC-80/ASYNC (e)(f) List Price $195

OS-BISYNC-80 SYNCHRONOUS INTERFACE ASSY List Price $395

NOTE: The prices shown in this catalog are estimates only; contact your OSI dealer for quotations. The "suggested installed price" reflects a typical business installation and includes reasonable allowance for software installation, minor program adaptation or customization, operator training, dealer support, back-up, etc. The "reference" or "list" price reflects a base price for the software for comparison purposes, exclusive of dealer installation and support.

Digital Technology, Inc., is the largest independent supplier of OSI software with hundreds of business packages in use around the world. Digital Technology software is sold by a growing number of conscientious OSI dealers and OEMs. Every package is backed by the finest support program in the microcomputer industry. All "bugs" are fixed free of charge. Updates (fixes to bugs, minor enhancements, new product announcements) are provided to all dealers and licensed users free of charge. And upgrades to new versions are encouraged (at nominal charge).

Digital Technology's software is user-obtained. In fact, no one else provides such expansive features as on-line documentation, idiot-proof prompting, and operator's manuals that are comprehensive, detailed, and accurate.

All Digital Technology software systems allow the operator to "set" the programs to the type of video terminal and printer used. The operator selects the terminal and printer types from the list provided in the "TERMINAL & PRINTER OPTIONS" program. Screen formatting and printer control are provided automatically yet may be redefined through user subroutines.

digital technology
inc.
P.O. BOX 178590
SAN DIEGO, CA 92117
(714) 270-2000

REQUIREMENTS

(a) BUS-II LEVEL I or LEVEL II G/L req'd
(b) BUS-II LEVEL I or LEVEL II A/R req'd
(c) Corresponding BUS-II Level I module(s) req'd
(d) H/D/E req'd
(e) C3 CPU W/ 56K RAM & OS-CP/M or Lifeboat Associates CP/M req'd
(f) SYNCHRONOUS INTERFACE ASSY req'd

If you disassemble 3.3 you will quickly see that it is a patchquilt affair. The code jumps all over the place. The watchword here, though, was compatability, that is why it was patched and not re-written. But if you add the value of compatibility, the new features, and the small memory overhead I think you'll agree 3.3 is a real bargain.

★

TURN YOUR OSI SUPPORT ROM INTO RAM

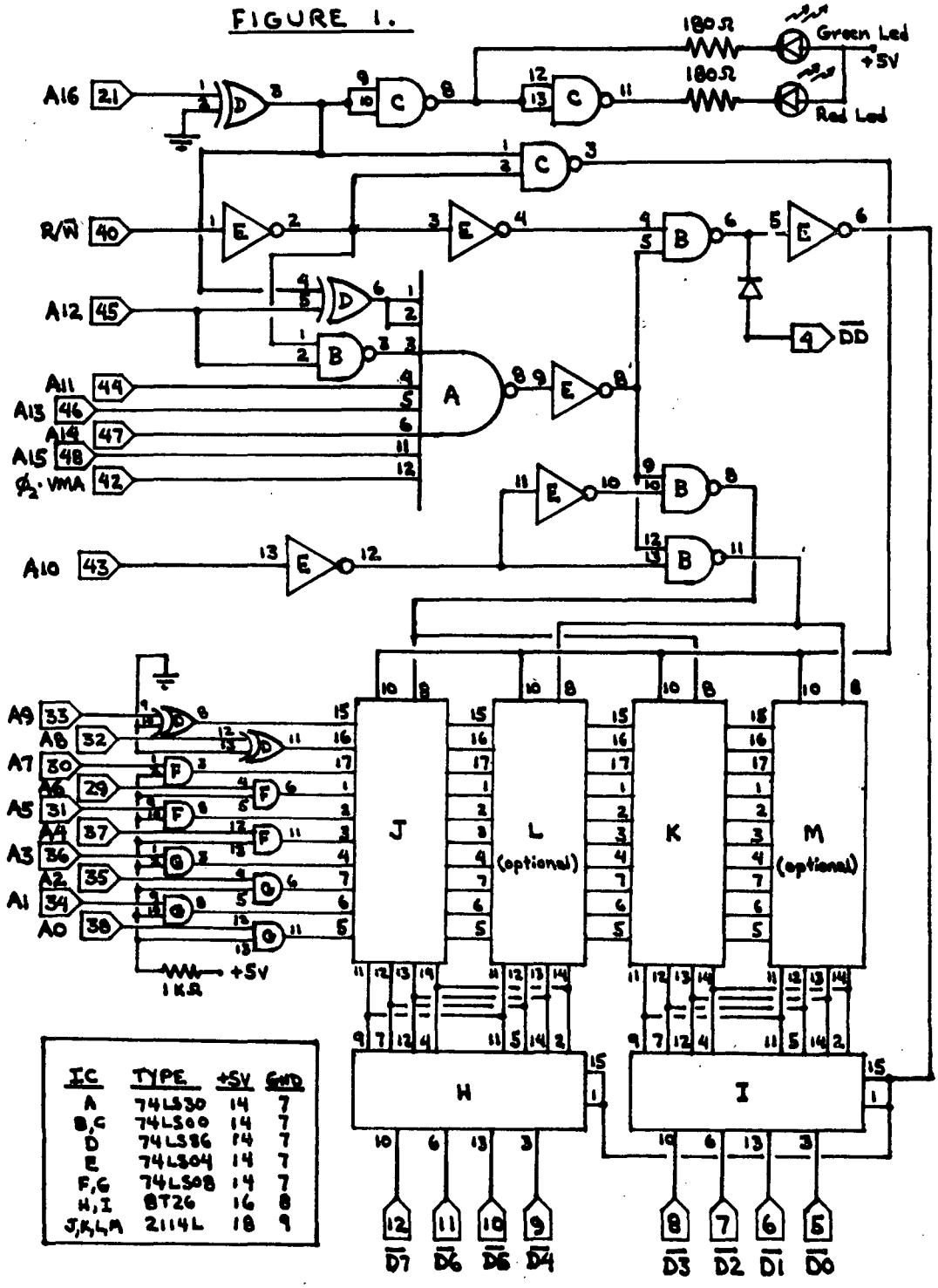
By: Michael J. Keryan
713 Locust Drive
Tallmadge, OH 44278

OSI's BASIC-in-ROM support chip contains a number of parameters and routines that cannot be changed. Described here is a method in which any (or all) of these can be changed at will, without replacing the chip.

Basic in ROM adds a great deal of convenience to small personal computers that are not equipped with disks. However, for this convenience, you give up some flexibility. Certain utility routines, vectors, and constants are untouchable by you the user, because they are in ROM. For example, OSI Basic in ROM machines have a 2K support ROM which contains the following:

(at $FCxx) Floppy bootstrap (for the C1P).

FIGURE 1.



IC	TYPE	+5V	GND
A	74LS30	14	7
B,C	74LS00	14	7
D	74LS86	14	7
E	74LS04	14	7
F,G	74LS08	14	7
H,I	8T26	16	8
J,K,L,M	2114L	18	9

(at $FDxx) Polled keyboard input routine.

(at $FExx) Machine language monitor. This area is independent of Basic. It contains video screen clear, initialization, memory input, load, go, ASCII to hexadecimal conversion, etc.

(at $FFxx) Basic support. This area contains various routines, pointers, and constants used continually by Basic. Routines include screen clear, menu, keyboard and tape load, input, save, output, and control C (break). The last 32 bytes contain constants and vectors:

Video parameters--cursor home location, line length, and screen size.

Pointers to memory for Basic source and variable storage areas.

I/O jump vectors to the input, output, load, save, and break routines.

Hardware vectors for non-maskable interrupt (NMI), interrupt request (IRQ), and reset (RES). These are set to $0130, $01C0, and $FF00, respectively.

There are several annoying problems caused by some of OSI's configurations:

1. IRQ vector. OSI defines this vector to point to location $01C0, which is near the top of the stack page. The reason for this is that very minimal systems can use interrupts (1K RAM or less). However, Basic initializes the stack to the top of the page; it doesn't take many FOR/NEXT loops or GOSUBs to destroy anything written in $01C0. It is virtualy impossible to use the IRQ while running Basic programs.

2. The video parameters set by OSI are much too restrictive. The 24 or 25 character widths on some systems make very poor tabulated listings. Most TV's can display approximately 30 characters (by running the TV at reduced voltage), but the computer will never use this extra space unless you POKE into it. Commercially available video modifications require a new support ROM to change these parameters.

3. The break key (actually the RESET line) is located too close to the other keys on some OSI keyboards. Accidentally hitting this switch causes a jump out of Basic to the system menu. An alternative to moving the switch would be to redefine the RES vector.

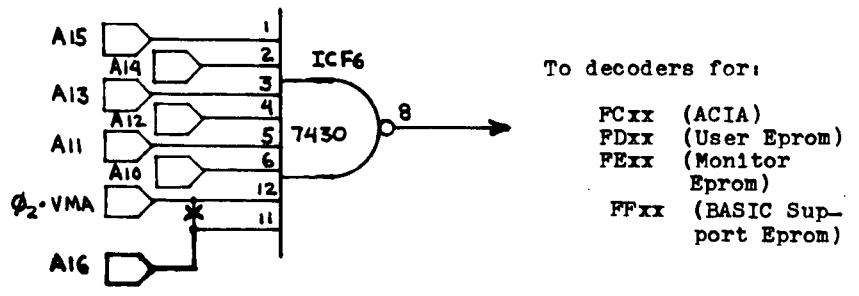
4. The C1P uses indirect I/O jump vectors (the vectors point to page zero pointers, which point to the routines), and are easy to change. However, older OSI systems use direct jump vectors and thus cannot be changed.

5. Although the support ROM contains two screen clear routines, you can't call either in your programs because they aren't written as subroutines. It should be possible to change both to make one good and one that can even be called by a Basic USR statement.

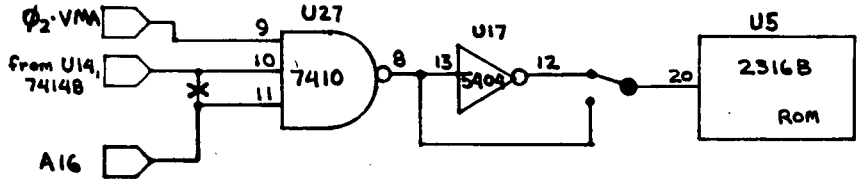
Also due to inherent limitations of many of the other routines, it is highly desirable to change them. To do this, you must physically pull out the support chip and replace it with another. You can program your own EPROM, or you can buy one for about $50. But any further changes will require a new or reprogrammed chip. It is desirable to have the support ROM present during system start-up and with only a few lines of code, change several vital locations, i.e. a read/write ROM.

While these new types of chips are becoming available (EAROM or EEPROM), the hobbyist does not yet have access to cheap, fast, reliable chips. But with some additional hardware, you can simulate this with RAM. This article describes a hardware modification, with which the entire support ROM contents ($FC00 - $FFFF) are transferred to RAM located 4096 bytes lower in memory ($EC00 - $EFFF). This can be done by either a simple block move machine language program

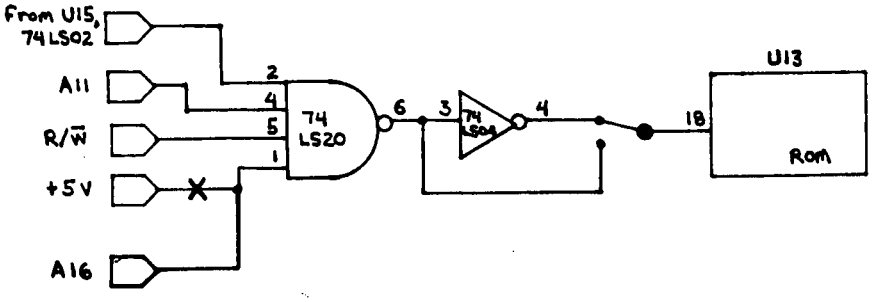
Figure 2.



a) C2-4P--OSI 500 Board Modification



b) C4P--OSI 502 Board Modification



c) C1P--OSI 600 Board Modification

Z-FORTH IN ROM by Tom Zimmer
5 to 10 times faster than Basic. Once you use it, you'll never go back to BASIC! $ 75.00
source listing add $ 20.00

OSI FIG-FORTH True fig FORTH model for 0S65D with fig editor named files, string package & much more $ 45.00

TINY PASCAL Operates in fig-FORTH, an exceptional value when purchased with forth.
TINY PASCAL & documentation $ 45.00
FORTH & TINY PASCAL $ 65.00

SPACE INVADERS 100% machine code for all systems with 64 chr. video. Full color & sound on C2, 4P & 8P systems. The fastest arcade program available. $ 14.95

PROGRAMMABLE CHARACTER GENERATOR $ 99.95
Use OSI's graphics or make a complete set of your own! Easy to use, comes assembled & tested.
2 Mhz. boards $109.95

PROGRAMMABLE SOUND BOARD $ 74.95
Complete sound system featuring the AY-3-8910 sound chip.
Bare boards available. $ 29.95

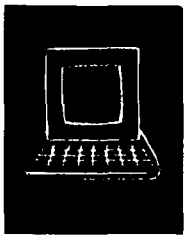
32/64 CHARACTER VIDEO MODIFICATION $ 39.95
Oldest and most popular video mod. True 32 chr. C1P, or 32/64 chr. C4P video display. Also adds many other options.

ROMS!!!
Augment Video Mod with our Roms. Full screen editing, print at,selectable scroll, disk support and many more features.
Basic 4 & Monitor $ 49.95
Basic 3 $ 18.95
All 3 for $ 65.00

65D DISASSEMBLY MANUAL. by Software Consultants
First class throughout. A must for any 65D user. $ 24.95

NUMEROUS BASIC PROGRAMS, UTILITY PROGRAMS AND GAMES ALONG WITH HARDWARE PROJECTS. ALL PRICES ARE U S FUNDS. Send for our $1.50 catalogue with free program (hardcopy) Memory Map and Auto Load Routine.

CHARGEX VISA



OSI Software & Hardware

3336 Avondale Court
Windsor, Ontario, Canada N9E 1X6
(519) 969-2500

3281 Countryside Circle
Pontiac Township, Michigan 48057
(313) 373-0468

progressive computing

or a one line Basic program. Then write over only those locations you want changed. But don't try writing the changes to the ROM; write them to the image in RAM. Then after everything in the RAM is the way you want it, change the state of a control line, a 17th address line (A16) by either flipping a switch or by software. This new address line then does several things:

1. It disables the support ROM.
2. It changes the RAM address from $Exxx to $Fxxx.
3. It disallows writing to the RAM.

You then have exactly what you need. RAM cleverly disguised as your support ROM. Return A16 back to its initial state and everything will switch back to normal.

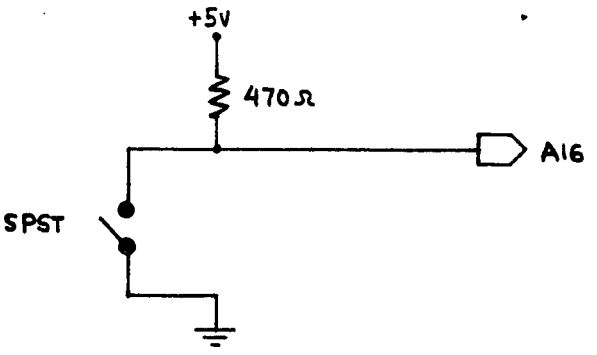
The additional hardware required is shown in figure 1. OSI bus numbers are shown next to each appropriate line. These signals can be obtained through the 40 pin expansion socket on the bus-less C1P (see your manual for pin numbers). For most systems, only two 2114L RAMs will be needed, allowing $FC00 - $FFFF to be revised. By using the two additional RAMs, the entire 2K space ($F800 - $FFFF) can be used, although most systems use only the upper 1K.

The circuitry can be built on an OSI-compatible prototype board. Two sources are OSI (model 495) and D&N Micro Products (model 96). Obviously, this isn't a project for the novice. Good wiring practices should be observed. Test out all lines with a meter before plugging in any chips. Test the circuit as normal read/write RAM located at $Exxx (A16=high) before proceeding. The optional LED circuitry will tell you when you have switched modes (green to red).

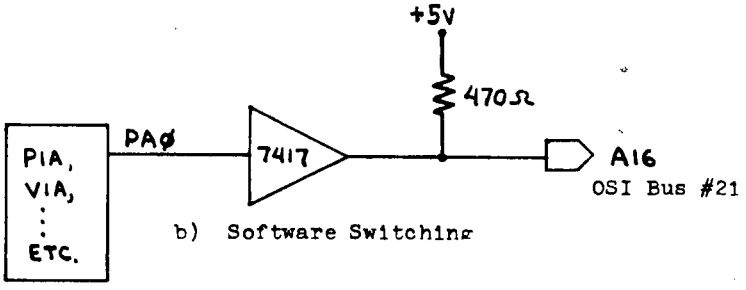
The support ROM must be disabled when A16 goes low; otherwise, two sources will try to supply data. This is done by cutting one trace to a decoder IC that enables the ROM. The circuitry is slightly different for the various OSI systems. Figure 2 shows appropriate methods for the 500 board (C2-4P), the 502 board (C4P), and the 600 board. Cut the foil at the X and wire in the A16 line as shown.

continued

Figure 3.



a) Manual Switching



b) Software Switching

So far we've done everything but define A16. Two methods of generating this signal are shown in figure 3. The manual method is merely an SPST switch and a resistor. Opening the switch is the "normal" mode; closing it puts the RAM in the "ROM" mode. Software switching is also shown in figure 3. One line of a PIA, VIA, or any parallel output bit drives the A16 line through a buffer. This circuit is already configured on the 500 board.

A summary of the functions and addressing is shown in Table 1. When not being used for the intended purpose, the RAM can be used as normal read/write memory located at $Exxx, an area that cannot be written over by Basic. One important thing to remember when writing machine language code for the RAM: although the code will be written into $Exxx, when A16 is switched low, the same code will be located at $Fxxx, so any non-relocatable code (jumps, subroutines, etc.) should be written as if it were located at $Fxxx. Also remember to fill the contents of the RAM before switching A16 to low, or your 6502 will get stuck in a non-existent routine, and even a RESET won't bring it back; only shutting off the power will restore it to normal.

This circuit was designed to allow IRQ to be used while using Basic, and to change some support routines. If nothing else, by allowing very easy changes to the utility functions and parameters, this modification will be helpful by allowing you to test your software improvements, before burning them into EPROM.

Table 1. Memory Map

	Support ROMS		Added RAM: 2K	
A16	READ	WRITE	READ	WRITE
1	FC00-* FFFF	WRITE PROTECT	E800-** EFFF	E800-EFFF
0	DISABLED	DISABLED	F800-FFFF	WRITE PROTECT

Notes:

* Beginning of decoded area may vary from F800 to FD00, depending on system.

** Optionally, only EC00-EFFF and FC00-FFFF may be used for 1K added RAM.

★ ★ ★

LETTERS

ED:

OSI even strikes out with their screwy documentation on the SAM's manual, at least for the 527 board. They have the memory chips identified in the wrong order (transposed end to end) on the photo. Don't think that didn't lead to frustration when I was looking for the cause of memory lapse in my C8 when it got warm. The schematic and board seem to agree at least to the point I could find the problem.

Question, what all does "CEGMON" change in page 1 & 2 compared to "SYNMON"? Something is surely different as some commercial software interferes with it or vice versa.

Neil Dennis
Bliss, NY 14024

Neil:

CEGMON is a character generator only! SYNMON is a full monitor ROM and does lots of vital things.

If you are talking about a 4PMF that is user converted for serial operation, later releases of SYNMON have different pages 4-7 and are missing page 6 (serial monitor). In short, you cheated on the cost of a C3-OEM and got caught.

SYNMON CONTENTS

page
0 65 U with ASCII keyboard routine
1 Basic support for keyboard
2 Polled keyboard routine
3 65U monitor for polled
4 ROM BASIC support for 540 polled
5 Hard disk
6 Serial monitor
7 Reset vector

Information obtained from M/A-COM OSI

* * * * *

ED:

I have been an OSI C1P series II owner for about two years. I have nothing but good things to say about it. I'm sure there are others like me. I hope your magazine can keep me in contact with other OSI owners and up on the new developments, products or what ever, for the OSI. Who knows, some day I might be able to contribute!

I would like to give you a brief description of my computer set-up. I own a C1PMF with 32K of memory. I just recently (1 week ago) bought the 610 board, the CD3P disk drive and OS65DV3.3. So you can see, I am very new to the disk scene. Boy, is there a lot to learn! I also have a model 33 teletype and a quest $60 modem. Maybe I can contact someone in your area sometime. I understand there is a users group in your area. Is PEEK (65) published by this group?? If there is a users group, it would be the closest one to me that I know. I wouldn't mind joining such a group and making a contact via modem. One other item that I have is a library of 40 cassettes.

I am heavy into hardware modifications. That is why I like OSI so much. They seem to encourage it.

David L. Kuhn

P.S. How can I put a 'NULL' command in so that OS65DV3.3 is more patient with my model 33?

David:

The best source of information about your machine is right here at PEEK (65). If you don't have the January 1981

OSI

AARDVARK NOW MEANS BUSINESS!

OSI

WORD PROCESSING THE EASY WAY— WITH MAXI-PROS

This is a line-oriented word processor designed for the office that doesn't want to send every new girl out for training in how to type a letter.

It has automatic right and left margin justification and lets you vary the width and margins during printing. It has automatic pagination and automatic page numbering. It will print any text single, double or triple spaced and has text centering commands. It will make any number of multiple copies or chain files together to print an entire disk of data at one time.

MAXI-PROS has both global and line edit capability and the polled keyboard versions contain a corrected keyboard routine that make the OSI keyboard decode as a standard typewriter keyboard.

MAXI-PROS also has sophisticated file capabilities. It can access a file for names and addresses, stop for inputs, and print form letters. It has file merging capabilities so that it can store and combine paragraphs and pages in any order.

Best of all, it is in BASIC (OS65D 51/4" or 8" disk) so that it can be easily adapted to any printer or printing job and so that it can be sold for a measly price.

MAXI-PROS — $39.95

NEW-NEW—NEW TINY COMPILER

The easy way to speed in your programs. The tiny compiler lets you write and debug your program in Basic and then automatically compiles a Machine Code version that runs from 50-150 times faster. The tiny compiler generates relocatable, native, transportable machine code that can be run on any 6502 system.

It does have some limitations. It is memory hungry — 8K is the minimum sized system that can run the Compiler. It also handles only a limited subset of Basic — about 20 keywords including FOR, NEXT, IF THEN, GOSUB, GOTO, RETURN, END, STOP, USR(X), PEEK, POKE, -,=,*,/, (,), <>, Variable names A-Z, and Integer Numbers from 0-64K.

TINY COMPILER is written in Basic. It can be modified and augmented by the user. It comes with a 20 page manual.

TINY COMPILER — $19.95 on tape or disk

THE AARDVARK JOURNAL

FOR OSI USERS — This is a bi-monthly tutorial journal running only articles about OSI systems. Every issue contains programs customized for OSI, tutorials on how to use and modify the system, and reviews of OSI related products. In the last two years we have run articles like these!

1) A tutorial on Machine Code for BASIC programmers.
2) Complete listings of two word processors for BASIC IN ROM machines.
3) Moving the Directory off track 12.
4) Listings for 20 game programs for the OSI.
5) How to write high speed BASIC — and lots more —

Vol. 1 (1980) 6 back issues - $9.00
Vol. 2 (1981) 2 back issues and subscription for 4 additional issues - $9.00.

ACCOUNTS RECEIVABLE — This program will handle up to 420 open accounts. It will age accounts, print invoices (including payment reminders) and give account totals. It can add automatic interest charges and warnings on late accounts, and can automatically provide and calculate volume discounts.

24K and OS65D required, dual disks recommended. Specify system.

Accounts Receivable. $99.95

*** * * SPECIAL DEAL — NO LESS! * * ***

A complete business package for OSI small systems — (C1, C2, C4 or C8). Includes MAXI-PROS, GENERAL LEDGER, INVENTORY, PAYROLL AND ACCOUNTS RECEIVABLE — ALL THE PROGRAMS THE SMALL BUSINESS MAN NEEDS. $299.95

P.S. We're so confident of the quality of these programs that the documentation contains the programmer's home phone number!

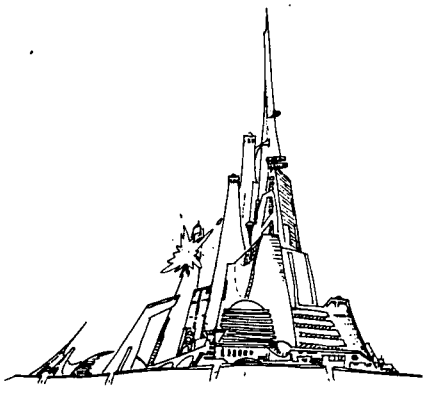
SUPERDISK II

This disk contains a new BEXEC* that boots up with a numbered directory and which allows creation, deletion and renaming of files without calling other programs. It also contains a slight modification to BASIC to allow 14 character file names.

The disk contains a disk manager that contains a disk packer, a hex/dec calculator and several other utilities.

It also has a full screen editor (in machine code on C2P/C4)) that makes corrections a snap. We'll also toss in renumbering and program search programs — and sell the whole thing for — SUPERDISK II $29.95 (5 1/4") $34.95 (8").

AND FUN, TOO!



BOOKKEEPING THE EASY WAY — WITH BUSINESS I

Our business package 1 is a set of programs designed for the small businessman who does not have and does not need a full time accountant on his payroll.

This package is built around a **GENERAL LEDGER** program which records all transactions and which provides monthly, quarterly, annual, and year-to-date PROFIT AND LOSS statements. GENERAL LEDGER also provides for cash account balancing, provides a BALANCE SHEET and has modules for DEPRECIATION and LOAN ACCOUNT computation.

GENERAL LEDGER (and MODULES) $129.95.

PAYROLL is designed to interface with the GENERAL LEDGER. It will handle annual records on 30 employees with as many as 6 deductions per employee.

PAYROLL - $49.95.

INVENTORY is also designed to interface with the general ledger. This one will provide instant information on suppliers, initial cost and current value of your inventory. It also keeps track of the order points and date of last shipment.

INVENTORY - $59.95.

GAMES FOR ALL SYSTEMS

GALAXIAN - 4K - One of the fastest and finest arcade games ever written for the OSI, this one features rows of hard-hitting evasive dogfighting aliens thirsty for your blood. For those who loved (and tired of) Alien Invaders. Specify system — A bargain at $9.95

NEW — NEW — NEW

LABYRINTH - 8K - This has a display background similar to MINOS as the action takes place in a realistic maze seen from ground level. This is, however, a real time monster hunt as you track down and shoot mobile monsters on foot. Checking out and testing this one was the most fun I've had in years! — $13.95.

NIGHT RIDER - You've seen similar games in the arcades. You see a winding twisting road ahead as you try to make time and stay on the road. NIGHT RIDER uses machine code to generate excellent high speed graphics - by the same author as MINOS.

NIGHT RIDER — $12.95 cassette only

THIEF - Another machine code goody for the C1P cassette only. You must use mobile cannon to protect the valuable jewels in the middle of the screen from increasingly nasty and trigger happy thiefs. Fast action and fun for one or two players. THIEF $13.95 on C1 cassette only!

SUPPORT ROMS FOR BASIC IN ROM MACHINES — C1S/C2S. This ROM adds line edit functions, software selectable scroll windows, bell support, choice of OSI or standard keyboard routines, two callable screen clears, and software support for 32-64 characters per line video. Has one character command to switch model 2 C1P from 24 to 48 character line. When installed in C2 or C4 (C2S) requires installation of additional chip. C1P requires only a jumper change. — $39.95

C1E/C2E similar to above but with extended machine code monitor. — $59.95

Please specify system on all orders

This is only a partial listing of what we have to offer. We now offer over 100 programs, data sheets, ROMS, and boards for OSI systems. Our $1.00 catalog lists it all and contains free program listings and programming hints to boot.

OSI

AARDVARK TECHNICAL SERVICES, LTD.
2352 S. Commerce, Walled Lake, MI 48088
(313) 669-3110

OSI

and December 1981 back issues, order them. They contain alphabetical indexes of all PEEK (65) articles from our first two years of publication, and will help you find more information.

The users' group is called OSIO. Contact them at 9002 Dunloggin Rd., Ellicott City, MD 21043. They do not publish PEEK (65), but we are friends! You might also want to contact Don Shay, The Challenger News, P.O. Box 18335, Philadelphia, PA 19120.

Readers, who knows how to ad Nulls under 65D 3.3?

Al

* * * * *

ED:

This subroutine is for a program that requires internal conversion and offers the User conversion from the Menu etc. The routine loads in 411 bytes.

(1) All inputs must be set to A$. (2) Hex inputs to Lines 100, 105 must be prefixed with "H" (personal preference, one stroke is easier than 2) (3) There are no program limits on input values other than integers only. (4) There is no validation of Hex input. (5) After any conversion, A$ = Hex without prefix and D = decimal. (6) U is POKE control, U=0 = no pokes. (7) The screen POKEs are set for a C1P with a 24*24 display, they will probably look funny on a "real" machine.

```
100 INPUT" #";A$:U=1:IFASC(A$)
    =72THENPOKE 54089,36  :REM
    get user's input; set POKE
    to screen; if Hex POKE "$"
    on "H"
105 IFU=1THENPOKE54100,61
    :REM POKE "=" to screen if
    POKE is called
106 POKE54117,32:IFASC(A$)=72
    THENA$=MID$(A$,2):GOTO130
    :REM POKE out cursor; If
    hex strip prefix "H" and
    goto Hex/Dec
110 D=VAL(A$):E=D:A$=""  REM:
    Set D to dec input; set E
    to D to preserve D; Null
    Dec to Hex operator
115 X=INT(E/16):Y=E-X*16+48:Y=
    Y-7*(Y>57):A$=CHR$(Y)+A$:
    E=X:IFXTHEN115  REM: Dec
    to Hex converter
120 IFU=0THENRETURN  REM: If
    POKE to screen not called,
    return
125 U=0:T=0:H$="$"+A$:GOTO150
    REM: Reset POKE request;
    Null POKE operator; copy
    A$ to preserve it and
    attach "$"; goto POKE to
    screen
130 H$=A$:D=0 Rem; copy A$ to
    preserve it; Null Hex to
    Dec operator
135 X=ASC(H$)-48:X=X+7*(X>9):
    D=D*16+X:H$=MID$(H$,2):IFH
    $<>""THEN 135  REM: Hex to
    Dec converter
140 IFU=0THENRETURN  REM: If
    POKE not called, return to
    caller
145 U=0:T=0:H$=STR$(D)  REM:
    Reset POKE request; null
    POKE operator; set H$ =
    Dec value for POKE to
    screen
150 IFLEN(H$)>6THEN?:T=-5 REM:
    if screen would overflow,
    scroll one line, back-up
    POKE start
155 POKE54102+T,ASC(H$):H$=
    MID$(H$,2):IFH$=""THEN
    RETURN REM:POKE ASCII
    value of digit; strip
    that digit; if done return
    to caller
160 T=T+1;GOTO155  REM: not
    done do next digit
```

INPUTS:

```
100 User input, Hex or Dec, if
    Hex user must prefix with
    "H"
105 Pgm input, Hex or Dec, if
    Hex user or program must
    prefix "H"
110 Pgm input, Dec only
130 Pgm input, Hex only (no
    prefixes)
```

Keep up the good work, PEEK must grow and grow. Incidentally, I timed Line 115, above, with a fixed input of 65535 at .111 secs per conversion. I don't know if this is good or bad.

Harry Hawkins
Burton, SC 29902

* * * * *

ED:

VIDEO MOD TO 65D3.3 FOR OSI C1P 32 CHR.

In BASIC (the easy way) add this line to your BEXEC*

POKE 13042,25:REM BOTTOM OF SCREEN $32F2

POKE13043,31:REM LEN OF LINE ON SCREEN $32F3

POKE13044,96:REM HOME LOCATION OF CURSOR $32F4

AFTER BEXEC* RUNS HIT ESC AND #1 TO GIVE 32 CHR SCREEN

For the more adventurous, you may save these changes in the operating system as follows. Make the POKEs above in the immediate mode then execute the following command:

DISK!"SA 13,1=3274/8"

The next time you boot this disk the 32 CHR mod will be activated.

We are still working on the 64 CHR 2K screen mod for users with that modification and will share this information as soon as it is available.

Charles A. Stewart &
David J. Larson
Adrian, MI 49221

* * * * *

ED:

RE: COMPUTERCUBE

Sincere apologies, some errors crept into my lisitng of COMPUTERCUBE in PEEK (65), March 1982. Fortunately they shouldn't have caused much strife - the program would still run.

INSERT:

285 POKER,13:POKED,0

this switches on the PSG when a new cube face is called.

DELETE line 680

1490 J=0:GOTO300:PRINTCR$

should (of course !) read:

1490 J=0:PRINTCR$:GOTO300

Colin Law
New Zealand

* * * * *

ED:

There have been several notes in PEEK (65) regarding the 'flakey' random number generator in the OSI BASIC-in-ROM systems. Several of these pointed out that the period of this generator is 1861. I have no quarrel with these findings, but would like to add some more fuel to the fire.

The one letter that I recall most clearly included a simple BASIC program that saved the first five random numbers. Each of the succeeding numbers was compared to the first five to determine if the sequence was repeating. When this program was loaded in my C2-4P the results were exactly as advertised; the sequence repeated after 1861 random numbers. It then appeared that 1861 was a 'magic' number. Not so!

Another writer suggested that the only way to assure a reasonably true random sequence was to 'seed' the generator

with a negative number. Also, not true!

I have attached a modification of the BASIC program which tests the repeat count of the random number generator. This is no different from the original letter. The program requests the user to enter a 'seed' value. This value is used as the argument in the random number function. This is no big deal, since the argument is meaningless unless negative. The user also has the option of entering a negative seed. If selected the negative value of the original seed value is used. This option does make a difference. The program will run through 1861 random numbers (unless it finds a repeat sequence). Then it stores a new series of five random numbers and starts the whole test sequence over again. Each time the sequence is restarted, an '*' is output to the display. If a repeat sequence of 1861 or less is detected, the repeat count is printed.

Using this technique some interesting results were obtained. Depending on the negative seed value used, several different repeat counts can be demonstrated. The system may sequence through several iterations of 1861 without repeating (?). Eventually it will lock into some sequence. In addition to the sequence of 1861, I have also found sequences of 279 and 813.

The reason for the question mark in the preceding paragraph is that I am not sure if other sequences exist. These could be short repetative series that are buried within the major sequences. It would require a more complex program and more memory than I have available to test for this. Probably a better approach would be to disassemble the code and determine how the algorithm really works.

Once the system develops the repeat count mentioned above, it will continue with that sequence until reseeded with a negative value. If you wish to test this for yourself; load the program listed below and try seed values of 3, 6, 13, 17, 20, 123 or 133. Answer the 'NEGATIVE SEED; prompt with 'YES'. In each case the system will cycle through several sequences of 1861 but will finally lock onto a sequence of 279. Unless reseeded with a negative value, the repeat count will continue to be 279 for any seed value entered. Seed values of 41, 71, 78, 122 or 124 will produce a cycle of 813. There are certainly other seed values that will produce the same results and possibly some other repeat sequences. I have personally tested seed values up to 137 with the results mentioned above.

I hope that all of this has been of some help to those who require random numbers in their work. I should mention that no computer or mathematical sequence can produce truly random numbers. I have done some additional work with Chi-Squared tests to determine if the sequences produced appear statistically random. I would be happy to submit these if there is any interest.

```
RANDOM NUMBER TEST

10 INPUT"ENTER SEED VALUE";S
15 PRINT:INPUT"WANT NEGATIVE
   SEED";A$
20 IF LEFT$(A$,1)="N" THEN30
25 G=RND(-S)
30 FOR I=1 TO 5:N(I)=RND(S)
   :NEXT
35 C=5
40 X=RND(S):C=C+1:IF C>1866
   THEN PRINT"*";:GOTO 30
45 IF N(1)<>X THEN 40
50 R(1)=X
55 X=RND(S):C=C+1
60 IF N(2)<>X THEN 40
65 R(2)=X
70 X=RND(S):C=C+1
75 IF N(3)<>X THEN 40
80 R(3)=X
85 X=RND(S):C=C+1
90 IF N(4)<>X THEN 40
95 R(4)=X
100 X=RND(S):C=C+1
105 IF N(5)<>X THEN 40
110 R(5)=X
115 PRINT
120 FOR I=1 TO 5:PRINT N(I);
    :NEXT
122 PRINT:FOR I=1 TO 5:PRINT
    R(I);:NEXT
125 PRINT:PRINT "REPEAT
    COUNT="C-5
130 END

Harry B. Pye
Lansdale, PA   19446

Harry:

Of course there is interest!

Al

* * * * *

        PRESS RELEASE
* * * * * * * * * * * * * * *
```

DP Directory, a new data processing reference magazine, publishes the tables of contents of over 100 DP periodicals each month. In addition to PEEK (65), DP Directory covers dozens of data processing magazines dealing with hardware, software, systems development, telecommunications, graphics, word processing and personal computing. 12 monthly issues are available for $48.00 from DP Directory, P.O. Box 562, Bloomfield, CT 06002.

AD$

Dual ss,sd floppies, CPU with additional 8K for CPM, CA-10, C-23 hard disk completely rebuilt from factory OS65U, OS65D, Abacus G/L, CPM-Basic, Cobol, Fortran, Malibu 160 printer, Hazeltine 1500 terminal, fast floppy loader, memory test, Unisoft G/L, etc. Works beautifully. No problems. Will sell whole system for $11,000 or will sell without hard disk for $6800. R.J. Murray, 916/628-5513.

FOR SALE: C4PMF in excellent condition, with joysticks, software, manuals, and other technical information, $1450. For more information call 713/776-9821 or 713/437-4156.

Spring cleaning our inventory creates great bargains. ATV Research MICROVERTER, $27.50; LEEDEX 12" B&W Monitor with documentation, $100.00; EPSON RS-232 serial interface, $50. All items new. Software available for Investment Counselors, CPA's, and other business applications. Electronic Information Systems, P.O. Box 5893, Athens, GA 30604, 404/353-2858.

FOR SALE: C3-OEM, excellent condition. Includes 48K memory (one board), CA-14A voice output board (ready to plug into your speaker or stereo receiver), CA-20 8 port I/O board with real time clock and calendar (on-board batteries), and CA-18 "jungle" board with 4K memory (usable by UCSD Pascal), Centronics parallel interface, 24 bit parallel interface, and two RS-232 ports ready to go. Will configure baud rates of RS-232 ports for you. SOFTWARE: OS-65U, V1.2, OS-65D, V3.2, and UCSD Pascal. All manuals included. CALL: Brian Goodhart. Work 301/984-8000. Home: 202/265-4560. I travel frequently for my employer (that's why I'm selling), so if I am on the road please leave message. Will return all calls.

FIG FORTH UNDER OS-65U

OK. You've read about Fig Forth after Fig Forth, and you're probably thinking they're all about the same. You're probably right too, with one exception, the Fig Forth from SOFTWARE CONSULTANTS.

Our Fig Forth is the only alternative language available to OSI Level III users that want to run more than just BASIC. Our version of this "user-defined" language is the only one that's especially designed for the hard-disk, multi-user system.

If you're an assembler programmer, you'll love our Forth because it offers you the flexibility and execution speed you need, all the advantages of a high-level language, and the freedom to construct your own programming "tools." Since our Forth is totally extensible, it can do much more than BASIC ever could. And, under Level III, you can run it simultaneously with any BASIC program you may not want to rewrite.

Try Fig Forth from Software Consultants. We think you'll like it as much as we do.

FEATURES

- Supports multi-user, hard-disk systems
- Includes a number of helpful utilities and applications, such as:
  - *Saves & executes compiled code, eliminating compilation time for completed programs*
  - Full screen terminal-oriented editor
  - Printer and terminal tools
  - Case command
- Runs simultaneously with BASIC

PRICE $89.95

OTHER GREAT PRODUCTS

REF COMMAND UNDER BASIC
Lists line numbers, variables, constants for 65D or 65U. $31.95

SPOOLER/DESPOOLER UTILITY
Super fast. Frees up screen, feeds data to serial or parallel printers. $69.95

OS 65D V3.2 DISASSEMBLY MANUAL
60 page manual complete with cross reference listing. Fully commented. Now a classic. $25.95.

VIDEO ROUTINE
Convenient control of variable screen parameters. May be connected to graphics resolution booster. $25.95 alone. With extensions, $29.95.

GRAPHICS RESOLUTION BOOSTER
Hardware to boost screen resolution by 8 times to 128 x 128. $49.95 alone. With video routine and software extensions, $79.95.

Orders shipped postpaid from Memphis. Foreign orders please add $10 postage fee. Source code for purchased products is available for $12 each.

Dealer inquiries welcome.

Write or call us today with your order, or ask for our free product catalog and get all the details.

SOFTWARE CONSULTANTS
HARDWORKING *software*

6435 Summer Avenue
Memphis,TN 38134
901/377-3503

PEEK (65)
The Unofficial OSI Users Journal

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

BULK RATE
U.S. POSTAGE
PAID
Owings Mills, MD
PERMIT NO. 18

DELIVER TO: