PEEK (65)

The Unofficial OSI Users Journal

$1.75
April 1983
Vol. 4, No. 4

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

INSIDE

Column One

I just got off the phone with Ken Worktz, president of OSI. It is a different company these days, as Ken is the first to point out.

Ken has just returned from Europe, where he had "very educational -- very worthwhile" meetings with European dealers and distributors.

Perhaps the most exciting events in the OSI world are the Product Seminars now underway throughout the US. Dick McGuire is at the Washington DC seminar today, and by the time you read this issue of PEEK(65), seminars will have been held in most major cities in the US. Dick will describe the seminar in more detail for us next month, but some of the major features Ken pointed out were:

Joe Sorrentino, the owner of OSI, has attended every seminar, indicating the support of highest management for the new products;

Ads are appearing in the Wall Street Journal and other national publications, backing up the seminars;

Most exciting is the 300 series itself: they are true multiprocessor machines, with each terminal given its own separate computer, complete with CPU, RAM, operating system (MasterKey, a Turbodos system) and I/O, sharing expensive resources such as hard disks and printers;

The Beta-tests are complete, a few minor changes have been made as a result of these tests, and now production units are being shipped, on schedule as promised.

Ken feels that this is particularly important. "I made a lot of promises to the industry, and we're keeping all of them," he notes.

Among the promises kept have been the development of the 300 series itself, "from the drawing board to delivery in 90 days," 65U compatibility of MasterKey with simultaneous CP/M compatibility, and the installation of a system of regional sales offices with dealers on contract.

The result of all this is that the new OSI has contracted to sell $14 million dollars worth of computers in 90 days, double total OSI sales for the previous year.

Concerning the changes, Ken says, "It's nothing new in the industry. All simply fundamentals. It's something OSI should have been doing all along..."

I have yet to see a 300 series machine. Reports are good, and Dick will be able to tell us first hand what they look and work like. Still, it is hard not to get excited and catch some of Ken's obvious enthusiasm. Let's hope he can be as cheery a year from now as he is today!

Ken added that OSI "has a huge inventory of hobbyist equipment, and a large customer base of hobbyists." He promised that there are no plans to abandon that business, though no new machines are at present planned. "We don't plan at present to get into the business of competing with Commodore and [the smaller] Apple, though of course we reserve the right to re-enter that market agressively in the future."

There will be, he added, a new low-priced machine announced later this year, which some may interpret as a new hobbyist machine, but he hastened to add that it is actually intended as an intelligent workstation for the series 200 and 300 machines, "though it could certainly be used by hobbyists."

al

AN OS65D DISK READER

by: Steven P. Hendrix
Route 8, Box 81E
New Braunfels, TX 78130

Among the many industry standards OSI chose to ignore in their computer line, their disk format alone guaranteed incompatibility with all other manufacturer's systems. To make matters worse, the only available alternative operating system not produced by OSI is HEXDOS. HEXDOS uses a format which is different from even OSI's format. As the author of HEXDOS, I can say in hindsight that it may have been a poor choice, but I had my reasons at the time for the disk format I chose. If you are wondering about designing your own disk format, let me say that the hardware allows you to easily change it, since most of the formatting is done in software, but it does not allow you to change it to match any standard format, as it will always have either 10 or 11 bits per byte, and the other standard formats all use less.

For those who would like to adapt some of OSI's published software to HEXDOS, I will present in this article a routine to copy BASIC files from a standard OSI disk to a HEXDOS disk. The routine comes in two parts: a machine-language routine which will read a specified sector from a disk into a specified area of memory, and a Basic program which uses this routine to copy Basic programs from OSI disks to HEXDOS disks. For those who want to get into more detail, the machine code routine is set up to be position-independent, so you can place it anywhere in memory to use it with other routines of your own. The code I present here will work with HEXDOS 4.0. I will copy

published monthly

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

Subscription Rates

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

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

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

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


```
                        HEXASM ASSEMBLY LISTING      PAGE  # 1

10    0000              .OPTION L 2 S 2 E 2

                        ; OS65D SECTOR READER FOR HEXDOS

                        ; USR(-7) track, sector, memloc
                        ; WILL READ A SECTOR INTO MEMLOC
                        ; RETURNING AS ITS VALUE THE NUMBER
                        ; OF PAGES IN THAT SECTOR.

100   0000              MEMLOC = $AD
110   0000              PC     = $2FD
120   02FD 00           TRACK  .BYTE 0
130   02FE 00           SECTOR .BYTE 0
140   02FF 00           BCDTRK .BYTE 0

160   0300              PC = 0

180   0000 20 A5 06     JSR  GETVAL
190   0003 8D FD 02     STA  TRACK
200   0006 20 01 AC     JSR  $AC01
210   0009 20 A5 06     JSR  GETVAL
220   000C 8D FE 02     STA  SECTOR
230   000F 20 01 AC     JSR  $AC01
240   0012 20 A5 06     JSR  GETVAL
250   0015 85 AD        STAZ MEMLOC
260   0017 A5 AE        LDAZ $AE
270   0019 85 AE        STAZ MEMLOC+1

290   001B AD FD 02     LDA  TRACK
300   001E C9 1E        CMP# 30
310   0020 90 04        BCC  PC+5
320   0022 69 11        ADC# 17
330   0024 90 0E        BCC  PC+15
340   0026 C9 14        CMP# 20
350   0028 90 04        BCC  PC+5
360   002A 69 0B        ADC# 11
370   002C 90 06        BCC  PC+7
380   002E C9 0A        CMP# 10
390   0030 90 02        BCC  PC+3
400   0032 69 05        ADC# 5
410   0034 8D FF 02     STA  BCDTRK

430   0037 A9 FF        LDA# $FF
440   0039 85 E6        STAZ CKSEEK
450   003B AD FD 02     LDA  TRACK
460   003E 20 F3 04     JSR  SEEK
480   0041 E6 E6        INCZ CKSEEK
500   0043 20 9C FC     JSR  READSK
510   0046 C9 43        CMP# $43
520   0048 D0 F9        BNE  PC-6
530   004A 20 9C FC     JSR  READSK
540   004D C9 57        CMP# $57
550   004F D0 F5        BNE  PC-10
560   0051 20 9C FC     JSR  READSK
570   0054 CD FF 02     CMP  BCDTRK
580   0057 F0 03        BEQ  PC+4
590   0059 4C 58 05     JMP  TRKERR
600   005C                        SLOOP
610   005C 20 9C FC     JSR  READSK
620   005F C9 76        CMP# $76
630   0061 D0 F9        BNE  PC-6
640   0063 20 9C FC     JSR  READSK
650   0066 CD FE 02     CMP  SECTOR
660   0069 F0 12        BEQ  FNDSECT
670   006B 20 9C FC     JSR  READSK
680   006E A8           TAY
690   006F A2 01        LDX# 1
695   0071 CA           DEX
700   0072 20 9C FC     JSR  READSK
710   0075 CA           DEX
720   0076 D0 FA        BNE  PC-5
730   0078 88           DEY
```

Listing continued on pg.4.

High Resolution Color Graphics

Finally, low-cost high-resolution color graphics is available for your OSI computer. With Color-Plus from Generic Computer Products, you can have the following features:

- Color — 15 unique colors plus transparent
- High Resolution — 256 × 192 with 2 different colors in each group of 8 horizontal dots
- Medium Resolution — 48 × 64
- Text — 24 × 40 characters
- Sprites — 32 programmable animation patterns that move smoothly across the screen without disturbing the background
- Joystick interface — Supports up to 2 joysticks or 4 game paddles with 8-bit resolution
- Software — Extensions for OS65D which provide a superset of Apple][graphics instructions
- Video switch — Software selects the Color-Plus or standard 540 video display

Color-Plus does not need user memory, leaving the full 48K memory space available for user programs.

Two versions are available:

CP-16 — Connects to C4P and C8P systems with the 16-pin bus or to any system equipped with the OSI CA20 board. Comes in ABS plastic case.

CP-48 — Connects to the standard 48-pin bus.

Cost: $279

Low Power Memory Board

Our popular MEM + board is ideal for:

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

Options include:

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

MEM + includes the following features:

- Low power consumption — A 48K board draws about 1/4 amp. A fully populated board draws about 3/4 amp
- Accepts 2K × 8-bit memory chips — Compatible with 2716-type EPROMs
- High reliability — All memory chips in machine-screw sockets
- Versatile addressing — Divided into 3 16K blocks and 2 individually addressable 4K or 8K blocks

Bare	$100
16K	$275
24K	$325
32K	$370
40K	$410
48K	$450
56K	$490
64K	$530

Disk controller	$95
Real time clock	$65
Centronics interface	$45

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

To order, or for more information, contact:

Fial Computer
11266 SE 21st Avenue
Milwaukie, Oregon 97222
(503) 654-9574

Generic Computer Products

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

these routines onto a disk if you send it with return postage.

Listing 1 is an assembly language listing of the sector read routine. It is in the format for HEXASM, which uses slightly different designations for the 6502 addressing modes than are used by OSI's assembler. If you prefer to just type in the object code, see listing 2 for a hexadecimal dump. Place the code in memory starting at some address B, and place the high-byte of B in $00F1 and low byte in $00F0 to link this routine to USR(-7) under HEXDOS.

The calling format for the USR routine looks like:

P=USR(-7) T, S, M

where T is the desired track number, S is the sector number, and M is address of the beginning of the memory area to receive the data from the disk. P will be set to the number of pages (256 bytes each) of data which were contained in the sector.

Listing 3 is the Basic program which uses the above routine to pick a Basic program off of an OS65D disk and write it to a HEXDOS file. I would suggest using the write-protect feature on the disk to be copied, though the program will stop before it tries to write on the 65D disk if you get them mixed up, since it will not find the HEXDOS directory where it expects to. Note that you must create a file to accept the copied program before running this program, since it is used as a data file. There will still be a few incompatibilities remaining in the program after you transfer it, due to differences between ROM Basic and Disk Basic. The verb DISK will be changed to SAVE, and the verb EXIT will show up as LOAD. Anything that relies on the 9 digit precision of disk Basic will have to contend with the 6 digit precision of ROM Basic, but it will run correspondingly faster. And of course, PEEKs and POKEs may need to be changed, depending on what area they were addressing. The two biggest changes in transferring to HEXDOS will be that the program will run somewhat faster and that it will have some 12K more memory available to work with.

Basic program is to load the block of code in listing 2 into memory, and set the pointers in page zero as shown in the last line of the listing. If you then warm start Basic and do a LIST, you will see a line 0 which consists of REMark full of garbage. DON'T try to edit it! You can add lines to the program by typing them in as usual. Since the code is carefully set up to avoid any zero bytes, Basic will treat the whole thing as a remark, with no ill effects. You can save the completed program as an ordinary Basic program on a HEXDOS disk (though not on tape), and you can use the same technique for other short machine language routines in hybrid programs, as long as you avoid zeroes in the code.

In summary, the procedure for loading this program is as follows:

Boot up under HEXDOS

Use BREAK or USR(-6) to get to the monitor

Continued on page 5.

```
HEXASM ASSEMBLY LISTING      PAGE  # 2

 740   0079 D0 FB     BNE  PC-4
 750   007B F0 DF     BEQ  SLOOP
 760   007D                      FNDSECT
 770   007D 20 9C FC  JSR  READSK
 780   0080 48        PHA
 820   0081 AA        TAX
 830   0082 A0 01     LDY# 1
 835   0084 88        DEY
 840   0085 20 9C FC  JSR  READSK
 850   0088 91 AD     STA@Y MEMLOC
 860   008A C8        INY
 870   008B D0 F8     BNE  PC-7
 880   008D E6 AE     INCZ MEMLOC+1
 890   008F CA        DEX
 900   0090 D0 F9     BNE  PC-6
 910   0092 68        PLA
 912   0093 A8        TAY
 914   0094 4C D0 AF  JMP  $AFD0
 920   0097

                      .PAGE
                      ; EQUATES FOR HEXDOS ROUTINES

 950   0097           GETVAL = $06A5
 960   0097           CKSEEK = $E6
 970   0097           SEEK   = $04F3
 980   0097           READSK = $FC9C
 990   0097           TRKERR = $0558


                        SYMBOL TABLE

ADDR  SYMBOL                    ADDR  SYMBOL

02FF  BCDTRK                    00E6  CKSEEK
007D  FNDSECT                   06A5  GETVAL
00AD  MEMLOC                    0097  PC
FC9C  READSK                    02FE  SECTOR
04F3  SEEK                      005C  SLOOP
02FD  TRACK                     0558  TRKERR
```

LISTING 2.

```
.T0B00,0B9F
0B00> 00 9E 0B 00 00 8E 20 A5
0B08> 06 8D FD 02 20 01 AC 20
0B10> A5 06 8D FE 02 20 01 AC
0B18> 20 A5 06 85 AD A5 AE 85
0B20> AE AD FD 02 C9 1E 90 04
0B28> 69 11 90 0E C9 14 90 04
0B30> 69 0B 90 06 C9 0A 90 02
0B38> 69 05 8D FF 02 A9 FF 85
0B40> E6 AD FD 02 20 F3 04 E6
0B48> E6 20 9C FC C9 43 D0 F9
0B50> 20 9C FC C9 57 D0 F5 20
0B58> 9C FC CD FF 02 F0 03 4C
0B60> 58 05 20 9C FC C9 76 D0
0B68> F9 20 9C FC CD FE 02 F0
0B70> 12 20 9C FC A8 A2 01 CA
0B78> 20 9C FC CA D0 FA 88 D0
0B80> FB F0 DF 20 9C FC 48 AA
0B88> A0 01 88 20 9C FC 91 AD
0B90> C8 D0 F8 E6 AE CA D0 F9
0B98> 68 A8 4C D0 AF 00 00 00
>
T0079,0080
0079> 01 0B A0 0B A0 0B A0 0B
>
```

LISTING 3.

```
1 P$="      ":P=PEEK(123)+256*PEEK(124)+3
2 BP=PEEK(124)+1-(PEEK(123)>128)
10 INPUT"FILENAME";FL$
15 NL=LEN(FL$)
20 FL$=LEFT$(FL$+"      ",6)
30 POKE240,6:POKE241,11
40 PRINT"INSERT 65D DISK"
50 PRINT"AND PRESS RETURN"
60 IFUSR(0)<>13THEN60
70 T=USR(-7)12,1,BP*256:IFT<>1THENPRINT"ERROR IN DIRECTORY":END
80 T=USR(-7)12,2,(BP+1)*256:IFT<>1THENPRINT"ERROR IN DIRECTORY":
                                                          END
90 FORI=BP*256TOI+511STEP8
92 POKEP,IAND255:POKEP+1,I/256
95 T=LEN(P$)
100 IFP$<>FL$THENNEXT:PRINTFL$;" NOT FOUND":END
110 DEFFNB(X)=(XAND240)/1.6+(XAND15)
120 ST=FNB(PEEK(I+6)):ET=FNB(PEEK(I+7))+1
130 IF2048*(ET-ST)+BP*256>PEEK(133)+256*PEEK(134)THENPRINT"TOO
                                                          BIG":END
140 FORI=STTOET-1:T=USR(-7)I,1,BP*256+2048*(I-ST):NEXT
144 PRINT
146 PRINT"INSERT HEXDOS DISK"
148 PRINT
150 PRINT"NAME OF EXISTING FILE":PRINT"TO SAVE THIS IN";:INPUTFD$
155 LOAD*5,FD$
158 BA=PEEK(BP*256)+256*PEEK(BP*256+1)-(50*256+121)+BP*256
160 FORI=PEEK(578)TOPEEK(579)-1
170 SAVE#I,BA+2048*(I-PEEK(578)):NEXT
```

Continued from page 4.

Enter the block of code in Listing 2

Set the pointers on Page 0

Warm start Basic

Type in the Basic program in Listing 3

Save the resulting program on disk

I hope this satisfies those of you who would like to use some programs published in the 65D format with HEXDOS. Now you can have the best of both worlds!

★

THE NEW 300 SERIES REVIEWED

by; Gary Gesmundo, V.P.
Research & Development
Kalamazoo Software Systems
9703 E. M-80/Box 363
Richland, MI 49083

The new 300 series is here! After spending a week in Bedford, Mass. converting our Data Base Manager(Keybase) to Keybasic, we thought we'd respond to you folks who are out there waiting for a new machine. It took two days to convert about 300K of program from 65-U to Keybasic. Since we did our conversion in Bedford, OSI has written new utilities that will actually do the conversion of your exsisting software as you transfer it from the 65-U machine to the new 300 series. We used Keyword to make the changes because we didn't have the transfer programs available. Keyword was an extremely powerful tool for changing syntax and doing global search and replace routines.

We were able to get all functions up and running and were extremely pleased with the performance of the 300 machine. At the time I was doing the conversion there were two other users on and off the system, but I couldn't tell whether they were there or not; the 300 worked equally fast either way. I was real pleased with the new commands that have been provided in Keybasic, such as the CRT commands that allow you to directly control the terminal from Basic with new Keywords or verbs rather than write subroutines for each function you need from the terminal. The 300 series also allows 16 channels for data files vs. 8, which means larger applications using more files. Next month I hope to write a longer article on the actual steps in conversion and the new features of Basic.

★ ★ ★

CASSETTE CORNER

ASSEMBLY LANGUAGE CAN COEXIST WITH BASIC

by: Harry B. Pye
2406 Hillock Court
Lansdale, PA 19446

The Microsoft BASIC supplied with all OSI Basic-in-ROM systems is really very good. If you compare your system performance with some of the 8080 or Z80 benchmarks that appear from time-to-time in the popular publications, we OSI users are quite competitive. Compare the efficiency of this chip with an 8080 or Z80 and you will find that in most applications the 6502 will get the job done quicker and with more readable code. Consequently, our BASIC runs faster than the BASIC on some of the so called "modern systems".

If BASIC is fast enough for your applications, skip to the next article. But, if you have some programs that just poke along (no pun intended), I will try to show you some simple ways of combining assembly language with BASIC. No big deal you say, we have the "USR" function to conveniently link BASIC to assembly language routines. You are right. There have been a number of articles written on the "USR" function and how it is to be used. I intend to outline two methods for conveniently saving and reloading the necessary code.

METHOD #1 DATA STATEMENTS

BASIC has the ability to "POKE" a value into a specific memory location. When this capability is combined with a "DATA" statement, we have a method of storing assembly language programs within a BASIC program. The decimal values associated with the assembly language subroutine are stored in "DATA" statements. The initialization portion of the BASIC program "READs" through the "DATA"

WEST COAST DISTRIBUTOR

Ohio Scientific LIQUIDATION UP to 60% off

Model	Description	Retail	Cash
	Showroom Demonstrators		
C4PMF	24k w/5" floppy & color	1995	1299
C4PDMF	48k w/dual 5" floppy	2495	1499
CD-3AP	5" disk drv w/pwr sup/cabl	699	350
C8PDF	48K dual 8" floppy/color	3495	1750
C2-OEM	48K 1mhz dual 8" 65U sys	3250	1600
C2-OEM2	48K 2mhz stat mem 65U	3750	1800
C3-OEM	48K 2mhz dual 8" 3 proc	4200	2100
C3-OEM	56k 2/4mhz CP/M compatbl	4400	2200
C3-DTS	56k 10 Mbyte 5 ser ports	9300	4650
C3-D/5	52k 5 Mbyte HD 2mhz	8000	4000
C3-C12	104k 36 MB 2 user 2/1mhz	14900	8950
C3-B33	152k 74 MB 3 user 2mhz	19500	9990
	Brand New Computers		
C3-OEM	56k 2mhz dual 8" 65U 3 proc	4400	2950
C3-DTS	56k 10 M/B HD up to 4 user	9300	5750
C3-C'	52k 36 MB HD 16 slot	13500	8500
C3-D/5	52k 5 Mbyte HD 2mhz - slick	7000	5000
	Accessories and Spare Parts		
CM-2	4k stat ram at D000 L3/CPM	125	49
CM-10	8K stat mem at D000 L3/CPM	200	79
CM-6	48K 1mhz dynam mem board	550	249
CM-9	24k 2mhz stat mem board	450	179
CM-3	16k 2mhz lo pwr stat mem	399	149
CM-11	48K 2mhz stat lo pwr ram	995	499
CA-9	Centr prl ptr intfc w/cabl	235	99
CA-9D	Diablo prl intfc w/ribcabl	200	99
CA-10-1	RS-232 ser intfc w/1st port	200	99
CA-15	Modem/Telephone intfc board	500	299
FD100-8	Siemens 8" disk driv A or B	500	299
PS5-3	5 volt 3 amp power supply	69	29
PS24	24 volt 2.5 amp disk pwr sup	110	49
PS-1	5/12/-9 volt triple pwr supp	270	129
590/5	Hard disk controller pair	600	399
510c	CPU w/3 proc 2/4 mhz	600	299
470b	Floppy disk controller board	175	79

Discounts for quantity purchase/dealers/clubs/schools
Offer limited to qty on hand. Payment by cashiers ck
Demos tested and sold as is. Inspection available.

Toll Free 1-800-854-7165 Call today

SPACE-COM International

22991 La Cadena Drive, Laguna Hills, CA 92653 (714) 951-4648

statements and "POKEs" these values into memory.

In my opinion, this is a horrible waste of memory. The data is stored twice, once in the BASIC program and a second time as assembly language code in another area of memory. Actually, the "waste ratio" is more than two-to-one. The "DATA" statements and "FOR ....NEXT" loop required to load the program will probably occupy three to four times the amount of memory required by the assembly language routine.

As you may have guessed, I do not like "DATA" statements. They served a useful purpose when BASIC was used primarily as a teaching tool. In current applications of the language, they are rarely needed. There are, however, exceptions.

Just so you don't think that I am totally prejudiced, Listing #1 is a BASIC program that will reproduce a segment of memory in the form of "DATA" statements. It also will output the "FOR...NEXT" loop and "READ" statements required to reload the program. The output of this program can be added to a larger BASIC program, but is most useful in loading those short assembly language programs which are often stored in locations from $0222 through $02FF. These locations are not used by BASIC (on a C2/C4 ROM system) and are not modified by a "COLD" or "WARM" start. I use this technique to load the excellent BASIC line editor that was published in the July 1981 MICRO.

The procedure is to "LOAD" the output of the program in Listing #1, "RUN" the program and then execute "NEW". The result - your assembly language program is loaded and the BASIC program is wiped out of memory. Fairly efficient for short programs.

Type the BASIC program shown in listing #1 into your system and "SAVE" it on a cassette. I have attempted to document the listing with numerous "REM" statements. Obviously, these may be omitted. Actually, I prefer to keep two copies of a program. One with "REM's" and spaces for readability, and the other, a working copy with all the nonessentials omitted.

The program will prompt for the starting and ending memory locations, the first line number for the output and the line number increment to be used. Before pressing <CR> following the last input, make sure that the cassette is running and is in the "RECORD" mode. You should see the instructions for your loader program written on the display at the normal 300 baud "SAVE" rate. Remember, all input to this program is in decimal.

I considered adding a hex to decimal converter for the addresses, but decided to keep the program as short as possible. Another addition would be "POKES" to locations 129 and 130 (decimal) to adjust the top of the BASIC work space protecting assembly language programs which are stored in upper memory.

Line #250 outputs the decimal value of the current memory location. The use of "RIGHT$" eliminates the leading space (reserved for a minus sign). Line #320 is used to maintain a realistic line length. The "POS(C)" in line #320 returns the current cursor position and assures that no line will exceed 62 characters in length.

METHOD #2 MONITOR LOADER (OS-65V format)

Another method of loading an assembly language program is with the ROM Monitor "L" command. Unfortunately, OSI did not supply us with a corresponding OS-65V dumper. Aardvark and others advertise replacement ROM's with this capability, but since my system has the standard ROMs, I cannot comment further.

Listing #2 is a BASIC program which dumps portions of memory in the Monitor load (OS-65V) format.

This program has several useful capabilities. It can output several non-contiguous blocks of code, the blocks can be defined to load into a memory area different from the original memory location and the user may specify a self-start address or exit to the monitor as desired.

I found these capabilities mandatory when writing some extensions to the Assembler/Editor. All of the extentions were to be loaded into page zero and a portion of the stack. By using a "memory offset" with the Assembler/Editor, the source code was assembled into upper memory. The program in Listing #2 was used to dump the object code in a form that would load into page zero and page one using the monitor loader. Try that with your replacement ROM!

The program seems to be quite convoluted, but it gets the job done. The output is dumped at 300 baud, so it will load as fast as the monitor is capable of inputting data. As noted earlier, Listing #2 is fully commented. It is expected that the user will delete all "REMs" from the working copy of the program.

Some comments on the program: Line #110 defines the addresses associated with the serial port on a C2/C4. Change these values for a C1 or Superboard. Lines 150 through 215 are a subroutine to convert decimal addresses to their two byte hexadecimal equivalent. This subroutine was put at the beginning of the program to improve the execution speed. Lines 235 through 330 solicit input from the user to determine the memory locations to dump, where this code is ultimately to be loaded and a self-start address if required. Line #350 and on do the actual dump(s). Lines 420 and 445 do another decimal to hexadecimal conversion on the data retrieved from memory. In addition, they build a "STRING" (H$) that is eventually output as three bytes to the cassette.

Incidentally, this is the reason that the monitor loader is so slow. Each byte of memory is output as three ASCII bytes; two data bytes followed by a carriage return character. This means that we have a data rate of slightly less than 100 baud. The "slightly less" is due to the method of recording on the cassette. Eleven bits are stored on the tape for each byte that is output. The eleven bits are comprised of two start bits, eight data bits, and one stop bit. The extra three bits are supplied by the ACIA.

CONCLUSIONS

Two methods of saving and entering assembly language programs have been presented. The first uses "DATA" statements. The output of this program can be used alone for short programs or it can produce output which can be combined with a larger BASIC program. The second method produces tapes which will load through the monitor in OS-65V format. While this program could be used with any application, it is most useful for longer (two or three pages) programs or those that will be loaded into an unusual

area of memory. In all cases, when executing these programs, make sure that the assembly language program is protected from the BASIC dumper program.

I am sure that some of the readers of PEEK(65) can suggest some improvements to these two utility programs. If you have comments or improvements, share them with the other readers. If I have touched on an area that you don't understand, ask a question. You are probably not alone. One answer to one question could possibly help a large number of readers.

LISTING #1

```
100 REM Get the following User Information:
105 REM      -Starting Memory Location (in decimal) Variable-'F'
110 REM      -Ending Memory Location (in decimal) Variable-'L'
115 REM      -Starting Line Number for the Output-Variable 'N'
120 REM      -The Line Number Increment to be Used-Variable 'I'
125 :
130 INPUT "FIRST, LAST, LINE#, INCREMENT";F,L,N,I
135 :
140 REM Turn-On the Cassette Port. ('SAVE' uses less Memory
145 REM                                 than 'POKE 517,255')
150 :
155 SAVE
160 :
165 REM Output the 'FOR....NEXT' Statement
170 :
175 PRINT N "FOR I="F"TO"L":READ A:POKE I,A:NEXT";
180 :
185 REM Step the Line Number Variable 'N' to the next line.
190 REM            Output the Next Line Number
195 REM            Output the Word 'DATA'
200 REM      (Semicolon Prevents Output of Carriage Return)
205 :
210 N=N+I:PRINT:PRINT N "DATA ";
215 :
220 REM Get the Value of the Memory Location to be Output
225 REM      -Convert it to a String Variable (V$)
230 :
235 V$=STR$(PEEK(F))
240 :
245 REM Output the Significant Digits. (The Left-Most Digit
250 REM                                 of 'V$' is a Space)
255 REM       (Semicolon will keep Output on the same Line)
260 :
265 PRINT RIGHT$(V$,LEN(V$)-1);
270 :
275 REM If the Last Character was output, Clear the 'SAVE' Flag
280 REM     and Terminate.
285 :
290 IF F=L THEN POKE 517,0:END
295 :
300 REM Else Increment the Memory Address Pointer & Test
305 REM       if more than 59 Characters Outputted on the Line
310 REM          Then Output a Carriage Return & Start New Line
315 :
320 F=F+1:IF POS(C)>59 THEN 210
325 :
330 REM       Else Print a Comma and Go Back for the Next Digit
335 :
340 PRINT",";:GOTO 235
```

LISTING #2

```
100 REM Define Serial Port Addresses & HEX Constants
105 :
110 A=64512:B=A+1:A$="0123456789ABCDEF
115 :
120 GOTO 235:REM Hop Over the Subroutine
125 :
130 REM Subroutine to Convert Two Byte Addresses to HEX
135 REM Clear Result String & Set Divisor Constant to Maximum
140 REM 'D' is the Input Variable to be Converted
145 :
150 H$="":K=4096
155 :
160 REM Get High-Order Value & Convert to HEX
165 :
170 D1=INT(D/K):D=D-D1*K:D1=D1+48:IF D1>57 THEN D1=D1+7
175 :
180 REM Add the Converted Value to the String 'H$'
185 REM Reduce the Divisor, and Test for the End of Subroutine
190 :
195 H$=H$+CHR$(D1):K=K/16:IF K<1 THEN RETURN
```

```
200 :
205 REM If not the End, Repeat the Sequence
210 :
215 GOTO 170
220 :
225 REM User must Define how many discontinuous blocks to output
230 :
235 INPUT"Enter the number of non-contiguous blocks";N
240 :
245 REM Dimension Variable 'M' for the Number of Blocks times 3
250 :
255 PRINT:PRINT:PRINT:DIM M(N,3)
260 :
265 REM Set-up Loop to get Data for each Block of Data
270 REM We need to know where the Data starts & ends
275 REM     and where it will be stored in memory.
280 REM     This is necessary if the code is Assembled with
285 REM     an Offset.
290 :
295 FOR I=1 TO N:PRINT "Input data for block" I
300 PRINT:INPUT"Beginning & End of Source";M(I,1),M(I,2)
305 PRINT:INPUT"Destination";M(I,3):PRINT:PRINT:NEXT
310 :
315 REM If the Program is to Self-Start, get the Starting Address
320 :
325 INPUT"Is program to self-start";SS$:PRINT
330 IF LEFT$(SS$,1)="Y" THEN INPUT"Self-Start Address";SS
335 :
340 REM Set the System Flag to 'SAVE'
345:
350 POKE 517,255
355 :
360 REM Set up a Loop for the Number of Blocks to be Output
365 REM     Convert the Starting Address to HEX
370 :
375 FOR I=1 TO N:D=M(I,3):GOSUB 150
380 :
385 REM Set 'Address' mode & Output Starting Address
390 REM     Then set 'Data' Mode & Set Loop to Output Data
395 :
400 PRINT "." H$ "/";:FOR J=M(I,1) TO M(I,2)
405 :
410 REM 'PEEK' Data from Memory & Convert to two Nibbles
415 :
420 D=PEEK(J):LO=D AND 15:HI=(D-LO)/16
425 :
430 REM Build a String 'H$' with the ASCII of the two Nibbles
435 REM    plus a 'Carriage Return' symbol
440 :
445 H$=MID$(A$,HI+1,1)+MID$(A$,LO+1,1)+CHR$(13)
450 :
455 REM Set up loop to Output the three (3) characters
460 REM    Character will be converted to it's ASCII equivalent
465 REM    Test to see if the ACIA output port is ready
470 REM    and send the character to tape via the ACIA
475 :
480 FOR K=1 TO 3:V=ASC(MID$(H$,K,1)):WAIT A,2:POKE B,V
485 :
490 REM Now bump the indices for the loops
495 :
500 NEXT K,J,I
505 :
510 REM All Data blocks have been output-Test for Self-Start
515 REM    If a self-start is requested, convert it to HEX
520 :
525 IF LEFT$(SS$,1)="Y"THEN D=SS:GOSUB 150:GOTO 580
530 :
535 REM No self-start address input, Use 'Monitor' start address
540 :
545 D=65024:GOSUB 150
550 :
555 REM In either case---
560 REM    Set 'Address' Mode
565 REM    Output starting address
570 REM    Output a 'G' to start execution
575 :
580 PRINT "." H$ "G"
585 :
590 REM Clear the System 'SAVE' flag before ending
595 :
600 POKE 517,0
```

★ ★ ★

THE GENERIC 'MEM+' BOARD

By: David T. Sigafoos
P.O. BOX 19024
Portland, OR 97219

The system that I am using presently is an OSI C3-S1 with a Generic Computer Products 'MEM+' board with the parallel printer port, and real time clock installed (hopefully by this time I will have the disk controller configured for a 5 1/4 inch disk which with some tri-state logic will allow me to access either 5 1/4 or 8). Along with the standard serial terminal (ADM-3a) I am running the 540b video board by OSI with keyboard. The reason that I have included the 540b is that I am using the Dwo Quong WP6502 processor and I do not have lower case on my ADM.

HISTORY

The first commercial board was brought by Bob, a fellow member, to the OSI-Northwest group. The board was installed into his C4P-DMF. This allowed the user to remove the POWER-HUNGRY 527 memory board by OSI. His board was configured for 48K, printer port, and real-time clock. To this day, he has not had a problem with any of the functions on the board.

Around September I received my board with 56k, printer port, and real-time clock. This allowed me to replace my 3 530 boards and go to 2 meg operating speed without all of the phony mod the 530's require.

I had, in the beginning, some troubles with the RTC keeping track of the days. Even with my machine being the only one with this problem, Bob spent a large amount of time to diagnose the problem. There seemed to be a problem in the powerdown to the RTC, not caused by the board design, but inherent with the N.S. chip. Bob proceeded to update the etch to include the necessary changes to correct the problem. Since my board has been modified, I have not had a single problem with the time.

SPECS:

The board itself can hold 64k of low power ram, a floppy disk controller (5 1/4 or 8 inch), a real-time calendar clock, and a Centronics compatible parallel printer interface. There is a micro hex switch which is used to specify users in a multi user environment. The board is a 2 layer assembly with silk screen of components over a dark green solder mask. Not only does the board look professionally done, but anyone who has built many boards knows the importance of a solder mask.

The memory is populated with Augat machine screw sockets for extreme reliability. There is also a provision on the board to allow for power-on reset. This is accomplished with a wire added to the CPU board.

Of course, the memory is fully compatible with the 2716 Eproms on the market.

CONCLUSIONS:

One of the main concerns a person has when he/she is about to invest a large amount on upgrading a computer is not only whether the board works, but what is the documentation like. I have spent the last 13 years in manufacturing of 'STATE OF THE ART' computer systems, and I have found that the major down-fall to any product is its documentation. I would like to relieve the fears anyone may have by stating that I find the documentation to be first class. Included with the circuit description are reduced 8 1/2 x 11 schematics. These schematics have been reduced from 'c' size inked originals, and if you have never seen the results of such a process you are in for a real treat. The documentation package also includes the listings for the RTC chip (read and write), and you can also get the assembly listing of the update needed by the printer routine.

These listings allow the user to investigate the workings of the hardware that you spend so much on. I find this to be a blessing in light of the amount of information that you get from either OSI or D&N.

I have been using this board for about 5 months solid and have no problems (other than the original RTC problem) with this board. With the board being able to handle a full 64k of ram this will allow me to get the Proxy-80 board and use CPM to its full advantage (it needs all the help it can get).

There is one problem that may bother people, and this deals with the printer port. The OSI design used a 6820 PIA for

their printer port, and Bob uses a 6522 VIA in his. The difference is in the way each device is programmed. Bob sends a disk (either 5 1/4 or 8 inch) with the order to do a simple patch in DOS (either 65D or 65U). I have not had any problems with the patch even though I have installed it on 65D 3.0, 3.2, 3.3, and 65U 1.0, 1.2, 1.3, 1.42. I have also installed the patch on my Dwo Quong WP65D and WP65U.

I would recommend this board to anyone needing more/better memory, and/or disk controller, RTC or printer port. This board has proven itself to me to be a real bargain.

We have installed three of these boards in one of the new 250J systems that I specified to a tire dealership in Hillsboro running in a multi-user environment. There has, as yet, been no problems in the boards at all.

As a final note: Bob also has a few new projects in his bag of treats (no tricks) coming, dealing with an operating system for the assembly programmer, a graphics add on, and at present I am testing a printer queue routine which allows a single user system to be printing a listing while you go back and start either writing more code or changing the code, or whatever you wish to do.

★ ★ ★

AN IMPROVED COLD-START ROUTINE FOR BASIC IN ROM

by: Gerdt Vilholm
Prinsessagade 4B,St.
DK 1422 Copenhagen K
Denmark

The Coldstart routine shown here gives some advantages over the original Coldstart. I have deleted some superflous text (apologies to Mr. Weiland) and made some extra features.

On Coldstart, you see on the screen:

```
MEMLO?

MEMHI?

WIDTH?

xxxx BYTES

BASIC

OK
```

By responding with a number to MEMLO, you can set Basic Sourcecode to start anywhere in memory. This makes it possible to run a Basic program while other programs, such as EXMON or Assembler are in the machine. If you respond with CR, MEMLO is set to 0300 hex.

MEMHI is the usual MEMORY SIZE.

You can give a hex number, such as &23AB in response to both MEMLO and MEMHI. Note, that I use "&" as the hex-prefix to avoid confusion with string-variable names.

One more feature: When Basic crashes, there is a chance that the Sourcecode and its pointers are still OK. In this case you can do a Coldstart and answer R (CR) to MEMLO. This will reset most of page zero, but retain the Sourcecode. It works when EXMON clobbers Basic.

Coldstart still starts at addr. BD11, but the first part through BD70 is not changed.

You may wonder why the hex to binary conversion routine at BE64 takes the DEF-token into account. It is because this is a general conversion routine, which can be used to convert hex numbers embedded in Basic-Sourcecode, where the tokenize-routine has converted DEF to 95 hex.

Listing starts on page 12

IMPROVED COLDSTART FOR BASIC IN ROM
ADDR.BD11-BD70 ARE UNCHANGED

```
BD71 A9AD   LDAIM   AD
BD73 A0BE   LDYIM   BE
BD75 20C3A8 JSR     A8C3  Print "MEMLO
BD78 2046A9 JSR     A946  INPUT
BD7B 86C3   STXZ    C3    Set Pointer
BD7D 84C4   STYZ    C4
BD7F 20BC00 JSR     00BC  Get 1st. ch.
BD82 C952   CMPIM   52    Is it "R"
BD84 D003   BNE     BD89
BD86 4C41BE JMP     BE41  Then retain prog.
BD89 A8     TAY           Is it CR
BD8A D004   BNE     BD90
BD8C A003   LDYIM   03    Then set to 0300
BD8E D00A   BNE     BD9A  Branch always
BD90 2057BE JSR     BE57  Fetch number
BD93 A8     TAY           Valid number?
BD94 D0DB   BNE     BD71
BD96 A511   LDAZ    11    Set MEMLO
BD98 A412   LDYZ    12    pointer
BD9A 8579   STAZ    79
BD9C 847A   STYZ    7A
BD9E A9B3   LDAIM   B3
BDA0 A0BE   LDYIM   BE
BDA2 20C3A8 JSR     A8C3  Print "MEMHI
BDA5 2046A9 JSR     A946  INPUT
BDA8 86C3   STXZ    C3    Set pointer
BDAA 84C4   STYZ    C4
BDAC 20BC00 JSR     00BC  Get 1st. ch.
BDAF A8     TAY           Is it CR?
BDB0 D021   BNE     BDD3
BDB2 A579   LDAZ    79    Then check
BDB4 8511   STAZ    11    MEMsize
BDB6 A57A   LDAZ    7A
BDB8 8512   STAZ    12    Check all
BDBA A000   LDYIM   00    RAMbytes from
BDBC E611   INCZ    11    MEMLO and up
BDBE D002   BNE     BDC2
BDC0 E612   INCZ    12
BDC2 A992   LDAIM   92
BDC4 9111   STAIY   11
BDC6 D111   CMPIY   11
BDC8 D00F   BNE     BDD9
BDCA 0A     ASLA
BDCB 9111   STAIY   11
BDCD D111   CMPIY   11
BDCF F0EB   BEQ     BDBC  When nonvalid byte
BDD1 D006   BNE     BDD9  Then branch
BDD3 2057BE JSR     BE57  Fetch MEMHI number
BDD6 A8     TAY           valid number?
BDD7 D0C5   BNE     BD9E
BDD9 A511   LDAZ    11    Set MEMHI pointer
BDDB 8585   STAZ    85
BDDD 8581   STAZ    81    and High String
BDDF A512   LDAZ    12    pointer
BDE1 8586   STAZ    86
BDE3 8582   STAZ    82
BDE5 A9B9   LDAIM   B9
BDE7 A0BE   LDYIM   BE
BDE9 20C3A8 JSR     A8C3  Print "WIDTH
BDEC 2046A9 JSR     A946  INPUT
BDEF 86C3   STXZ    C3
BDF1 84C4   STYZ    C4
BDF3 20BC00 JSR     00BC  Get 1st. ch.
BDF6 A8     TAY           Is it CR?
BDF7 F01C   BEQ     BE15  Then default
BDF9 207FA7 JSR     A77F  Else compute
BDFC A512   LDAZ    12    Terminal Width
BDFE D0E5   BNE     BDE5
BE00 A511   LDAZ    11
BE02 C910   CMPIM   10
BE04 90DF   BCC     BDE5
BE06 850F   STAZ    0F
BE08 E90E   SBCIM   0E
BE0A B0FC   BCS     BE08
BE0C 49FF   EORIM   FF
BE0E E90C   SBCIM   0C
BE10 18     CLC
```

Listing continued on page 14

OHIO SCIENTIFIC, Inc.

With our new management team, OSI is proud to announce the addition of the **KeyFamily 300** series —

MULTI-PROCESSING BUSINESS SYSTEMS

to our complete line of 200 series timesharing business computers. Utilizing state-of-the-art microprocessor technology OSI now offers the highest performance microprocessor based business system available. Each user has his own Z80A 4MHZ CPU, 64K memory, 4 channel DMA and two serial ports. A system master processor with a separate CPU, 56K of memory, 4 channel DMA and 2 serial ports handles all disk and system I/O tasks. Our separate, proprietary, 8 Megabit inter-processor communications bus provides nearly instantaneous inter-processor data transfers. Running OSI's proprietary version of the KeyOperator-1 Multiprocessing operating system allows most of the over 3000 CP/M based packages to run together with OSI's ...

KEYBASIC Version 2.0

KeyBasic 2.0 is the 65U BASIC version 1.43 compatible SUPER-BASIC language, the culmination of **your** input on 65U extensions and has many, many features unavailable in any other language. These include;

• Enhanced Extended Input • Character oriented Disk I/O • FIND command with limit • CRT Command • SWAP • WHILE WEND • KILL MultiByte to MultiByte input translation • Semaphore WAIT FOR with time limit • Enhanced Extended Output • Key Map • RANDOMIZE • TIMER • Selectable Dynamic File Allocation • RESUME • Invisible SPOOLING on 1 to 16 Queues onto 1 to 16 printers • **Record Locking** • Extended EDITOR • 4 types of Program Chaining with COMMON Verb • Up to 15 Disk Channels with individual buffers • Subroutine CALL • SuperTrace • TIME • DATE • RENAME • INSTR$ • Delete, Resequence and Renumber In Basic • PRINT USING • ON TIMER GOTO • ! and !! editor commands • ON ERROR GOTO • ERASE (delete file) • OPEN (creates file) • FIX • 16 Digit Precision • DEV$

The KeyFamily 300 series will initially be available in 4 models, the 10MB 330E and 40MB 330I (up to 4 users) and the 350J/JJ (up to 8 users). These systems will include **KeyOperator-1**, **KeyWord** Word Processing System and **KeyBasic**.

ORDER YOUR SYSTEMS NOW!!!

from your dealer or

OHIO SCIENTIFIC, Inc.
6515 Main Street
Trumbull, CT 06611
(203) 268-3116

Listing continued from page 12

```
BE11 650F    ADCZ    0F  and comma
BE13 8510    STAZ    10  column width
BE15 A000    LDYIM   00
BE17 98      TYA         Put 00 in
BE18 9179    STAIY   79  1st. source
BE1A E679    INCZ    79  and increase
BE1C D002    BNE   BE20  pointer
BE1E E67A    INCZ    7A
BE20 A579    LDAZ    79
BE22 A47A    LDYZ    7A
BE24 201FA2  JSR   A21F  Check free MEM
BE27 206CA8  JSR   A86C
BE2A A585    LDAZ    85
BE2C 38      SEC
BE2D E579    SBCZ    79
BE2F AA      TAX
BE30 A586    LDAZ    86
BE32 E57A    SBCZ    7A
BE34 205EB9  JSR   B95E  Print number
BE37 A9BF    LDAIM   BF
BE39 A0BE    LDYIM   BE
BE3B 20C3A8  JSR   A8C3  Print "BYTES
BE3E 2063A4  JSR   A463  Do New
BE41 2077A4  JSR   A477  Do CLEAR
BE44 A9C3    LDAIM   C3  Set OK pointer
BE46 8504    STAZ    04
BE48 A9A8    LDAIM   A8
BE4A 8505    STAZ    05
BE4C A974    LDAIM   74  Set WARM pointer
BE4E 8501    STAZ    01
BE50 A9A2    LDAIM   A2
BE52 8502    STAZ    02
BE54 4C0000  JMP   0000  WARM START
BE57 20C200  JSR   00C2  GET 1st. ch.
BE5A C926    CMPIM   26  Is it "&"?
BE5C F006    BEQ   BE64  Then get Hex
BE5E 20C200  JSR   00C2  Else get decimal
BE61 4C7FA7  JMP   A77F
BE64 A900    LDAIM   00  Convert hex-string
BE66 8511    STAZ    11  pointed to by C3-4
BE68 8512    STAZ    12  to binary in 11-12
BE6A 20BC00  JSR   00BC  Get 1st. ch after &
BE6D C995    CMPIM   95  Is it DEF-token?
BE6F D011    BNE   BE82
BE71 A90D    LDAIM   0D  Then make D-E-F
BE73 209ABE  JSR   BE9A
BE76 A90E    LDAIM   0E
BE78 209ABE  JSR   BE9A
BE7B A90F    LDAIM   0F
BE7D 209ABE  JSR   BE9A
BE80 F0E8    BEQ   BE6A  Branch always, next ch.
BE82 38      SEC         Conv. hexch. to bin.
BE83 E930    SBCIM   30
BE85 3022    BMI   BEA9  If less than zero
BE87 C90A    CMPIM   0A
BE89 300A    BMI   BE95  If 0-9
BE8B C911    CMPIM   11
BE8D 301A    BMI   BEA9  If between 9-A
BE8F C917    CMPIM   17
BE91 1016    BPL   BEA9  If greater than F
BE93 E906    SBCIM   06  It was A-F
BE95 209ABE  JSR   BE9A  Shift number
BE98 F0D0    BEQ   BE6A  Branch always, next ch.
BE9A 0A      ASLA        Shift binary number
BE9B 0A      ASLA        in Ac into
BE9C 0A      ASLA        11-12
BE9D 0A      ASLA
BE9E A004    LDYIM   04
BEA0 0A      ASLA
BEA1 2611    ROLZ    11
BEA3 2612    ROLZ    12
BEA5 88      DEY
BEA6 D0F8    BNE   BEA0
BEA8 60      RTS
BEA9 20C200  JSR   00C2  Finished.  Get ch.
BEAC 60      RTS         following hexnumber.
```

Continued on page 15

Continued from page 14

```
BEAD 4D454D Coldstart MEMLO
BEB0 4C4F00 texts
BEB3 4D454D           MEMHI
BEB6 484900
BEB9 574944           WIDTH
BEBC 544800
BEBF 204259           BYTES
BEC2 544553
BEC5 0D0A0A           CR LF LF
BEC8 424153           BASIC
BECB 494300
```

```
10 REM DIRECTORY UTILITY FOR OS-65D VERSION 3.0
12 REM MODIFIED 7/81 BY D.B.BAKER
14 REM  TO PRINT VOLUME ID IN DIRECTORY HEADING.
16 REM  VOLUME ID MUST BE FIRST ENTRY IN DIRECTORY,
18 REM  WITH TRACK RANGE '00-00'.
20 REM
30 NF=0
40 PN=;1897
50 DEF FNA(X)=10*INT(X/16)+X-16*INT(X/16)
80 DV=1 : Y=1 : X=PEEK(8994)
90 IF X<Y THEN 110
100 DV=DV+1 : Y=Y+Y : GOTO 90
110 PRINT "LIST ON LINEPRINTER INSTEAD OF DEVICE #";DV;
120 INPUT A$
128 REM DV=1 MODIFIED FOR C1P 7/81 DBB.
130 IF MID$(A$,1,1)="Y" THEN DV=1
10000 REM
10010 REM PRINT A DIRECTORY OUT
10020 REM
10025 FL=0
10030 PRINT @DV : PRINT @DV,"OS-65D VERSION 3.0"
10035 PRINT @DV, "-- DIRECTORY ";
10037 GOSUB 11150
10039 PRINT @DV, "--"; PRINT @DV
10040 PRINT @DV,"FILE NAME    TRACK RANGE"
10050 PRINT @DV,"------------------------"
10060 DISK ! "CA 2E79=12,1"
10070 GOSUB 11000
10080 DISK ! "CA 2E79=12,2"
10090 GOSUB 11000
10130 PRINT @DV : PRINT @DV,NF;"ENTRIES FREE OUT OF 64"
10135 PRINT @DV
10140 END
11000 REM
11010 REM READ DIRECTORY OUT OF BUFFER INTO ARRAYS
11020 REM
11040 FOR I=PN TO PN+248 STEP 8
11050 IF PEEK(I)=35 THEN NF=NF+1 : GOTO 11130
11060 N$=""
11070 FOR J=1 TO I+5
11080 N$=N$+CHR$(PEEK(J))
11090 NEXT J
11100 PRINT @DV,N$;TAB(12);FNA(PEEK(I+6));TAB(16);"-";
11110 PRINT @DV,TAB(17);FNA(PEEK(I+7))
11130 NEXT I
11140 RETURN
11150 REM READS FIRST ENTRY IN DIRECTORY
11160 REM IF TRACK RANGE IS '00-00', THEN ENTRY
11170 REM IS A VOLUME ID.
11180 REM IF NOT, STANDARD OS65D DISKETTE.
11190 DISK ! "CA 2E79=12,1"
11200 FOR I=PN+6 TO PN+7
11210 IF PEEK(I)<>0 THEN RETURN
11220 NEXT
11230 N$=""
11240 FOR I=PN TO PN+5
11250 N$=N$+CHR$(PEEK(I))
11260 NEXT
11270 PRINT @DV, "FOR "+N$+" ";
11280 RETURN
```

```
OS-65D VERSION 3.0
-- DIRECTORY FOR SYS@@1 --

FILE NAME    TRACK RANGE
------------------------
SYS@@1       0  - 0
OS65D3       0  - 12
BEXEC$       14 - 14
SCRTCH       25 - 29
VOLID        30 - 31
CHANGE       15 - 16
CREATE       17 - 19
DELETE       20 - 20
DIR          21 - 21
TRACE        22 - 22
ZERO         23 - 24
MODEMA       36 - 36
MODEMS       35 - 35

51 ENTRIES FREE OUT OF 64
```

```
100 REM VOLUME IDENTIFICATION UTILITY
110 REM  D.B.BAKER  7/81
120 REM
130 REM THIS PROGRAM WRITES A "VOL-ID"
140 REM  AS THE FIRST ENTRY OF
150 REM  A DISK'S DIRECTORY
160 REM
170 REM THE FORMAT IS: 6 CHAR VOLUME ID
180 REM  WITH A TRACK RANGE OF "00-00"
190 REM
200 DV=2:REM SET TO VIDEO OUTPUT
210 PRINT@DV,"-VOLUME IDENTIFICATION-"
220 PRINT@DV,"      - UTILITY -"
230 PRINT@DV,"-----------------------";PRINT@DV
240 PRINT@DV,"IS THE DISK INITIALIZED"
250 PRINTTAB(5)"(Y/N)";:INPUTA$
260 IF LEFT$(A$,1)="N"THEN PRINT@DV:GOSUB620
270 PRINT@DV,"DOES THE DISK HAVE"
280 PRINT@DV,TAB(5)"A DIRECTORY(Y/N)";:INPUT A$
290 IF LEFT$(A$,1)<>"Y" THEN GOSUB 660
300 DISK!"CA 41@@=12,1":DISK!"CA 42@@=12,2"
310 REM READ LAST ENTRY
320 IF PEEK(17144)<>35 THEN GOSUB 560
330 REM LAST ENTRY IS OPEN, CONTINUE
340 REM WRITE 2ND SECTOR BACK TO DISK,
350 REM  OFFSET BY ONE ENTRY.
360 DISK!"SA 12,2=41F8/1"
370 REM SOLICIT VOLUME ID
380 PRINT@DV,"ENTER VOLUME ID,"
390 PRINT@DV,TAB(5);"6 CHAR, MAX";:INPUT A$
400 LG=LEN(A$):IF LG>6 THEN390
410 REM WRITE ID TO BUFFER
420 FST=16631:LST=16637
430 FOR I=1 TO 6:POKE FST+I,32:REM ASCII BLANKS
440 NEXT
450 FOR I=1 TO LG:POKE FST+I,ASC(MID$(A$,I,1))
460 NEXT
470 REM WRITE SPECIAL TRACK RANGE
480 POKE 16638,0:POKE 16639,0
490 REM WRITE 1ST SECTOR TO DISK
500 DISK!"SA 12,1=40F8/1"
510 PRINT@DV,"THE DISK DIRECTORY"
520 PRINT@DV," HAS BEEN WRITTEN"
530 PRINT@DV," WITH A VOLUME ID OF"
540 PRINT@DV," -- ";A$;" --"
550 END
560 REM DIRECTORY FULL, CANNOT ADD
570 REM  1ST ENTRY.
580 REM
590 PRINT @DV,"THE DIRECTORY IS FULL --"
600 PRINT @DV," CANNOT ADD THE VOLUME ID."
610 END
620 REM INITIALIZATION
630 REM
640 DISK!"INIT":PRINT@DV
650 RETURN
660 REM WRITES A DUMMY DIRECTORY
670 REM  ON A NEWLY INITIALIZED
680 REM  DISK. WILL ALSOOVER-WRITE
690 REM  ANY EXISTING DIRECTORY!
700 REM
710 AD=16639
720 PRINT@DV,"EXISTING DIRECTORY WILL"
730 PRINT @DV," BE OVER-WRITTEN ! "
740 PRINT@DV,"CONTINUE (Y/N)";:INPUT A$
750 IF LEFT$(A$,1)<>"Y" THEN END
760 FOR I=1 TO 256
770 POKE AD+I,35:REM FILL BUFFER W/NULL
780 NEXT
790 DISK!"SA 12,1=41@@/1":DISK!"SA 12,2=41@@/1"
800 REM WRITE DIRECTORY ENTRY TO
810 REM  DIRECTORY. (DIR$)
820 REM
830 DISK!"CA 41@@=12,1"
840 FOR I=0 TO 5
850 POKE16640+I,ASC(MID$("DIR$  ",I+1,1))
860 NEXT
870 POKE16646,12:POKE16647,12
880 DISK!"SA 12,1=41@@/1"
890 RETURN
```

ADDING VOLUME IDENTIFICATION TO OS65D Diskettes

By: Donn Burke Baker
3128 Silver Lake Road
Minneapolis, MN 55418

Not long after I added a disk to my C1P system, I ran headlong into one of life's little mysteries..."You never have enough disk space, and even if you did, you couldn't keep track of what you had there."

OS65D provides a "directory" utility that will list the files contained on any given disk. The difficulty is that the directory is based upon the DRIVE in use, not the disk. It is easy to confuse listings, especially when several disks are in use.

I mentioned my problem to Leroy Erickson, who had an immediate answer... "Put a Volume ID on each disk. That way, when you do a list, its for that disk. Someday we'll have software to handle it."

Being somewhat impatient of nature, I decided to write my own software for the Volume-ID oriented directory listing. The format of the Volume-ID is the same as a standard 'file-name', with the 'track-range' set to '00-00'. The Volume-ID MUST be the first entry in the directory to be considered as the Volume-ID. These changes to the standard directory utility will work for either OS-65D formatted directories, or the 'Volume-ID' format.

The second program provides a utility which will generate the Volume-ID fields for a disk. If the disk is new, it will initialize it, write a directory, request a file name, and write it as the first entry.

Error conditions checked for include a full directory (unlikely), and a Volume-ID that is longer than six characters.

Two words of caution! The utility uses the standard 'INIT' routine. Don't answer 'Y' when asked "Are You Sure" if you don't want the disk initialized. Also, if you say that no directory exists on the disk, anything that is there will be overwritten. You could lose files if you're careless.

★ ★ ★

LETTERS

ED:

Enclosed is the new errata sheet I received with my C8P Manual (essentially the same manual as the C4P's). It contains corrections for nearly all the typo's and otherwise erroneous information that create confusion, disgust, and lack of confidence in the entire manual (or in one's self in the case of us trusting but ignorant rookies who try to learn from it).

Here are some OS65D V3.3 tidbits I've learned the hard, time-consuming way that were not addressed in either the manual or B.I.T.'s supplement. Maybe they'll save someone else from wasting their time discovering them.

The FIND command doesn't work in Buffer/Device #7.

Comma's used as data separators and Carriage Return's count as characters when figuring the record length you want to set up for a Random file, i.e., a record with eight fields entered into a file setup for '32 characters' will actually only hold 24 data characters (32 char. record length-7 commas-1 CR=24 data char's).

If an error occurs within a subroutine while the TRAP command is activated, the RETURN address is lost. Therefore, trying to branch directly back into the routine will result in an 'RG' error when it tries to RETURN to the main program. One way to get around this without a major disruption is to setup your TRAP's goto line with a GOSUB (the one that triggered the TRAP):GOTO(the line following the original GOSUB).

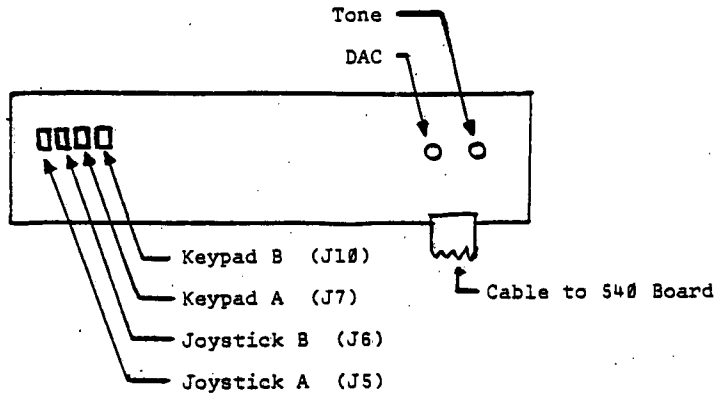
```
ie, 10   GOSUB 100
    20   INPUT#6,A$
    30   . . .
   100   TRAP 1000:DISKOPEN,
         #6,A$:TRAP0
  1000   PRINT" The drive is
         not ready or "A$"is
         not on that disk.
         Fix, then hit any
          key and (RETURN)"
  1010   INPUT z$:GOSUB 100:
         GOTO 20
```

A TRAP'd error will not disrupt the count if in a FOR/NEXT loop. Thus, you can correct or circumvent the cause of the error and branch right back into the loop's remaining iterations.

C3P USERS MANUAL ERRATA

(C4P HAS THE SAME ERRORS)

Page Number	Correction
3	Some keyboards have J5, J6, J7, and J10 on the back of the keyboard instead of on the computer, therefore, eliminating the A-15 cable.



28	Line 80 in the program should read: POKE TUNES,1 not POKE TUNES,I
30	The second address in the list should be 0301 not 0300.
35	In the description of key labels: "SL" in number one should be SHIFT LOCK.

Page Number	Correction
58	The next page contains a sample transmit and receive program for modem operations.

Universal Modem Program

This is a BASIC program which will set up a machine code modem routine designed for use with a standard modem (with RS-232). The routine will operate with the modem ports on the Ohio Scientific C1P, C4P and C8P computers. The 630 and UTI board modem ports are exceptions to this and are not supported by this routine.

This is basically a dumb terminal routine with only two local commands:

CONTROL-D - Toggles the output back and forth between Full and Half duplex mode. (Sometimes echoed as a comma.)

CONTROL-B - Returns to BASIC if the routine is operating on a cassette system, or runs BEXEC* if it is operating on a disk system, effectively terminating the call.*

Shift-O is still used to output a delete character code. Since ROM BASIC doesn't process a backspace, the previous character will be omitted from the text but not on the video screen. The delete

Continued on page 18

TRAP 0 is not necessary if a new TRAP is defined. Each new one simply overwrites the current one.

I've yet to come across a PRINT statement that needs all those ";'s" shown in the manuals. Being basically lazy and RAM-frugal, I never bother and haven't hit a snag...so far.

Typing in programs completely in lower case aids in catching missing ""'s",":'s", and variable names which include key words. Since any of these conditions can easily be spotted on a LISTing due to letters not being capitalized where they should and vice versa.

When setting up a window, don't exceed x-2 & y-2 for your values in the !(22, _, _) command, where x=64-CURSOR position & y=24-CURSOR position.

Sorry, that's the best I can do. For example, the largest window you can define is !(22,62,22) ; you lose one row and column for the zero one and another for ??. For example, say you wanted a window defined as the lower half of the screen, you move the cursor to row 12 and column 0 to establish the upper left hand corner and PRINT!(22,62,10);. Anything larger than these dimensions would cause the command to be ignored completely and the window would never be formed.

There's a lot more they didn't tell us or told incorrectly, but these are the major ones that have wasted my time. Sure hope I've saved some of you the grief I went thru. For a complete list of 65V3.3 hints, omissions, and typo's I've discovered, send a buck and a S.A.S.E.

Since I, by no means have all the answers, how about some answers to the following, if any of you can help:

1) How can you get out of a small, 'secondary' window without wiping out the entire screen?

2) Has anyone redone the screen-to-printer dump to work with an MPI, specifically the 88G (graphics dot matrix model)?

3) How can you add or update a Sequential file short of INPUTing everything, making the additions or changes to the variables, and then PRINTing

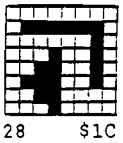
code will be displayed as a graphic backspace, a forward space and another graphic backspace on the ROM BASIC computers.

NOTE: If this program is run on a disk system, create two buffers using the change utility before entering the program.

*You must physically hang up the telephone to complete call termination.

```
10 REM MODEM PROGRAM
20 FORI=1TO30:PRINT:NEXT:PRINT"MODEM ROUTINE LOADING"
30 Y=PEEK(2):Z=PEEK(64774)
40 IFZ=32THENGOSUB3000:GOTO60
50 GOSUB4000
60 FORI=1TO32:PRINT:NEXT:PRINT"MODEM READY"
70 X=USR(X)
80 RESTORE:GOSUB500:IFY=4THENRUN"BEXEC*"
90 END
500 PS=1:IFPEEK(9800)=32THENPS=2
510 IFY<>40RZ<>32THENFORI=1TO48:READP:NEXT:RETURN
520 READP,C(1),C(2):IFPTHENPOKEP,C(PS):GOTO520
530 RETURN
540 DATA 9730,8,16
550 DATA 9743,7,15
560 DATA 9723,31,63
570 DATA 9736,31,63
580 DATA 9725,4,10
590 DATA 9738,29,59
610 DATA 9800,32,64
620 DATA 9636,101,75
630 DATA 9766,101,75
640 DATA 9770,101,75
650 DATA 9815,101,75
670 DATA 9670,125,123
680 DATA 9783,125,123
690 DATA 9682,95,164
990 DATA55296,0,1,0,0,0
1500 FORI=0+FTO216+F:READX
1510 IFX=-1THENX=INT(I/256)
1520 POKEI,X:NEXT
1530 RETURN
2000 DATA 32,13,37,173,0,240,74,144,6,173,1,240,32,67,35
2010 DATA 32,93,-1,240,239,201,2,240,22,201,4,240,21,72,32
2020 DATA 67,35,173,0,240,74,74,144,249,104,141,1,240,76,37
2030 DATA -1,76,13,37,173,63,-1,73,12,141,63,-1,208,225,138
2035 DATA 72,152,72
2040 DATA 169,1,32,190,252,32,198,252,208,5,10,208,245,240,83
2050 DATA 74,144,9,42,224,33,208,243,169,27,208,33,32,200,253
2060 DATA 152,141,19,2,10,10,10,56,237,19,2,141,19,2,168,138
2070 DATA 74,240,69,136,200,74,144,252,208,42,234,185,207,253,205
2080 DATA 21,2,208,38,206,20,2,240,43,160,5,162,200,202,208,253
2090 DATA 136,208,248,240,67,201,1,240,53,160,0,201,2,240,54,160
2100 DATA 192,201,32,240,48,169,0,141,22,2,141,21,2,169,2,141
2110 DATA 20,2,208,36,162,150,205,22,2,208,2,162,14,142,20,2
2120 DATA 141,22,2,169,1,32,190,252,32,207,252,74,144,3,76
2130 DATA 143,253,208,194,160,32,76,167,253,169,0,76,183,253
3000 GOSUB500
3005 IFY=4THENPOKE574,34:POKE575,66:F=16930:GOTO1500
3008 F=546:GOSUB1500
3010 POKE546,44:POKE592,96
3020 POKE559,251:POKE560,2:POKE576,251:POKE577,2
3030 POKE763,41:POKE764,127:POKE765,76:POKE766,45:POKE767,191
3040 POKE11,34:POKE12,2:RETURN
4000 GOSUB3000
4010 POKEF+65,141:POKEF+66,0:POKEF+67,223
4020 POKEF+68,174:POKEF+69,0:POKEF+70,223
4030 POKEF+193,141:POKEF+194,0:POKEF+195,223
4040 POKEF+196,173:POKEF+197,0:POKEF+198,223
4050 POKEF+1,68:POKEF+2,38
4060 POKEF+47,68:POKEF+48,38
4070 POKEF+5,252:POKEF+11,252:POKEF+34,252:POKEF+42,252
4080 IFY=4THENPOKE63235,52:POKE64512,2
4090 RETURN
```

CONTINUED ON PAGE 19

Page Number	Correction
59	Prior to using DISK!"IO ,Ø1" or DISK!"IO ,Ø3", one must POKE 63235 with either a 6Ø for the printer port or 52 for the modem port to avoid system problems.
64	The 48 PIA lines range from 5Ø948 to 5Ø959 (C7Ø4 to C7ØF hex) not 5Ø948 to 5Ø958 (C7Ø4 to C7ØE hex)
69	For an explanation of number 8 under troubleshooting, see pages 78-8Ø besides the BASIC User's Manual.
73	New disks are initialized not initiated.
74	The caption should read: 8 INCH FLOPPY DISK
76	ABS(X) FOR X>=Ø ABS(X)=X FOR X<Ø ABS(X)=-X
78	>=,=> B=>A B greater than or equal to A
78	NOT IF NOT(B<>A) THEN 7 If B=A then 7
79	ID Illegal Direct: INPUT or DEF statements cannot be used in direct mode.
81	The example of a POKE should be: 1Ø POKE 11904,1 loads $2E8Ø with 1.
82	22ØØ 898 - The monitor ROM directs track Ø to load here at $22ØØ should be: 87Ø4 22ØØ - The monitor ROM directs track Ø to load here at $22ØØ.
84	Note, the re-entry point from the machine monitor at hex. 2547 will not always work.
1Ø4	Character number 28 ($1C) should be:  28 $1C

Page Number	Correction
115	Device number 3 for both tables for input and output should be: 3 - 43Ø ACIA UART port
117	NHEX)NNNN,MMMM should be: NHEX>NNNN,MMMM "HEX" referring to 1-3 bytes not 1-4 bytes.
117	WTEXT)MMMM,NNNN should be: WTEXT>MMMM,NNNN "TEXT" is 1-6 characters long.
118	Under the section DISKETTE COPIER the "GØØ2ØØ" should be "GO Ø2ØØ".
121	POKE 896Ø,94 should be: POKE 133,94 for disk based machines or POKE 132,94 for ROM BASIC machines.
122	HEX LOCATION / DECIMAL LOCATION / MACHINE CODE / ASSEMBLER CODE / COMMENT 5E02 / 24Ø66 / AØØ8- / LDY #8 / Load page count

Continued on page 20

it all back out ? (I've tried PEEKing ad POKEing the buffer pointers without success so far).

4) Any warm start routines THAT WORK on a C8P DF w/48k? By the way, the one in the V3.3 manual (Monitor .2547G) is somewhat more useful than they indicate: once you're back to BASIC you can LIST#1 your program and/or values before rebooting so at least you don't have to re-think everything.

Ken Thurman
6706 Abbey Road
B'vil, OK 74003

* * * * *

ED:

In an article which appeared in the Arpil 1982 issue of Peek (65) I discussed changes which could be made to OS65U release 1.3 to tailor the system for use with terminals other than the Hazeltine 1420 when either the Line Editor or Extended Input were active. I recently upgraded to release 1.43. During my study of this release I found that the changes which could be made to release 1.3 were also applicable to release 1.43. I also discovered some additional which may be of interest. This new information is discussed below as well as the previous information (for the benefit of new readers).

Memory locations 23699 and 23700 are both used to identify the delete character code (destructive backspace). The initial value of location 23699 is 127, the DELete character. The initial value of location 23700 is 95, the underscore "-". Location 23700 may be changed (why not to 127?) to enable the underscore to be used.

Location 23701 is used to identify the line delete code. The initial value is 64, the commercial "at" sign ("@"). Changing this memory location (e.g., to 5, the value for Control-E to mean "erase line") permits use of the "@" character.

Locations 23702 and 23703 contain the codes to be recognized as incoming forward space and incoming backspace commands respectively. Locations 23734-23740 contain the code(s) to be echoed to cause a forward space. Locations 23741-23747 contain the code(s) to be echoed to cause a backspace. For most term-

Page Number

122 The BNGs in the assembler code should be BNEs at 5E0A and 5E10.

126 Lines 10 and 20 in the BASIC program should be:

```
10 POKE 8955,0
20 POKE 8956,64
```

126 Lines 20 through 110 in the assembler program should be:

```
20                ; N=USR(H)
30                ; H=character number 0<=H<=255
40                ; N=count of how many times the
50                ;   character appears on the screen.
60                ;
70  3FFC          *=$3FFC
80  3FFC 6C0600   CALL  JMP(6)
90  3FFF EA             NOP
100               ;
110 4000 20FC3F   START JSR CALL integerize H
```

126 Comp in line 210 of the assembler programs should be COMP.

C8P USERS MANUAL ERRATA

Correction

121 The first paragraph should refer to 22FC hexadecimal (8956 decimal) as the "high half" of the hexadecimal address and 22FB hexadecimal (8955 decimal) as the "low half".

Correction

127 Lines 50 and 60 in the BASIC program should be:

```
50 POKE 8955,0
60 POKE 8956,64
```

138 The general form of I/O distribution should be:

```
IO nn     to assign input devices only
IO ,mm    to assign output devices only
IO nn,mm  to assign both input and output devices
```

139 Device number 3 for both tables for input and output should be:

3 - UART on 430 Board ★

finals these echo codes would be one character. The structure appears to provide for an echo command of up to six (6) characters. A zero (0) in a memory location denotes that the previous location was the last character in the command sequence. All of these values are contained in the terminal parameter file "CRT 0", and are placed in memory during initialization of the Line Editor or Extended Input Mode.

Location 23704 contains the code to be recognized for "toggling" between character insert/overstrike modes. The initial value is 20, Control-T. Location 23721 is used to indicate which mode is currently in effect. (See the reference manual for more information).

Location 23705 contains the code entered from the terminal to request a non-destructive cursor move to the front of the line. The initial value is 6, Control-F.

Location 23706 contains the code entered from the terminal to request tabbing eight (8) character positions to the right". The initial value is 9, Control-I.

Location 23707 contains the code entered from the terminal to request a non-destructive cursor move to the rear of the line. The initial value is 18, Control-R.

Location 23708 contains the code to be PRINTed to the terminal to "ring the bell". The value is 7 or Control-G.

Locations 23709 and 23710 are used to specify lower and

upper bounds for acceptable characters when entering alphanumeric data. These locations contain 32 and 128 respectively. Either or both locations may be changed to restrict or expand the range of accceptable characters. Note: any changes effect entry of data whether in the immediate mode or in response to an INPUT command from a program.

Location 23711 is used to contain the character to be recognized as terminating data entry. It contains 13, carriage return.

Locations 23713 and 23714 are used together for entry of data via the terminal. When using Extended Input, after data is entered in response to an INPUT command (e.g., INPUT [20,"A"] A$), location 23713 contains the length of the data field entered (A$), and location 23714 contains the maximum permitted length of the reply (in the case of the example, 20). Normally, location 23714 contains 71, the maximum number of characters permitted when entering program text, immediate mode commands or non-extended input commands (e.g., INPUT A$). Location 23698 also contains 71. The value at this location is used to refresh location 23714 prior to an INPUT command. If extended input is not being used with an INPUT command, the maximum number of characters permitted to be entered can be specified by altering location 23711.

Location 23715 contains the number of positions minus one that the cursor is to be moved to the right in response to a tab request (Control-I). This location contains a 7 for tabbing right eight positions. This value may be changed to suit your needs.

Location 23732 is used to specify the character to be recognized as the "lead-in" for Escape Control commands. A 27 is in this location, ASCII for the ESCape character. If the terminal being used has the program function keys (PFKs), this value may be changed to the lead-in character generated when a PFK is depressed, permitting the PFKs to be used.

David Weigle
Morton, IL 61550

* * * * *

ED:

Regarding Robert L. Dingle's request for a Morse Code program (in the January issue), I highly recommend Rodnay Zaks' book, 6502 APPLICATIONS, published by Sybex, Inc. I think it'll solve this problem and many others.

Stephen B. McGinnis
Crawfordsville, IN 47933

* * * * *

ED:

Under 65U V 1.3 (or higher) with extended I/O enabled, a statement such as

```
Input [L(I), "A"] D$ (I)
```

displays the existing value of D$ (I) at the current cursor position and then expects an input value, meeting the bracketed length and type specifications, which becomes the new value of D$ (I).

This is fine when you are using an unformatted screen and want input prompts, but messes up a pre-formatted screen with protected-field prompts because the display of the existing value of D$ (I) starts filling up the unprotected input field. Even if D$ (I) is null the cursor spaces L(I) spaces - often the entire length of the screen input field.

Does anybody know a good way to combine the input edit capabilities of extended I/O with the kind of screen-

managed input that many users want? In other words, how can the "prompt" aspect of extended I/O be disabled while keeping the utility's advantages?

I have poked around a little but haven't found what I'm looking for. A few interesting things came to light, though. For instance, extended I/O moves the input buffer length control (default = 71) from 1398 to 23698.

W. P. Ausman
Indianapolis, IN 46241

BILL:

The specific answer to your question is that you can't disable the "prompt" while retaining the rest of it.

I don't know what your problem is, however, I do it all the time and have no difficulty.

Dick

* * * * *

ED:

Here's a good one for you. I recently purchased a Hayes Chronograph in the hope of date/timing reports I output on the printer. I am using port 1 of CA10X board to Hayes unit which is RS232, wired with pins 2, 3, 7 and 2x3 crossed per instructions. The attached program works flawlessly with 65D v3.3 (output to screen is: DATE SUN. 83/01/23 TIME 13.56.59 (in hr,min,sec format). The commands in lines 110-150 "AT-- are Hayes commands. In OS65U v. 1.2 - where I need this more than anything (DMS) is where (flag 7 showed me) the system hangs up on line 2010.

I have NO 65U (1.2) system manual - except for a few loose sheets of paper that aren't worth a red cent, plus my own sparse listing of peeks and pokes derived from various sources...but according to my recollection PRINT#8 and INPUT#8 should work. PRINT#8 works from immediate mode - I used it to set the clock to correct date or time. Same can be accomplished by POKE 11686,129 for the serial board output and then the 'AT' commands with the word print, but without the quotes; in 65D, Input #8 doesn't work from immed. mode either but in a program it does. Clock baud is set at 1200 (300 or 1200 OK per documentation). I tried to use (65U 1.2) POKE 11668, 8 or 11668,129 since those are input device pokes, but result

```
10 REM HAYES CHRONOGRAPH
110 OST$="ATVT.":GOSUB 2000:REM PERIOD IS TIME SEPARATOR
120 OST$="ATVD/":GOSUB 2000:REM SLASH IS DATE SEPARATOR
130 OST$="ATRT":GOSUB 2000:TIM$=IST$:REM READ TIME
140 OST$="ATRD":GOSUB 2000:DTE$=IST$:REM READ DATE
150 OST$="ATRW":GOSUB 2000:W=VAL(IST$):REM READ WEEKDAY
160 GOSUB 3000
170 GOSUB 4000:REM OUTPUT TIME,WEEKDAY,DATE TO SCREEN
180 END
2000 PRINT#8,OST$:REM OUTPUT CLOCK COMMAND
2010 INPUT#8,IST$:REM GET INPUT FROM CLOCK
2020 RETURN
3000 IF W<1 THEN DAY$="MON":RETURN
3001 IF W=1 THEN DAY$="TUE":RETURN
3002 IF W=2 THEN DAY$="WED":RETURN
3003 IF W=3 THEN DAY$="THU":RETURN
3004 IF W=4 THEN DAY$="FRI":RETURN
3005 IF W=5 THEN DAY$="SAT":RETURN
3006 IF W=6 THEN DAY$="SUN":RETURN
4000 PRINT"DATE: ";DAY$;", ";DTE$;" TIME: ";TIM$
4020 RETURN
5000 REM NOTE THIS PROGRAM WORKS ON 65D BUT LINE 2010 HANGS IN
                                                        65U
```

was that upon INPUT A$ in a subsequent line, the entire garbage in disk program printed out, and I mean GARBAGE garbage!

Any ideas why INPUT#8 should not work in 65U v.1.2?????

Then, a couple of days later, PEEK(65) received....

I just answered my own question regarding why Input #8 doesn't work in OS65U. I found the answer in PEEK(65) Vol 1 No. 2 in the System Map for 65U. We now have the following line ahead of the first line of the subroutine I sent you-- as follows:

```
IF PEEK(19798)=>1 THEN POKE 19798,0
```

But, I just hope that doesn't mess anything else up! (So far I haven't noticed anything out of the ordinary).

I finished with a program which I wrote from scratch to help me with my office work in the airline industry. Specifically, I have to process large group reservations with the same itinerary, running weekly, e.g. 25 consecutive departures. I have to report back to the client what is (or is not) confirmed which means writing/typing a long tabulation of dates, e.g.

```
FL# FROM TO   FL# FROM TO, etc.
 01 JUN 83      02 JUN 83
 08 JUN 83      09 JUN 83
```

etc. for 26 weeks, and this has to be done for 70 or 80 such 'series'. Lots of work, consulting calendars, etc. Thought I'd let my computer do this for me. The program is too long for presentation in PEEK(65) -6 pages on 8 1/2" x 11" single spc, about 400 lines. A real intriguing problem. Dates are not the easiest things to work with. This is for non-leap year so I'll have to redo parts of it for 1984. What it gives me is data like above but includes the day of the week. Inputs are start date, duration of the tour itinerary, number of weeks, type of flight/connections and the program does the rest - a real time saver - somewhat slow (many GOTOs/GOSUBs) but more than adequate. What used to take 20 minutes now takes 2!

A friend of mine knows someone who has managed to create a routine in assembly language (my weak point) that allows him to run TRS formatted minidisks in some form and to some degree on a OSI C4P. This brings up some interesting ideas. If I ever wanted to swap to IBM 3740 disk format (via a D+N 80 board, for instance) then I'd have to find a way to run my OSI formatted disks (many data disks) and transfer them to the IBM format. Also, I have a CP/M program which runs under the IBM format which I'd like to run under OSI's (current) track format.

Trying to find notes on how the OLD OSI (OSU) disk track format is constructed is impossible. Do you know a source for that information? My feeling is - if Lifeboat can do it, so can we, with the proper guidance of course. I've asked my dealer to order "Reformatter," but it doesn't seem all that easy. Recently bought "Pmate" (a text editor) from Lifeboat. Thus far, I'm not impressed. Documentation is good in some aspects and very inadequate in other ways, particularly in interfacing. It's going to take lots of

effort, I can see that. I wanted it to help me program in C-BASIC but now I'm beginning to wonder. I got it because the "ED" utility under CP/M 2.24 isn't all its cracked up to be either. In fact, I understand that even less, despite the CP/M USER'S GUIDE by Hogan!

Frederick S. Schaeffer
Jamaica, NY 11435

ED:

I'm afraid I have run out of ideas --- HELP! Here is a sample of the problem that has certainly been solved by some OSI owner but has escaped me.

I am attempting to save graphic symbols (actually complex shapes) on disk files as strings. These shapes were originally constructed using DATA statements but, in my particular application, would take too much room. Various pokes to OS65D3.2 have been published which allow the operating system to pass graphic characters to the screen. Unfortunately, it seems that a similar routine is accessed on data returning from a disk file. The CA D200=XX,1 (XX=track of file DATA 1) shows that the graphic symbols got out to disk. They just don't come back. Perhaps the Software Consultants OS65D Disassembly Manual (recently ordered) will provide the answer, but, in the meantime, I could still use some help.

Am currently running an OSI C4P 48K with dual 5 1/4" drives using 65D V3.2, a DCAT modem, BASE 2 printer and an old C2-4P cassette unit.

1. Run program - observe

2. Run 2 - observe graphic characters are now displayed but are still not returned from diskfile DATA 1.

3. Reboot - Run 9 - no better

```
1 GOTO100
2 REM
3 REM
4 REM  FROM AARDVARK JOURNAL
5 FORX=9657TO9664:POKEX,234:
NEXT:POKE9633,234:POKE9634,234
6 GOTO100
7 REM
8 REM
9 REM FROM PEEK(65) VOL1.NUM 6
10 POKE9634,255:POKE9660,0:
   POKE9664,0:POKE9656,0
11 REM
12 REM construct string of
   graphic characters &
   write to disk
13 REM
100 FORX=32TO255
110 IFX=93THENNEXT
120 A$=A$+CHR$(X)
130 NEXT
140 PRINTA$
150 DISKOPEN,6,"DATA1
160 DISKGET,0
170 PRINT#6,A$
180 DISKCLOSE,6
240 PRINT:PRINT"DONE WITH THE
    WRITE TO DISK
300 REM
301 REM recall information
     from disk
302 REM
400 DISKOPEN,6,"DATA1
420 DISKGET,0
440 INPUT#6,B$
460 PRINT:PRINT;PRINT"READ=
    ";B$
480 DISKCLOSE,6
```

Fred Schwierske
Cedarburg, WI 53012

* * * * *

ED:

The reason I'm writing is twofold. One, do you have any recommendations for "FORTH" support, preferably not too expensive. I presently have a copy of "FORTH" with tiny "PASCAL" from Progressive Computing and frankly, the documentation stinks. I hope there is something better. The other reason is, I was intrigued by Jeff Easton's comments in the FEB issue about putting the 6809 on the OSI bus. I think that is a super idea and hope you can convince him to continue and then publish an article on "how to". I can't think of a better combination than OSI's video for games and "FLEX" with "SS-50" bus software in the business area where OSI is lacking. That would be a really super setup.

Neil Dennis
Bliss, NY 14024

* * * * *

WANTED: The Wrath of Khan by Cygnus Software. Anyone owning this software, please contact David Whipp, 1014 E. 400 So., Salt Lake City, UT 84102.

AD$

USED OSI - BUY SELL SERVICE. C3-B 6K. Dale King, P. O. Box 5412, Arlinton, TX 76011 (817) 265-3760.

16K OSI C4P with CA-20 board, D&N BMEM-CM9F with 8K, socketed disk controller, D&N 96 protoboard, Lambda 10-A supply. SAMS/OSI manuals, $300. Paul K. Pagel, 4 Roberts Rd., Enfield, CT 06082.

FOR SALE: 1 C1P w/610 Board 32K configured with C1S V4.0 Monitor Rom. 2. (2) MPI-005 Disk Drives w/Data Separators. 3. 48 Pin Backplane 8 slot. 4. Centronics parallel printer interface board. 5. AMDEX Video 100 monitor. 6. One complete w/disks OS-65D Tutorial and reference manual. 7. Internal and external power supplies +/- 5volt and +/- 12volt 10 amp. 8. Manuals: The First Book of Ohio Scientific Vol 1, The Second Book of Ohio Scientific, Understanding Your Ohio Scientific C1P and C4P, 65V Primer, Basic Reference Manual, Assembler Editor and Extended Monitor Reference Manual, Sams for Computer Boards 600 & 610, Superboard II, Challenger 1P Technical Report, Line Printer Interface Manual, C1S V4.0 Monitor Rom Manual, Instructions for Wire Wrapping A Sound Board For The C1P, Joystick Instructions, Maxi-Pros Instructions, Warranty Cards on Equipment. 9. Software on Cassette: 1. Fighter Pilot, 2. Asteroids, 3. Catterpillar, 4. Memory Test, 5. OSI-Term CIP, 6. SCX-102 C1P Sampler, 7. SCX-106 Series 2 Video Swap, 8. Mortgage Programs. Price: $800.00 or best offer. CALL: 301-792-2387 after 5pm or 301-654-6616 between 9am-5pm ask for John Bowman.

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

GOODIES for OSI Users!

PEEK (65)
The Unofficial OSI Users Journal

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

( ) **C1P Sams Photo-Facts Manual:** Complete schematics, scope waveforms and board photos. All you need to be a C1P or SII Wizard, just — $7.95 $ _____

( ) **C4P Sams Photo-Facts Manual.** Includes pinouts, photos, schematics for the 502, 505, 527, 540 and 542 boards. A bargain at — $15.00 $ _____

( ) **C2/C3 Sams Photo-Facts Manual.** The facts you need to repair the larger OSI computers. Fat with useful information, but just — $30.00 $ _____

(X) **OSI's Small Systems Journals.** The complete set, July 1977 through April 1978, bound and reproduced by PEEK (65). Full set only — $15.00 $ _____

( ) **Terminal Extensions Package** - lets you program like the mini-users do, with direct cursor positioning, mnemonics and a number formatting function much more powerful than a mere "print using." Requires 65U. — $50.00 $ _____

( ) **RESEQ** - BASIC program resequencer plus much more. Global changes, tables of bad references, **GOSUBs** & GOTOs, variables by line number, resequences parts of programs or entire programs, handles line 50000 trap. Best debug tool I've seen. MACHINE LANGUAGE - VERY FAST! Requires 65U. Manual & samples only, $5.00 Everything for — $50.00 $ _____

( ) **Sanders Machine Language Sort/Merge** for OS-65U. Complete disk sort and merge, documentation shows you how to call from any BASIC program on any disk and return it or any other BASIC program on any disk, floppy or hard. Most versatile disk sort yet. Will run under LEVEL I, II, or III. It should cost more but Sanders says, "...sell it for just..." — $89.00 $ _____

( ) **KYUTIL** - The ultimate OS-DMS keyfile utility package. This implementation of Sander's SORT/MERGE creates, loads and sorts multiple-field, conditionally loaded keyfiles. KYUTIL will load and sort a keyfile of over 15000 ZIP codes in under three hours. Never sort another Master File. — $100.00 $ _____

BOOKS AND MANUALS (while quantities last)

( ) **65V Primer.** Introduces machine language programming. — $4.95 $ _____

( ) **C4P Introductory Manual** — $5.95 $ _____

(X) **Basic Reference Manual** — (ROM, 65D and 65U) — $5.95 $ _____

( ) **C1P, C4P, C8P Users Manuals** — ($7.95 each, please specify) — $7.95 $ _____

( ) **How to Program Microcomputers.** The C-3 Series — $7.95 $ _____

( ) **Professional Computers Set Up & Operations Manual** — C2-OEM/C2-D/C3-OEM/C3-D/C3-A/C3-B/C3-C/C3-C' — $8.95 $ _____

( ) Cash enclosed ( ) Master Charge ( ) VISA

Account No. _____ Expiration Date _____

Signature _____

Name _____

Street _____

City _____ State _____ Zip _____

TOTAL	$_____
MD Residents add 5% Tax	$_____
C.O.D. orders add $1.50	$_____
Postage & Handling	$ 3.00
TOTAL DUE	$_____

POSTAGE MAY VARY FOR OVERSEAS