PEEK (65)

The Unofficial OSI Users Journal

$1.75

SEPTEMBER 1983
Vol. 4, No.9

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

INSIDE

Column One

Our plans for the "software issue" in October are progressing. Actually, of course, the entire issue will not be software alone -- we will not abandon hardware freaks for a whole month! But we do plan to devote a large portion of the October issue to descriptions of software available by/for PEEK(65) readers. If you haven't sent in your description of YOUR favorite gem yet, do so right away (use the form in the back of the August issue or a reasonable photocopy).

If the response is as good as we expect, we will continue to present descriptions of PEEK(65) subscriber's software in future issues as well. Let us know about your goodies.

Speaking of goodies ... have a look at this issue! We have the usual (and appreciated) technical articles for OSI personal computer users, plus a review of the Denver Board modification to OSI's "old" line of business computers and the first of a series of articles on optimizing Microsoft BASIC. This is the kind of all-size-machine mix we love. Keep it up.

More info from OSI: New Users' Guides for 230 E and I machines are at the printers. Details will be released next month on Keysort (OSI special form of Cosort), Key File Loader and Microsoft Utilities for the 300's. M80, Microsoft's famous macro assembler, will be available for about $200, we're told.

A new "advanced" software course for the 300's was run in the spring and will be repeated this fall. It is for experienced programmers only, and covers assembly language techniques for the 300 series. Contact your dealer to get into the course.

We now have our multiuser computer here at PEEK(65) running OS-65U V 1.43. We have noted a couple of changes not mentioned (that we can find) in the documentation. First, the new directory format, instead of just saying "XXXX Bytes recoverable" (meaning some files have been deleted but not repacked to recover the disk space) actually LISTS the deleted files, marked as deleted.

Secondly, and more importantly, CREATE now scans the directory for deleted files, and when it finds some space into which the new file you are creating will fit, sticks it in there, rather than always appending it to the end of the disk as before. This removes one of the major objections to 65U file handling, the need to periodically repack the disk to eliminate space no longer used by deleted files, while retaining the major advantage of 65U file handling, in that files are still contiguous on disk, allowing super-fast searching without a lot of "thrashing" back and forth on the disk. Be sure to write us and let us know what else, good and bad, you find as you use the new releases of the OS.

Publishing PEEK(65) is like owning a motorcycle and a van (or maybe an 18-wheeler!). Many of our loyal readers and contributors own C1P's. Others own multi-user business machines with huge hard disks. How do we address both groups? We almost don't. The hackers (I use the term as a compliment) send us lots of stuff; the business users are so busy trying to make a living they rarely do.

We are about to change all that, with the following startlingly generous offer. Any OSI user who writes an article on a real live business application which is published in PEEK(65) will win a free one-year subscription to PEEK.

There is only one limitation. It must be a new subscription, or a renewal for next year on your own. This means you can't write 8 business articles and get your subscription extended 8 years (since who knows what inflation may have done to our prices by then).

However, the subscription is assignable. This means if you are a dealer or consultant or programmer, as many of our business system users are, you can use your prize as a gift to one of your clients, and we will be sure he/she gets the issue in which your article is published. That should be good publicity for you! So keep those articles and letters coming, folks. *ac*

QMAKER

by: Steve Rydgig
#4 Frontenac, R.R. 2
Collinsville, IL 62234

I thought it was about time for me to put in my two cents worth. My machine is a C4P with an additional OSI power supply, a D&N memory and floppy board, 32K, and MPI B-51 drive and a Shugart SA-400, a serial MX-80 printer and OS65D V3.3.

The programs I have enclosed have to do with my school work and are written for OS65D V3.3.

Listing 1 is a program I call qmaker. I use it to prepare test questions. It requires 1 buffer and that the files already be created on the diskette. For multiple choice questions, six inputs are required. The first is the question. The second is the second part of the question. If the question fits in 80 characters a <CR> must be entered for the second part of the question. It then prompts for four answers, clears the screen and shows the complete question and waits for a response. A <CR> or number >6 causes the items to be sent to disk. A 1 or 2 lets you enter a new 1st or second part of the question and 3-6 allows you to change an answer.

When selecting matching items, you first enter the term <CR> and the question. If the question is longer than 80 characters in the input buffer, a * as the last character of that question allows another question line to be input. The true false section is similar in that if you can't get the statement in one line, ending that line with '*' will allow input of a second line. The multiple choice holds about 10-15 items per track, the matching 20-25 and I've just added the true false so I'm not sure how many of those items will fit on a track, perhaps 20-25.

Listing 2 is my program to print tests from the data files generated from listing 1. Somewhere between line 1 and 100 an INPUT T$ line is needed to get the name of that test printed. Line 280 has a kind of GET that I dug out of a BEXEC* program from 3.3. The pokes in 320 allow me to send info to my serial MX-80 printer at 4800 baud. The pokes to 64513 generate underline characters which I haven't been able to print any other way yet. 370-440 are for statements to open files and send the program to subroutines. 370-390 are spaced as they are to allow me to "edit" those lines using CTRL I's to get me between the quotes and put the names of the files in those lines. Sometimes I add other kinds of test items and line 400 is there to remind me to add the number of those items to NO which is my question counter. By running this once then switching the locations of A1$-A4$ in lines 500-530, you can get two different tests to look identical. For the most part I am pleased with this program but on occasion the spacing gets out of whack on long questions. That's why I'm peeking and poking 22 which, as far as I can tell does the same thing as 14 in ROM basic. I originally wrote this for my ROM basic but it required much memory and a good garbage collection as well as typing numerous data statements which becomes confusing late at night. The print CHR$ statements are control statements for the printer.

Listing 3 is a program to edit files made by listing 1. It allows changing spelling errors to be corrected, etc.

Hope these programs are of some use to someone. Keep up the good work at Peek(65).

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

LISTING 1

```
 90 DISK!"SE B
 100 REM QMAKER
 110 PRINT" 1>Multiple choice"
:PRINT" 2> Matching
 120 INPUT" 3> True false";I
 130 ONIGOSUB150,340,480
 140 PRINT!(28):GOTO110
 150 INPUT"FILE NAME";A$:POKE2
888,0:POKE8722,0
 160 POKE2972,13:POKE2976,13
 170 INPUT"HOW MANY ITEMS";I
 180 DISK!"SE B"
 190 DISK OPEN,6,A$
 200 PRINT#6,I:FORA=1TOI:FORB=
1TO6:IFB=1THENPRINT"1ST PART O
F QUESTION
 210 IFB=2THENPRINT"2ND PART O
F QUESTION
 220 IFB=3THENPRINT"ANSWER A
 230 IFB=4THENPRINT"ANSWER B
 240 IFB=5THENPRINT"ANSWER C
 250 IFB=6THENPRINT"ANSWER D
 260 INPUTA$(B):IFB=1THENA$(B)
=A$(B)+" "
 270 NEXTB
 280 PRINT!(28):FORB=1TO9:PRIN
T:NEXT
 290 PRINTA$(1);A$(2);:FORI=3T
O6:PRINT" ("CHR$(I+62)") "A$(I
):NEXT
 300 INPUT"ANY CORRECTIONS";C:
IFC>0ANDC<7THENINPUTA$(C):GOTO
280
 310 FORB=1TO6:PRINT#6,CHR$(34
)A$(B)CHR$(34):NEXT
 320 NEXTA:DISK CLOSE,6
 330 RETURN
 340 INPUT"HOW MANY MATCHING I
TEMS";M
 350 INPUT"FILE NAME";CH$:DISK
 OPEN,6,CH$
 360 PRINT#6,M
 370 FORI=1TOM
 380 INPUT"TERM";A$(1):INPUT"Q
UESTION";A$(2)
 390 IFRIGHT$(A$(2),1)="*"THEN
INPUT"2ND PART OF QUES";A$(3)
 400 PRINT!(28):FORQ=1TO10:PRI
NT:NEXT
 410 PRINT1A$(1):PRINT2A$(2):I
FRIGHT$(A$(2),1)="*"THENPRINT3
A$(3)
 420 INPUT"IS THIS ITEM OK";A:
IFA>0ANDA<4THENINPUTA$(A):GOTO
400
 430 FORT=1TO2:PRINT#6,CHR$(34
)A$(T)CHR$(34):NEXT
 440 IFRIGHT$(A$(2),1)="*"THEN
PRINT#6,A$(3)
 450 NEXTI
 460 DISK CLOSE,6
 470 RETURN
 480 INPUT"HOW MANY TRUE FALSE
 ITEMS";TF
 490 INPUT"FILE NAME";N$:DISK
OPEN,6,N$
 500 PRINT#6,TF
 510 FORI=1TOTF
 520 PRINT"QUESTION":INPUTQ$(1

 530 IFRIGHT$(Q$(1),1)="*"THEN
INPUT"SECOND PART";Q$(2)
 540 PRINT!(28):FORQ=1TO10:PRI
NT:NEXT
 550 PRINT"1"Q$(1):IFRIGHT$(Q$
,1)="*"THENPRINT"2"Q$(2)
 560 INPUT"Is this item Ok";a:
IFa>0ANDa<3THENINPUTa$(a):GOTO
540
 570 PRINT#6,q$(1):IFRIGHT$(q$
(1),1)="*"THENPRINT#6,q$(2)
 580 NEXT:DISK close,6:RETURN
```

LISTING 2

```
 1 REM TESTV4
 100 POKE2972,14:POKE2976,13:G
OTO270
 110 L=LEN(Q$)
 120 FORA=1TOL:P=PEEK(22)
 130 IFP<60THEN170
 140 IFQR=2THEN160
```

High Resolution Color Graphics

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

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

Pricing:

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

Generos Operating System

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

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

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

Low Power Memory Board

Our popular MEM + board is ideal for:

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

Options include:

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

MEM + includes the following features:

- High reliability — All memory chips in machine screw socets
- Low power consumption
- Uses 2K x 8-bit memory chips — will accept 2716-type EPROMS.
- Versatile addressing

NEW LOW PRICES!!

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

16K	$200
24K	$250
32K	$300
40K	$350
48K	$390
52K	$415
56K	$440
64K	$490

Disk controller	$85
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
5221 S.W. Corbett
Portland, Oregon 97201

(503) 227-7083

Generic Computer Products

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

```
 150 IFP=>64ANDMID$(Q$,A,1)="
"THENGOSUB210:GOTO180
 160 IFP=>64ANDMID$(Q$,A,1)="
"ANDQR<>1THENPRINT:POKE22,3
 170 PRINTMID$(Q$,A,1);
 180 NEXT
 190 IFQR=1ANDP=>65THENGOSUB21
0
 200 RETURN
 210 PRINTTAB(72);N$;
 220 IFPEEK(8994)<>2THENFORQ=1
TO4:POKE64513,95:NEXT:PRINT
 230 IF PEEK(8994)=2THENPRINT
 240 IFMA<>0THENIFLEN(CH$)>23T
HENPRINTMID$(CH$,24);TAB(26)"
";:GOTO260
 250 IFMA<>0THENPRINTTAB(26)"
";
 260 PRINT"    ";:QR=2:RETURN
 270 W=80:DISK!"SE B"
 280 PRINT"PRINTOUT (P) OR SCR
EEN";:DISK!"GO 2336":A$=CHR$(P
EEK(9059))
 290 PRINTA$
 300 IFA$="P"THEN320
 310 GOTO330
 320 W=80:POKE64512,3:POKE6451
2,16:DISK!"IO 02,03"
 330 PRINT"PHYSIOLOGY TEST "TA
B(16) T$TAB(W-21)"NAME";
 340 IFPEEK(8994)<>2THENFORx=1
TO16:POKE64513,95:NEXT:PRINT
 350 PRINTTAB(W-21)"Hour";
 360 IFPEEK (8994) <>2THENFORx
=1TO5:POKE64513,95:NEXT:PRINT
 370 REM DISK OPEN,6,"":INPUT#
6,MC:GOSUB350
 380 :   DISK OPEN,6,"":INPUT#
6,MA:GOSUB580:DISK CLOSE,6:REM
 MATCH
 390 :  DISK OPEN ,6,"":INPUT#
6,TF:GOSUB780
 400 REMno=no+4  increment ite
m counter when ?ing other kind
s items
 410 REM
 420 REM
 430 REM
 440 DISK!"IO 02,02":DISK!"se
a":END
 450 ca$="*** multiple choice
***":GOSUB560
 460 POKE23,w:FORI=1TOMC:REM m
ultiple choice
 470 no=no+1:n$=MID$(STR$(no),
2) +". "
 480 INPUT#6,a$,B$,A1$,A2$,A3$
,A4$
 490 Q$=N$+A$+B$:QR=1:GOSUB110
 500 Q$=" (A) "+A1$:GOSUB110
 510 Q$=" (B) "+A2$:GOSUB110
 520 Q$=" (C) "+A3$:GOSUB110
 530 Q$=" (D) "+A4$:GOSUB110
 540 PRINT:PRINT
 550 NEXTI:RETURN
 560 PRINT:P=20:p=p-(LEN(CA$)/
2):PRINTCHR$(14)TAB(p)CA$
 570 PRINT:RETURN
 580 ca$="*** matching ***":PR
INT:GOSUB560
 590 FORi=1TOma:no=no+1:N$=MID
$(STR$(NO),2)+". "
 600 INPUT#6,ch$,Q$:L=LEN(Q$)
 610 IFRIGHT$(Q$,1)="*"THENI$=
LEFT$(Q$,L-1):INPUT#6,Q1$:Q$=I
$+" "+Q1$
 620 L=LEN(Q$)
 630 IFCH$="*"THENQ$=N$+Q$:QR=
1:GOTO750
 640 IFLEN(CH$)>23THENPRINTCHR
$(I+64)". "LEFT$(CH$,23)"-";:G
OTO660
 650 PRINTCHR$(I+64)". "ch$;
 660 Q$=N$+Q$:QR=1
 670 L=LEN(Q$)
 680 FORA=1TOL:P=PEEK(22)
 690 IFP<60THEN740
 700 IFQR=2THEN730
 710 IFP=>64ANDMID$(Q$,A,1)="
"THENGOSUB210:GOTO730
 720 IFP=>64ANDMID$(Q$,A,1)="
"ANDQR<>1THENPRINT:PRINTTAB(26
)" ";
 730 PRINTMID$(Q$,A,1);
 740 NEXT
 750 PRINTTAB(26)" ";:GOSUB110
 760 IFQR=1THENGOSUB210
 770 PRINT:NEXTI:MA=0:RETURN
 780 CA$="*** TRUE-FALSE ***":
PRINT:GOSUB560
 790 PRINT"Put + in the answer
 blank for true, O for false":
PRINT
 800 FORi=1TOtf
 810 NO=NO+1:N$=MID$(STR$(NO),
2)+". "
 820 INPUT#6,q$:IFRIGHT$(q$,1)
="*"THENINPUT#6,Q1$
 830 IFRIGHT$(q$,1)="*"THENq$=
MID$(Q$,1,(LEN(Q$)-1))+" "+q1$
 840 Q$=N$+Q$:QR=1:GOSUB110:IF
QR=1THENGOSUB210
 850 PRINT:IFL>66THENPRINT
 860 NEXTi:RETURN
 870 GOSUB 450:REM MULTIPLE CH
OICE
 880 GOSUB 580:REM MATCHING
 890 GOSUB 780:REM TRUE FALSE
```

LISTING 3

```
 1 REM         DATACG         WI
TH 2 BUFFERS
 5 DISK!"SE B
 10 PRINT!(28):PRINT:PRINT:PRI
NT:PRINT"1. multiple choice it
ems
 20 PRINT"2. matching items
 30 INPUTi$:ON VAL(i$)GOSUB100
,300
 40 CLEAR:POKE2888,27:POKE8722
,27:GOTO10
 100 PRINT:PRINT:PRINT"What is
 the name of the ";:POKE2888,0
:POKE8722,0
 110 INPUT"multiple choice fil
e you wish to change";f1$
 120 PRINT:PRINT:PRINT"What is
 the name of the ";
 130 INPUT"name of the file in
 which this will go";f2$
 140 DISK open,6,f1$:DISK open
,7,f2$
 150 INPUT#6,n:PRINT#7,n
 160 DIMN$(N,6)
 170 FORI=1TON:FORA=1TO6:INPUT
#6,N$(I,A):NEXTA
 175 PRINT!(28):PRINT:PRINT:PR
INT:PRINT:PRINT
 176 FORa=1TO6:PRINTa;n$(i,a):
NEXTa
 180 INPUT"Is this item ok";q$
:t=VAL(q$)
 190 IFt>0ANDt<7THENINPUTn$(i,
t):GOTO175
 200 FORA=1TO6:PRINT#7,CHR$(34
);N$(I,A);CHR$(34)
 210 NEXTa,i
 299 DISK CLOSE,6:DISK CLOSE,7
:RETURN
 300 PRINT:PRINT:PRINT"What is
 the name of the ";:POKE2888,0
:POKE8722,0
 310 INPUT"matching file you w
ish to change";f1$
 320 PRINT:PRINT:PRINT"What is
 the name of the ";
 330 INPUT"name of the file in
 which this will go";f2$
 340 DISK open,6,f1$:DISK open
,7,f2$
 350 INPUT#6,n:PRINT#7,n
 360 DIMN$(N,3)
 370 FORI=1TON:FORA=1TO2:INPUT
#6,N$(I,A):NEXTA
 372 IFRIGHT$(N$(I,2),1)="*"TH
ENINPUT#6,N$(I,3)
 375 PRINT!(28):PRINT:PRINT:PR
INT:PRINT:PRINT
 380 FORA=1TO2:PRINTA;N$(I,A):
NEXT
 390 IFRIGHT$(N$(I,2),1)="*"TH
ENPRINT3;N$(I,3)
 400 INPUT"Is this item ok";q$
:t=VAL(q$)
 410 IFT>0ANDt<4THENPRINT"chan
ge";n$(i,t):INPUTN$(I,T):GOTO3
75
 420 FORa=1TO2
 440 PRINT#7,CHR$(34)n$(i,a)CH
R$(34)
 450 NEXTa
 455 IFRIGHT$(N$(I,2),1)="*"TH
ENPRINT#7,CHR$(34)N$(I,3)CHR$(
34)
 457 NEXTI
 460 DISK close,6:DISK close,7
:RETURN
```

★

MORE EX/MON ADDITIONS

by: Bill Woodland
6000 Turtle Dove Drive
Austin, TX 78744

First of all, thanks PEEK(65) for such a wonderful job of distributing all of this handy OSI information. I have been reading your magazine from the start and have found many valuable pieces of information which were of great help. I suggest to all of you OSI and computer neophytes that you buy all of the back issues of PEEK(65) and read them one by one. These will probably answer about 99% of your questions.

Here are a few goodies for you, both of which are additions to the 65D Extended Monitor. The first is a Hex Dump with ASCII on the right, the second will renumber the assembly code which is presently in the workspace.

The ASCII dump routine was written for the Serial Terminal but can be set up for the 540 board. Lines 80 and 90 add the address of the routine to the EX/MON's jump table in place of the 'J' command. Use this just like the 'D' command (J3200,3300 will dump with ASCII the block of memory from

$3200-$32FF). Line 110 gets the starting address from the OS input buffer. Lines 120 through 250 print the least significant digit of the address across the top of the hex part, 260 through 320 print it over the ASCII part. Lines 330 to 410 print the start address of the next 16 bytes to dump.

The hex dump is performed in lines 420 to 680. This part also checks for the end of dump address. If the dump is for a number of bytes which is not evenly divisible by 16 then spaces are printed till we point to the ASCII part. Lines 690 to 810 print the ASCII of the byte then JSR's to a routine in the EX/MON which increments the starting address and compares with the ending address. If the starting address is greater, then the routine jumps back to the command mode of the EX/MON, if not, it just returns to the calling routine. This is how several of the EM routines return to the command mode.

In my ASCII dump, hex codes less than $20 or greater than $7E are printed as a period. I figured it would be easier to count them than to count those invisible spaces when figuring which hex code was which ASCII character. If you want another character, put the hex code for it in line 820.

Since the video board only has 64 characters across, those users need to delete lines 220 and 570. This makes the hex part a little dense but it will all fit on the screen fairly nicely. If you need to look at the hex part only then use the 'D' command, but the ASCII routine will sure come in handy for checking the end of source in the workspace or making changes to source without having to retype entire lines.

The second routine will renumber the source which is currently in the workspace. For reasons of simplicity the increment value is limited to 1 through 9. Use the following format for the renumber command: ZDDDDD,I, where Z is the command, DDDDD is the line number to start renumbering with, and can be from 1 to 65535, and I is the one digit increment.

Lines 80 and 90 add the routine address to the JUMP TABLE as in the DUMP ROUTINE, but this time at the 'Z' location. Lines 120 through 170 initialize some storage locations. Lines 180 through 370 get the starting line number from the OS input buffer one digit at a time and perform an overflow and non-digit entry check. When a comma is found, the program jumps to line 380 to input the increment value and also check for legal digit entry. The actual renumbering is done at lines 450 through 830. When the job is done it jumps back to the assembler. This can be easily changed to jump to the command mode of the EX/MON if you want. Lines 920 to 1140 do the decimal to hex conversion of the starting line number.

Here is an example of how to combine source programs which have conflicting line numbers:

```
A*AS          GO TO ASSEMBLER
.!LO OS40     LOAD PROGRAM 40
              OF 40 INTO THE
              WORKSPACE
.!RE REM      GO TO THE EX/MON
:Z5000,5      RENUMBER SOURCE
              FROM 5000 INCRE-
              MENTING BY 5 AND
              RETURN TO THE
              ASSEMBLER
.P[ SHIFTK    LIST WORKSPACE
              FILE INTO INDIR-
              ECT FILE
.] SHIFTM     CLOSE INDIRECT
              FILE
.!LO OS39     LOAD NEXT PROGRAM
              INTO WORKSPACE
.<CNTL>X      INDIRECT FILE IS
              ADDED TO THE
              WORKSPACE FILE
.RE EM        BACK TO EX/MON
:Z5000,5      SAME AS BEFORE
.P            SAME
.             SAME
.!LO OS38     CONTINUE TILL ALL
              FILES ARE ADDED
              TOGETHER
.!PU SOURCE   SAVE FINISHED
              FILE TO DISK
```

Be sure that the workspace file doesn't get too big for the memory available for the indirect file or you will get hung up when you try to bring the indirect file back down to the workspace.

I hope these routines will be of some service to all of you and maybe bring to mind some more routines that would be handy. Stay tuned for an easy way to add commands to the OS (like DELETE, CREATE, DIR, RENAME, COPY) while using only 512 to 768 bytes at the top of RAM (at least I think I can keep it that small). I will write it up in full detail as soon as I have it completed.

Also, if you don't want to type your little fingers off, I will sell a copy of the source for the ASCII dump and assembly renumber and the object code linked up to the EX/MON for 10 dollars. Send S.A.S.E. which will hold a disk, and check or money order to the above address.

```
:!IO ,01

.P

10 GETADR = $1A6B
20   CRLF = $1A56
30  SPACE = $1BAF
40    HEX = $19F2
50  P2HEX = $19E9
60  PRINT = $2343
70 INCADD = $1A85
80      * = $18B3
90 ORIGIN . WORD $1F46
100       * = $1F46
110       JSR GETADR
120       JSR CRLF
125 ; PRINT 6 SPACES
130       LDY #$06
140 SP    JSR SPACE
150       DEY
160       BNE SP
165 ; PRINT LOW DIGIT OF AD-
          DRESS
166 ; ACROSS TOP OF HEX PART
170       LDX $CC
180       LDY #$10
190 LOOP1 TXA
200       JSR HEX
210       JSR SPACE
220       JSR SPACE LEAVE
          THIS OUT FOR VIDEO
          BOARD
230       INX
240       DEY
250       BNE LOOP1
255 ; PRINT LOW DIGIT OF AD-
          DRESS
256 ; ACROSS TOP OF ASCII PART
260       LDX $CC
270       LDY #$10
280 LOOP2 TXA
290       JSR HEX
300       INX
310       DEY
320       BNE LOOP2
325 ; PRINT THE ADDRESS AND A
          SPACE
330 PADD  JSR CRLF
340       LDA $CD
350       STA $C3
360       JSR P2HEX
370       LDA $CC
380       STA $C2
390 OK2   JSR P2HEX
400       JSR SPACE
410       LDY #$00
415 ; DO HEX PART OF DUMP
420 LOOP3 LDA ($CC),Y
425       JSR P2HEX
426       JSR SPACE
430 ; ARE WE PAST END ADDRESS?
440       INC $C2
450       BNE OK
460       INC $C3
470 OK    SEC
480       LDA $C2
490       SBC #CE
500       LDA $C3
510       SBC $CF
530       BCC FINISH
535       BCS ENDHEX
540 ; PRINT SPACES FOR REMAIN-
          DER OF LINE
```

3 USERS-80 Mega Bytes —$8990.00*

INTRODUCTORY SPECIAL

WITH DUAL FLOPPIES
BRAND NEW —
1 YEAR WARRANTY ON HARD DISK!
REGULAR $10,990.00

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

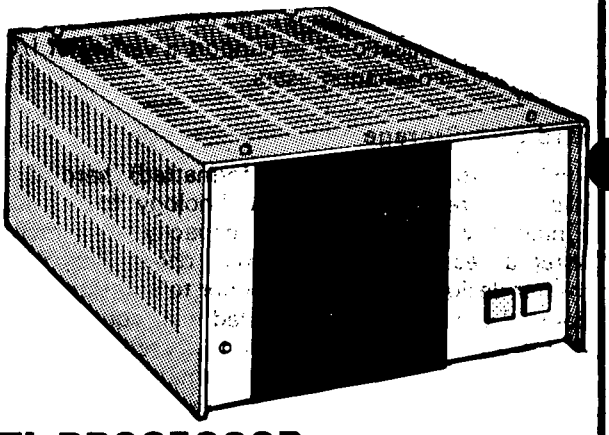
ALSO AVAILABLE WITH 3 MULTI-PROCESSOR

Denver Boards with 64K each user and Centronics Parallel Printer Port at

$9990.00

*DEALER DISCOUNTS AVAILABLE

8" HARD DISK SYSTEMS



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

$5990.00

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

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

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

MULTI-PROCESSOR DEVELOPMENT SYSTEM SPECIAL

SPECIAL
ONLY $5990.00

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

CLOSE OUTS

C4P—$350.00, C4PMF—$699.00, C4P DMF 48K—$1199.00, C8P-DF—$1499.00 C2-OEM—$1799.00, CM-2—$69.00, CM-10—$89.00, CM-11—$499.00, CA-9—$129.00, CA-10-1—$149.00, 510 CPU—$299.00, OSI C4P Disk Programs, Regular $29-$49 — *NOW* $9.00 each.

Call or Write for Bargains List!

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

Space-Com International

22991 LA CADENA DRIVE, LAGUNA HILLS, CALIFORNIA 92653

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

```
550 LOOP7 JSR SPACE
560       JSR SPACE
570       JSR SPACE TAKE OUT
          FOR VIDEO BOARDS
580 ENDHEX INY
590       CPY #$10
600       BMI LOOP7
630       BPL PASCII
660 FINISH INY
670       CPY #$10
680       BNE LOOP3
690 PASCII JSR SPACE
700       LDY #$00
710       LDX #$10
720 LOOP4 LDA ($CC),Y
730       CMP #$20
740       BMI DOT
750       CMP #$7E
760       BPL DOT
770 ASCII JSR PRINT
780       JSR INCADD
800       BNE LOOP4
810       BEQ PADD
820 DOT   LDA #$2E
830       BNE ASCII
```



```
10                ; ASMNUM : RENUMBER ASSEMBLY SOURCE CODE
20 000D=                CR = $OD
30 0000=                INDEX = $00
40 3179=                BEGIN = $3179
50 317B=                LAST  = $317B
60 1303=                STASM = $1303
70 2CE4=                BUFBYT = $2CE4
80 1A4E=                ERROR1 = $1A4E
90 18D3               * = $180D3
100 18D3 E31F           . WORD $1FE3
110 1FE3              * = $1FE3
120 1FE3 D8             CLD
130 1FE4 A900           LDA #$00
140 1FE6 BDCA20         STA LNMULT
150 1FE9 8DCB20         STA LNMULT+1
160 1FEC 8DC820         STA LINNBR
170 1FEF 8DC920         STA LINNBR+1
180 1FF2 20E42C LININP  JSR BUFBYT
190 1FF5 C930           CMP #$30
200 1FF7 3004           BMI XLEGAL
210 1FF9 C940           CMP #$40
220 1FFB 3007           BMI GOOD
230 1FFD C92C   XLEGAL  CMP #',
240 1FFF F01B           BEQ INPINC
250 2001 4C8220         JMP ERROR
260 2004 290F   GOOD    AND #$0F
270 2006 48             PHA
280 2007 D8             CLD
290 2008 208F20         JSR MULTLN
300 200B 6B             PLA
310 200C 6DC820         ADC LINNER
320 200F 8DC820         STA LINNER
330 2012 90DE           BCC LININP
340 2014 18             CLC
350 2015 EEC920         INC LINNBR+1
360 2018 B068           BCS ERROR
370 201A 90D6           BCC LININP
380 201C 20E42C INPINC  JSR BUFBYT
390 201F C931           CMP #$31
400 2021 305F           BMI ERROR
410 2123 C939           CMP #$39
420 2025 105B           BPL ERROR
430 2027 290F           AND #$0F
440 2029 8DCC20         STA INCR
450 202C A500           LDA INDEX
460 202E 48             PHA
470 202F A501           LDA INDEX+1
480 2031 48             PHA
490 2032 AC7931         LDY BEGIN GET LO BYTE OF SOURCE ADDR.
500 2035 A900           LDA #$00
510 2037 8500           STA INDEX
520 2039 AD7A31         LDA BEGIN+1  GET HI BYTE
530 203C 8501           STA INDEX+1
540 203E ADC820 LOOP    LDA LINNBR  GET LO BYTE OF LINE NUMBER
550 2041 9100           STA (INDEX),Y STORE IT IN SOURCE
560 2043 208620         JSR INCREM
570 2046 ADC920 L1      LDA LINNBR+1 GET HI BYTE OF LINE NBR.
580 2049 9100           STA (INDEX),Y STORE IT IN SOURCE
590 204B 208620 L2      JSR INCREM
600 204E B100           LDA (INDEX),Y LOOK FOR END OF LINE
610 2050 C90D           CMP #CR
620 2052 D0F7           BNE L2
630 2054 208620         JSR INCREM
640 2057 CC7B31 L4      CPY LAST
650 205A D010           BNE L5
660 205C A501           LDA INDEX+1
670 205E CD7C31         CMP LAST+1
680 2061 D009           BNE L5
690 2063 68             PLA         RESTORE ORIGIN DATA
700 2064 8501           STA INDEX+1
710 2066 68             PLA
720 2067 8500           STA INDEX
730 2069 4C0313         JMP STASM
740 206C ADC820 L5      LDA LINNBR  INCREMENT LINE NUMBER
750 206F D8             CLD
760 2070 18             CLC
770 2071 6DCC20         ADC INCR
780 2074 8DC820         STA LINNBR
790 2077 9006           BCC L6
800 2079 18             CLC
810 207A EEC920         INC LINNBR+1
```

Listing continued.

```
820 207D B003           BCS ERROR
830 207F 4C3E20 L6      JMP LOOP
840 2082 18     ERROR   CLC
850 2083 4C4E1A         JMP ERROR1
860 2086 C8     INCREM  INY
870 2087 D005           BNE INCRTS
880 2089 18             CLC
890 208A E601           INC INDEX+1
900 208C B0F4           BCS ERROR
910 208E 60     INCRTS  RTS
920 208F 20BE20 MULTLN  JSR MULTX2
930 2092 ADC920         LDA LINNBR+1
940 2095 8DCB20         STA LNMULT+1
950 2098 ADC820         LDA LINNBR
960 2098 8DCA20         STA LNMULT
970 209E 20BE20         JSR MULTX2
980 20A1 20BE20         JSR MULTX2
990 20A4 6DC820         ADC LINNBR
1000 20A7 8DC820         STA LNNBR
1010 20AA 9006           BCC SECOND
1020 20AC 18             CLC
1030 20AD EECB20         INC LNMULT+1
1040 20B0 B0D0           BCS ERROR
1050 20B2 ADCB20 SECOND  LDA LMULT+1
1060 20B5 6DC920         ADC LINNBR+1
1070 20B8 8DC920         STA LINNBR+1
1080 20BB B0C5           BCS ERROR
1090 20BD 60             RTS
1100 20BE 18     MULTX2  CLC
1110 20BF 0EC820         ASL LINNBR
1120 20C2 2EC920         ROL LINNBR+1
1130 20C5 B0BB           BCS ERROR
1140 20C7 60             RTS
1150 20C8 00     LINNBR  .BYTE $00,$00
1150 20C9 00
1160 20CA 00     LNMULT  .BYTE $00,$00
1160 20CB 00
1170 20CC 0A     INCR    .BYTE $0A
```


★ ★

OS65D EXTENDED MONITOR FIXES AND ODDS & ENDS FOR C1P MINI FLOPPY

By: Robert Madoux
1577 Poag Rd.
Edwardsville, IL 62025

As many of you know the Load, View, and Save functions of the OS65D EM do not necessarily work as supplied on some versions of the Disk Operating System (I have OS65D3.1 and 3.3). I brought this to the attention of the old OSI and was authorized to return my disk. A week later my disk was returned COD for $12, $10 recopying & $2 postage and handling. I tried my newly recopied disk and lo and behold View, Load, and Save locked up just as before. I compared track 0 through track 11 with my other disk (it was not sent to OSI) and both disks were identical. Looks like OSI has a problem with their master disk! I was wanting to use the View, Load, and Save functions so my only recourse was to look at the disassembled EM and try to fix what was wrong. The rest of this article is what I have found to fix these functions plus other useful fixes, subroutines, and miscellaneous information.

VIEW, LOAD, & SAVE FIXES

View and Load have two major problems. The first is that the ACIA is not accessed at all. The second problem is that the keyboard is not accessed properly. The fix for the first problem is to change the subroutine at $1F0D to the DOS input device #1 subroutine - the ACIA routine at $24F6. See listing #1 lines 40 and 50 for this change. The keyboard is a bit more complex to fix than the ACIA was. First the keyboard access routine has been rewritten. See lines 70 through 150 in listing #1 for the keyboard access changes. For a good example of the keyboard access routine see the OSI graphics manual. Just a reminder that this article is written for a C1P or Superboard, other OSI computers use different hardware for the keyboard. The five NOP's remove a second access to the keyboard that is not needed. The change at $1E9C also fixes the Save routine so that it will work properly (it did to a certain extent on OS65D3.3); this change is also part of the keyboard fix. The rest of the keyboard fix just changes pointers. See listing #1 lines 70 through 280 for the complete keyboard fix.

View has a minor problem, it accesses the keyboard twice for every read to the ACIA. All that is done is remove the extra keyboard subroutine and move the jump routing below it up in its place. See lines 310 and 320 in listing #1.

EXTRA BONUS

The way the fix above works is to check the keyboard for a pressed space bar just before outputting any character to the Operating System output devices, i.e., CRT driver, ACIA, any & all selected devices. The extra bonus is that all EM functions that use the O.S. output devices can be terminated by pressing the space bar. Those major functions are Dump, Q (disassembler), Save, Load, and View.

There is no need to press the "L" key to terminate Save, pressing the space bar is sufficient. Load may terminate if there is a checksum error in which "ERR" will print or if a character is detected that is not an ASCII encoded hex digit in which only a "?" will print. In either case, just rewind the tape a little and press "L" and the RETURN key. One last note on Load. The disk Load will not respond to "$XXXXG" at the end of a checksum program, that is to say, it is not able to be a self starting program.

BREAK POINT PROBLEM

Ever get bit (pun intended) by the Break Points? You enter Break Points in your program trying to debug it and upon reentering the EM, after running your program, you find the Break Point table has been reset to $FFFF's. The problem is that some Break Points may still be set in your program.

TWO FIXES

If you reenter the EM at $1738 the Break Point table will not be destroyed. To do this, type (after A*) GO 1738 instead of RE EM. If you wish to make "RE EM" occur at $1738 instead of $1700, make the following change. The byte of $00 at $1721 should be changed to $38 and then saved back on disk via listing #2. See lines 370 and 380 in listing #1. Now when the EM is first loaded and run, the Break Point table will be set to $FFFF's, subsequent reenters will not change the table. Even if you enter the ASM first and then "RE EM", for the first time, it will still take one pass at the EM for this change to work. It works easier than it sounds. The only problem with this fix is that the Assembler Editor also uses the same memory that the EM uses for Break Point storage. If you use the ASM, then the Break Point table will be destroyed, if you don't use the ASM then the BP table will stay intact. If you do use the ASM then eliminate all Break Points by using the E command (E1,E2, ...E8 as needed). Now the ASM can be entered and used. When the EM is to be used again, then enter it $1700 to clear the break table. By the way, the BP table is 0-page locations $D8 through $FF. Locations $F0 through $FF store the address of the Break Point. Locations $D8 through $DF store the byte that is replaced with $00 for the Break Point. For those of you that don't like this plan (for the Break Point fix), I have plan B. If the ASM and EM are loaded, try the command !RE BA. You will note that error #7 is printed. This is the syntax error message. What has happened is that the EM (also the ASM) loads the Basic reentry vector to the error message. What plan B does is to change that RE BA vector from an error message to - you guessed, to $1738. RE EM will be left to the original $1700 start that will clear the Break Point table. RE BA will enter the EM at $1738 where the BP table will not be reset. There will be no problems when Basic is reloaded as it will reset all the reentry vectors. Basic will set the RE EM and RE ASM vectors to the error #7 message. To implement this fix, make the following changes. The byte #$2A at $1726 change to #$17 and the byte #C0 at $172B to #$38. Use Plan A or Plan B but not both. See lines 410 through 440 in listing #1 for Plan B.

MAKE THE CHANGES PERMANENT

IF you would like to make the change permanent, use the listing #2 for reference. It would be wise to do this on a duplicate disk first. I believe this method is free from errors as I have done this on both OS65D3.1 and 3.3.

ODDS & ENDS ON THE EM

The maximum number of data bytes per line to be transmitted, using the Save function, is stored at $1E31. As supplied, it is #$18 but can be changed to anything from #$01

(why, I don't know) to #$80.

Need a subroutine to do a carriage return and a line feed to all selected devices? Try the one at $1A56.

How about a subroutine to print one space to all selected devices? Try $1BAF.

Need more spaces? Load the X-register with the number of spaces you need (up to #$FF) and JSR $19C8.

Do you use the disassembler and want more or less lines disassembled than what is currently used? Then try changing the data at $18DA from #$16 to whatever you need, up to #$80.

CREDIT WHERE CREDIT
IS DUE DEPT.

I never could have found the bugs in the system without the help of Software Consultants Disassembly Manual 3.2 of the Disk Operating System. Version 3.1 that I have, and the 3.2 manual are similar enough that it was easy to see how the DOS works. Many thanks to their superb documentation.

```
 10                    ;LISTING #1
 20                    ;
 30                    ; ACIA ACCESS
                       ROUTINE
 40 1F0D               * = $1F0D
 50 1F0D 20F624        JSR $24F6
 60                    ;
 70                    ; KEYBOARD ACC-
                        CESS ROUTINE
 80 1F18               * = $1F18
 90 1F18 A9FD          LDA #$FD
100 1F1A 8D00DF        STA $DF00
110 1F1D AD00DF        LDA $DF00
120 1F20 C9EF          CMP #$EF
130 1F22 D003          BNE SP
140 1F24 4C3817        JMP $1738
150 1F27 60            SP RTS
160                    ;
170 1F10               * = $1F10
180 1F10 EA            NOP
190 1F11 EA            NOP
200 1F12 EA            NOP
210 1F13 EA            NOP
220 1F14 EA            NOP
230 1E9C               * = $1E9C
240 1E9C 20181F        JSR $1F18
250 1EA0               * = $1EA0
260 1EA0 4C4323        JMP $2343
270 17A0               * = $17A0
280 17A0 4C9B1E        JMP $1E9B
290                    ;
300                    ; VIEW - MINOR
                         FIX
310 1EA6               * = $1EA6
320 1EA6 4CA31E        JMP $1EA3
330                    ;
340                    ; BREAK POINT
                         FIX
350                    ;
360                    ; PLAN "A"
370 1720               * = $1720
380 1720 A938          LDA #$38
390                    ;
400                    ; PLAN "B"
410 1725               * = $1725
420 1725 A938          LDA #$38
430 172A               * = $172A
440 172A A917          LDA #$17
```

LISTING #2

```
THIS IS HOW TO SAVE THE EM
                    BACK ON DISK.
THIS IS FOR THE 5 1/4 IN DISK.
FIRST EXIT EM BY TYPING "EXIT",
(THIS IS THE SAFE WAY)
AND THEN DO ONE OF THE FOLLOW-
                           ING:

FOR OS65D3.1
(AFTER "A*") SAVE 10,1=1700/8
(AFTEER"A*") SAVE 11,1=1F00/4

FOR OS65D3.2
I DO NOT HAVE THIS VERSION.
SOMEONE ELSE WILL HAVE TO
                     CHECK THIS.
I DO HAVE A HUNCH THAT IT IS
THE SAME AS OS65D3.1 BUT NOT
                           SURE.

FOR OS65D3.3
(AFTER "A*") SAVE 09,1=1200/8
(AFTER "A*") SAVE 10,1=1A00/8
```

★

NEW DBI BOARDS

by: Tom Stover
Star Route
Gering, NE 69341

Readers of Peek(65) must have seen recent advertisements for a new multi-processing board made and sold by DBI, P. O. Box 7276, Denver, CO 80207. Their ads claim multi-user capability with little speed degradation.

It's true - and they work. I installed three of the boards in a C3-B several months ago and just added a fourth board connected to a modem a couple of weeks ago. There is virtually no degradation regardless of the number of users or the application being run. Even with intensive disk operations, there is almost no perceptible speed loss.

The user of this system is a multi-division company running custom-written multiple general ledgers, seven different accounts receivable applications, and inventory control -simultaneously, including using record lock-out techniques capable within 65U. They have used their OSI for over two years but had been generally unhappy with the speed degradation under the "old" time-sharing method. Individual file sizes run to several million characters. An Alloy Engineering tape back-up system is also used daily. Two serial printers are run at 9600 baud. There is no degradation in printing speed, either.

The physical changes required to implement these new boards involve pulling all the old memory, processor and "multi-user" serial boards out of the system - just leaving the disk and tape controllers. Each user gets his own processor/ memory board - simply connected to a terminal or modem via a three wire RS-232 connection. Each board has DIP switches to set the baud rate and board user number. The documentation is very clear and each board can be set up in less than a minute. No soldering, jumpers, etc. with the possible exception of a resistor on the back- plane- missing from some systems from OSI. Each board has 64K memory (not all usable in your programs, though) running its 6502 at 2 MHz.

The software changes are a little more involved - but still "not bad." I have had no problems with assembly language programs such as Sanders Sort, although the tape back-up programs must be run under "single- user" mode. There are a number of new FLAG commands added which greatly simplify the 65U paging techniques, for example, but do require the elimination of the traditional "pokes" in your programs. DBI provides a list of changes necessary in 65U utilities such as the editor, etc. If you already own Version 1.43, you ought to be able to coax DBI into providing the changes already made on disk!

Support is excellent. Since we had an older computer - and the fear of new operating system glitches had kept us several versions behind, we encountered an unforeseen problem. The 72 meg hard disk directory wasn't the same as it is under newer versions and it needed to be changed. Although they had not encountered this problem before, Art Hughes, the board designer, spent three hours on the phone with me one Saturday afternoon helping get the problem resolved. I had one defective board which was replaced with a new one.

These new boards are breathing new life into the "old" OSI hard disk systems. Especially since OSI has failed to provide upgrades to the new Turbodos versions, the DBI boards are an excellent choice. The beauty of the 65U file system with the FIND command is hard to match; applications can be written which are virtually impossible to write under any other operating system-language of which I am aware. These boards provide the upgraded speed that has been lacking in multi-user systems as well as continuing the record and file locking capability. The old OSI equipment with new boards is still viable for serious small business applications. Even more impressive, they mean that even lowly C2-OEM's could be upgraded to multi-user systems!

★

OS-65U SMART TERMINAL PROGRAM

By: Rick Trethewey
8 Duran Court
Pacifica, CA 94044

In the June issue of PEEK(65), Al again asked for articles and further asked authors to describe their programming

techniques. I wrote this article in part to respond to that request. As some of you may know, I am the SYSOP of the Ohio Scientific Special Interest Group on CompuServe. There, we exchange programs, help each other with problems, and discuss ideas for new ventures. CompuServe users can access a database of programs and articles, post messages on a bulletin board, and even talk directly with other members from every state in the union and Canada. The immediacy of this medium allows for a rapid flow of information so we can often get the answers to questions overnight. I think this service compliments PEEK(65) quite well. PEEK(65) can cover a given subject in depth and provide much needed information, but it is more convenient to use a bulletin board to ask a question or discuss an idea without the time lag of a monthly publication. So, I am submitting this article to help PEEK(65) readers running OS-65U gain more use from not only CompuServe, but other bulletin boards.

TRM65U allows capturing of received data in OS-65U data files and to transmit BASIC programs. TRM65U supports the standard ASCII X-ON (<CTRL> 'Q') and X-OFF (<CTRL>'S') in both directions to prevent loss of data. There are two parts to this program. The first part is the executive program, written in BASIC, which handles initialization and all disk communication. The second part is the machine code which handles communication between the user and the host system. TRM65U runs on either video or serial systems with 48K of RAM. TRM65U DOES NOT support the UTI board, however. I made an attempt to configure this program for 32K, but there just isn't enough room.

To begin, you will need to create four files on an OS-65U diskette. Name the first file "TRM65U" and make it at least 4000 bytes long. Name the second file "TRMCOD" and make it at least 2048 bytes long. Name the third file "DISK" and make it 3500 bytes long. Name the final file "PRINT" and make it 4096 bytes long. The executive program in BASIC goes in the file TRM65U. The machine code goes in the file TRMCOD. And finally, the disk read/write utility program goes in the file DISK. The program DISK is a general purpose utility program that

TRM65U

```
10 POKE133,159:CLEAR:REM- RESERVE HIGH MEMORY
20 K0=0:K1=1:K2=2:K3=3:K4=4:K5=5:K6=6:K7=7:K8=8:K9=9:KT=10
30 PG=256:F1=PG^1:F2=PG^2:F3=PG^3:O1=16:FS=275968
40 DIMF$(64),FT(64),FA(64),FL(64):CB=9889
50 BUFFER=KT*4096:POKE2073,96:REM BUFFER=$A000  DISABLE ^C
60 LE=3584:U1SER=PEEK(8778):U2SER=PEEK(8779)
70 TA=11*4096:TH=INT(TA/PG):TL=TA-TH*PG
80 DR$="A":GOSUB5020:FORK=K1TOEC:IFF$(K)="TRMCOD"THEN100
90 NEXTK:PRINT"CAN'T FIND TERMINAL CODE":GOTO140
100 DA=FA(K):K=EC:NEXTK:RA=11*4096:RW$="R":NB=2048
110 GOSUB6010:INPUT"PRESS <RETURN> WHEN CARRIER DETECTED ";Y$
120 POKE8778,TL:POKE8779,TH:CM=K0
130 X=USR(CM)
140 IFX=K0THENPOKE133,191:GOTO50040
150 IFX=K1THEN1010:REM TURN SAVE ON
160 IFX=K2THEN2010:REM TURN SAVE OFF
170 IFX=K3THEN3010:REM WRITE OUT TRACK FOR "SAVE"
180 IFX=K4THEN4010:REM BUFFER DIRTY ON QUIT
181 IFX=K5THEN7010:REM SELECT FILE FOR TRANSMISSION
182 IFX=K6THEN8010:REM FETCH NEXT TRACK FOR TRANSMISSION
190 :
200 X=K0:PRINT"UNRECOGNIZED RETURN COMMAND":GOTO140
210 :
1000 REM- TURN SAVE ON
1010 PRINT:PRINT"SELECT FILE TO SAVE IN":PRINT:GOSUB5010
1020 PRINT:PRINT" DATA FILES AVAILABLE"
1025 PRINT" ---- ----- ---------"
1030 X=K0:FORK=K1TOEC:IFFT(K)<>K0THEN1050
1040 PRINTTAB(X*13);F$(K);:X=X+K1:IFX=K5THENX=K0:PRINT
1050 NEXTK:IFX<>K0THENPRINT
1060 PRINT:INPUT"YOUR CHOICE ";F$:IFLEN(F$)>K6THEN1060
1070 IFLEN(F$)<K6THENF$=F$+" ":GOTO1070
1080 FORK=K1TOEC:IFF$(K)=F$ANDFT(K)=K0THEN1110
1090 NEXTK:PRINT:PRINT"FILE NOT FOUND":PRINT
1100 CM=1+O1:GOTO130
1110 DA=FA(K):FL=FL(K)+DA:RA=BUFFER:NB=LE
1120 IF(FL-DA)<(LE)THENPRINT"FILE TOO SMALL":CM=1+O1:GOTO130
1130 CM=K1:PRINT"SAVE IS NOW ON":PRINT:GOTO130
1140 :
2000 REM- TURN SAVE OFF
2010 RW$="W":GOSUB6010:PRINT
2020 CM=K2:PRINT"SAVE IS NOW OFF":PRINT:GOTO130
2030 :
3000 REM- WRITE OUT DISK BUFFER FOR SAVE
3010 RW$="W":GOSUB6010:DA=DA+LE:CM=K0:IFDA>FLTHENCM=O1
3020 CM=CM+K3:IFCM>127THENPRINT"FILE OVERFLOW - SAVE OFF"
3030 GOTO130
3040 :
4000 REM BUFFER DIRTY ON QUIT
4010 RW$="W":GOSUB6010:POKE133,191:GOTO50040
4020 :
5000 REM GATHER DIRECTORY FOR FILE SELECT
5010 INPUT"DRIVE (A/B/C/D)";DR$:D=ASC(DR$)
5015 IFD<65ORD>68THEN5010
5020 DEVDR$:EC=K0:FORK=K0TO64:F$(K)="":NEXTK:GOSUB5240:OF=O1
5050 GOSUB6010
5060 RT=RA+OF:EC=EC+K1:IFPEEK(RT)=K0THEN5210
5080 N$="":FORI=K0TOK5:N$=N$+CHR$(PEEK(RT+I)):NEXTI
5090 F$(EC)=N$:TM=PEEK(RT+K8)
5110 TM=INT((TMAND28)/K4):FT(EC)=TM
5120 AD=PG*(PEEK(RT+9)+PG*(PEEK(RT+10)+PG*PEEK(RT+11)))
5130 SZ=PG*(PEEK(RT+12)+PG*(PEEK(RT+13)+PG*PEEK(RT+14)))
5140 FA(EC)=AD:FL(EC)=SZ:IFDA+SZ>HATHENHA=DA+SZ
5160 OF=OF+O1:IFOF<PGTHEN5060
5180 EA=EA+PG:OF=K0:DA=DA+PG:IFEA<ENTHEN5050
5210 EC=EC-K1:RETURN
5220 :
5230 REM  SETUP FOR DIREC* READ
5240 DA=25088:RW$="R":RA=9970:NB=PG:A=9899:S=9902
5250 OPEN"DIREC*","PASS",K1:CLOSE1
5260 EA=PG*(PEEK(A)+PG*(PEEK(A+K1)+PG*PEEK(A+K2)))
5270 ES=PG*(PEEK(S)+PG*(PEEK(S+K1)+PG*PEEK(S+K2)))
5280 EN=EA+ES:HA=EN:RETURN
5290 :
6000 REM- EXECUTE DISK READ/WRITE
6010 POKE8778,192:POKE8779,36:POKE9435,232:POKE9436,40
6020 DH=INT(DA/F3):RM=DA-DH*F3
```

Listing continued.

allows you to read from or write to disk (except track 0) to or from any RAM address. The program PRINT provides a directory and output of TRM65U data files (more on this next month). After you have created all four of these files and entered the three BASIC programs, run a DIRectory program and write down the disk address of the file TRMCOD. You'll need that address when you run DISK to save the machine code portion of TRM65U.

Next month we will address and list the machine code program TRMCOD. But for the moment, let's assume that the code has been entered, assembled and saved.

You are now ready to run the program, but first a few notes on how this all hangs together. This program really does demonstrate how I program much of the time. I tend to develop code piece-by-piece, checking each part as I go. I also try to keep many tasks uniform throughout my work so that I can take pieces from old projects and append them to new ones. TRM65U actually is a translation of a similar program I wrote for OS-65D. Since BASIC is virtually identical in both operating systems, this worked out fine.

```
6030 DM=INT(RM/F2):RM=RM-DM*F2
6040 DL=INT(RM/F1):RM=RM-DL*F1
6050 POKECB+K1,RM:POKECB+K2,DL:POKECB+K3,DM:POKECB+K4,DH
6060 POKECB+K5,NB-INT(NB/PG)*PG:POKECB+K6,INT(NB/PG)
6070 POKECB+K7,RA-INT(RA/PG)*PG:POKECB+K8,INT(RA/PG)
6080 RW=K0:IFRW$="R"THENRW=K0
6090 IFRW$="W"THENRW=K1
6100 ER=USR(RW):POKE8778,TL:POKE8779,TH:IFER=K0THENRETURN
6110 GOTO50000
6120 :
7000 REM- SELECT FILE FOR TRANSMISSION
7010 PRINT:PRINT"SELECT FILE TO SEND":PRINT:GOSUB5010
7020 PRINT:PRINT"BASIC FILES AVAILABLE"
7025 PRINT" ----- ----- ---------"
7030 X=K0:FORK=K1TOEC:IFFT(K)<>K1THEN7050
7040 PRINTTAB(X*13);F$(K);:X=X+K1:IFX=K5THENX=K0:PRINT
7050 NEXTK:IFX<>K0THENPRINT
7060 PRINT:INPUT"YOUR CHOICE ";F$:IFLEN(F$)>K6THEN7060
7070 IFLEN(F$)<K6THENF$=F$+" ":GOTO7070
7080 FORK=K1TOEC:IFF$(K)=F$THEN7110
7090 NEXTK:PRINT"FILE NOT FOUND":PRINT
7100 CM=K4+O1:GOTO130
7110 IFFT(K)<>K1THENK=EC:NEXTK:PRINT"NOT A BASIC FILE":GOTO7100
7120 DA=FA(K):FL=FL(K):RA=BUFFER:NB=LE
7121 IFFL<NBTHENNB=FL
7130 K=EC:NEXTK:RW$="R":GOSUB6010:FL=FL-NB:CM=K4:GOTO130
7140 :
8000 REM FETCH NEXT TRACK FOR TRANSMISSION
8010 DA=DA+NB:IFFL<NBTHENNB=FL
8020 FL=FL-NB:RW$="R":GOSUB6010:CM=K5:GOTO130
8030 :
50000 REM COMMON EXIT ROUTE
50010 REM  ERROR PRINTER
50020 T=PEEK(9832):IFT>127THENT=T-124:IFT>63THENT=T-58
50030 PRINT"** DEVICE ";CHR$(65+T);" ERROR #";ER;" AT";DA
50040 POKE 8778,U1SER:POKE8779,U2SER:REM- RESTORE USR(X)
50050 POKE2073,76:END
```

However, writing machine code to handle the disk drives under OS-65U was a bit beyond my experience, so I relied on BASIC to handle that. Fortunately, BASIC makes provisions for passing parameters back and forth between machine code and BASIC programs. This lets one part tell the other part what to do and what has happened. You should be aware of a couple of things about the SAVE function of the program. First of all, the destination file MUST be an OS-65U data file. This is a safety measure and could be removed from the program by deleting the check of "FT(K)" in the IF statement in line 1080 of the executive program. Second, TRM65U saves data one track at a time, so all files to be used must be at least 3584 bytes long and for maximum efficiency, the file should be a multiple of 3584 in length as well. Last, TRM65U masks off bit 7 when it saves data so machine code transfers will require extra code. The file transmit routine is set up to only send and translate BASIC programs, but most users should have no trouble in adding the ability to also transmit other file types. While the program is running, you have 4 local commands. They are:

<CTRL>'B' - Exit the program and stop.
<CTRL>'D' - Toggle the duplex.
<CTRL>'I' - Toggle "SAVE" on and off.
<CTRL>'O' - Initiate file send routine.

The program has its own keyboard polling routine for video systems which is an adaptation of the routine from OS-65D V3.3, but without the latches for BASIC and 3.3's video driver and the delay loop has been shortened. The HEX-decimal conversion routine has 24-bit accuracy and could be used by those who want to get ambitious and write the disk handlers in machine code.

To start communication, connect your modem to your system and run the executive program. Dial up the host computer, wait for the carrier detect light to light on the modem and when it does, press <RETURN> (video systems may have to hit 2 <RETURNS>). You should now be able to talk to the host system. I hope you enjoy using this program and if you have any questions, please feel free to contact me on CompuServe either via EMAIL to 70001, 1265 or in OSI SIG on page PCS-125.

DISK

```
10 REM- *** OS-65U DISK READ/WRITE UTILITY      ***
20 :
30 REM- SAVE USER'S USR(X) ADDRESS
50 UL=PEEK(8778) : UH=PEEK(8779)
60 :
70 REM- SET UP DOS READ/WRITE VECTOR
90 POKE8778,192 : POKE8779,36  :REM- $24C0
100 :
110 REM- SET UP ISR PUT IN SUBROUTINE
130 POKE 9432,243 : POKE 9433,40
140 :
150 REM- SET UP ISR GET IN SUBROUTINE
170 POKE 9435,232 : POKE9436,40
180 :
190 CB=9889:REM- CONTROL BLOCK  $26A1
210 Q=256:REM- ONE PAGE
220 :
230 REM- GET DISK ADDRESS FROM USER
250 INPUT"ENTER DISK ADDRESS FOR READ/WRITE";DA
260 :
270 REM- GET NUMBER OF BYTES FROM USER
290 PRINT"HOW MANY BYTES ARE TO BE READ/WRITTEN":GOSUB840:NB=A
300 :
310 REM- GET RAM ADDRESS FROM USER
330 PRINT"FOR THE MEMORY ADDRESS":GOSUB840:RA=A
340 :
350 REM- GET OPERATION TYPE FROM USER
370 INPUT"READ OR WRITE (R/W)";RW$: RW$=LEFT$(RW$,1)
380 IF RW$ <> "R" AND RW$ <> "W" THEN PRINT : GOTO 370
390 :
400 REM- CHECK DA, RA, AND NB FOR VALIDITY
420 IF DA < 0 OR DA > 275967 THEN 250
430 IF RA < 0 OR RA > 65536 THEN 330
440 IF NB < 0 OR NB > 65536 THEN 290
450 :
460 REM- NOW PERFORM CALCULATIONS FOR OPERATION
480 DH=INT(DA/16777216) : RM=DA-DH*16777216
490 DM=INT(RM/65536)    : RM=RM-DM*65536
500 DL=INT(RM/256)      : RM=RM-DL*256
510 :
520 POKECB+1,RM : POKECB+2,DL : POKECB+3,DM : POKECB+4,DH
540 POKECB+5,NB-INT(NB/Q)*Q : POKECB+6,INT(NB/Q)
560 POKECB+7,RA-INT(RA/Q)*Q : POKECB+8,INT(RA/Q)
570 :
580 REM- NOW DO IT
600 IF RW$ = "R" THEN RW = 0
610 IF RW$ = "W" THEN RW = 1
630 DEV "A"
650 ER = USR (RW)
660 :
670 REM- CHECK FOR ERRORS
690 IF ER THEN GOSUB810
700 :
710 REM- RESTORE USER'S USR(X) VECTOR
730 POKE 8778,UL : POKE 8779,UH
740 :
750 REM- RESTORE OLD <CTRL> 'C' STATUS
770 POKE 2073,CC
780 :
790 END
800 :
810 PRINT"*** DEVICE A ERROR #";ER;" AT ADDRESS";DA
820 PRINT:RETURN
830 :
840 PRINT"ENTER THE DECIMAL VALUE OR HEX VALUE PRECEEDED"
850 INPUT"BY A '$' ";A$:IFLEFT$(A$,1)="$"THEN880
860 FORX=1TOLEN(A$):C$=MID$(A$,X,1):IFC$<"0"ORC$>"9"THEN840
870 NEXTX:A=VAL(A$):RETURN
880 A=0:IFLEN(A$)<2THEN840
890 FORX=2TOLEN(A$):C$=MID$(A$,X,1):IFC$<"0"THEN840
900 IFC$<="9"THENA=A+VAL(C$)*(16^(LEN(A$)-X)):GOTO930
910 IFC$<"A"ORC$>"F"THEN840
920 A=A+(ASC(C$)-55)*(16^(LEN(A$)-X))
930 NEXTX:RETURN
```

Next month the source code and installation instructions.

"PRINT" Listing on page 16.

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

```
10 X=PEEK(133)-14:POKE133,X:CLEAR
20 K0=0:K1=1:K2=2:K3=3:K4=4:K5=5:K6=6:K7=7:K8=8:K9=9:KT=10
30 PG=256:F1=PG^1:F2=PG^2:F3=PG^3:O1=16:FS=275968
40 DIMF$(64),FT(64),FA(64),FL(64):CB=9889
50 BUFFER=(PEEK(133)+K1)*PG
60 LE=3584:U1SER=PEEK(8778):U2SER=PEEK(8779)
70 DV=PEEK(11686):PD=K1:IFDV=K1THENPD=K5
80 PRINT"FILE PRINTER":PRINT:GOSUB290
90 PRINT:PRINT" DATA FILES AVAILABLE"
95 PRINT" ---- ----- ---------"
100 X=K0:FORK=K1TOEC:IFFT(K)<>K0THEN120
110 PRINTTAB(X*13);F$(K):X=X+K1:IFX=K5THENX=K0:PRINT
120 NEXTK:IFX<>K0THENPRINT
130 PRINT:INPUT"YOUR CHOICE ";F$:IFLEN(F$)>K6THEN130
140 IFLEN(F$)<K6THENF$=F$+" ":GOTO140
150 FORK=K1TOEC:IFF$(K)=F$ANDFT(K)=K0THEN190
160 NEXTK:PRINT:PRINT"FILE NOT FOUND":PRINT
170 GOTO720:REM QUIT
180 :
190 ZZ=K:DA=FA(K):AA=DA:RA=BUFFER:NB=LE:K=EC:NEXTK
200 INPUT"PRINTER OR CONSOLE OUTPUT ";Y$
210 IFLEFT$(Y$,K1)="P"THENDV=PD
220 GOSUB580
230 FORK=BUFFERTOBUFFER+LE-1
240 C=PEEK(K):IFC=255THEN720
250 PRINT#DV,CHR$(C);:NEXTK
260 AA=AA+LE:DA=AA:IFAA<(FA(ZZ)+FL(ZZ))THEN220
270 GOTO720
280 :
290 INPUT"DRIVE (A/B)";DR$:IFDR$<>"A"ANDDR$<>"B"THEN290
300 DEVDR$:EC=K0:FORK=K0TO64:F$(K)="":NEXTK
310 GOSUB520
320 OF=O1
330 GOSUB580
340 RT=RA+OF:EC=EC+K1
350 IFPEEK(RT)=K0THEN490
360 N$="":FORI=K0TOK5:N$=N$+CHR$(PEEK(RT+I)):NEXTI
370 F$(EC)=N$
380 TM=PEEK(RT+K8)
390 TM=INT((TMAND28)/K4):FT(EC)=TM
400 AD=PG*(PEEK(RT+9)+PG*(PEEK(RT+10)+PG*PEEK(RT+11)))
410 SZ=PG*(PEEK(RT+12)+PG*(PEEK(RT+13)+PG*PEEK(RT+14)))
420 FA(EC)=AD:FL(EC)=SZ
430 IFDA+SZ>HATHENHA=DA+SZ
440 OF=OF+O1
450 IFOF<PGTHEN340
460 EA=EA+PG
470 OF=K0
480 DA=DA+PG:IFEA<ENTHEN330
490 EC=EC-K1:RETURN
500 :
510 REM  SETUP FOR DIREC* READ
520 DA=25088:RW$="R":RA=9970:NB=PG:A=9899:S=9902
530 OPEN"DIREC*","PASS",K1:CLOSE1
540 EA=PG*(PEEK(A)+PG*(PEEK(A+K1)+PG*PEEK(A+K2)))
550 ES=PG*(PEEK(S)+PG*(PEEK(S+K1)+PG*PEEK(S+K2)))
560 EN=EA+ES:HA=EN:RETURN
570 :
580 REM- EXECUTE DISK READ/WRITE
590 POKE8778,192:POKE8779,36:POKE9435,232:POKE9436,40
600 DH=INT(DA/F3):RM=DA-DH*F3
610 DM=INT(RM/F2):RM=RM-DM*F2
620 DL=INT(RM/F1):RM=RM-DL*F1
630 POKECB+K1,RM:POKECB+K2,DL:POKECB+K3,DM:POKECB+K4,DH
640 POKECB+K5,NB-INT(NB/PG)*PG:POKECB+K6,INT(NB/PG)
650 POKECB+K7,RA-INT(RA/PG)*PG:POKECB+K8,INT(RA/PG)
660 ER=USR(0)
670 IFER=K0THENRETURN
680 REM
690 REM  ERROR PRINTER
700 T=PEEK(9832):IFT>127THENT=T-124:IFT>63THENT=T-58
710 PRINT"** DEVICE ";CHR$(65+T);" ERROR #";ER;" AT";DA
720 POKE 8778,U1SER:POKE8779,U2SER:REM- RESTORE USR(X)
730 MEMSIZ=PEEK(133)+14:POKE133,MEMSIZ
740 POKE2073,76:END
```

★ ★

OPTIMIZING MICROSOFT BASIC PROGRAMS - PART I

By: Dave Rich
Courtesy of Osmosus News
Minneapolis, MN 55418

OSI BASIC, like all Microsoft BASICs, has one thing in common with all interpreted languages; it is slow. As a result, many articles and letters have appeared in computer magazines showing ways to wring the most speed out of Microsoft BASIC, that is, how to optimize the speed.

If you are an expert BASIC programmer, this series of articles may not contain anything new; but for those new to BASIC, I would like to point out some ways that OSI BASIC programs can be written to make them run faster and, often, take up less memory.

Optimization of computer code has two facets, which often do not complement one another: speed of execution and memory size. This article will explore ways to achieve the fastest execution times for your programs. Often, the same techniques will reduce memory size as well.

BRANCHING (or, how do we GOTO there from here?)

Microsoft BASIC has a fairly limited means of locating a line number referenced on a GOTO or GOSUB. At the beginning of each line in memory is encoded the current line number and address of the start of the following line. This creates a one-direction chain, or 'singly-linked' list, for use in locating lines. When Microsoft BASIC encounters a GOTO or GOSUB, BASIC must begin the search from the first line of the program. This is true even if the branch is to the current line.

The above search sequence has many implications. If BASIC has to chase down a list of line numbers, the fewer line numbers to check the better. This leads to these suggestions:

1. Put all high-use subroutines, those called often and from all over, at the very beginning of the program, even ahead of the main program code.

2. Use multiple-statement lines wherever possible. This one also reduces memory size.

3. Avoid REM-only lines. Their line numbers have to be checked too. This also reduces memory size. (Put your documentation on a separate piece of paper, or keep a documented version in addition to the 'run-time' version).

4. Put low-use, one-time-only subroutines at the end of the program. The loss in speed when calling them is less important than avoiding having to search past them on more frequent branches.

5. If you have code which requires repeated execution, i.e., a loop, use FOR...NEXT. No line search is done. If a loop is to be executed a variable number of times, dependent on some variable within the loop (a WHILE loop), try something like the following:

```
FOR X= 0 TO -1 STEP -1
--
--
X=(T>25):REM WHILE T<=25
NEXT X
```

(Incidentally, spaces, as in the above, should be avoided!)

GOTO a REM

Many programmers have a habit of using REM statements as the destination of branches. This causes two problems. First, removal of REMS to speed up the program can cause unexpected errors. Second, every time BASIC branches to the line it has to interpret the REM and scan over the rest of the line. Avoid branches to REM lines, branch to executable code.

Next month, we will discuss constants, structuring programs, FOR loops and string concatenation.

★

NOTES FROM OSI

OS-65U VERSION 1.44 FEATURES DISK SYSTEMS MANAGER

Ohio Scientific has unveiled OS-65U Version 1.44 to supply users with a powerful Disk Systems Manager. Completely compatible with the 1.43, it is the latest version of 65U in a long series of operating system enhancements.

The Disk Systems Manager, or Hard Disk Manager, allows a hard disk to be broken into separate areas protected from each other by the operating system. Utilities facilitate the transfer of programs and data files between systems as well as to and from floppies or through the network. The DSM will support 65U and CP/M

systems on one or multiple hard disks.

Part No. Description

SUG 1 OS-65U upgrade to current version for CD-7/28, C2-D, C3-D, 220E, 230E, and 230I. Retail Price $150.

SUG 2 OS-65U upgrade to current version for C3-C(CD-23) Retail Price $150.

SUG 3 OS-65U upgrade to current version for CD-36/74 for C3-A,C3-B,C3-C, C3-OEM,C2-OEM,220C,230C 250I,250J,250JJ Retail Price $150.

NEW KEYCALC (PLANNER PLUS) MANUAL

The new 215 page Keycalc manual is now available. This new, completely revised manual is the result of a years work in our Technical Publications Department. With this new manual, users can get started with their models almost immediately without extensive training.

All the information a new user may need to create and use a simple model can be found in the first three chapters. This new manual has an extensive tutorial section along with a reference guide for the experienced user.

The Keycalc manual is applicable for all previous releases of Planner Plus. Keycalc, together with the new manual, makes a great support tool for those users who have traditional Challenger or Masterkey 200 line systems. The SUG-5 software upgrade to Planner Plus version 4.02 includes this manual for $150.00 retail. Should you desire extra copies of just the manual, they can be purchased separately for only $50.00 retail.

AD CAMPAIGN IN FULL SWING

On March 14 Ohio Scientific's three-quarter of a million dollar advertising campaign began with weekly half page ads in the national issue of the Wall Street Journal. Following this image building thrust additional ads are scheduled to appear in Info World, Computerworld, ISO World and Inc. during the upcoming months.

The response to the Wall Street Journal ads and to the recent insertion in Computerworld has been excellent. The leads received from these ads are being forwarded to our Regional Sales Managers for distribution.

FORTRAN, COBOL AND BASIC NOW AVAILABLE FOR 300 LINE

Three new languages are now available for the 300 multiprocessing line. These languages, FORTRAN v 3.34, COBOL v 3.0 and BASIC v 4.51, were written by Microsoft and are available separately or combined in one package (MS-Languages). Ohio Scientific supplies these languages on eight inch diskettes with documentation. Running under TurboDOS, these languages offer a multitude of new opportunities.

Retail prices for these languages are:

FORTRAN	$440
COBOL	$440
BASIC	$285
MS-Languages	$825

WP-3.3 VERSION 1.01 FIX FOR LEVEL 2

In OS-65U versions 1.43 and 1.44 under Level 2, the new code to handle the "busy" signal sent by the node to the work station caused the work station to do a delay and then

resend its original message. The delay routine which was called had been overlayed by WP-3.3.

A problem would appear when the node was busy and WP-3.3 made a request from the node (i.e.,!LOAD,!PUT,!MERGE,etc.). The work station would then hang and never return from the request.

Due to the loss of this subroutine, the delay routine calls have to be changed to call another delay routine. The following fix solves this problem.

First load the WP-3 program by entering:

```
LOAD "WP-3","PASS"
```

Enter the following two new lines:

```
211 IF PEEK(16317)<>THEN 220

212 POKE 14025,184:POKE14026,
    62:POKE14044,184:
    POKE14045,62
```

Now save the WP-3 program by entering:

```
SAVE"WP-3","PASS"
```

This fix will be effective the next time WP-3.3 is used.

No, the IF PEEK (XXXX) <> THEN is not an error, we are assured.

OS-65U CD-28 CYLINDER "WRAP-AROUND" FIX

Systems utilizing cylinder offsets that accessed cylinder 254 or greater would "wrap-back" to cylinder zero. This can be fixed by installing the following correction using the CHANGE program.

To install this correction, follow the CHANGE conversation below and install the correction on BOTH the floppy based and hard disk based versions.

```
RUN"CHANGE","PASS"

OS-65U Disk Change Utility

MODE:HEX(H),DEC(D)?H
DEVice?x
ADDRESS OFFSET?C00
ADDRESS     ?   34C6
000034C6        D8?75
000034C7        A2?03
000034C8        10?6D
000034C9        A9?03
000034CA        00?34
000034CB        2C?60
000034CC        80?00
000034CD        C2?00
000034CE        9D?00
000034CF        FF?00
000034D0        C1?00
000034D1        95?00
000034D2        80?00
000034D3        CA?00
000034D4        D0?00
000034D5        F5?00
000034D6        E6?.
ADDRESS     ?   37CE
000037CE        85?A9
000037CF        03?00
000037D0    m   6D?6A
000037D1        03?4C
000037D2    4   34?C6
000037D3        60?34
000037D4        A0?X
OK
```

The CHANGE conversation is now complete. If timesharing is implemented, then make the following corrections to the program TSCD07. This correction applies to version 3.8 of TSCD07 ONLY.

```
LOAD"TSCD07","PASS"

Change line 20873 as follows.

20873 DATA 13512,044:  REM
$34C8 =BIT $3403(kill ofst)
*CD-28

SAVE"TSCD07","PASS"

OK
```

This completes the corrections to TSCD07. Should you have any questions and/or problems, please call your dealer.

WP-3.3 VERSION 1.01 FIX PRINT DEVICE #5

In OS-65U versions 1.43 and 1.44 when device #5 (the line printer) was "not ready", the system would ring the bell to alert the user. The user then had a choice of entering a <CONTROL>-C to abort the print or any other key to try the print again.

A problem would appear when this "not ready" occurred and the user typed a <CONTROL>-C to abort the print. The system would hang, trying to error out through BASIC's error routines.

In order to solve this, an error routine for aborting the print to device #5 must be added and used instead of BASIC's error routine. The following fix adds this error routine to WP-3.3 and instructs WP-3.3 to use this routine.

FIX:

First load the WP-3.3 program by entering:

```
LOAD"WP-3","PASS"
```

Enter the following new line in the program to set up the error path:

```
145POKE15945,76:POKE15946,164:
   POKE 15947,65
```

After this line is entered, the following lines should be typed in the immediate mode. These lines are the new error path for WP-3.3.

```
POKE36260,162:POKE36261,254

POKE36262,154

POKE36263,169:POKE36264,130

POKE36265,32:POKE36266,61:
  POKE36267,13

POKE36268,32:POKE36269,220:
  POKE36270,17

POKE36271,4:POKE36272,32:
  POKE36273,6
```

Now save the WP-3 program by entering:

```
SAVE"WP-3","PASS"
```

This fix will be effective the next time WP-3.3 is used.

★

READER PROFILE

ED:

My machine is a C4P with an additional OSI power supply, a D & N memory and floppy board, 32K, and MPI B-51 drive and a Shugart SA-400, a serial MX-80 printer and OSD65 V3.3.

I have been using a Commodore 64 at the school where I work and yes, the graphic and sound capabilities are neat, BUT it is a pain to use with two opens for any device and the disk seems almost as slow as my cassette used to be. I think that the OSI machines are as good or better than Commodore's for most of my applications, which revolve around my school work and a bowling league secretary program that I've written.

I have a few questions. Is it possible for me to run COBOL on my machine? I have seen the ads in Peek(65) for Technical Products' version of OSI FORTH. What are the advantages and disadvantages of this language? How much more memory can I add to my machine? What kind(s) of boards are necessary?

What about CP/M, I know that it is for Z80's but I have read about OSI with it? How compatible is it with other manufacturer's disk formats?

How hard would it be to use a different keyboard with my machine?

Is OSI going to do anything with the personal market?

Steve Rydgig
Collinsville, IL 62234

Steve:

1. I haven't heard of a COBOL which will run on a 6502 under 65D.

2. We could write a book about FORTH. It is fast, logically organized and seems to really appeal to hard-core hackers; but it is just about as hard to program in as assembler.

3. You can add memory until you get up to the locations used for I/O by any of the other boards you use (some of which are non-OSI). Where does the D+N floppy controller address the disk? Where does your serial I/O port live? My guess is you can add another 16K. D+N could tell you more and recommend boards, as could Generic and the other fine PEEK(65) advertisers.

4. CP/M runs only on 8080, 8085 and Z80 machines. It will not run on a C4P at all. It is written in 8080 Assembler, and therefore, requires a CPU which will run 8080 instructions. Your 6502 will not. OSI machines which have a Z80 can (usually) run CP/M. As to disk formats, CP/M is available in many different disk formats, but that isn't your problem.

5. I know of no other manufacturer making OSI Polled-key board compatible keyboards, and don't know enough about them to tell you how hard it would be for you to make/modify one.

6. As of now, OSI plans to continue selling/servicing its present line of "personal" computers and concentrate on new development on the "business" market, but that includes "executive personal" computers, so who knows?

7. Obviously, my answer to many of your questions is, "I dunno." I expect many readers know more than I do on many of these topics. I welcome correction of errors and disagreement with opinions!

Al

* * * * *

ED:

You asked for comments and ideas for PEEK(65) so here goes:

Any and all information about OSI and their products is eagerly consumed. Please have more. How about Cleveland Consumer Computers and Components? For a while they were selling bare boards and all sorts of OSI equipment. Are they still around and what do they have to offer OSI users?

For several years I have been corresponding with several dozen OSI users around the world. In the last few months I have received too many letters starting:

"Dear Earl,

Thanks for the program you sent, but I have sold my OSI machine and can no longer use it."

The ranks of OSIers are thinning rapidly!!

I will send you another graphics program as soon as I use up the rest of the roll of film.

Concerning programming, I use two techniques. The first is to block out ranges of line numbers with REM statements describing the function of what will be the purpose of the code in that block. This forces me to break up the total program into discrete blocks or units. Then I later fill in the code for each block.

Another technique is to write the main program first like this:

```
10 REM Program Title
100 GO SUB 1000 : REM ...
200 GO SUB 2000 : REM ...
300 GO SUB 3000 : REM ...
400 GO SUB 4000 : REM ...
500 IF FLAG=1 THEN GOTO 100 :
    REM REPEAT IF NOT DONE
600 END
```

Note, this is a general purpose do everything main program. You need only fill in the REM statements to describe what each subroutine does. It is also easy for someone else to follow. The program flow is obvious, you do not need to dig through pages of code. Now that the main program is done, fill in the required subroutine.

Earl Morris
Midland, MI 48640

Earl:

We called Cleveland Consumer Computers & Components. They still have various equipment for sale. Check their ad in Dec. 82 issue of PEEK(65). They also have some bare boards available.

Peek Staff

* * * * *

ED:

This letter is in response for your call for information on a Reader Profile.

My computer is an OSI C1P-MF, with 32K of memory. It is an early 1977 model, and is interfaced with a NCR thermo printer. I have also connected two Atari joysticks, but do not get maximum use from them, because the system sometimes "hangs" when they are used. I suspect that this may be somehow related to the incomplete description of the "AND" operator given in OSI documentation.

I write much of my own software (this was written on a highly enhanced version of the Aardvark Letter Writer, combined with the OSI Word Processor), and am very interested in other utility programs suitable for the C1P. I am mainly interested in programs for personal use. Since PEEK(65) seems to be the only publication devoted to OSI, we all depend quite heavily on you!

As to content, my highest interest is software. I don't mind ads, as long as they are paid ads. I don't feel the reader should subsidize free ads (called Product Reviews) by his subscription price. Articles on "programming techniques" are fine, as are tutorials. Hardware articles would be OK, if only the author could write them so that a non-electronics major could understand them. This goes double for those writers who like to play guessing games as to which machine they apply. Another idea might be a question and answer column, like the Readers Feedback in COMPUTE!

In short, you seem to be doing a great tightrope job in trying to satisfy a wide range of interests and skills. While some of your articles are a bit heavy, none are too basic.

Stanley Harshfield
Memphis, TN 38115

* * * * *

LETTERS

ED:

I am responding to C. D. Lombard's letter in the July issue. My guess is he should look for an IC that is not functioning when it is cold... maybe UF-9 or UC-9 (74LS138). He might try heating either of those chips with a hair dryer to see if the LOAD problem goes away sooner. Or, he might try interchanging those two chips with two others (UD-9, UE-9). Also, there is a possibility that the problem is in RAM, chips UA/B-5 thru UA/B-8.

In answer to Don Bruechert, I'm convinced that the best way to choose 300/600 baud is to cut the foil trace between pin 9 of U-63 and W5. Then, insert a SPDT switch as follows: Left switch pin to U-63 pin 11; center switch pin to W5; right switch pin to U-63 pin 9. This mod does not require any adjustment of R57. However, the length of any line being SAVEd must be 1 less than the screen width because the CPU doesn't have time to scroll the screen

OSMOSUS provides users of Ohio Scientific computers a facility through which they may exchange information, hardware and software ideas; or otherwise improve their understanding of computers an their ability to use them. Members' interests range from beginner to hard-disk systems. A monthly newsletter provides technical exchange and short articles of interest to members. Recent topics included *"Disk RPM Timer," "OSI ROM Routines* (8 parts)," and *"The Unofficial, Reasonably-Complete OSI Directory* (4 parts)."

OSMOSUS is interested in establishing contact, exchanging newsletters, etc., with other OSI user groups. Memberships in OSMOSUS are available to those outside our immediate area. For further information and a sample newsletter, please send a stamped, self-addressed ***BUSINESS-SIZED*** envelope to:

OSMOSUS NEWS
D. B. Baker, Editor
3128 Silver Lake Road
Minneapolis, MN 55418

before the next data is received when LOADing. For an early model C1, this means 23 characters.

Has anyone ever produced a version of Missile Command for OSI cassette machines? This is the game where you try to defend 3 cities from falling nukes.

Bruce Showalter
Abilene, TX 79601

* * * * *

ED:

By the way, Dick Bramain called you a while back regarding advice on how to change IBM 3740 disk format for CP/M to (old) OSI format. Actually, this is a project we're both working on. I read a review in the March issue of INFOWORLD about a Proxy 80 board sold by Software Federation which is said to have a switch whereby the 470/510 boards can be placed or retained in the card cage along with the new 4 mhz Z80 board, so that old (e.g. DMS) software can still be run. But the article was a bit vague since they might just as well have meant something else. I'm intensely interested because I have too much highly customized software that runs under the 6502 CPU to switch to the D+N 80 board - and swapping boards constantly is not appealing - particularly since I don't trust the strength of the backplane (& 48 pin Molex connectors). Have you heard anything about this, and can you elaborate? I'm confused by the article primarily because OSI data transfer is asynchronous and IBM data transfer is snychronous - I heard someplace, and that was the major reason for incompatibility.

Fred S. Schaeffer
Jamaica, NY 11435

Fred:

Many questions!

1. Many people (including Westico) can copy IBM<-->OSI disk format for a small fee per disk.

2. Proxy - 80 (D&N-80) boards will allow both disk controllers to be in the computer at once, though not hooked to the same disk drives at once - and can, therefore, read CP/M - IBM format and/or CP/M-OSI (old) format. However, the D&N-80 board has no 6502 chip on it, and, therefore, cannot run 65D or 65U. The 510 board (CPU board) couldn't be in the computer at the same time unless a switch disabled at least all power supply lines .....Readers?

3. OSI (and most other MICROS) data transfer between computers or between computer and terminal is asynch. Some IBM is synchronous - this has nothing to do with internal transfer "inside the box."

Al

* * * * *

ED:

The dealer I purchased my OSI C2-OEM system from went out of business so I have had difficulty obtaining much needed information about its operation.

Your Journal is of considerable help as I have spent many hours seeking facts and learning through trial and error.

I use my system primarily for preparing programs for a client who uses a OSI system. I use the 65U operating system almost exclusively. I find it useful compared to WANG BASIC that I use on a WANG 2200. I miss PRINT-USING, however.

When I use the computer long periods of time making keyboard entries, I would like to turn off the Siemans disk drives to eliminate the noise and save drive wear. I placed a switch in the AC line to the motors so I could turn them off after start-up. This disabled access to the computer so I was not able to continue. I do not know if this was caused by a transient in the AC power line when the motors were turned off or by the lack of feed-back information that might be needed by the computer to show the disk drives are functioning. I bought a SAMS service manual for the C2-OEM boards but it gives nothing about this question that I can discern. Can anyone help me with this?

Some of my programs take a long time to run, even with a 2MHz clock, due largely to many disk transfers required. I would be interested in learning if a BASIC compiler is available for my applications that might make a considerable improvement in running time.

Earle L. Kent
Elkhart, IN 46516

Kent:

Disablement of the floppies has been done many times, -successfully! We don't know precisely what you have done, but you may be picking up transients that "blow" the CPU. A cheap transient suppressor may be installed at the feed to the CPU power supply. A capacitor across the floppy switch may also do the job. In any case, the "Hot" side of the A.C. to the floppy should be switched.

A compiler is available (see PEEK ad this issue), but it has some limitations, and it appears that it is the disk that is slowing things down, not program execution. Consider programming and/or file structure changes to reduce disk accesses, i.e., read/write 10 records at a time by keeping them in variables (matrix) until 10 are accumulated. You might also "play" with the floppy stepping rate. This is accomplished by POKEing the delay (in miliseconds) into the right location. The range is generally between 3 and 9 msec. If yours is a Shugart drive, better leave it at 9. Most others run well at 5, some faster at 3. For OSU release 6/79 - POKE 11895, MS. for earlier version it must be determined if location 11895 currently contains 9. IF PEEK(11895)=9 THEN POKE 11895, MS, otherwise POKE 11886,MS.

Peek Staff

* * * * *

ED:

I often write or acquire BASIC programs that, for the purpose of readability, contain many embedded spaces. These spaces each take a byte of memory that may be needed for data. They are, moreover, not necessary for correct execution of most programs.

This program, written to run under the HEXDOS disk operating system, will eliminate all spaces from BASIC programs except those occurring in REMarks and in literals ("ed strings). The compressed source program is saved in the same file space that was used to store the original program. Each line number and the associated original (O-add) and new (N-add) execution addresses are displayed as the line is processed. At the end of the compression the number of bytes saved is displayed. CTRL C is disabled during execution of the program.

2 REM-COMPRESS FOR HEXDOS-
7/10/83 continued.

```
4 POKE530,1:PRINTCHR$(26):DIMB
  (72):INPUT"Filename:;F$
6 LOAD*4,F$:LOAD*5,F$:POKE227,
  255:AD=2817:AT=AD
8 PRINT:PRINT" Line# O-add
  N-add"
10 TA=USR(5)+256*USR(5):IFTA=0
   THEN30
12 LL=USR(5):HL=USR(5):LN=LL+
   256*HL:P=0:FL=1
14 D=USR(5):IFD=34THENFL=FL*-1
16 IFD=142THENFL=0
18 IFD=32ANDFL=1THEN14
20 P=P+1:B(P)=D:IFD>0THEN14
22 PRINTCHR$(13)LN;TAB(7)AT;
   TAB(14)AD;:AT=TA
24 AD=AD+P+4:HA=INT(AD/256):
   LA=AD-HA*256
26 PRINT#4,CHR$(LA)CHR$(HA)CHR
   $(LL)CHR$(HL);
28 FORI=1TOP:PRINT#4,CHR$(B(I)
   );:NEXTI:GOTO10
30 PRINT#4,CHR$(0)CHR$(0);:
   SAVE*4:PRINTCHR$(13)" *"TAB
   (7)AT;TAB(14)AD;
32 PRINT:PRINT:PRINTAT-AD"
   Bytes saved":POKE227,127:
   POKE530,0:end
```

Jim Hayes
Seattle, WA 98116

SECOND ANNUAL ROCKY MOUNTAIN COMPUTER SHOW AND SOFTWARE EXPOSITION

The Second Annual Rocky Mountain Computer Show and Software Exposition will be held Thursday-Saturday, Sept. 22-24, 1983, at the Denver Merchandise Mart. Show hours are 10:30 AM to 5:30 PM daily.

The show features microcomputers including business computer systems, home and personal computers, and video games, plus business, personal, educational, and entertainment software, accessories, publications, services and peripherals. Admission is $5.00 for adults and $3.00 for children. For more information, call or write Northeast Expositions, 822 Boylston Street, Chestnut Hill, MA 02167. Telephone: 800-841-7000 or 617-739-2000 (within Massachusetts).

EPROM PROGRAMMER POWER SUPPLY

The last issue of PEEK featured a construction article for a new EPROM programmer - and there will probably be another in a coming issue. There is, however, just one little hitch -

Virtually all EPROM programmers require a 26 volt supply to power the system. That all well and good if you have an 8" floppy system, because you can "steal" it from the floppy power supply.

But what about those who don't have an 8" floppy? It's back to the store for more parts ... transformer, diodes, capacitors, etc. Or is it?

PEEK wouldn't leave you in a dilemma like that! So, here is Brian's handy little circuit that makes the required 26 v. out of the readily available +12 and -5 volts that everyone has. Basically, it's a DC to DC converter with a voltage doubling output.

It is only little, but it will give sufficient drive for one EPROM at a time, but then most EPROM programmers are or can be made this way.

To make things easier for you, a 1 1/2" x 2" bare circuit board is now available through PEEK (65) - see the ad. Better yet, for those of you who cannot get to the store easily, or want the whole thing at one time, there is also a bag of parts to complete the kit.

Good luck and happy programming.

PEEK(65)

AD$

Must sell. OSI C2-8PDF (two 8-inch disk drives) complete system, keyboard, 48K RAM, BMC green monitor, Centronics printer #779. OSU, OSD V.3.3, manuals, OSI's best WP, other software. System used infrequently and in excellent condition. Original price over $5000. Will sell for $1500 or best offer. In Maryland. (301) 292-4883.

* * * * *

FOR SALE: New 1-C3OEM, $1,500.00 / 1-C3D, $3,500.00 / 3-C2OEM, $1,203.00 / 1-C2D, $2,900.00 / 2-C8P, $1,500.00 / 1-C4P 2 drive, $1,500.00. Immediate delivery, freight COD, Micro Software International, Inc., - Phone 1-800-843-9838.

* * * * *

"Need to sell: O.S.I. - C3D CPU with hard disk and Okidata printer. For more information, please call (206) 248-3377, and ask for John."

* * * * *

Souped up C8P DF with Intertec Intertube II Terminal. 2 MHz. Excellent condition. Unit has 2 Siemens 8" SSSD drives, 1 505 board, 1 CA10-X board, 3 520 boards for 48 K RAM total, 2 RS232 serial ports for terminal and printer, lots of software and blank disks, 9 notebooks full of documentation on software and hardware. Hardware, software, blank disks, and documentation for $2000.00 plus freight. Bob Bernard, 2253 Ringling Blvd., Sarasota, FL 33577. (813) 953-5363.

OSI—AFFORDABLE DATA BASE MANAGER

B&W FILE MASTER

FULL FEATURED VERSION
NOW RUNS IN 32K

B&W FILE MASTER runs under OS65D V3.3, (video only). Single or dual drive.

FEATURES: User and/or pre defined files with coding options, formatted screen viewing and inputting, find, edit, update, delete & page. 'Screen', 'quick' and 'format' dump. Manual included. only $55.00

Manual only (price applied towards purchase) $10.00

ADD ON FEATURES:
Label print option $45.00
Report generator $45.00

For more information contact:

BUNIN & WARD COMPUTER SERVICES
P.O. BOX 895 CHURCH STREET STA.
NEW YORK, NY 10008
(212) 434-5760

EPROM POWER SUPPLY

12-5=26

Provides 26 volts from available +12 and -5 volts Sufficient to drive programmer for one Eprom

Bare board	**$2.50**
Bag of parts	**$3.50**
Post & Handling	**$3.00**

Md. residents add 5% tax. Send U.S. dollars (drawn on a U.S. Bank to:

PEEK (65)
P.O. BOX 347
Owings Mills, Md. 21117
(301) 363-3268

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:

Subscription rates for 12 issues (one year), effective with the July, 1981 issue. All rates quoted in U.S. dollars. Due to U.S. bank surcharges, all funds payable to PEEK (65) must be in **U.S. Dollars** and be drawn on a **U.S. Bank** or be an **International Money Order.**

Please fill out and return with check or money order.

( ) $15.00 Enclosed. U.S. (Maryland residents add 5% sales tax)
( ) $23.00 Enclosed. Canada and Mexico, 1st class, surface.
( ) $35.00 Enclosed. South and Central America. Air Mail.
( ) $35.00 Enclosed. Europe. Air Mail.
( ) $40.00 Enclosed. All other. Air Mail.

NAME.....STREET.....

CITY.....STATE .....

ZIP CODE.....COUNTRY.....

Please send the following back issues. I enclose:

( ) $2.00 ea. U.S. Surface. (MD. residents add 5% sales tax.)
( ) $2.50 ea. Canada and Mexico. Surface.
( ) $3.00 ea. South and Central America. Surface.
( ) $3.00 ea. Europe. Surface.
( ) $3.50 ea. All other. Surface.

Vol 1. 1980

( ) JUN #6 ( ) SEP #9 ( ) NOV #11( ) DEC #12

Vol 2. 1981

( ) JAN #1 ( ) FEB #2 ( ) MAR #3 ( ) APR #4 ( ) MAY #5 ( ) JUN #6
( ) JUL #7 ( ) AUG #8 ( ) SEP #9 ( ) OCT #10 ( ) NOV #11 ( ) DEC #12

Vol 3. 1982

( ) JAN #1 ( ) FEB #2 ( ) MAR #3 ( ) APR #4 ( ) MAY #5 ( ) JUN #6
( ) JUL #7 ( ) AUG #8 ( ) SEP #9 ( ) OCT #10 ( ) NOV #11 ( ) DEC #12

Vol 4. 1983

( ) JAN #1 ( ) FEB #2 ( ) MAR #3 ( ) APR #4 ( ) MAY #5 ( ) JUN #6
( ) JUL #7 ( ) AUG #8

INDEXES ARE INCLUDED IN THE JAN. & DEC. 1981 AND DEC. 1982 ISSUES.