April, 1986
Volume 7, No. 4

PEEK [65]

The Unofficial OSI Journal

Inside This Month

Column One

I'm finally becoming comfortable with publishing. I don't mind telling you that it's a lot of work and requires attention each and every day, but I love it.

My goals for PEEK[65] remain unchanged. I want to help move the OSI 8-bit community out of the dark ages and into the 1980's... like maybe 1982 or so, anyway. Fortunately for all of us, PEEK has steadfastly let us keep on being a community through some very hard times.

We are, and have always been, a very diverse community. There have always been the divisions between C1P's, C4P's, C8P's, and the serial systems. But beyond those hardware related differences, our needs have always been different too. There remains a steady stream of mail from people who want elementary programming instructions. In addition, the serial system people beg for anything on the innards of OS-65U.

By the same token, we share many common uses for our systems. Word processing is at the top of that list. The Reader Survey published last month has shown that. See the article in this issue for some preliminary results of the survey. And if you haven't filled out your form and mailed it in yet, GET TO IT!!

In order to move forward, I intend to emphasize some of the more sophisticated aspects of programming. I hope this will inspire more people to work on polishing their skills. Past issues of PEEK contain a wealth of information for helping you along this road. I certainly expect that this higher level of sophistication will help maintain the interest of all of you.

That's not to say that I'm going to abandon the novices. As long as their are generous people like Leo Jankowski, there will always be a Beginner's Corner. Check out Leo's fine function graphing program in this month's issue.

As I said, I want to move the OSI community into the 1980's. Part of that trip is going to mean some new hardware. I truly believe that we can make some significant improvements to our systems with a minimum investment. That may mean attatching what amounts to a second computer to your current system, but it seems to me that the time is right to make some bold moves.

Thank goodness that OSI machines have always attracted the hardware buffs. Somewhere out there is the knowledge and experience to build what we need. It will require some real commitments from many of us before anything concrete can be actually etched on a circuit board, but by defining our needs and demonstrating our interest, I am sure that we can convince one or more vendors that such a project would be worthwhile.

Speaking of hardware, Steve McGuinnis continues his work on adding a disk drive to a C1P in this month's issue. In the coming months, you'll see a lot of work in this area for the other OSI systems.

Last, but not least, for the big boys, I have developed a proposal for assigning semiphores to files under OS-65U Level 3. If this technique fills the void I think it does, I hope it will become the first of many standards that will be developed and maintained by PEEK[65]. For too long now, the only organization with the clout to set a standard has been ISOTRON and many good ideas have fallen by the wayside because of this.

I have great hopes that whatever we can come up with in my "new" OS-65D version will become accepted and used. If we stand together and say "this is the standard from now on", it will take hold and grow.

Don't forget that this process is a two-way street. It cannot possibly succeed without your close cooperation and active participation. Drag out that utility program, write up a short description and send it in to PEEK. Have you had good luck with a commercial program? We want to hear about that too. Again, fill out your User Survey and send it in. The page can be cut out without damaging the rest of the magazine. Do it now! Thanks a lot. Let me end by repeating myself one more time. I love this job!

Rick

Beginner's Corner: Draw-A-Graph for OSI BASIC

by Leo Jankowski
Otaio Rd 1
Timaru, New Zealand

OSI BASIC is compact. That's what makes it so fast. Nevertheless, it would be nice to have a SWAP command. A great deal of time and code would be saved if the "INSTR" function was available. (Editor's Note: Leo - INSTR is easy. Just point USR(X) at 574 and 575 to the following code:

```
JSR $0587
TAY
LDA #$00
JMP $1218
```

This code waits for a keypress and will return it's ASCII value in "X" in the equation "X=USR(X)". See Earl Morris' article on the SWAP command.)

Another desirable function is "EVAL". With this function, BASIC could evaluate a mathematical expression. For example, if Y$="SIN(X)" then after Y=EVAL(Y$), variable Y would hold a numerical value. The following program would simulate a calculator:

```
10 INPUT Y$: PRINT EVAL(Y$)
20 GOTO10
```

Without an "EVAL" function, a graph drawing program is difficult to write, but not impossible. The program listed here will draw the graph of any equation - examine the accompanying printouts. It will input an equation as a string, evaluate the string if it is legal, and then draw the graph of the equation. Several equations can be graphed in succession without the need to reRUN the program for each new equation. Different screen displays are possible. They are: one full screen graph, or two superimposed full screen graphs, or two graphs plotted one above the other.

PRINTER
Copy to printer is also possible. The program contains it own machine code, 64x32, screen dump program. It is POKEd into the directory buffer beginning at $2E79 - see line 1065. The source code for this routine was published PEEK[65], p17, July 1985. See the April 1984 issue for a non-disk screen dump.

Flexible
The program is very flexible and can be easily adapted to any screen format, and, if need be, to any non-OSI computer. Only 5 constants need to be changed. They are in line 110. "R" contains the lowest-left screen address. "GW" is the graph width, "GH" is the graph height, and "SW" is the screen width. "SC" is the screen corner, the position of the cursor after the graphs have been POKEd to the screen.

C1P or C4P
As it stands, the program is written for OS-65D v3.3 and a C4P screen. Notice the screen clear in line 80, the cursor positioning command in line 800 (PRINTs could be substituted), and the disk-BASIC halting get-key subroutine in line 950.

To convert to C1P non-OS-65D BASIC, the following changes also need to be made: omit line 1050 and the TRAP in line 1060 and also the two POKEs and the "PRINT..." in line 1080. Also, change the "USR(X)" POKE values in line 890. The value of "G" in line 1190 would need to be changed to about "820" for a C1P where program RAM starts at $0300. The "PRINT#4" commands refer to output to printer via device #4 - change to suit.

Tokens
A few words about tokens. Every BASIC keyword (eg. SIN or LOG) is stored in RAM as a single number. That number is called a "token". For example, the token for "SIN" is "$B8" and for "LOG" it's $B5". Imagine that the equation "Y=SIN(X)" has been accepted by the program. It is stored in RAM in token form, in HEX as;

```
59 AB B8 28 58 29
```

where "AB" is the token for "=", and "B8" is the token for "SIN".

Notice lines 30 and 50. It is in these two lines that the input equations will appear. The BASIC tokens representing the equations will be POKEd into RAM to exactly coicide with the "$" string in line 30 and the "*" string in line 50. It is for this reason that lines 10 and 20 must be exactly as shown. Otherwise, the value of "G" must be changed in line 1190.

The Program

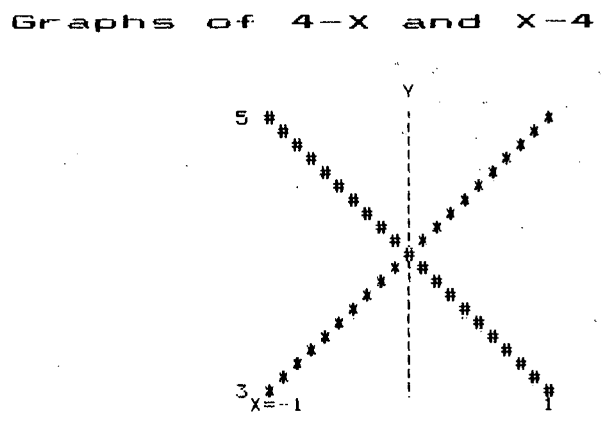
The program begins with a jump to line 1030 where constants and variables are defined. In line 1040, "M=163" is the first token value that the program can use; "P" is also a token value. String "Z$" stores "X" and equation operators. The two POKEs in line 1080 change screen size from 24 to 27 lines. Type these two POKEs in immediate mode, clear the screen with <ESC>-2, and then lean on the <RETURN> key until the cursor comes down to the 27th lines. At the moment, I can't see how this could be done from within a BASIC program - maybe 27 PRINTs in line 1080 following the screen clear!

Copyright 1986 **PEEK[65]** All rights reserved
published monthly
Editor: Richard L. Tretheway

Subscription Rates	Air	Surface
US		$22
Canada & Mexico (1st class)		$30
Europe	$42	$40
Other Foreign	$47	$40

All subscriptions are for one year and are payable in advance in US dollars. For back issues, subscriptions, or other information, write to:
PEEK[65]
P.O. Box 586
Pacifica, CA 94044 415-359-5708

When typing in the equation, line 1090, use "X" as the variable. The "Y=" part is taken care of by the program. After input of the equation, a search is made (line 1190) for the first free RAM address. That address is stored in G(N). Next, the equation in string E$ is evaluated (lines 1230-). If an illegal word or character is found, the program ends - see the error messages in lines 1360-. As each legal word or character is found, its token or ASCII value is POKEd into RAM, that is, into line 30 or 50. The BASIC program modifies itself! Once that stage is complete, the program jumps to line 110.



Article and Program Listing
Continued on Page 4

```
10 REM DRAW-A-GRAPH for OSI BASIC (c) 1986 L.Z. Jankowski
20 GOTO 1030
30 Y=$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
40 RETURN
50 Y=##########################################################
60 RETURN
70 :
80 PRINT !(28): RETURN
90 :
100 REM -------- Constants that determine size of Graph ------
110 R=55043: GW=58: GH=20: SW=64: SC=55168
120 :
130 REM ---------------- Other Variables --------------------
140 GOSUB 80: HG=GH: GR=42: LI=45: EL=33: R$=CHR$(13): M=R-(GH+7)*SW-4
150 M$="Graphs of "+E$(2)+" and "+E$(1): IF B THEN M$="Y= "+E$(1): GOTO 250
160 :
170 PRINT "Plot both Graphs ? (Y) ";: GOSUB 950: PRINT C$: IF C$="N" THEN 230
180 :
190 TWO=-1: GH=INT(GH/2): PRINT "Superimposed     ? (N) ";: GOSUB 950
200 PRINT C$: IF C$="Y" THEN SU=-1: GH=HG
210 GOTO 250
220 :
230 PRINT : PRINT "Display which Graph, 1 or 2 ? ";: GOSUB 950: PRINT C$
240 M$="Y = "+E$(1): IF C$="2" THEN PG=-1: M$="Y = "+E$(2)
250 IF F=-1GOTO 300
260 :
270 F=-1: PRINT "Equal Axes       ? (N) ";: GOSUB 950
280 PRINT C$: IF C$="Y" THEN GW=GH: R=R+15
290 :
300 PRINT : PRINT : INPUT "Lo-X value ";LX: INPUT "Hi-X value  ";HX
310 IF HX<=LX THEN GOSUB 80: PRINT "Hi X <= Lo X !": PRINT : GOTO 300
320 GOSUB 80: ST$=CHR$(14)+M$: GOSUB 920
330 :
340 REM Draw a graph ---------------------------------------
350 SX=(HX-LX)/GW: X=LX: IF PG THEN GOSUB 50: GOTO 370
360 GOSUB 30
370 LY=Y: HY=Y: FOR C=1 TO GW: X=X+SX: IF PG THEN GOSUB 50: GOTO 390
380 GOSUB 30
390 IF Y>HY THEN HY=Y : REM find highest Y value
400 IF Y<LY THEN LY=Y : REM find lowest Y value
410 NEXT C
420 K=ABS(HY): N=INT(K+.5): IF SGN(N-K)=1 THEN HY=N*SGN(HY)
430 K=ABS(LY): N=INT(K+.5): IF SGN(N-K)=1 THEN LY=N*SGN(LY)
440 :
450 VA=0: IF (LX<=0 AND HX<=0) OR (HX>0 AND LX<0) GOTO 470
460 POKE R-SW*(GH+2),89: M=R-SW*GH: GOTO 510
470 DX=HX-LX: VA=INT(ABS(LX)/DX*GW+1.5): M=SW*GH+1
480 POKE R+VA-M-2*SW,89: M=R+VA-M
490 :
500 REM ------------Poke vertical axis ------------------
510 FOR N=M TO R+VA STEP SW: POKE N,EL: NEXT
520 :
530 DY=HY-LY: IF DY=0 THEN PRINT "Division by 0 error": PRINT : GOTO 300
540 IF (HY>0 AND LY>0) OR (HY<0 AND LY<0) THEN F1=-1: GOTO 620
550 IF NOT (HY>=0 AND LY=<0) THEN M=R: GOTO 590
560 HA=INT(ABS(LY)/DY*GH+.5): M=R-HA*SW
570 :
580 REM -------------- Poke horiz. axis -------------------
590 FOR N=M TO M+GW: POKE N,LI: NEXT : POKE N+1,88: IF SUGOTO 690
600 :
610 REM --------- Print labels on graphs --------------------
620 ST$=STR$(LX): M=N+SW-GW-5: ST$="X="+ST$: IF F1 THEN M=R+SW-2
630 GOSUB 920: ST$=STR$(HX): N=N+6-LEN(ST$): M=N+SW-5
640 IF F1 THEN M=R+SW+GW-LEN(ST$)
650 GOSUB 920: F1=0: ST$=STR$(LY): M=R-4
660 GOSUB 920: ST$=STR$(HY): M=R-GH*SW-4: GOSUB 920
670 :
680 REM --------Calculate & poke graph to screen ------------
690 FOR N=0 TO GW: X=LX+SX*N: IF PG THEN GOSUB 50: GOTO 710
700 GOSUB 30
710 PR=INT((Y-LY)/DY*GH+.5): M=R-PR*SW+N
720 IF PEEK(M)>LI AND PEEK(M)<F8GOTO 740
730 POKE M,GR
740 NEXT
750 :
760 IF SU THEN GR=35: SU=0: TWO=0: PG=-1: GOTO 350
770 IF TWO THEN R=R-13*SW: GR=35: TWO=0: PG=-1: GOTO 350
780 :
790 REM --------- Graph(s) displayed - what next ? -----------
800 PRINT &(0,26);
810 PRINT "Printer, Values, Equations, Exit. Which (P V E or X) ?"R$;
820 GOSUB 950: IF C$="V" THEN GOSUB 80: GR=42: PG=0: GOTO 110
830 IF C$="E" THEN GOSUB 980: RUN
840 IF C$="X"GOTO 1340
850 IF C$<>"P"GOTO 820
860 :
870 PRINT SPC( 10)"SENDING TO PRINTER, NOW!"SPC( 20)R$;
880 FOR C=1 TO 2000: NEXT : PRINT SPC( 60)R$;: POKE SC,32
890 PRINT #4,CHR$(27)CHR$(49);: POKE 574,121: POKE 575,46: X=USR(X): GOTO800
900 :
910 REM ----------- 'PRINT AT' routine ----------------------
920 FOR PK=1 TO LEN(ST$): POKE M+PK,ASC(MID$(ST$,PK,1)): NEXT PK: RETURN
930 :
940 REM --------------- Get a key --------------------------
950 DISK !"GO 2336": Y=PEEK(9059): C$=CHR$(Y): RETURN
960 :
970 REM ------------ Modify lines 30 & 50 --------------------
980 FOR K=0 TO EQ: POKE G(1)+K,C(1): NEXT K: IF B GOTO 1000
990 FOR K=0 TO EQ: POKE G(2)+K,C(2): NEXT K
1000 RETURN
1010 :
1020 REM ============------- MAIN MENU --------===============
1030 GR=0: X=0: LX=0: SX=0: Y=0: LY=0: DY=0: M=0: P=0: PR=0: N=0: R=0: GH=0: F8=58
1040 F7=47: R=0: C(1)=36: C(2)=35: M=163: P=173: DIM K$(14): EQ=50
1050 POKE 2972,58: POKE 2976,44: POKE 2888,0: POKE 8722,0: POKE 2073,173
1060 TRAP 2000: Z$="X() .+-*/^": FOR C=1 TO 14: READ K$(C): NEXT
1065 FOR X=11897 TO 11897+55: READ N: POKE X,N: NEXT
1070 :
1080 POKE 13042,26: POKE 13048,26: PRINT CHR$(27)CHR$(28): FOR N=1 TO 2
1090 PRINT : PRINT "GRAPH" N ".  Type equation:  ";: INPUT "Y= ";E$(N)
1100 IF E$(N)="" OR LEN(E$(N))>EQGOTO 1090
1110 E$=E$(N): GR=C(N): GOSUB 1190: G(N)=G: IF K=0 THEN N=2: GOTO 1360
1120 GOSUB 1230: IF K=0 THEN N=2: GOTO 1370
1130 :
1140 IF N=1 THEN PRINT : PRINT "Another Graph ? (Y) ";: GOSUB 950: PRINT C$
```

The Graphs

Because the graphics are POKEd, the program can be adapted to either ROM of disk BASIC. But first, a series of questions must be answered and flags set appropraitely - see lines 170-310.

The next step, lines 350-410, is to determine the maximum and minimum "Y" values. They are used later to scale the graph. Lines 420-430 are an attempt to rounding silly decimal values. For example, .0000015E-17 should be 0! Lines 450-480 and 530-560 take care of where to draw the two axes. The graph is prperly labelled in lines 620-660 using the "PRINT AT" subroutine in line 920. Finally, the screen locations to which the graph is POKEd are calculated in lines 690-710. Line 720 prevents a graphics POKE that would overwrite a numerical value already present on the screen.

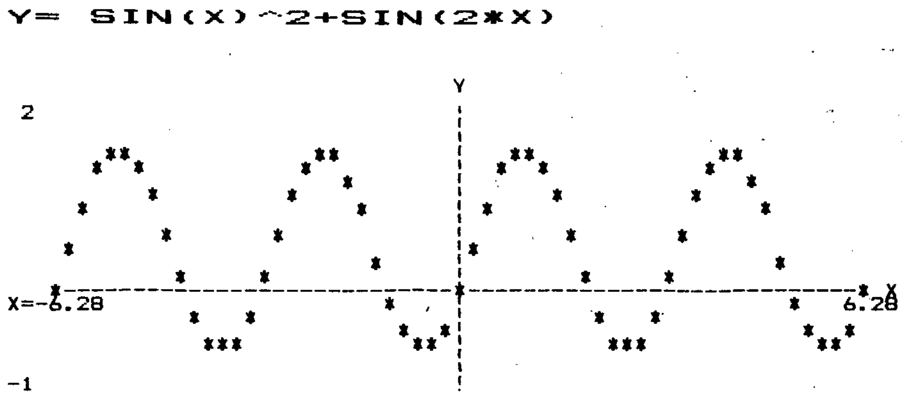
At last, after all that, a "*" or a "#" is POKEd to the screen in line 730.

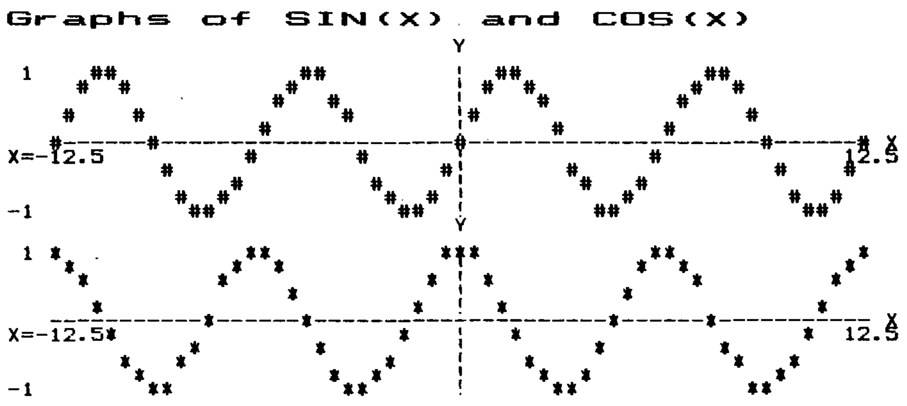
Options

The final program block, lines 810-850, presents the following options: exit, or send to printer, change the range of values for the graph plot, or change the equations themselves. The screen dump to printer routine is called in line 890 with "X=USR(X)". the MX-80 printer command in line 890 sets line spacing to 7/72 of an inch.

What Use Is It?

The program can be used in two ways. It could be used to show what the graph of a particular equation looks like. For example, show that the graphs of SIN(X) and SIN(X)*SIN(X) are both sine curves. When solving equations, it is a help if one knows the approximate solutions in advance. The graph of an equation can indicate the values and the number of solutions that the equation has!





```
1150 IF C$="N" THEN B=-1: N=2
1160 NEXT N: GOTO 110
1170 :
1180 REM -------------- Search for a '$' or a '#' ------------
1190 K=0: FOR G=15047 TO 15240: IF PEEK(G)=GR AND PEEK(G+1)=GR THEN K=G: G=15240
1200 NEXT G: G=K: RETURN
1210 :
1220 REM -------- Evaluate the given expression --------------
1230 L=LEN(E$): W=-1: FOR T=1 TO L: W=W+1: D$=MID$(E$,T,1): K=0
1240 IF ASC(D$)>F7 AND ASC(D$)<F8 THEN POKE G+W,ASC(D$): K=1: GOTO 1310
1250 FOR C=1 TO 5: S$=MID$(Z$,C,1): IF S$=D$ THEN POKE G+W,ASC(D$): K=C: C=5
1260 NEXT C: IF KGOTO 1310
1270 FOR C=6 TO 10: S$=MID$(Z$,C,1): IF S$=D$ THEN Z=M+C-6: POKE G+W,Z: K=C: C=10
1280 NEXT C: D$=MID$(E$,T,3): IF KGOTO 1310
1290 FOR C=1 TO 14: IF D$=K$(C) THEN Z=P+C-1: POKE G+W,Z: T=T+2: K=C: C=14
1300 NEXT C: IF K=0 THEN T=L
1310 NEXT T: POKE G+W+1,58: POKE G+W+2,142: RETURN
1320 :
1330 REM ------ Reset printer, EXIT, Error out ---------------
1340 PRINT #4,CHR$(27)CHR$(50);: POKE 2073,173
1350 GOSUB 80: PRINT TAB( 23)"Bye for now !": GOSUB 980: END
1360 PRINT : PRINT "Error. Cannot find '$' or '#'": LIST -60: END
1370 PRINT "Error in expression: " E$: LIST -60: POKE 2073,173: END
1380 :
1390 DATA SGN,INT,ABS,USR,FRE,POS,SQR,RND,LOG,EXP,COS,SIN,TAN,ATN
1400 DATA 72,138,72,152,72,162,0,160,0,189,0,208,32,159, 36
1410 DATA 200,192,64,208,12,169,10,32,159,36,169,13,32,159, 36
1420 DATA 160,0,232,208,230,238,132,46,173,132,46,201,216,208,220
1430 DATA 104,168,104,170,104,169,208,141,132,46,96
1440 :
2000 POKE 2073,173: PRINT "Error": LIST -60
```

Level 3 Semiphore Standard: A Modest Proposal

by Richard L. Trethewey
Editor

I am just beginning to work with OS-65U V1.44 under Level 3. As with any multi-user operating system, a primary concern with OS-65U is file contention. It has only been in the past few years that OSI began to truly address this issue with the advent of semiphores for file locking.

Using standard OSI software, the programmer has up to 256 semiphores, or "flags" if you prefer, which are used to indicate when a file is "busy" - that is, it is being used by another user on the system. Several of these 256 semiphores are reserved by OSI for the DIREC* file, DATA and PASVAL for OS-DMS, and some others. However, beyond these reserved semiphores, I have never seen any mention of how a programmer is supposed to select which semiphore to use for his own data files. I have poured over the 65U v1.44 manual without finding any advice on this issue. If a protocol for assigning semiphores exists, I hope someone out there will let me know.

The problem with not having a standard protocol to use for assigning semiphores is that unless all of your software is written by the same person, only one author's software can be safely run simultaneously under Level 3. While most end-users do in fact use only on author's software, I think we would all benefit from a standard protocol so that all of our programs will soon be able to co-exist.

The protocol I propose is simple in design, and the supporting software takes little additional code. In essence, I assign each file of type "DATA" a cardinal value based on it's physical location within the disk's directory. Drive location is also factored in to allow for files on floppy disks to be used as well as hard disks. The drive offset is equal to the ASCII value of the drive name minus 65 times 10 or;

DO = (ASC(DV$)-65) * 10

Thus, floppy disks are allotted up to 10 data files each for drives A through D and occupying semiphore numbers 1 through 49. Semiphores 50 through 200+ are allocated to hard disks.

The demonstration program shown in Listing 1 will list the semiphore numbers for all "DATA" files on a selected drive. Obviously, in a real program, this code would be changed to find a match for a file name in the directory and compute it's semiphore. I think you'll find that many of the subroutines within the demonstration program are already in some of your programs. Further, even if you have to install all of this code, the overhead is minor - in the range of 20-30 lines.

I wrote this code for maximum speed. Changing the variables holding constants back into their numeric equivilents really wouldn't hinder execution time significantly in most cases, so if you're worried about variable name conflicts, feel free. Soon, probably next month, I will present a subroutine which will compute a given file's semiphore using machine code. I ***think*** I can make the supporting code very small. I know it would be fast.

In designing this protocol, I thought about how it would be used in real applications. Again, I expect that most software would pass a file name and drive location and it would want a semiphore number passed back to it. I considered the possibility of writing a program which would compute the semiphores for all "DATA" files on all drives and store them in another data file for easy retrieval from BASIC. The problem is that this utility program would have to be run every time the disk is PACKed and every time a new "DATA" file is created. In real life, I don't think that's acceptable. The code needs to reflect the current state of any disk at all times.

Let me again emphasize that my experience with Level 3 is very limited. I know there are potential problems with this protocol in a networked environment. Further, this protocol would really be an improvement if it could easily and reliably provide record locking as well as file locking semiphores. I have two goals here. (1) If no standard exists for semiphores, let's use this one. (2) If there are problems with this proposal, I'd sure like to hear about them. What do you think of this? If I've made a stupid mistake, let me know. I can take it... I think. But by all means write me a letter.

continued on page 6

JOIN OSI SIG AND SEE THE WORLD!

Sign Up for CompuServe!

CompuServe subscription kits with up to 5 hours of free 300 baud non-prime connect time are now available directly from PEEK[65] for only $32.00 plus shipping. That's 20% off the regular price of $39.95. This kit includes a manual for using the service.

In addition to giving you access to OSI SIG, a CompuServe account can be your gateway to a wealth of information and communications services. Send for your kit now!

computer repair

Board level service on:
- OSI / Isotron
- TeleVideo
- IBM pc/xt

Floppy drive alignment:
- Siemens
- Shugart
- Teac

Terminal repair:
- TeleVideo
- Micro-Term

(1 week turnaround)

SEI
Sokol Electronics Inc.
474 N. Potomac St.
Hagerstown, Md. 21740
(301) 791-2562

```
10 REM- Multi-User Semiphore Utility
20 REM- Written by Richard L. Trethewey
30 :
40 GOTO 1000
50:
100 AT=AA
120 IF PEEK(AT)=K0 THEN 40000: REM- END OF DIRECTORY? --> 40000
130 IF PEEK(AT)=K1 THEN 170: REM- DELETED ENTRY? --> 170
140 TY = (PEEK(AT+K8)AND28)/K4: IF TY THEN 170: REM- DATA FILE?
150 F$="": FOR K=K0TOK5: F$=F$+CHR$(PEEK(AT+K)): NEXT K
160 SM=SM+K1: PRINT F$; " is semiphore *";SM
170 AT=AT+SX: IF AT<(AA+PG) THEN 120
180 DS=DS-PG: IF DS=K0 THEN 40000
190 DA=DA+PG: GOSUB 300: ER=USR(K0): IF ER THEN 50000
200 GOTO 100
210:
300 REM- Break down Disk Address & do POKEs
310 D0=DA: D3=INT(D0/F3): D0=D0-D3*F3: D2=INT(D0/F2)
320 D0=D0-D2*F2: D1=INT(D0/F1): D0=D0-D1*F1
330 POKE CB+K1,D0: POKE CB+K2,D1: POKE CB+K3,D2: POKE CB+K4,D3
340 RETURN
350:
1000 K0=0:K1=1:K2=2:K3=3:K4=4:K5=5:K6=6:K7=7:K8=8:K9=9:KT=10
1010 PG=256:SX=16:AA=ASC("A"):AE=ASC("E"):AZ=ASC("Z")
1020 U1SER=PEEK(8778): U2SER+PEEK(8779): CB=9889
1030 POKE 8778,192: POKE 8779,36: POKE 9435,232: POKE 9436,40
1040 AA=9970: POKE CB+K5,K0: POKE CB+K6,K1: REM- DIRBUF - 1 PAGE
1050 POKE CB+K7,AA-INT(AA/PG)*PG: POKE CB+K8,INT(AA/PG)
1060 T=PEEK(9832): IF T>127 THEN T=T-128: IF T>63 THEN T=T-58
1070 DD$=CHR$(T+AA): F1=PG^K1: F2=PG^K2: F3=PG^K3
1080:
1090 INPUT "DEVice ";DV$: IF DV$="" THEN DV$=DD$
1100 PRINT: C=ASC(DV$): IF C>AZ THEN C=C-32
1110 DV$=CHR$(C): IF C<AA OR C>AE THEN PRINT CHR$(K7);: GOTO 1090
1120 DA=25000: REM- 65U DIRECTORY TRACK DISK ADDRESS
1130 T=C-AA: REM- INSTALL CODE TO ADJUST FOR HARD DISKS HERE
1140 POKE CB,T: DEV DV$: REM- SELECT DEVICE TO READ
1150 GOSUB 300: REM- SET UP FOR 1ST READ
1160 ER=USR(K0): IF ER THEN 50000: REM- READ 1ST PAGE OF DIR
1170 DS=(PEEK(AA+12)*F1)+(PEEK(AA+13)*F2)+(PEEK(AA+14)*F3)
1180 SM=((C-AA)*KT): REM- DIFFERENTIATE BETWEEN DRIVES!
1190 GOTO 100
1200:
40000 REM- COMMON EXIT
40010 POKE 8778,U1SER: POKE 8779,U2SER: DEV DD$: END
40020:
50000 REM- ERROR HANDLER
50010 PRINT "DISK ERROR *";ER;" ON DRIVE ";DV$
50020 GOTO 40000
```

BACKISSUE BONANZA!

The backissues of PEEK(65) hold a wealth of information not available anywhere else. Programs, PEEKs, and POKEs, to solve that problem you've been running into lately. The early issues are especially valuable for you C1P owners. And now all backissues are on sale for $6.00 per year, plus $3.00 shipping for the first year ordered and $1.50 for each additional year. Individual issues are $1.00 each, plus 75 cents for shipping for the first issue ordered and 40 cents for each additional issue. Backissues are available through January 1981. Limited quantites of the 1980 issues are also available.

STRING SWAP FOR 65D BASIC

By: Earl Morris
3200 Washington
Midland, MI 48640

Whenever a string sort is done in BASIC, strings are placed in the proper order by interchanging pairs of strings until all are in sequence. Swapping strings is usually done by a three step process such as

T$=A$ A$=B$ B$=A$

This constant redefining of strings rapidly uses up string memory space, and calls the dreaded garbage collector. With many strings in memory, and perhaps the memory already nearly full, the constant garbage collection creates long delays and slows the sort. The Sept. '85 issue of PEEK presented a two part article on Sorts. The SWAP command for 65U BASIC was explained. Rather than actually moving the string data, the pointers to the two strings are interchanged. No free memory is consumed. This SWAP command has not been available for 65D. The following BASIC program will replace the BASIC "DEF" command with "SWP". Note that the "DEF" command can no longer be used until the system is rebooted.

An Assembly Language program for the patch is listed. The JSR to $0F2E is the key to the program. This subroutine returns the address of the BASIC variable in the A and Y registers. In the case of a string variable, a three byte pointer is pointed to. The first byte is the string length, and the second two bytes are the address where the actual string is stored. The pointers to the first string are found and stored on page zero in locations $40 and $41. A comma is checked for, and the address of the second string pointer is found and stored in $42 and $43. The SWAP routine then interchanges these three byte string descriptors and returns to BASIC.

The BASIC program given will POKE in the machine patch. Line 5020 changes the command from DEF to SWP. When the BASIC set-up program has been run, it can be newed and your sorting program loaded.

The patch program must be run before any lines of BASIC are entered which contain the SWP command. If the patch is not present when the letters SWP are typed, the command will not be tokenized and will give a syntax error. The purpose of the swap program was to interchange string variables. However, numeric variables (X,Y,Z etc.) can also be interchanged with a small modification to the program. A numeric variable is five bytes in length. To swap all five bytes, the LDY #$02 in source code line 130 must be changed to #$04. The patch program could be expanded to look at the string flag at $000E and swap the proper number of bytes for either string or numeric variables. This exercise is "left to the reader". Thanks to Ray Peterson and Harry Pye for several suggestions for the string swap program.

```
10 REM PROGRAM TO REPLACE  'DEF' WITH 'SWP'
20 REM   SWP A$.B$   or   SWP A$(2),A$(3)
30 REM    WILL SWAP STRINGS WITHOUT CAUSING
40 REM    STRING GARBAGE COLLECTION
50 REM     (FOR 65D DISK BASIC)
60 :             ~~1253~~  1235
5000 FOR X= 4661 TO 4696 :READJ:POKEX.J:NEXT
5002 DATA32.46,15,133.64,132,65,32.193,0
5004 DATA32.18.14,32,46,15,133,66.132,67
5006 DATA160,2,177,64,170.177,66,145,64,138
5008 DATA145.66.136.16,243.96
5010 REM POKE OUT DEF AND ADD SWP
5020 POKE 725.83:POKE726.87:POKE727,208
```

```
10              ; STRING SWAP ROUTINE FOR 65D
20              ;
30              ; REPLACES BASIC 'DEF' COMMAND
40              *=$1235
50          JSR $0F2E       ; GET POINTERS (PTRGET)
60          STA $40         ; SAVE LSB OF LOCATION
70          STY $41         ; SAVE MSB TOO
80          JSR $00C6       ; RE-FETCH LAST CHAR. SEEN
90          JSR $0E12       ; MAKE SURE IT WAS A COMMA
100         JSR $0F2E       ; GET PTRS TO DEST. VARIABLE
110         STA $42         ; SAVE IT'S LSB
120         STY $43         ; SAVE IT'S MSB
130         LDY #$02        ; INIZ COUNTER
140 SWAP    LDA ($40),Y     ; FETCH A BYTE
150         TAX             ; SAVE IT IN X REGISTER
160         LDA ($42),Y     ; FETCH REPLACEMENT BYTE
170         STA ($40),Y     ; DO THE REPLACEMENT
180         TXA             ; RETRIEVE ORIGINAL'S BYTE
190         STA ($42),Y     ; AND DO THE EXCHANGE
200         DEY             ; DECREMENT COUNTER
210         BPL SWAP        ; LOOP 'TIL Y GOES NEGATIVE
220         RTS             ; QUIT
```

compuwork WANTS:

PROGRAMMERS

Must be expert on application programming using 65U and/or 65E
Must be able to relocate to the Atlanta area

VERTICAL MARKET PACKAGES	HORIZONTAL MARKET PACKAGES
Must be field tested in your area	Must be easy to use and easy to learn
Must be comprehensive	Must be single program file and one or two data files
Must have a broad market	Must not require extended input (65U)
Must have established current users	Financial or insurance applications a priority
Must run on Isotron 250J (1.44)	
Must run on DBI series (1.00)	
Must be well documented	

Send resumes/salary requirements and marketing info/instruction manual to:
Leon Haverly
Compuwork, Inc.
1395 Marietta Parkway, Suite 706
Marietta, Georgia 30067

We sell and service OSI computers, DBI computers, Wyse terminals, ADDS VP terminals. Special note: we have 10 years factory experience on OKI drive and board repair. For information please call:
Roger Stone or Jim Smith (404) 426-5509

Interfacing a C1P-II to an MPI Disk Drive Using a D&N MEM-CM9 Board- Part II

by Steve McGuinnis
20 Curt Boulevard
Saratoga Springs, NY 12866

I have done my best to be as accurate as possible in compiling the following information. However, if you encounter anything that does not "compute" please share that information through PEEK[65].

My sources include the public domain, past articles in PEEK[65] and visual inspections of the drive. So, although I suspect that much of this information is valid for any brand of computer that uses MPI drives, I've confined my remarks to the one I understand - the OSI C1P-II!

If you intend to do any major repairs on your drive I highly recommend that you purchase MPI's product manual p/n 03021-001. Although the $30.00 I paid may sound a bit expensive, the amount of information contained within the manual is considerable and is (for me, at least) highly valuable.

I originally purchased an MPI B51 drive (single-sided) by mail order for $49.95 (a great source of such things is, by the way, The Computer Shopper magazine). Although it "banged" a bit when I first accessed it with HEXDOS, it did seem to work properly (although, I'll admit, noisily). Two days later it stopped working. It appears that in my haste to get everything running I didn't do a very good job of aligning the D&N controller board. So, the drive either would have stopped anyway or I hastened it's demise. As a direct result of this "fiasco" I became a little more informed in the world of 5-1/4" floppy disk drives. (I've purchased a double-sided MPI drive since then for only $39.95 and now use it with my C1P-II!)

The Electronics Board

The electronics board layout was prepared by using the board as a template and actually copying the location of the major components onto paper and keeping to scale as much as possible (therefore, integrated circuits, resistor packs, etc. are shown but discrete capacitors and resistors are not). You'll notice that in addition to the part numbers that the codes silkscreened by MPI on the board are also included (for example, at the upper left hand corner of the board next to the head connector is IC-1A which also happens to be a differential amplifier, part number CA3054).

Besides discovering that a disk drive is very similar in theory to a cassette tape recorder in the way that it reads and writes (keep this in mind as we go on), I noticed some interesting things while studying the drives:

(1) It appears that the MPI electronics board is the same regardless of whether it is for a single-sided drive (model B51) or a double-sided drive (model B52). The only difference that I have found is that on the B51 the only head is HD0 (on the bottom) whereas, the B52 has the upper head installed (HD1). The board has connections for both of them regardless of the drive model (as a matter of fact, I have switched the board from the model B51 to the model B52 without any problems to date.)

(2) The MPI drives have manufacturer-installed jumper wires on the back of the board. If you have any problems it is probably wise to make a drawing of the wire colors and connections since they are fine and tend to break. (the assembly number printed on the board tells MPI engineering the computer make the drive is configured for - mine is marked 3-29003-822-AM6 and I'm pretty sure that it came out of a Tandy computer.)

(3) There is a place on the board at connector J5 for a door "open" switch. It can be used to signal a computer whether or not the drive door is properly latched. Although I don't know of an easy way to implement this on the C1P, I thought it was interesting.

(4) The C1P doesn't know whether it is addressing a single-sided or double-sided drive. Therefore, it is possible to "fool" the computer into using both sides (although only one side at a time) of a model b52 by installing a SPST (single-pole-single-throw) switch between MPI connector J1, pin number 32 (side select) and ground. When the switch is "open" and there is no path to ground the drive automatically uses head HD0 and acts like the typical OSI drive. However, when the switch is "closed", pin 32 goes "low" and instead of head HD0 it uses head HD1 (the top head). This allows me to regularly use both sides of my single-sided diskettes (don't spend the money for double-sided diskettes unless you need them for another computer).

(5) When I was originally looking for help on getting the B51 to work, someone in the OSI SIG on CompuServe was kind enough to give me some very helpful information. Fortunately I checked some of it out with an ohmmeter before I plugged in the drive because as the result of some "typos" he accidently gave me the wrong power connector pinout (it was at this point that I ordered the MPI product manual). The correct pinout is shown on the drawing (please note that the "-" 's represent the ground connections and that they are the two center pins!).

Before we go on, the following items deserve some discussion.

(1) It should be noted that if you are using only one drive that the 1f (see drawing) termination resistor package should be installed. If you are using two drives only the last (second) drive should have this chip installed.

(2) I have found that replacing the shunt supplied by MPI with a 14-pin dip switch works well. The shunt settings I use with my drives are HS and DS0 switches closed, all others open.

(3) Don't forget that a data separator goes between pins 10 and 11 of the D&N board and pin 30 of the MPI connector J1. Just hooking them together won't work if you're using an

OSI computer since a separator was not included as part of the disk drive controller board.

The Disk Drive

If you're not already way beyond me, you might want to set your drive on your lap (outside of it's enclosure) and follow along:

(1) In front of the electronics board and to the left (the front of the drive should be facing you) are four small, blue "plug-in" sensors (a "sender" and a "receiver" - located in pairs). The pair on the left "look" for the write protect notch on an inserted disk and the pair on the right sense the small index hole and generate an "index" signal.

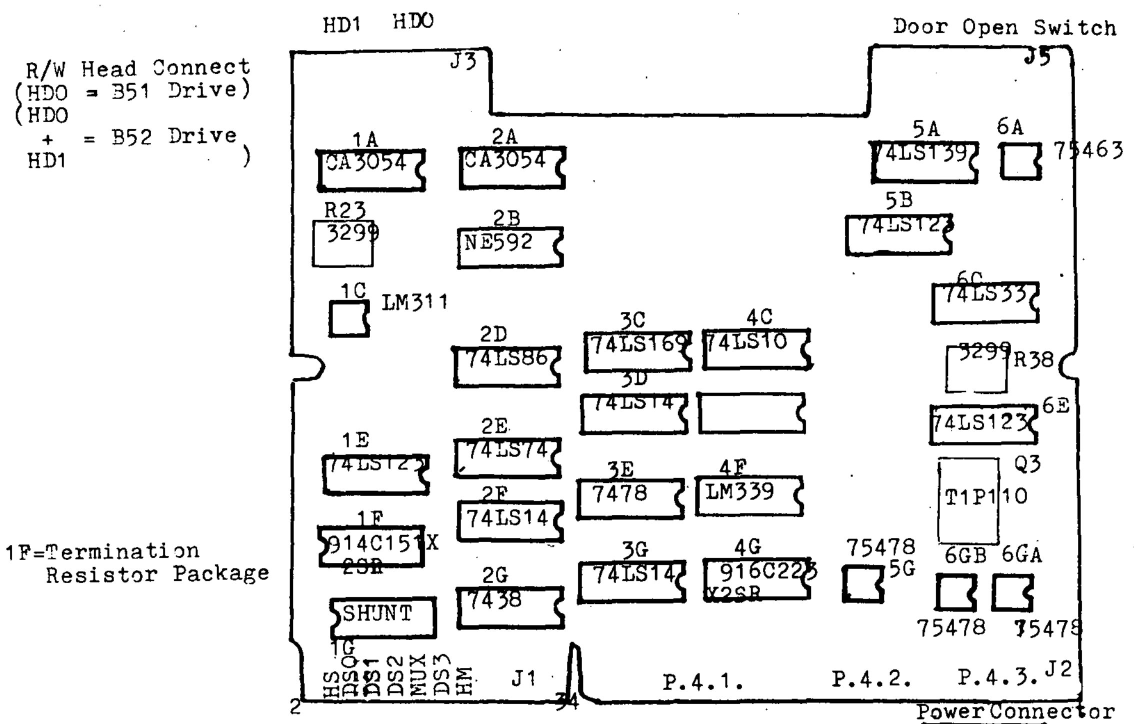
(2) The white plastic disk which is attached to the metal frame (it moves with the door latch) is part of the spindle assembly. When you insert a disk and close the door, it clamps the disk to the bottom (metal) rotating member (turn the drive over and you'll see that it is attached to a large pulley which is, in turn, attached to the motor/tachometer with a drive belt).

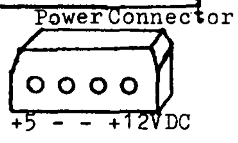
(3) Now, while the drive is upside down, notice that in the far left hand corner (the front of the drive is still facing you) is a square motor. This is the stepper motor.

(4) When the computer sends a "step signal" on pin 20, connector J1, the stepper motor will index a small amount (either forward or backward depending upon the logic level of the signal on the "step direction" line at pin 18, connector J1).

CONTINUED ON PAGE 10

MPI B51/52 ELECTRONICSBOARD LAYOUT





D&N J3		MPI J1	
17	Index	8	Index
3	Sel Dr 1	10	Drive Sel 1
18	Sel Dr 2	12	Drive Sel 2
12	Motor On	16	Motor On
6	Step-In	18	Step Dir
5	Step	20	Step
9	Write Data	22	Write Data
8	Write En	24	Write Gate
23	Track 0	26	Track 00
19	Write Prot	28	Write Protect
10	Sep Clock		
11	Sep Data	30	Read Data
2,4,7,20,21, 22,24	No Connect	2,4,6,14,32,34	No Connect

(12,13=Ground) (1 to 33 odd=Ground)

SHUNT	
HS	Head Load W/Select
DS0	Select Drive 1 (OSI)
1	Select Drive 2 (OSI)
2	Select Drive 3 (OSI)
MUX	Multiplex
DS3	Select Drive 4 (OSI)
HM	Head Load W/Motor On

OTHER	
R38	Motor/Tach Speed Adjust
Q3	Motor Current Source (Power Transistor)

CONTINUED FROM PG. 9

(5) Notice that to the right of the stepper motor is a rectangular black structure riding on a pair of metal "rails". A gray cable runs to it and is attached with a copper foil. This is the bottom head (HD0) on the B52 drive and the only head on the B51 drive. Also note that there is another blue sensor (it's "mate" is on the other side of the drive). When the head is over track zero a small plastic "flag" breaks the "beam" and a pulse is sent to pin 26, connector J1. Also look at the very back of the drive just to the left of the copper "clamp" which holds the head assembly "rails" in place. You should see a small, tapped hole with a set screw. This set screw is precisely adjusted so that the head will not go behind track 0. It can be a delicate adjustment so resist the temptation to change it.

(6) Now, turn the drive over and take a close look at the area around the head assembly and the shaft which protrudes up from the stepper motor. It's a bit difficult to see but the stepper is attached to the head assembly through a metal "band" mechanism. As the stepper shaft revolves in one direction, the band winds onto the shaft and pulls the head along the rails. If it revolves in the opposite direction, the head direction also reverses. It is precisely this mechanism which positions the head over the correct track on the floppy disk so that the information can be decoded and sent on it's way to the computer.

Basic Alignment

I couldn't resist the temptation to share at least a bit of my new found knowledge about drive track alignment, etc. with those of you who have even the poorest oscilloscope. Note however, that without the MPI product manual and a rather expensive oscilloscope that the job will be no more than a "kludge". Hopefully, however, the job should suffice until you can have the job done professionally or can purchase the manual and gain access to a better scope.

A workable track alignment procedure is as follows:

Remember earlier when I pointed out the location of IC-1A? Well, it was an interesting discovery but IC's 1A and 2A are the disk head amplifiers (it's really quite obvious since the circuit leads run straight to the head connector). IC-1A appears to be used during write operations and IC-2A amplifies the signal during read operations. You can get a reasonable idea of what the head is seeing during disk access by (carefully!) touching the scope probe to pins 1,7,8 and 14 of IC-2A (use an IC test clip if at all possible). Please note that the pins you find the signal on depend upon which drive head is active and in contact with the floppy disk!

The next step shouldn't be attempted unless your drive is definitely off track alignment and you're either quite confident in your own abilities or you had plans of having the drive aligned at the shop in the first place and I've talked you out of it. The reason is that if you turn the stepper motor too far you can position the head completely off of track 0 and might never get it re-positioned correctly. If this happens you're on your own!

First, turn the drive on it's side so you can see the stepper motor and program the computer to re-position the disk drive head to track 0 (in HEXDOS a "LOAD*0,8192" works nicely). Since the OSI keeps the head down and the motor running you can get a "look" at the signal from the head by attaching the scope probe to the pin of IC-2A where you previously found the strongest signal. Now look for the molded-in slot in the drive frame in front of the stepper motor. While watching the signal on the scope and being careful not to change it's amplitude in any way, carefully loosen the two screws holding the stepper motor but just enough that a screwdriver positioned between the slot and the screw "cut-out" in the stepper motor frame can turn the motor. If you make them "finger loose" the motor can turn unexpectedly and throw the drive completely out of alignment.

Okay, now for the alignment. Carefully rotate the screwdriver enough to cause a small change in the signal amplitude on the scope. If you lose it entirely don't panic but simply turn the screwdriver in the opposite direction until the signal reappears. When you discover the exact point where the signal amplitude is at a maximum, carefully re-tighten the screws. Program the computer to step out to track 39 ("LOAD*39,8192") and repeat the procedure. Then check track 20. Hopefully at this point you're finished and all is well.

Do you want to set the motor speed? On the bottom pulley you should find a strobe disc. If you put a neon light next to it with the drive motor turning, the "bars" should be reasonably stable and barely in motion. If not, adjust the variable resistor R38 until they become stable. I found that a neon AC outlet tester of the common household variety when plugged into the end of an extension cord worked very well.

What was wrong with my drive? Well, a blown capacitor in the motor control circuit allowed digital pulses from the motor-tachometer to feed through to the read/write circuits. If that sounds confusing to you then maybe it's an opening for someone else to carry this discussion a bit farther in another article!

User Survey: Preliminary Results

The results of the User Survey really amazed me. First of all, I had no idea how many of you would take the time to respond. I have been very pleased with the number of forms I have already received.

Despite my perception that the majority of PEEK subscribers are video system owners, roughly half of the responses I've received so far have been from serial system people. I believe that's because many more video system owners are satisfied with their current systems, or at least have become comfortable with their limitations, whereas serial system owners are constantly looking for help on software issues. As I promised, PEEK will be paying special attention to 65U and related topics from now on.

Not surprisingly, 100% of the respondents have a printer. I was surprised to find that 80% also owned a modem. Now if only 80% of them would sign up for CompuServe and OSI SIG! I was pleased to see a majority of respondents had upgraded to OS-65D V3.3 and OS-65U V1.44. That means we can write for these versions and know people can use what we write.

The clear winner in the commercial software field was OSI's imfamous OS-DMS software. General accounting software from various vendors came in second, with Dwo Quong's WP6502 word processor a close third. Also rans include my own Term-Plus and (believe it or not) WordStar.

The software most people wanted was a word processor. None of the respondents indicated why the word processors currently available were inadequate, and while I have some pretty good guesses, the fact remains that a void exists here. Second place went to 65U utilities of various sorts.

When asked about CPU upgrades, 16% were not interested, 16% would go $15.00, a whopping 44% said they would pay $100-$200, 16% said $200-$500, and 5% went all out to $500-$1000.

44% of the respondents said they had no interest in a graphics board upgrade, which reflects the serial system bias so far. But 22% would go $50-$100, 16% said $100-$200, and another 16% said $200-$500.

I really enjoyed reading your suggestions for articles for PEEK. First of all because many of the requests are already in the process of being fulfilled. Some of the topics suggested have already been covered in past issues of PEEK. Don't forget about the sale on backissues, people. Less than $50.00 would buy everything PEEK has ever printed.

Lots of people wanted more technical articles on OS-65U. Several people have requested we do a memory map including PEEK's and POKE's. I have an aversion to doing such lists for many reasons. First of all, 90% of the useful user-modifiable memory locations are documented within BEXEC*, DIR, CREATE, and LEVEL3. Secondly, as 65U matures, and even more patches are made to it, more of these POKEs are being replaced with FLAGs and as the patches are made, POKEs become ever more vulnerable. I could go on, but the bottom line is that I don't have any immediate plans to print another 65U memory map.

Overall, I think PEEK will satisfy most of the requests for articles in the coming months. But I still need your help, especially from you hardware people and you 65U guru's. For example:

Hardware
Floppy Disk Drive Interfacing
Adding Hard Disks to Video Systems
C1P hardware mods/upgrades
65C02/65816 upgrades
Reviews of 700 series systems
EPROM Programmer
16-pin Bus projects/programs
Amateur Radio Applications
Printer Installation Tips
Modem Installation Tips
Adding RAM to video systems

★★★

Software
65U-based accounting utilities
65U Level 3 utilities
65U Hard Disk Managers
UNIX topics
65D Expansion
6502 Interrupts

Again, these results are preliminary. If you haven't done so yet, go back to the March issue and fill out your survey form and mail it in. I really think that there are many possibilities for upgrading our systems that we can do if a large number of us are willing to make the investment. What those upgrades will consist of depends upon letting the hardware people have some idea of what you want and how much you're willing to spend in order to get it. There's no obligation, but please be as honest as you can be. Thanks again to all who have participated thusfar.

WRITE FOR PEEK!

ATTENTION: DEALERS!

PEEK(65) needs new subscribers and you need new customers, and together we can make it happen with our own Co-op advertising program. This program pays dealers for signing up new subscribers with free ad space in PEEK(65). Just five paid subscriptions will earn a 1/9th page advertising credit in PEEK(65).

Most dealers sell their own software with the systems they install. By advertising in PEEK, you vastly expand the potential market for your products. And how many sales have you lost because you couldn't find the application your customer wanted? Dealer ads can be our own Yellow Pages. Readers and customers win too by increasing the number of uses for their equipment.

Call or write today for details and your free promotional materials. Making a PEEK(65) subscription a part of every sale is painless and profitable. This time, "Co-op" pays you.

LETTERS TO THE EDITOR

Editor;

You said you wanted evaluations of software, so here is one.

Several years ago, I ran across a piece of software called FIND. This program modifies OS-65U upon startup so that the FIND command can also search the BASIC workspace for any string. To use it when programming, just go to the immediate mode and type;

FIND"ABC"

If there are any occurances of the string in your program, it will list the complete line from BASIC. This is very useful in finding where (or if) variables were previously used and if there are any lines that change the variable's value. To find BASIC tokens, you have to enter their ASCII value. For instance, FIND CHR$(140)+"1000" will find all occurances of GOSUB1000. The program will print a list of all tokens and their ASCII value if desired. By using a POKE, FIND will either include or exclude anything in a REM statement. This modification does not change the normal FIND%CH, A$ command used in file searches.

The only bug I have found in the program is that when listing a program on the printer, the last line does not generate a <CR> so it gets "stuck" in the printer's buffer. I understand that a fix is in the works.

The program is easy to use, requires no additional memory, and seems to work on all versions of OS-65U. As a part-time programmer I simply could not write or modify either my own or other people's programs without this FIND command.

The program can be purchased from Keith Brown at Brown/Collinson Associates at (503)-635-5055 for $75.00. I understand from Mr. Brown that there are many pirated copies of this program circulating and that if you have an illegal copy, he would appreciate it if you sent him some money and made it legal.

William Brown
503 N 13th
Cornelius, OR 97113

P.S. No relation to Keith

Dear Mr Brown,

Thanks for the review. Keith's reputation in the 65U community is tremendous, but I'm sure he appreciates the plug. I hope other people will also write in with their software experiences.

Rick

Editor;

Upon reading your "OS-65D Revisited" article in the February issue, I was glad to see your interest in improving OS-65D. While I have both versions 3.2 and 3.3, I seldom use the latter version due to it's higher memory requirement, lack of an assembler, and my much greater familiarity with version 3.2.

My own "wish list" therefore applies directly to v3.2, although many of the individual items would also apply to v3.3, perhaps with little or no modification. Some of the listed improvements have been incorporated into my own version of OS-65D (which I will call OS-65R for lack of a better name); other improvements may be added as time permits. With the exception of adding a decent polled keyboard routine to the 8" system, all existing modifications take the form of fairly short blocks of re-written code (not patches) that reside in the same memory space as the original code with the same external entry points. This works wonders for compatibility, and it makes some of the improvements independent of others and therefore, optional. I might add that the system boots modified; the changes don't have to be overlayed afterward.

Now for the specifics;

(1). All of us know the faults of the ROM keyboard polling routine, and the one written for the 5-1/4" system isn't really any better. The KB routine used in v3.3 represents a considerable improvement, but I prefer the conventional function of the <ESC> key and the "single-finger" method for repeating keystrokes. I am currently using a variation of the ROM routine with entry at $3180 for both 8" and 5-1/4" systems, with several improved features such as normal typewriter function with <SHIFT-LOCK> released, null-character exist if no key is pressed, and override of the <SHIFT-LOCK> key by means of a single POKE.

(2) By compacting some of the code in the 540 video routine, I was able to impliment clear-screen in response to a specific control character (I use $1C for this), and a flashing cursor, toggled on or off by "printing" a NULL. The latter is generated by a variation of the routine at $252B after a set number of null exits from the keyboard routine. I have also used another version of the video routine that impliments the horizontal TAB function, but I don't recommend it for general use unless you want to make corresponding changes to the input routines of ASM, BASIC, and OS-65D. One of these days I may look at some form of full-screen editing and/or windows, but this will likely require one or more patches occupying additional memory.

(3) The "stock" versions of OS-65D are actually single-drive systems that support two drives, one at a time. To be useful, the system should assign the drive location of an opened file to the disk buffer just like the current track, I/O pointers, etc. I saw an article in a recent PEEK issue describing a high-memory patch that is supposed to overcome this shortcoming. In my own version of the system, I re-wrote some of the disk I/O routines to accomplish the same results without patches. In fact, there was enough space left over for some code to skip over the directory track, so all remaining tracks except zero can be used in a single file if needed.

(4) The foregoing improvement works best if it is accompanied by some changes to the code used for drive selection so the OS knows the current

track in both drives. Then it doesn't have to send the head to track zero for re-synchronization every time you select the opposite drive. This took a little work, but I was able to re-write the code from $29C6 to function without need of the table at $29EB, and shrink the check-ready routine by 4 bytes, thus making available 12 bytes for the extra code needed. The modified set-drive routine should work with two-sided drives also, but I don't have a convenient way to try it out.

(5) While I was at it, I wrote another check-ready routine to use with the 5-1/4" system. It's not as foolproof as the hardware implimentation, but it works in most cases, which is more than I can say for the OSI version.

(6) The PUT/GET overlay in v3.2 can stand a number of improvements, and many of these have been addressed by other PEEK[65] readers and by OSI in v3.3. My general abhorrence of memory-consuming patches has led me to re-write the overlay so that all the features I want will fit into 256 bytes or less. These include means to assign the current drive to the disk buffer when a file is opened, random access for both disk buffers, elimination of unnecessary disk transfers during random access, and individual control of record size for each buffer. The directory track is skipped over during random access also if it lies in the middle of a file. Record size is stored in the resident portion of the system, so it won't be subject to change every time another file is opened. I haven't tried to impliment a DISK FIND process, primarily because I have yet to need one, but the modified overlay is 24 bytes short of a full page.

(7) A good way to recover space in the PUT/GET overlay is to eliminate some of the context swaps; they don't serve any useful purpose in my system, as I have found no apparent conflict when using the same context for both BASIC and the OS. For what it's worth, the 5-1/4" OSI system (v3.3 also) still performs disk I/O without swapping context to handle buffer overflow. The 8" system used to work this way, before OSI put the adaptive step rate at $EF and then had to write two more patches at $2F55 and $2f5B to make it stay there.

(8) A very useful feature of many "modern" small computers is their ability to allocate file space "on the fly", permitting a disk file to be created or extended on demand from within a running program. I believe something like this could be built into OS-65D, probably requiring more space for the code and certainly a modified directory structure. I might even try it myself someday, time permitting, if it can be done without requiring wholesale reassembly of the system or too much loss of compatibility.

(9) I did make a change to the code from $2DA6 through $2E1D used by the OS to find a file in the directory. As modified, if it encounters an empty drive or otherwise can't locate the file, it selects the opposite drive and checks again before reporting error #C. The drive already selected will always be the first checked, in case there are two disks with the requested file name. (Yes, the code all fits in the same memory space.)

(10) Speaking of memory space, the indirect file handler can be shrunk by at least 12 bytes without causing any operational changes. Any additional modification to this code would depend on personal preference, but I would suggest using non-printable control characters for all functions, permitting the brackets ("[" and "]") to be used normally in text.

(11) If other changes that you have in mind will require at least a partial reassembly of the system anyhow, you might want to consider a provision for up to four disk buffers. I can think of one practical use for this many, involving the simultaneous updating of separate master and detail sequential files, wherein the data structure does not provide a convenient key field that is common to both files.

(12) A resident directory management routine would probably be a nice feature for OS-65D, at least to display the directory on command from a running program. How much more it should do will depend on the extent of other improvements made to the system.

Well, that's an even dozen and I haven't even started to talk about the BASIC interpreter. I've done a little tampering there too, but I'll save that discussion for another time. All the OS changes that I have implimented relate specifically to the C4P-MF system, but most would probably apply in some form to the serial systems also.

Robert S. Runyon
7015 Brookview Road
Hollins, VA 24019

Dear Robert,

Wow! You have been busy, haven't you? I think we're largely on the same track, if you'll pardon the expression.

Any new version of 65D must incorporate a fully functional keyboard polling routine and video driver. At present, I lean toward keeping OS-65D v3.3's keyboard poll largely untouched. It debounces cranky keyboards better than anything I have seen and people are comfortable with it. It still needs some work because it is very slow, but I think it should remain functionally unchanged. The video driver is another matter. I would like to see it re-written so that the control codes can be altered by the user to obtain compatibility with industry standard terminals. I don't know how practical this is, but it's a goal I think worth pursuing.

The disk file manager portion of 65D needs wholesale re-writing. Top priority should be given to incorporating a directory display command, a file creation command, a file deletion command, a command to check for a file's existance, and new file I/O commands. I like the idea of letting the OS increase the size of a file at will, but I worry about the throughput efficiency of non-contiguous files on 8" diskettes, not to mention sheer wear and tear over the long haul. We won't know until we try it, of course, and I sure want to try it.

I am also leaning toward using a master OS buffer for all disk I/O as OS-65U performs. This allows many files to be accessed simultaneously without a lot of extra code overhead. In addition, I am considering the possibility of allowing for a transient code buffer to allow seldom needed routines to be called into memory only when needed. In addition, I foresee the benefits of a command block that stores data much like that now located at (roughly) $265C-$2661, although this block will not likely be at those locations. When combined, these last two changes will allow for easy extensibility and user customizing. It will not be compatible with programs that directly manipulate the operating system, but once such a standard is established, it will remain upwardly compatible from then on.

As you can see, I have virtually abandoned hopes of compatibility with older versions of 65D. Too much needs to be changed in order to make this project worthwhile. Retaining bad code just isn't worth the price. 65D is small enough and well known enough so that most users should be able to easily modify their current software to be compatible with whatever we come up with. Once we tackle BASIC, we'll really need such software to allow old programs to be read by the new version (ie. the values of keyword tokens are going to change).

I have to point out that people modifying OS-65D should NOT neglect to properly perform the page 0/1 context swapping if they hope their software will run on 8" systems. As you mention, the 8" versions of OS-65D use an adaptive stepping rate byte that must be present at $EF in order for the drives to be able to work. I know mini-floppies work without doing this swapping, but it's a real no-no on 8" systems.

Thanks for letting us know what you've been up to, Robert. I hope that you'll send in the source code for the changes you've been working on so that the rest of us can play with it too!

Rick

Editor;

In the February '86 issue, you mentioned the possibility of revising OS-65D. When I read that article, I felt that I ought to remind or advise you of the late Jim Kupperian's work of several years ago.

I also want to inquire about Term-Plus. Can you tell me if it will work with the D&N addresses for their ports?

J. Daly
1900 Torregrossa Court
McLean, VA 22101

Dear Mr. Daly,

I am aware of Mr. Kupperian's fine efforts which were originally distributed through OSIO. However, with the demise of that organization and Mr. Kupperian's death, the ownership of the code is in doubt as far as I know. I heard many good things about v5.1 and would love to see some source code if it is available without restrictions. I want to be able to freely distribute this code.

Term-Plus comes with software that allows you to set it for any memory address where a standard 6850 ACIA chip resides. This would include the D&N board, OSI's CA-10-X, or any other board that uses the 6850.

Thanks for writing. I hope you'll write again with some ideas for 65D!

Rick

Editor;

Recently, I needed some help with my Seimen's disk drive. I learned that Word Disk Drives are now the OEM of the Seimen's drives. After a couple of letters to them and a couple of replies, all was in good shape with me.

They say that their quality control has been improved dramatically and that they are ready to pursue their share of the disk drive market. They replied by return mail and were most ready to help. It is good to know that all of what we Challenger owners have has not been abandoned.

For information or help, write to:

Mr. T. Downey
World Disk Drives
23501 Ridge Route Drive Bldg. G
Laguna Hills, CA 92653
(714)-855-1761

Sincerely,
Dana Skipworth
2055 West 87th Street
Cleveland, OH 44102

SPECIAL OFFERS FROM T.O.S.I.E

TOSIE-IV PADDLE BOARD FLOPPY DATA SEPARATOR

This popular bare board kit is actually a multiplexer, disk switch, and very stable data separator - sufficient to allow mating most floppy drives with virtually any OSI computer with up to 4 surfaces. The 8 74LS chips required make for economic population, flexibility, and stability. Complete instructions and schematics are included. Although not fancy, it should help resolve many of the problems encountered in matching new drives to OSI machines. Since it is all on the board, minor mods are required on the OSI board to bypass some of the old circuitry. Available from Paul Chidley, RR*2, Ennismore, Ont., CANADA K0L 1T0 for $20.00 total. Non-Canadian orders should be made in US dollars.

OSI-CALC: SPREADSHEET PROGRAM

OSI-CALC is written entirely in BASIC by Paul Chidley of TOSIE, the program gives you a 26 column by 36 row spreadsheet with many features. Don't let the fact that it's written in BASIC fool you. It's VERY FAST.

In contrast to what PEEK said last month, the original version would not run on serial systems. However, I have developed a version just for serial systems. Requires 48K of memory and OS-65D V3.3. Specify video or serial system and mini-floppy or 8" disks. Price $10.00 plus $3.70 shipping.

Term-Plus

A smart terminal program running under OS-65D V3.3 which allows capturing and transmitting to and from disk. Term-Plus also supports error-free file transfers and cursor addressing on CompuServe. Memory size does not limit the size of files that can be captured or transmitted. Video systems get enhanced keyboard driver with 10 programmable character keys. 10 programmable function keys on both serial and video systems. Utilities included allow translating captured text files into OSI source format for BASIC and Assembler programs or into WP-2/WP-3 format, translating OSI source files into text files for transmitting to non-OSI systems, and printing captured text files. Runs on all disk systems, mini's or 8", except the C1P-MF. $35.00.

Term-32

Same as Term-Plus, but for OS-65D V3.2. Video system support includes enhanced keyboard driver, but uses V3.2 screen driver. $35.00.

Term-65U

Patterned after Term-Plus, Term-65U is a smart terminal program for OS-65U (all versions) running in the single user mode. Allows capturing text to disk files. Term-65U will transmit text files, or BASIC programs as text. The program will also send WP-3/Edit-Plus files as formatted text and can transmit selected fields in records from OS-DMS Master files with sorts. Includes utility to print captured text files and convert them into WP-3/Edit-Plus files for editing. $50.00

ASM-Plus

ASM-Plus is a disk-based assembler running under OS-65D V3.3 that allows linked source files enabling you to write very large programs, regardless of system memory size. ASM-Plus assembles roughly 8 to 10 times faster than the OSI Assembler/Editor and is compatible with files for that assembler. ASM-Plus adds several assembly-time commands (pseudo-opcodes) for extra functionality. Included is a file editor for composing files that allows line editing and global searches. $50.00

Edit-Plus

Word processor styled after WP-3-1, although not quite as powerful. Edit-Plus allows composing and editing WP-3 compatible files and to have those files printed as formatted text. Edit-Plus uses line-oriented editing, as opposed to the screen editing of WP-3, and also allows global search and replace. Edit-Plus fixes problems in WP-3 including pagination, inputs from the console, and file merging (selectable line numbers from the merged file). Edit-Plus can perform a trivial right-justification, but it does not support true proportional spacing. Requires OS-65D V3.3. $40.00

Data-Plus 65U Mail Merge

A program to insert fields from OS-DMS Master files into WP-3 documents. Output can be routed to a printer or to a disk file for printing later or for transmission via modem using Term-65U. Insertions are fully selectable and are properly formatted into the output. Perfect for generating form letters. $30.00

Data-Plus Nucleus

Data-Plus Nucleus is a replacement package to the OS-DMS Nucleus from OSI. All of the programs from the original except SORT have been duplicated and enchanced and new software, the MC-DMS Interface, has been added. The name "MC-DMS" stems from the extensive use of machine code support built into the utilities to replace slower, BASIC code. Features include; (1) MC-DMS Interface code supports up to 8 Master files simultaneously without requiring OPEN/CLOSE commands under Level 3 at every file access. The only 65U software support needed for Level 3 file access is semiphores. This produces a significant increase in speed. READ, WRITE, and FIND commands operate on the field level. FIND skips over embedded garbage between fields eliminating the need for embedded blanks, and automatically stops on the last record in the file. (2) Machine code DIR utility. Ultra-fast. Automatic paging. ^C interrupt. Can selectively list by file type or can search for file name matches with wildcards. (3) Machine code file manager. Creates, deletes, or renames files in a flash. The file manager is linked to the Master/Key file creation utility. (4) Machine code file transfer/merge. Grabs up to 30 records per pass. Single/dual drive. Fully selectable field specifications. Also allows searching for matches in source and destination files for linked merges. (5) Machine code single/dual drive floppy diskette copier. Moves up to 7 tracks per pass. (6) Disk-based mailing label printer. Stores printing format designs on disk. Selectable fields and record range, Key file access, searches, and more. (7) Disk-based report writer. Stores report format designs on disk. Same features as above, but with formatted columns by type and width. (8) Edit-Plus 65U. Most of the same features as the 65D version. Suitable for correspondence and form letters. (9) Data-Plus Mail Merge. Complete documentation allows implimenting the MC-DMS Interface into your own applications. $150.00

AD$

For Sale: 12 fully-populated, used, OSI 520 boards. Each provides 16K of static RAM. Totally unwarranted to do anything. $10.00 each or $75.00 for all 12 - plus postage. Contact PEEK[65]

For Sale: OSI C2-8P-DF, 48K, 540 video board, two 8" floppy drives, 12" monitor and cable for paralell printer, WP-3-1, OS-DMS Nucleus and Sort, OS-65D V3.0 and OS-65U v1.2 operating systems. Documentation and 60 disks. $300.00. Call 412-654-2381

C8P upgraded to multi-user C3-A, 56K, D&N 1600IO (serial and paralell), Visual 200 Terminal, Centronics 704 printer, also Corona TP1, Citizen graphics, much software, extra boards. Call 715-249-5323 evenings (Central Time).

Wanted: OSI 470 printer board with Centronics paralell cable for C2-OEM. Don Cotton, Burnic Enterprises, RR3, Scott City, KS 67871, 316-872-5826

FORTH $24.95, Utilities available also. Free catalog. Aurora Software, 37 South Mitchell, Arlington Heights, IL 60005

Have you got something to sell? Why not take out a classified ad in PEEK? Ads cost 35 cents per word, not including "price" words. Copy is due 30 days before the cover month.

Watch This Space Grow

PEEK [65]
PO Box 586
Pacifica, CA 94044
415-359-5708

Bulk Rate
U.S. Postage
PAID
Pacifica, CA
Permit #92
Zip Code 94044

DELIVER TO:

GOODIES for OSI Users!

PEEK [65]
The Unofficial OSI Users Journal

( )	**C1P Sams Photo-Facts Manual.** Complete schematics, scope waveforms and board photos. All you need to be a C1P or SII Wizard, just	$7.95 $ ______
( )	**C4P Sams Photo-Facts Manual.** Includes pinouts, photos, schematics for the 502, 505, 527, 540 and 542 boards. A bargain at	$15.00 $ ______
( )	**C2/C3 Sams Photo-Facts Manual.** The facts you need to repair the larger OSI computers. Fat with useful information, but just	$30.00 $ ______
( )	**OSI's Small Systems Journals.** The complete set, July 1977 through April 1978, bound and reproduced by PEEK (65). Full set only	$15.00 $ ______
( )	**Terminal Extensions Package** - lets you program like the mini-users do, with direct cursor positioning, mnemonics and a number formatting function much more powerful than a mere "print using." Requires 65U.	$50.00 $ ______
( )	**RESEQ** - BASIC program resequencer plus much more. Global changes, tables of bad references, **GOSUBs** & GOTOs, variables by line number, resequences parts of programs or entire programs, handles line 50000 trap. Best debug tool I've seen. MACHINE LANGUAGE - VERY FAST! Requires 65U. Manual & samples only, $5.00 Everything for	$50.00 $ ______
( )	**Sanders Machine Language Sort/Merge** for OS-65U. Complete disk sort and merge, documentation shows you how to call from any BASIC program on any disk and return it or any other BASIC program on any disk, floppy or hard. Most versatile disk sort yet. Will run under LEVEL I, II, or III. It should cost more but Sanders says, "...sell it for just..."	$89.00 $ ______
( )	**KYUTIL** - The ultimate OS-DMS keyfile utility package. This implementation of Sander's SORT/MERGE creates, loads and sorts multiple-field, conditionally loaded keyfiles. KYUTIL will load and sort a keyfile of over 15000 ZIP codes in under three hours. Never sort another Master File.	$100.00 $ ______
( )	**Assembler Editor & Extended Monitor Reference Manual** (C1P, C4P & C8P)	$6.95 $ ______
( )	**65V Primer.** Introduces machine language programming.	$4.95 $ ______
( )	**C1P, C1P MF, C4P, C4P DF, C4P MF, C8P DF Introductory Manuals** ($5.95 each, please specify)	$5.95 $ ______
( )	**Basic Reference Manual** — (ROM, 65D and 65U)	$5.95 $ ______
( )	**C1P, C4P, C8P Users Manuals** — ($7.95 each, please specify)	$7.95 $ ______
( )	**How to program Microcomputers.** The C-3 Series	$7.95 $ ______
( )	**Professional Computers Set Up & Operations Manual** — C2-OEM/C2-D/C3-OEM/C3-D/C3-A/C3-B/C3-C/C3-C'	$8.95 $ ______

TOTAL	$ ______
CA Residents add 6% Sales Tax	$ ______
C.O.D. orders add $1.90	$ ______
Postage & Handling	$ 3.70
TOTAL DUE	$ ______

POSTAGE MAY VARY FOR OVERSEAS

Name ______________________
Street ______________________
City ______________ State ________ Zip ________