PEEK (65)

The Unofficial OSI Users Journal

$1.75

JUNE 1983
Vol.4, No.6

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

INSIDE

Column One

While we were looking over this month's issue pasted to the wall my eye fell on a pile of back issues, with December 1982 on top. Column one last December was pretty grim, sort of like December weather. This month's Column One will be much more like June.

Maybe it's because the new OSI machines use my favorite architecture. Maybe it's because this month's PEEK(65) has a particularly good mix of hardware and software articles on everything from C1P's to 230E's. Or maybe it's because it finally stopped raining this week and I got my little sailboat into the water. For whatever reason, things seem so much cheerier now that the December Column One reads like a letter from a foreign country.

We recently issued another in our series of Calls for Articles. Let me be a bit more specific. Of course, we want articles and stories on whatever you are doing with your computer: but more specifically, we would like to start a dialog in PEEK(65) on programming technique.

What does "programming technique" mean? It means the process you use to develop a new application on your computer. How do you start? Where do you start? Do you write down the specifications for the job in detail? Do you generate a flowchart? Do you just start typing in program lines?

After you have started, written the specs or charted the flow, where do you go from there? How do you check your progress as you go along? How do you make sure your program will really do what you want it to, no matter what the input data might be? How do you document what you have done, so that when something goes astray 9 months later you will be able to figure out how to fix it?

Now here comes the kicker... we want to hear what you REALLY DO, not what the programming textbooks all SAY you SHOULD DO! Sure, we all know about top-down design, about structured programming, about standard subroutines, about complete documentation. I even know one guy who does all that stuff! But we all know lots of folks who don't. And some of them produce some very good programs. The question is, are these guys productive because they ignore the rules, or in spite of ignoring the rules, or do they maybe have some rules of their own which they intentionally or accidentally follow?

In other words, what makes a good programmer, really? Since I know that you personally are a very good programmer, what is your secret?

Share it with us in a letter or an article.What with all the changes at OSI, it's also time to update our Reader Profile. We need to know several things from you.

What equipment do you have, both OSI and otherwise?

What operating system(s) do you use?

What do you use your computer for? Business? Hobby? Development? Consulting? Other?

What do you want from us? Hardware articles? Software articles? Ads? Something we don't do at all now?

What would you like us to stop doing or do less?

Send your information, comments, suggestions and gripes to PEEK(65). Please do make a special effort to respond to this one ... it will help us all to help each other!

MORE THOUGHTS ON 65U 230E Time share & v1.43

by: Colin Law
c/o Box 3819
Auckland, New Zealand

My article of November 1982 (Vol.3 No.11) explained developments which had occurred in translating from C3 with 65U v1.2 up to 230E time share and 65U v1.42, and a brief piece in January 1983 (Vol.4 No.1) cleared up semaphore checking problems. In the 8 months since the 230E was installed I have made some further progress and now have 3 users instead of two.

In all of the program listings with this article I have added REM statements to explain the workings; in everyday use they are, of course, much smaller. My articles have resulted in interesting correspondence with PEEK(65) contributor Fred Schaeffer and I would welcome exchanges of ideas with anyone else using similar systems.

TEMPORARY DATA FILES

KYUTIL and a number of other utilities use temporary data files for parameters and for sorting - files such as PASVAL, WORK1 and WORK2. With a three user system I had to consider the problem of all users wishing to use KYUTIL: should I make them wait until the semaphore is clear, or give them their own files? Since key file loading is likely to take up to half an hour (better than the 24 hours before KYUTIL!) I decided that users don't need to wait when they wish to access different master files. I created new files PASVL0, PASVL1, and PASVL2, together with WORK10, WORK20, WORK11, WORK21, WORK12, WORK22. You will see that the last digit represents the User number and within KYUTIL, MOVE, SORT, BACK, and other utilities I have a routine like this:

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

```
10 :   REM   Demonstration of date routines
20 :   REM   Valid for dates after 01.01.01
21 :   REM   Change order where appropriate for USA format MM.DD.YY
30 :
40 FOR I=1TO7 :READ DW$(I) : NEXT
50 DATASUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY
60 :
100 PRINT"Please enter date in form DD.MM.YY  "; : INPUT DT$
101 :     REM will also accept D.M.YY  e.g. 1.1.83
102 :     REM will also accept DD/MM/YY or D/M/YY
110 DD$=DT$ : GOSUB4500 : ED$=DD$ : REM Encode
120 PRINTDT$;" Encoded form: ";ED$ : PRINT
130 GOSUB4000 : DE$=DD$ : REM Decode
140 PRINT"You entered ";DT$;"  Encoded to: ";ED$;
150 PRINT"  Decoded to: ";DE$ : PRINT
155 REM   **   Encode & decode to verify that the routines bring  **
160 REM   **  back same date !                                     **
170 GOSUB8500 : REM Find Julian day number
180 GOSUB9000 : W1=W : REM find day of week
190 GOSUB8600 : REM Now subtract one from date
200 GOSUB9000 : W2=W : REM find day of week
210 PRINT"You entered ";DT$;TAB(25);"the day before is: ";DD$
220 PRINT"That is day";W1;" ";DW$(W1);TAB(25);
230 PRINT"that is day";W2;" ";DW$(W2) : PRINT
300 END
4000 :           REM  Date decode DD$/IN&OUT
4001 :       REM This expands the date from YYMD format to DD.MM.YY
4010 IFASC(DD$)<48ORASC(DD$)>57ORLEN(DD$)<4THEN4080
4020 IFASC(MID$(DD$,2))<48ORASC(MID$(DD$,2))>57THEN4080
4030 T3$=LEFT$(DD$,2)
4040 T2=ASC(MID$(DD$,3,1)):T2$=RIGHT$(STR$(T2+36),2)
4050 T1=ASC(RIGHT$(DD$,1)):T1$=RIGHT$(STR$(T1+36),2)
4060 IFT1>90THENT1$=RIGHT$(STR$(T1+30),2)
4070 DD$=T1$+"."+T2$+"."+T3$
4080 RETURN
4500 :           REM  Date encode DD$/IN&OUT
4501 :       REM This compresses from DD.MM.YY or DD/MM/YY to YYMD
4510 L9=LEN(DD$):IF DD$="0"ORDD$=""ORDD$=" "THEN4610
4520 IFL9>8ORL9<6ORMID$(DD$,L9-2,1)"/"THEN4600
4530 T1=VAL(LEFT$(DD$,2)):IFT1<10RT1>31THEN4600
4540 T1$=CHR$(T1+64):IFT1>26THENT1$=CHR$(T1+70)
4550 TY$=RIGHT$(DD$,2)
4560 T2=VAL(MID$(DD$,L9-4,2))
4570 IFT2<1THENT2=VAL(MID$(DD$,L9-3,1))
4580 IFT2<10RT2>12THEN4600
4590 T2$=CHR$(T2+64):DD$=TY$+T2$+T1$:GOTO4610
4600 L9=99:ME$="Date invalid":GOSUB60000
4601 :       REM L9=99 is used as a flag for invalid date
4602 :       REM gosub 60000 is an error message & beep routine
4610 RETURN
4620 :
8500 REM Date minus one routine  T1$,T2$,T3$>IN   DD$>OUT
8520 D=VAL(T1$):M=VAL(T2$):Y=VAL(T3$)
8530 XD=INT(30.57*M)+INT(365.25*Y-395.25)+D
8540 IFM<3THEN N=XD:GOTO8570
8550 IFINT(Y/4)*4=YTHEN N=XD-1:GOTO8570
8560 N=XD-2 : REM Day number
8570 RETURN
8580 :
8600 N=N-1 :REM minus one day
8610 Y=INT(N/365.26)+1
8620 D=N+INT(396-365.25*Y)
8621 :       REM original amended from 395.25 to 396
8622 :       REM I don't know why but it works correctly for me !
8630 D1=2:IFINT(Y/4)*4=YTHEND1=1
8640 IFD>91-D1THEND=D+D1
8650 M=INT(D/30.57):D=D-INT(30.57*M)
8655 DD$=RIGHT$(STR$(D+100),2)+"."+RIGHT$(STR$(M+100),2)+"."
8660 DD$=DD$+RIGHT$(STR$(Y+100),2)
8670 RETURN
8680 :
9000 REM Find the day of the week
9001 :    REM I don't use this one, but it appears to work all right
9010 W=(N+1)/7 : W=INT((W-INT(W))*7+1.1)
9020 RETURN
```

10 UN = PEEK(55381) : PV$ = "PASVL" + RIGHT$(STR$(UN),1) to assemble the appropriate data file name. Thus all users can access KYUTIL and load and sort their own key files. Semaphores are used only on the master files.

DATES

My earlier articles explained how our need for many date fields led me to compress dates down from DD.MM.YY to YYMD which also brings added value of being able to sort dates with KYUTIL. A further problem came when I needed to subtract one day from a date - for example when a new salary allowance commences it is necessary to print that the existing allowance ceases one day before. With regular dates the program becomes longer and longer as you allow for:

- jumping to previous month if original date is 01

- include routine for 30 days hath September ... & etc

- jumping back a year if original date is January 1

- then of course you have to think what would happen in a leap year!

Another need for date manipulation was in a program to "cost" salary increases, i.e. if salary is increased from $20.123 to $21.456 at October 11, 1982 what will be the cost in the financial year to March 31, 1983?

I found two articles a couple of years ago, one came from Byte magazine, one from somewhere else, but neither of them worked properly for me!

The two needs led to development of the date routines shown in listing 1. The segment from the Byte article is around 8500-8699 with a note on the part that was changed to make it work. The demonstration routines show how to compress dates for storing to file in YYMD format and how to get the Julian day number. Converting back and forth with these subroutines is very easy. In our costing program these routines are added to salary scale data files so that all you do is enter date 1, date 2, grading 1 and grading 2 and the cost appears almost immediately on screen. Listing 1 will also calculate the day before a given date. The date is converted to Julian day, one subtracted, then converted back

Listing 2a

```
1 REM LEAVE PROGS 10/82 (LVREAD)
10 CLOSE : CLEAR : PG$="LVREAD" : PW$="PASS" : SL=55
20 CM=2: CM$(1)="LVS830": CM$(2)="LVR830": CP$(1)="PASS":CP$(2)=CP$(1)
30 PRINT TAB(20); "Please wait" : PRINT
40 RUN ["TVCOM","PASS",10] : REM COMMONS
50 REM Semaphores not locked - start line 55
55 DIM GZ(30),GZ$(30)
200 PRINTSC$; TAB(20); "LEAVE PROGRAM - READ" : PRINT
```

Listing 2b

```
1 REM (TVCOM)  TVNZ Common terminal, flag and file set-up
2 IF PG$ = "" OR SL=0 THEN 50000 : REM Faulty call
10 CLOSE: PRINTCHR$(27);CHR$(28);CHR$(27);CHR$(17);"`w";Please wait";
15 FLAG2:FLAG5:FLAG9:FLAG11:FLAG15:FLAG18:FLAG21:FLAG23:FLAG27
19 :       REM below are the Hazeltine codes we use most
20 S$=CHR$(27) : SU$=S$+CHR$(31) : SD$=S$+CHR$(25) : NC$="" : EE$="/"
30 SP$=S$+CHR$(17) : SE$=S$+CHR$(23) : CL$=S$+CHR$(15) : BP$=CHR$(7)
40 SB$=S$+CHR$(23) : SC$=SP$+"``"+SB$ : SR$=CHR$(13) :PRINTSR$;
45 BB$="          " : BB$=BB$+BB$+BB$+BB$ : DT=55919 : DT$=""
50 DT$ = DT$ + RIGHT$(STR$(PEEK(DT+3)+100),2) + "."
55 DT$ = DT$ + RIGHT$(STR$(PEEK(DT+4)+100),2) + "."
60 DT$ = DT$ + RIGHT$(STR$(PEEK(DT+5)+100),2)
61 :       REM above here routine to get date
65 UN = PEEK(55381) :PD=5 :IF UN>0 THEN PD=8 :GOTO80
66 :       REM user 0 normal to printer #5, other users to #8
70 POKE15908,66 :POKE14457,66:POKE23,163:POKE24,INT(163/14)*14+1
71 :       REM paging & terminal width for #5 printer
80 PRINTSP$;"lw"; : IF CD$ <> "A" THEN CD$ = "E"
85 DEV CD$ : E0 = 23730 : E1 = 23731
86 :       REM E0 and E1 used for INP$ terminate input with escape
90 GOSUB 400 : GOSUB 200 : GOSUB100
95 DEV CD$ : RUN[PG$,"PASS",SL]
96 :     REM  SL is return line to calling program
97 :     REM  50 is normal, but if 55 then semaphores are not set,
98 :     REM  user is simply warned if files are in use - thus SL
99 :     REM  is 55 for READ only programs - e.g. report writer
100 CLOSE : OPEN"DIREC*","PASS",1 :PRINTSP$;"lw";
101 :   REM  Find out semaphore numbers - item 16 of DIREC* entry
110 FOR CH = 1 TO CM : SM(CH)=0 : INDEX(1)=0 : FIND CM$(CH),1
120 A2=PEEK(520) : A1=PEEK(528) : AD=A1*256+A2
130 SM(CH)=PEEK(AD+15)  : NEXTCH :CLOSE :PRINTSR$;
150 :   REM Verify file(s)
155 FOR CH = 1 TO CM : IF SM(CH)=0 THEN 195
160 FOR L=1TO3: POKE19632,1 :PRINTSP$;"`w";"Please wait";:POKE22,0
165 WAIT FOR SM(CH): IF PEEK(19633)=0 THEN 175
170 FORL=1TO1:NEXT: GOTO190
175 PRINTSR$;:NEXTL:PRINTBP$;SP$;"`";CHR$(2+CH*3);"The file you ";
180 PRINT"require is in use at another terminal  (";CM$(CH);")"
181 IFSL=55THEN SM(CH)=0 :PRINTBP$; :GOTO195 :REM warning only
182 PRINT"Perhaps you could try again later.  "
183 FORQ=1TO200:NEXTQ
185 PRINT:PRINT:PRINT:GOTO50260 : REM lockout is semaphore set
190 IF SL=55 THEN WAIT CLEAR  SM(CH) :SM(CH)=0
195 NEXT CH: RETURN
200 :
210 FOR CH = 1 TO CM
220 CLOSE: OPEN CM$(CH), "PASS", CH : PRINTSR$; :POKE22,0
225 IF RIGHT$(CM$(CH),1)<>"0" THEN 290
230 INPUT%CH,T$ : IF T$ <> LEFT$(CM$(CH),5) THEN ER=1 : GOTO40000
240 INDEX(CH)=9 : INPUT%CH,EODF(CH) : INDEX(CH)=20 : INPUT%CH,BODF(CH)
250 INDEX(CH)=31 : INPUT%CH,RL(CH) : INDEX(CH)=42 : INPUT%CH,RN(CH)
260 INDEX(CH)=53 : N(CH)=1
270 PRINTSP$;"lw";"("; :INPUT%CH,T$ : INPUT%CH,T
280 IF INDEX(CH)<BODF(CH) THEN N(CH) =N(CH)+1: PRINTSR$; :GOTO270
290 IF N(CH) > NN THEN NN=N(CH)
295 NEXT CH
300 CLOSE:RETURN
400 :  REM CHECK PRINTERS peek to see if in use
410 P5=PEEK(56427):P8=PEEK(56430):PRINTSP$;"``";
420 PRINT"Spinwriter Printer ";:IFP5<>127THENPRINT"IN USE":GOTO430
425 PRINT"FREE"
430 PRINT:PRINT"C-ITOH Printer ";:IFP8<>127THENPRINT"IN USE":GOTO450
435 PRINT"FREE"
450 RETURN
40000 IFER=1THEN PRINT"FILE HEADER ERROR":PRINT
50000 :      REM usual error routines below here .....
50230 :
```

Listing 2b continued

to Gregorian (normal) date again. I am sure that anyone handling dates in BASIC by tedious "30 days hath September....." routines will be able to adapt this to their needs.

Note: Gregorian calendar is the one you are used to, Julian calculates the day number from a fixed past date - no years, months, etc, just day number nnnn.

COMMON PROGRAMS

Listing 2 represents one of my most powerful applications of the "common" applications with v1.42. Originally at least 30 programs used lines 10 to at least 300 to set-up standard screen addressing, check files, set semaphores, flags and other standard jobs. With TVCOM I now do this in an absolutely standard form through the one program! When I need to write a new program I simply write the first few lines as shown in listing 2a and all of my normally used routines are guaranteed to standard format. I should mention that I ignore the OSI supplied routines for CRT independence since they are too complex and don't include a complete set of screen commands for Hazeltine terminals. We use only Hazeltine Esprit terminals and I can't imagine ever knowingly setting up a system with several different terminals since the operators would certainly not appreciate having different key layout, screen size (& colour), etc. If I change ALL of our terminals then it will be a simple matter to amend TVCOM as appropriate.

DIRECTORY

My DIRectory now comes in two forms, one standard and the other sorted. I also list the passwords, semaphore numbers and reference numbers for back-up floppy discs and tracks. With 144 files on the 7meg hard disc it's clear why I don't back up one file at a time! My back up system stores 64 hard disc sectors (=64 floppy tracks) on each floppy without regard to file boundaries (refer PEEK(65) Vol.2 No.11). Listing 3a gives extracts from DIR and a sample segment of directory. Listing 3b is extracts from DIRSRT and a sample of sorted directory. These extracts should be enough to show you what it's all about. You will note that I have abbreviated some of the information - when the sector boundary is always Y/Y I seriously wonder about omitting

Listing 2b continued

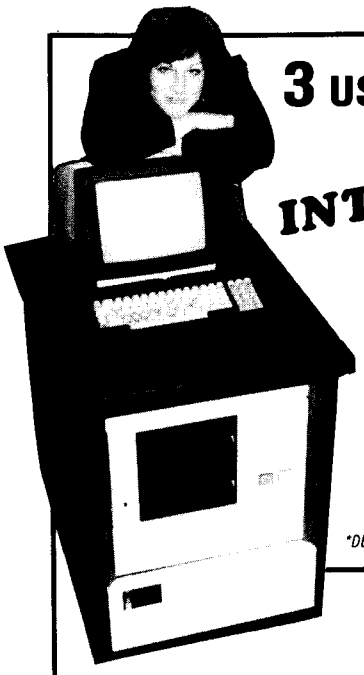
```
50240 CLOSE:DEVCD$(1):POKE15006,0
50260 FORCH=1TOCM :IF SM(CH)=0 THEN 50262
50261 WAIT CLEAR SM(CH)
50262 NEXT
50265 PRINT:INPUT"(Return) to TV MENU ";QA$: FLAG26
50270 FLAG24:FLAG28:CLOSE:DEV"E":IFQA$="STOP"THENEND:NEW
50280 :
50290 CLOSE:DEV"E":FLAG26:FLAG28:RUN"TVMENU"
63999 DEV"E":SAVE"TVCOM","PASS"
```

Listing 3a

Extracts from modified DIR

```
10 REM 65U DIRECtory V1.53
15 REM (C) OSI 1982


110 PRINT:PRINT:PRINT"--------------------------------"
115 PRINT": OS-65U File Directory Utility :"
120 PRINT"--------------------------------":PRINT
125 CC=PEEK(2073):IM=PEEK(2888):POKE15006,0
130 U1SR=PEEK(8778):U2SR=PEEK(8779)
140 GOSUB 150 :PRINT TAB(55);DT$ :GOTO 200
145 :
150 D=55919: FORI=0TO5: D$(I)=MID$(STR$(PEEK(D+I)+100),3,2): NEXT
160 DT$=D$(2)+";"+D$(1)+":"+D$(0)+"  "+D$(3)+"."+D$(4)+"."+D$(5)
170 RETURN
180 :
200 LVL = PEEK(16317) : REM curr lev


300 DEV DV$: OPEN"DIREC*","PASS",1: CLOSE 1
305 PRINT:PRINT"Directory Page to start from:  ";
310 PG=1:INPUTPG$:IFPG$()""THENPG=VAL(PG$):PRINT
315 IFPG(1ORPG)25THENPRINT"???":GOTO300
320 :


390 GOSUB1170:GOSUB150:IFDV=1THENPRINTCHR$(27);CHR$(28)
400 PRINT#DV,TAB(4);"OS-65U File Directory For DEVice ";DV$;
405 PRINT#DV,TAB(44);"Time      Date" :TB=64:IFDV$="E"THENTB=72
410 PRINT#DV,TAB(4);:FORX=1TO34:PRINT#DV,"-";:NEXT
415 PRINT#DV,TAB(44);DT$
420 PRINT#DV:PRINT#DV:T$="Sect  Backup":IFDV$()"E"THENT$="Track"
430 PRINT #DV, TAB(0);"Name"; TAB(9);"Type"; TAB(14);"Access";
440 PRINT #DV, TAB(22);"Address"; TAB(31);"Length";
450 PRINT #DV, TAB(39);"S/B";TAB(44);T$;TAB(TB-9);"P/W  Sem"
460 FOR I=0 TO TB : PRINT #DV, "-";: NEXT: PRINT #DV
470 :


665 REM  read dir bit
670 TM=PEEK(RT+8)
675 D$="   ": IF (TM AND 128)()0 THEN D$ = "DIR"
680 PX(2)=PEEK(RT+6):PX(4)=PEEK(RT+7):PX(1)=INT(PX(2)/16)
685 PX(3)=INT(PX(4)/16):PX(2)=PX(2)-PX(1)*16:PX(4)=PX(4)-PX(3)*16
689 IFPX(1)=15ANDPX(2)=15ANDPX(3)=15ANDPX(4)=5THENPX$="PASS":GOTO700
690 FOR I=1TO4:IFPX(I)=15THENGOTO (1AND1)+692
691 NEXT:GOTO698
692 PX(I)=-13:GOTO691
693 PX(I)=25:GOTO691
698 PX$="":PX$=PX$+CHR$(PX(1)+65)+CHR$(PX(2)+78)
699 PX$=PX$+CHR$(PX(3)+65)+CHR$(PX(4)+78)
700 SM=PEEK(RT+15)
710 REM ty


910 REM data
915 IF PEEK(RT)=1ANDIT=2THENN$="[----]":TY$="Deleted":AR$=""
920 IF PEEK(RT)=1 AND IT()2 GOTO 1000
930 IFDV$="E"THENGOSUB2000
940 :
950 PRINT #DV, TAB(0);N$; TAB(9);TY$; TAB(15);AR$;
955 PRINT#DV,TAB(21);RIGHT$(STR$(DA),8);TAB(30);RIGHT$(STR$(SZ),7);
960 SB$="N": IF DA / 3584 = INT (DA / 3584) THEN SB$ = "Y"
965 SL$="N": IF SZ / 3584 = INT (SZ / 3584) THEN SL$ = "Y"
```

Continued on page 6



3 USERS-80 Mega Bytes —$8990.00*

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

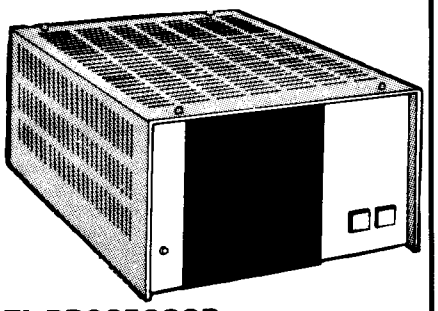
INTRODUCTORY SPECIAL

- 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

it altogether in future. Another facility is to start from a particular "page" of the directory. Listing through 144 files can be tedious when you know that you want to look only at the last two entries. You are asked "start from page ?" and the listing starts at directory address (page number * 256). The only thing to watch here is the end of DIRectory routines which tell you what space is left, how many files in use, etc. Since I use the "page" facility only for quick console references to DIREC* entries (usually the most recent) it doesn't worry me that my "paging" routine throws them astray.

MEMORY MAP

Roger Clegg's 65U memory map (PEEK(65) Vol.4 No.3) led me to compare with my own map compiled over the years as various bits & pieces were revealed. Listing 4 gives additional locations which may be of use.

LOG ENTRIES

I have amended my TVLOGG (LOGLIS program PEEK(65) Vol.3 No.11). With increased use of the system I found that log entries could be made shorter, thus the log need only be cleared out every couple of weeks. Instead of a full time/date entry it enters the date only at start-up and log-off (8.30am and 5pm). For the rest of the day the entries have HHMM time plus user number, operator initials and NN number accessed on the main menu (up to 60 entries). This is enough to allow identification of how much the system is used and to retrace who used what and when. When the system is rebooted it searches for the last log-off and offers this as time and date for modification. (Why didn't OSI include battery clock). Log entries are stored as 15 character strings which are expanded by the log printer when required. Listing 5 shows how the string is assembled and how it is printed.

★

NOTES ON DISK OPERATING SYSTEM

By: Charles Stewart
3033 Marvin Drive
Adrian, MI 49221

The reader will note that this may seem to have no logical order and the writer's intended it that way. These are

Listing 3a continued

```
970 PRINT #DV, TAB(39); SB$;"/";SL$;: ZC=INT(DA/3584*100)/100
975 PRINT#DV,TAB(43);ZC;TAB(50); :IFDV$()"E"THEN990
980 IFZC<9THEN PRINT#DV,"BU 33";:GOTO990
985 PRINT#DV,Z$(1);":";Z$(2);TAB(55);")";TAB(56);Z$(3);":";Z$(4);
990 PRINT#DV,TAB(TB-9);PX$;TAB(TB-4);SM;
992 PRINT#DV
995 :


1340 PRINT#DV:IF PEEK(9832))3THENSS=HS
1345 IFPG)1THENPRINT#DV,"Directory from Page ";PG;"   ";
1350 IFHA<(PG*7168)THEN1365
1355 PRINT#DV,SS-HA"Bytes Free  ";
1360 PRINT#DV,INT((SS-HA)/3584);"Sector(s) Free"
1365 IFRE<10THEN1380
1370 PRINT#DV,RE;"Bytes Recoverable  ";
1372 PRINT#DV,INT(RE/3584);"Sector(s) Recoverable":PRINT#DV
1380 PRINT#DV,EC-1-DC;"File(s) in use  ";
1382 PRINT#DV,DC;"File(s) deleted":IFPG)1THEN1400
1384 PRINT#DV,EC-1;"Total file(s) defined of";ES/16-1;"possible"
1390 :


2000 REM CALC BACK FLOPTR
2010 Z(9)=INT(DA/3584) : Z(9)=1+(Z(9)-9)/64 : REM Z(9)=DISC&FRACT
2020 Z(1)=INT(Z(9)) : Z(2)=9+(Z(9)-Z(1))*64 :REM DISC&TRCK
2030 Z(9)=INT(SZ/3584)+INT(DA/3584) : Z(9)=1+(Z(9)-9)/64 :REM DISC & FR
2040 Z(3)=INT(Z(9)) : Z(4)=9-1+(Z(9)-Z(3))*64 : REM DISC & TRACK
2050 FORT=1TO4:Z$(T)=MID$(STR$(Z(T)),2):NEXT
2060 RETURN
```

```
    OS-65U File Directory For DEVice E      Time      Date
    ----------------------------------      13:28:00  11.04.83

Name     Type Access  Address  Length  S/B  Sect  Backup       P/W  Sem
-------------------------------------------------------------------------
DIREC*   Other None    25088    7168    Y/Y  7     BU 33        PASS  0
BEXEC*   Basic Read    32256    7168    Y/Y  9     1:9  )1:10   PASS  0
DIR      Basic Read    39424    7168    Y/Y  11    1:11 )1:12   PASS  0
RENAME   Basic Read    46592    7168    Y/Y  13    1:13 )1:14   PASS  0
COPYFI   Basic Read    53760    10752   Y/Y  15    1:15 )1:17   PASS  0
COPIER   Basic Read    64512    10752   Y/Y  18    1:18 )1:20   PASS  0
DELETE   Basic Read    75264    3584    Y/Y  21    1:21 )1:21   PASS  0
//       Basic Read    78848    3584    Y/Y  22    1:22 )1:22   PASS  0
EDITOR   Basic Read    82432    10752   Y/Y  23    1:23 )1:25   PASS  0
SEMCHK   Basic Read    93184    7168    Y/Y  26    1:26 )1:27   PASS  0
INP$     Basic Read    100352   10752   Y/Y  28    1:28 )1:30   PASS  0
```

Listing 3b

```
60 DV=5: POKE15908,60:   REM Leave as line 60
70 :
72 LV=PEEK(16317):IFLV=3THENUN=PEEK(55381)
73 IFUN()0ANDDV()5THENRUN
75 T=PEEK(9832):IFT)127THENT=T-124:IFT)63THENT=T-58
80 SD$=CHR$(65+T) :        REM current device
85 CLOSE :DIM S1$(450),S2%(450),S3%(10,450),PK(10) :KK=1:SI=0
90 FLAG2:FLAG5:FLAG9:FLAG11:FLAG16:FLAG18


300 DEVDV$:OPEN"DIREC*","PASS",1: CLOSE 1
350 HS=7 311 360:FS=275968:SS=FS
360 GOSUB1170: GOTO500    REM setup xfrs then start DIR
380 :
390 :                       REM HEADING SUB
395 :
400 D=55919:FORI=0TO5:D$(I)=MID$(STR$(PEEK(D+I)+100),3,2):NEXT
410 DT$=D$(2)+";"+D$(1)+":"+D$(0)+"  "+D$(3)+"."+D$(4)+"."+D$(5)
420 PRINT#DV,TAB(4);"OS-65U Sorted Directory for DEVice ";DV$;
430 PRINT#DV,TAB(50);"Time        Date"
```

Listing 3b cont. on p.7

notes of things we have discovered while working with our systems.....

After boot up answering PASS will open system on almost all OSI operating systems in BASIC this includes the game disks...

Available basic commands.....

EXIT puts user into command kernel

DISK!"IO ,03" sets save mode for printer or tape

DISK!"IO ,02" returns to normal video output only

DISK!"IO 01" (note no comma) load from tape until error detect such as OK message. Use to load tapes into 65D system...any noise will lose the IO command...

DISK OPEN.N,"NAME" open random or sequential file

DISK CLOSE N close file

DISK!"PUT NAME" save user program "name"

DISK!"LOAD NAME" load user program....

RUN"NAME" run program...

Listing 3b continued

```
435 PRINT#DV,TAB(4);:FORX=1TO34:PRINT#DV,"-";:NEXT
440 PRINT#DV,TAB(50);DT$ :PRINT#DV:PRINT#DV:T$="Sect  Backup"
445 IFDV$<>"E"THENT$="Track"
450 PRINT#DV,TAB(0);"Name";TAB(9);"Type";TAB(16);"Access";
455 PRINT#DV,TAB(23);"Address";TAB(33);"Length";
460 PRINT#DV,TAB(41);"Sec:B/L";TAB(50);T$
465 T=55:IFDV$="E"THENT=70
470 FORI=0TOT:PRINT#DV,"-";:NEXT:PRINT#DV :GOTO680
475 :
480 :

500 OF=16:REM skip embd header
510 REM read p of dir
520 DH=INT(EA/16777216):RM=EA-DH*16777216
530 DM=INT(RM/65536):RM=RM-DM*65536
540 DL=INT(RM/256):RM=RM-DL*256 :DB=RM
560 POKECB+1,DB:POKECB+2,DL:POKECB+3,DM:POKECB+4,DH
570 EL=PEEK(134)*PEEK(135):ER=USR(0)
580 IFER<>0THEN50080
600 RT=RA+OF:EC=EC+1 :REM ram adr cur ent, entry no.
620 IFPEEK(RT)=0THENN$="-ZERO-":KK=KK-1:GOTO4650: REM empty de
630 IFPEEK(RT)=1THENN$="[----]":GOTO655: REM deleted
650 N$="":FORI=0TO5:N$=N$+CHR$(PEEK(RT+I)):NEXT
655 S1$(KK)=N$: PRINTKK;S1$(KK);CHR$(13);
660 FORI=1TO10:S3%(I,KK)=PEEK(RT+I+5):NEXT:S2%(KK)=KK
665 KK=KK+1 :GOTO1080
670 :
675 :
680 SI=SI+1 :IFSI>KKTHEN1340 :REM now print all
690 FORI=1TO10:PK(I)=S3%(I,S2%(SI)):NEXT
695 TM=PK(3):D$="    ":IFTMAND128<>0THEND$="DIR"
730 TM=INT((TMAND28)/4):TY$="Other":REM read ty

900 SZ=256*(PK(7)+256*(PK(8)+256*PK(9))):REM rd sze
910 IFS1$(SI)="[----]"THENTY$="Deleted file":AR$="":REM data
930 GOSUB1070 :IFDV$="E"THENGOSUB2000
950 PRINT#DV,TAB(0);S1$(SI);TAB(9);TY$;TAB(16);AR$;
955 PRINT#DV,TAB(22);DA;TAB(33);SZ;
```

Listing cont. on page 8

DISK!"CALL 4000=01,1" gets track specified and loads to specified memory location starting with specified sector in the above example would load track number 1 to hex $4000 starting with sector 1.

DISK!"SAVE 01,1=4000/8" saves the information starting at the specified hex location to the track and sectors specified for the number of pages specified. In the above example would save the information starting at hex location $4000 to track number 1 starting at sector 1 for eight pages.

DISK GET,RECORD utilized in random or sequential files will get the specified record number.

DISK!"GO 4000" similar to the USR(X) function in ROM Basic. Will execute the machine language program specified by the four digit number given after the GO command.

X=USR(X) Will function as in ROM Basic but the vectors are in locations 8955 and 8956 HI LO same as 11 and 12 in ROM.

DISK!"EX 4000=12" Reads track number specified into the memory locations $4000 for 8 pages may also put track specified to screen memory starting at $D000 so you can view the entire disk.

DISK!"DI 12" Displays sector directory for the specified track number, note tracks 0 thru 9 must be prefixed with a 0 i.e. for track 1 type 01.

DISK!"IN" Initialize a new disk i.e. format for operating system use, --warning, completely erases a disk.

DISK!"XQ NAME" Gets file listed in directory under name and executes as a machine language program starting at $327E.

DISK!"XQ TT" loads machine language program at track specified (TT) and executes with $327E as load vector.

DISK!"HOME" Homes disk head at track 0.

Work space pointers 121,120 indicate bottom of work- space .... loc 132 and 133 indicate top of work space (in RAM) ... loc 23 terminal width - these locations in HEX.

COMMAND KERNEL---

All commands (DISK!) are available from the command

Continued from page 7

```
960 SB$="No":IFDA/3584=INT(DA/3584)THENSB$="Y"
965 SL$="No":IFSZ/3584=INT(SZ/3584)THENSL$="Y"
970 PRINT#DV,TAB(42);SB$;"/";SL$;
975 ZC=DA/3584:PRINT#DV,TAB(49);ZC;TAB(56);:IFDV$()"E"THEN990
980 IFZC<9THENPRINT#DV,"BackupDisc 33";:GOTO990
985 PRINT#DV,Z$(1);":";Z$(2);TAB(62);")";TAB(64);Z$(3);":";Z$(4);
990 PRINT#DV
995 :


4600 :               REM SORT ARRAY
4650 PRINT"Please wait - sorting ";KK
4655 MS=KK/156
4660 M=KK
4665 M=INT(M/2):PRINTCL$;:FORMT=1TO1+(M/MS):PRINT")";:NEXT
4670 PRINTCHR$(13);:IFM=0THENSI=0:PRINTCL$;CU$;CL$;:GOTO400:REM FINISH
4675 J=0:K=KK-M
4680 I=J
4685 L=I+M:IFS1$(I)<S1$(L)THENPRINT")";CHR$(13);:GOTO4715
4695 T$=S1$(I):S1$(I)=S1$(L):S1$(L)=T$
4700 T%=S2%(I):S2%(I)=S2%(L):S2%(L)=T%
4705 I=I-M:PRINT")";CHR$(13);:IFI>0THEN4685
4715 J=J+1:IFJ>KTHEN4665
4725 GOTO4680
4730 :
```

OS-65U Sorted Directory for DEVice E Time 13:38:56 Date 11.04.83

Name	Type	Access	Address	Length	Sec:B/L	Sect	Backup
/	Basic	Read	143360	3584	Y/Y	40	1:40 > 1:40
//	Basic	Read	78848	3584	Y/Y	22	1:22 > 1:22
1983-0	Data	Read	3300864	240128	Y/Y	921	15:25 > 16:27
1983-1	Data	Read	3186176	100352	Y/Y	889	14:57 > 15:20
A83REP	Basic	Read	3540992	10752	Y/Y	988	16:28 > 16:30
ACOST	Basic	Read	5236224	7168	Y/Y	1461	23:53 > 23:54
AFINAL	Basic	Read	5193216	14336	Y/Y	1449	23:41 > 23:44
ALABEL	Basic	Read	5221888	14336	Y/Y	1457	23:49 > 23:52
APRIL	Basic	Read	874496	3584	Y/Y	244	4:52 > 4:52
AREPOR	Basic	Read	5207552	14336	Y/Y	1453	23:45 > 23:48
BACK	Basic	Read	741888	3584	Y/Y	207	4:15 > 4:15
BASIC	Data	None	1989120	25088	Y/Y	555	9:43 > 9:49
BASIC2	Data	None	2014208	25088	Y/Y	562	9:50 > 9:56
BEXEC*	Basic	Read	32256	7168	Y/Y	9	1:9 > 1:10
BLOGGS	Data	Read	2110976	14336	Y/Y	589	10:13 > 10:16
BLUEFA	Data	Read	845824	7168	Y/Y	236	4:44 > 4:45
BLUEFW	Data	Read	835072	7168	Y/Y	233	4:41 > 4:42
BLUEP	Basic	Read	852992	7168	Y/Y	238	4:46 > 4:47
BLUESR	Basic	Read	842240	3584	Y/Y	235	4:43 > 4:43
BOOT	Basic	Read	189952	21504	Y/Y	53	1:53 > 1:58

Listing 4

Memory map additions

```
2884     32 means INP$ is on line
9824     6 bytes give name of last loaded or saved file
9842     6 bytes give name of last opened data file
11690-99   Dev # references  0=on line  32=off line
13316    Disk reference 255=floppy  129=7meg  1=23meg  16=36meg  0=74meg
14518    Single character input  (v.1.42 manual p4-A)
18176    Error type number
23700    255 to restore underline (95) when in line editor
23704    Toggle code for editor insert/overstrike
23705    Code for front of line (6=control F)
23706    Code for tab 8 spaces  (9=control I)
23707    Code for rear of line  (18=control R)
23721    Editor toggle location  0=insert  255=overstrike
23723-29  Editor extended control codes  (v1.42 manual p22)
23730-31  Editor escape character  (v1.42 manual p23)
52736-??  Multi user ACIA locations in steps of 2 - equivalent to
          64512,64513 for single user (or user 0)
55333-64  Semaphore locations in time share
56554    System level number
55556    0=C3 type no interrupts at console port $FC00
         255=C200/300 type, interrupt allowable
57217    Current cylinder
57218    Current track
57219    Current sector
```

Listing 5 on page 10

High Resolution Color Graphics

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

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

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

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

Call for new lower pricing.

Low Power Memory Board

Our popular MEM + board is ideal for:

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

Options include:

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

MEM + includes the following features:

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

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

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

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

To order, or for more information, contact:

Fial Computer
5221 S.W. Corbett
Portland, Oregon 97201
(503) 227-7083

Generic Computer Products

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

Listing 5

```
4165 OPEN"TVLOGO","PASS",1
4170 INDEX(1)=9:INPUT%1,EODF:INDEX(1)=20:INPUT%1,BODF
4175 INDEX(1)=31:INPUT%1,RL
4200 :
4210 T$="":FORI=2TO1STEP-1:T$=T$+RIGHT$(STR$(PEEK(I+55919)+100),2):NEXT
4220 RX=EODF:UT$=RIGHT$(STR$(UN),1)+" "+LEFT$(PI$,2)+RIGHT$(STR$(PG),2)
4230 IFPEEK(24570)=0THENUT$=UT$+"!!!"
4250 INDEX(1)=RX:PRINT%1,[RL-1,"R"]T$+UT$
4260 INDEX(1)=9:PRINT%1,[10,"R"]STR$(EODF+RL)
4290 CLOSE:WAIT CLEAR 99:POKE19632,60
4299 :   REM  PG is menu item number from main menu
4300 RETURN


2215 OPEN"TVLOGO","PASS",1:IX=BODF:K=1:TB=0
2220 INDEX(1)=IX:INPUT%1,ENTRY$:K=K+1
2221 IFMID$(ENTRY$,6,1)=" "THEN2225: REM Standard entry
2222 IFMID$(ENTRY$,5,1)>"9"THEN2260: REM Backup entry
2223 IFASC(MID$(ENTRY$,13,1))>90THEN2300:REM Log on/off entry
2225 PRINT#PD,LEFT$(ENTRY$,2);":";MID$(ENTRY$,3,2);"  ";
2230 PRINT#PD,MID$(ENTRY$,5,4);"  ";MID$(ENTRY$,9);
2240 GOTO2500
2250 :
2260 T$=":":PRINT#PD,LEFT$(ENTRY$,2);":";MID$(ENTRY$,3,2);"  ";
2265 PRINT#PD,MID$(ENTRY$,5,1);")";MID$(ENTRY$,6,1);" ";
2270 PRINT#PD,MID$(ENTRY$,7,2);" ";
2275 PRINT#PD,MID$(ENTRY$,9,3);"/";MID$(ENTRY$,12);
2280 GOTO2500
2300 PRINT#PD,LEFT$(ENTRY$,2);":";MID$(ENTRY$,3,2);":";
2310 PRINT#PD,MID$(ENTRY$,5,2);"  ";MID$(ENTRY$,7,2);".";
2320 PRINT#PD,MID$(ENTRY$,9,2);".";MID$(ENTRY$,11,2);"  ";
2330 PRINT#PD,MID$(ENTRY$,13);
2340 :
2500 IFINDEX(1)>=EODFTHEN2610
2520 TB=TB+1:IFTB>2THENTB=0:PRINT#PD
2525 PRINT#PD,TAB(TB*26);
2550 IFPD<>1ORK<60THEN2590
2560 FLAG27:PRINT:INPUT"<Return> to continue ";QA$:FLAG28:K=0:TB=0
2570 IFQA$="STOP"THEN WAIT CLEAR 99: CLOSE: GOTO2670
2580 IFQA$="ABORT"THENPRINT#PD,"#ABORT#";:WAIT CLEAR 99:CLOSE:GOTO2630
2590 IX=IX+RL:GOTO2220
2600 :
2610 PRINT#PD:PRINT#PD,"END OF LOG":PRINT#PD:CLOSE:WAIT CLEAR 99
```

★ ★ ★

kernel (entered from BASIC, Extended Mon., and Assembler by the command EXIT) without the prefix (DISK!)

RE BA - Re enters Basic after exit.

RE AS - Re enters assembler after exit.

RE EM - Re enters extended monitor after exit.

RE M - enters ROM monitor and sets DOS up for a re-entry with a 2547G command from the monitor. Alternate re-entry is $2A51.

★

SOFTWARE REVIEW

OSI GREATEST HITS Vol 2

by: Earl Morris
3200 Washington St.
Midland, MI 48640

OSI GREATEST HITS Vol 2 is a cassette tape of four game programs which include Roach Trap, African Escape, Moon Base Alpha and Hospital Adventure.

Roach Trap is a 3 Dimensional maze game. A bird's eye view is first given of each layer of the maze. Then the player is placed in the maze and a rat's eye view displayed. At intersections the player can turn up and down in addition to the normal left and right. The openings at each intersection are displayed using the OSI graphics characters. The screen is drawn using a machine language subroutine for speed. The screen can also be rotated by 90 degrees if you decide to walk on the walls or ceiling. The 3-D maze takes a good deal of concentration to keep track of where you are. The program is written assuming ROM basic and a 1P screen. Contrary to the advertisement, the program does NOT work for 2P machines.

The three remaining programs are mini - adventures. The Basic code for each is about 3K in length and each adventure has ten or less rooms. The adventures require the usual two word - noun verb - input to move about. Because of the short length of program, elaborate room descriptions and subtle hints and objects to examine are largely absent. After solving the adventures, I enriched the programs by adding my own descriptions and hints. The adventures do include a number of challenging situations to figure out. These three programs are written in common Basic and will run on any OSI machine including disk Basic.

Documentation includes playing instructions and full listings for all the programs.

★

MICROSOFT BASIC VS FBASIC ON THE ALASKAN PIPELINE PROBLEM

By: Robert Van Clampitt
210 Market Apt #H304
Galveston, TX 77550

The Alaskan pipe problem was posed to me when I was in college. The problem goes like this:

Since oil has been found in Alaska, Exxon U.S.A. has been attempting to find a way to pipe the oil to Baytown. A young control and routing engineer found that there existed twenty-one salt domes between Baytown and Alaska which had been connected with pipe lines. These salt domes would make ideal storage tanks, hence his problem was how to route the oil through each salt dome, but not through it twice. Can you find the route?

Figure #1 shows the pipeline and the salt dome relationship. The short Basic program to find the solution is shown in listing #1. The solution requires the formulation of a look-up matrix (22x22). The matrix tells the computer which salt domes are connected by pipelines by putting a true value in the correct position. Some initial basic analyses tell you that the first move must be through salt dome #12. Therefore, the other possibilites were not put into the matrix. The algorithm simply involves choosing the first empty pipeline and doing the same at the next salt dome. It does this until it cannot go any further. At this point it jumps back to the last salt

dome and picks the next available pipeline. The computer continues to do this until the solution is found.

Since this program took approx. 13.5 minutes in Microsoft basic, even with telling the computer that the first move had to be through salt dome 12. I wanted to see what other languages would do. When I purchased FBASIC, one of the first programs I translated was this pipeline problem. Since FBASIC is a subset of Microsoft Basic, it did require some modification. The resulting modified program is shown in listing #2. The major modification involved changing the initial two dimensional array to a one dimensional array that FBASIC can accept. The integer and lack of string function that FBASIC requires did not seriously hinder the programming of this problem. After compiling this program the answer appeared in outstanding time of 31 seconds. This is better than 25 times faster than Microsoft. Therefore, for speed with some restrictions in the use of string functions, I highly recommend FBASIC. Eliminating the initial move through salt dome #12 and programming in all possibilities, the FBASIC took 3 minutes, 25 seconds to find the solution. I did not try the Microsoft version, but suspect that it would take well over one hour to come up with the solution.

I wanted to program this program in FORTH. but I am not a very good FORTH programmer. I was unable to get my FORTH program error free. I would be interested in hearing from anyone that could get this program running in FORTH. I would like to thank PEEK(65) for keeping me informed on what is happening with OSI while I've been busy with medical school and unable to contribute to the magazine. In the way of introducing myself to the OSI community, I would like to say that I own a C8P with 48K and two 8" floppies video/serial. I would like to hear from anyone who knows how to change a 16K 520 board to an 8K board that will work under a serial system. That is my current project during my break from medical school. I'm also involved with a project that requires an A/D converter. Does anyone in PEEK(65) land have one that they are willing to sell a poor student?

I hope to contribute more to PEEK(65) in the future, but for now it is back to the books.

LIST #1

```
10 DIM A(25.25),B(22)
20 A(1,12)=1:A(12.1)=1
22 A(2,3)=1:A(3,2=1
24 A(2,5)=1:A(5,2)=1
26 A(2,7)=1:A(7,2)=1
28 A(2.9)=1:A(9,2)=1
30 A(3,8)=1:A(8,3)=1
32 A(3,13)=1:A(13.3)=1
34 A(3,14)=1:A(14.3)=1
36 A(4,5)=1:A(5.4)=1
38 A(4,8)=1:A(8,4)=1
40 A(4.10)=1:A(10.4)=1
42 A(4.16)=1:A(16,4)=1
44 A(5,9)=1:A(9,5)=1
46 A(5,11)=1:A(11.5)=1
48 A(5,16)=1:A(16,5)=1
50 A(6,12)=1:A(12.6)=1
52 A(6,20)=1:A(20.6)=1
54 A(7,13)=1:A(13,7)=1
56 A(8,9)=1:A(9,8)=1
58 A(8,14)=1:A(14.8)=1
60 A(8,15)=1:A(15,8)=1
62 A(10.15)=1:A(15.10)=1
64 A(10.19)=1:A(19.10)=1
66 A(11.16)=1:A(16,11)=1
68 A(13.17)=1:A(17,13)=1
70 A(14.15)=1:A(15.14)=1
72 A(14.17)=1:A(17,14)=1
74 A(14.18)=1:A(18.14)=1
76 A(16,19)=1:A(19,16)=1
78 A(15.22)=1:A(22.15)=1
80 A(17,21)=1:A(21.17)=1
90 A(18,22)=1:A(22.18)=1
92 A(20.21)=1:A(21.20)=1
94 A(21,22)=1:A(22,21)=1
210 B(1)=1 : J=1 : Z=1
300 M=1
400 FOR I = M TO 22
450 IF A(I,J)=1 THEN 500
```

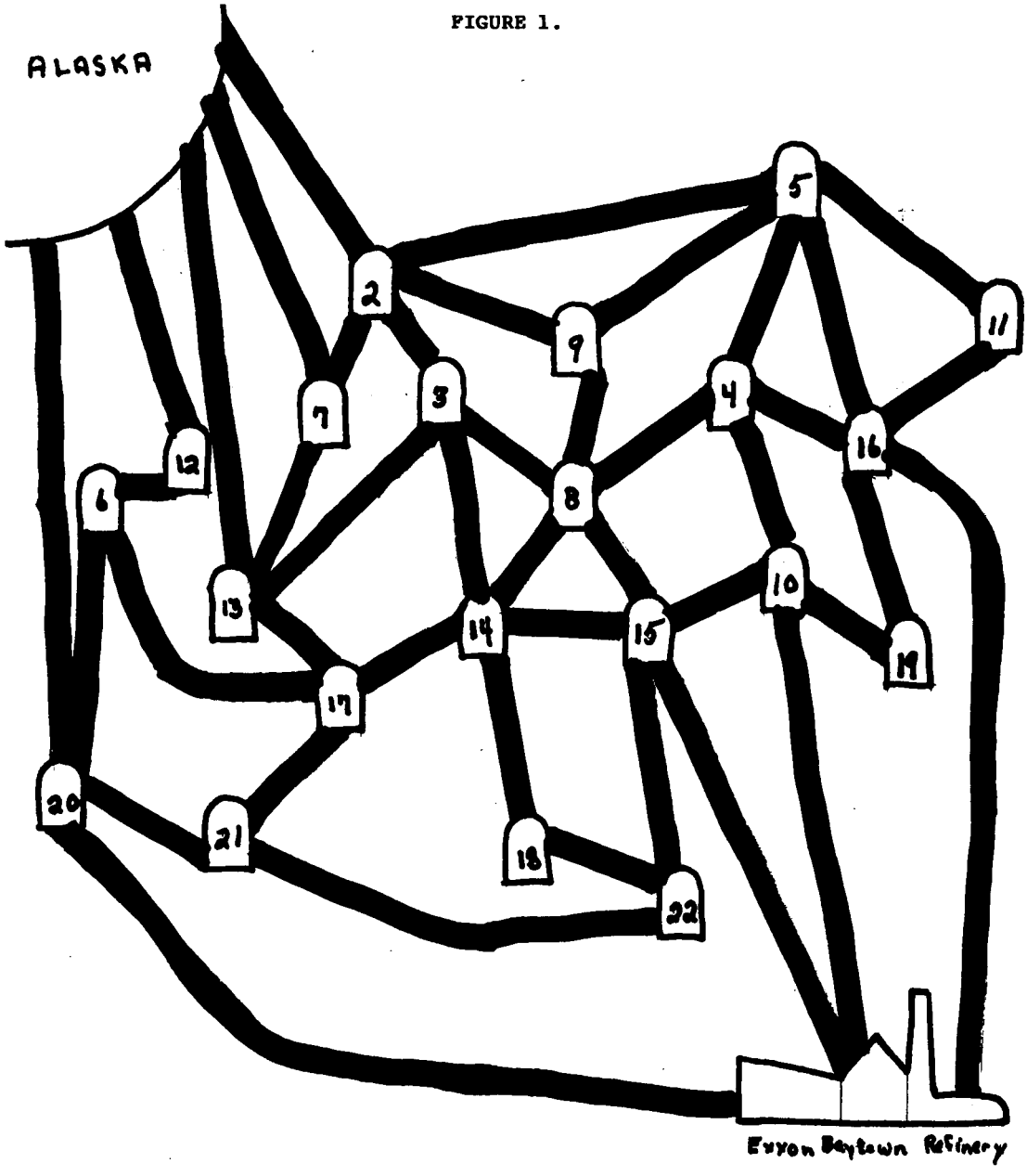
List 1 cont. on page 12

```
460 GOTO 900
500 FOR K = 1 TO Z
550 IF I = B(K) THEN 900
600 NEXT K
700 R = I
800 IF Z = 21 THEN 2000
850 GOTO 1000
900 NEXT I
950 M = B(Z)+1
960 J = B(Z-1)
970 Z = Z-1
980 GOTO 400
1000 Z = Z+1
1100 B(Z) = R
1200 J = R
1300 GOTO 300
2000 Z = Z+1
2100 B(Z) = R
2300 IFB(Z)=10 OR B(Z)=150 OR
     B(Z)=16 OR B(Z)=20THEN
     3000
2601 PRINT"HELP I'M STUCK AT";
     B(Z)
2700 GOTO 950
3000 REM * SOLUTION FOUND *
3003 PRINT:PRINT"SAFE ALONG
     PATH.": FOR I = 1 TO 22:
     PRINT B(I);:NEXTI:PRINT:
     PRINT
3250 GOTO 950
```

LIST #2

```
10 DIM A(625),B(22)
20 A(37)=1:A(301)=1
22 A(53)=1:A(77)=1
24 A(55)=1:A(127)=1
26 A(57)=1:A(177)=1
28 A(59)=1:A(227)=1
30 A(83)=1:A(208)=1
32 A(88)=1:A(328)=1
34 A(89)=1:A(353)=1
36 A(105)=1:A(129)=1
38 A(108)=1:A(204)=1
40 A(110)=1:A(254)=1
42 A(116)=1:A(404)=1
44 A(134)=1:A(230)=1
46 A(136)=1:A(280)=1
48 A(141)=1:A(405)=1
50 A(162)=1:A(306)=1
52 A(170)=1:A(506)=1
54 A(188)=1:A(332)=1
56 A(209)=1:A(233)=1
58 A(214)=1:A(358)=1
60 A(215)=1:A(383)=1
62 A(265)=1:A(385)=1
64 A(269)=1:A(485)=1
66 A(291)=1:A(411)=1
68 A(342)=1:A(438)=1
70 A(365)=1:A(389)=1
72 A(367)=1:A(439)=1
74 A(368)=1:A(464)=1
76 A(419)=1:A(491)=1
78 A(397)=1:A(565)=1
80 A(446)=1:A(542)=1
90 A(472)=1:A(568)=1
92 A(521)=1:A(545)=1
94 A(547)=1:A(571)=1
210 B(1)=1 : J=1 : Z=1
300 M=1
400 FOR I=M TO 22
450 IF A(25*I+J)=1 THEN 500
460 GOTO 900
500 FOR K=1 TO Z
```

FIGURE 1.



```
550 IF I=B(K) THEN 900
600 NEXTK
700 R=I
800 IF Z=21 THEN 2000
850 GOTO 1000
900 NEXT I
950 M=B(Z)+1
960 J=B(Z-1)
970 Z=Z-1
980 GOTO 400
1000 Z=Z+1
1100 B(Z)=R
1200 J=R
1300 GOTO 300
2000 Z=Z+1
2100 B(Z)=R
2300 IF B(Z)=10 OR B(Z)=15 OR
     B(Z) = 16 OR B(Z) = 20
     THEN 3000
2601 PRINT"HELP I'M STUCK AT";
     B(Z):GOTO 950
3000 REM * SOLUTION FOUND *
3003 PRINT:PRINT"SAFE ALONG
     PATH.":FOR I = 1 TO 22:
     PRINT"  ";B(I);
3005 NEXTI:PRINT:PRINT:
     GOTO 950
3400 RETURN
```

TABLE #1.

```
RUN
     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 1   0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
 2   0  0  1  0  1  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 3   0  1  0  0  0  0  0  1  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0
 4   0  0  0  0  1  0  0  1  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0
 5   0  1  0  1  0  0  0  0  1  0  1  0  0  0  0  1  0  0  0  0  0  0  0
 6   0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0
 7   0  1  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
 8   0  0  1  1  0  0  0  0  1  0  0  0  0  1  1  0  0  0  0  0  0  0  0
 9   0  1  0  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
10   0  0  0  1  0  0  0  0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0
11   0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
12   1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
13   0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
14   0  0  1  0  0  0  0  1  0  0  0  0  0  0  1  0  1  1  0  0  0  0  0
15   0  0  0  0  0  0  0  1  0  1  0  0  0  1  0  0  0  0  0  0  0  1  0
16   0  0  0  1  1  0  0  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0
17   0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  1  0  0
18   0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1  0
19   0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0
20   0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
21   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  1  0
22   0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  1  0  0
23   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Ok
```

★ ★ ★

RIGHT-JUSTIFIED TEXT FOR WP6502 ON C1P/C4P

By: Leo Z. Jankowski
Otaio RD1 Timaru
New Zealand

When I bought WP6502 on tape, the tax I had to pay was 99.3% of the purchase price. I did not rush to purchase any enhancements. The government with its rapacious tax saw to that.

But right-justified text became more and more desirable and also the commands contained in #S and #K. My version of WP6502 does not support #S and #K. I decided to have a go at getting the job done in BASIC. It worked!

To run the program in BASIC first save the text from WP6502 in case of accidents. Next, COLD start and answer memory size with a number, e.g. 16384 for 16K RAM etc. This prevents BASIC overwriting the text in RAM. Now enter the Monitor and change 007A from 03 to 34. Next, change contents of 3400 to 00. Warm start and type NEW. Typing PRINT FRE(8) should return 3068 bytes free. Load the program and RUN. Incidentally, the method described is a way of placing in RAM several like-numbered BASIC programs.

The POKEs in line 1 speed up the output to the printer by a

Continued on page 16

TEXT FORMATTER FOR WP6502

```
1 POKE61440,3:POKE61440,16:POKE15,255:CLS:GOTO88
2 N=1:C$="A":A$="PpFfMmDdTtCcSsKk":L=LEN(A$):W$="":
  F=1:T$=CHR$(127)
3 K=2812:HH$=H$:POKE517,1:PRINTTAB(P):IFTTHENGOSUB46
4 GOTO7
5 IFN<LEN(T$)THEN10
6 T$=""
7 N=1:U=0:FORQ=KTOK+180:C$=CHR$(PEEK(Q)):T$=T$+C$:U=U+1
8 IFU>135ANDC$=" "THENK=K+U:GOTO10
9 NEXT:T$=T$+" ":K=K+U
10 Q=FRE(X):IFC$="#"THEN61
11 IFASC(C$)=64THENGOSUB48:GOTO91
12 IFASC(C$)=127THENGOSUB48:GOSUB40:W=Q-P+J:GOSUB54:GOTO31
13 B=0:X=LEN(L$)+LEN(W$):
   IFLEN(W$)>WANDGANDLEN(L$)>WTHENGOSUB28
14 IFX>WTHENGOSUB17:L$="":GOTO29
15 IFX>W-2THENB=1
16 GOTO29
17 M=P:U=1
18 IFMID$(L$,U,1)=" "THENU=U+1:GOTO18
19 R=LEN(L$):IFR=OTHENRETURN
20 IFMID$(L$,R,1)<>" "THENQ=0:GOTO23
21 R=R-1:IFR=OTHENRETURN
22 GOTO20
23 FORX=UTOR:IFMID$(L$,X,1)=" "THENQ=Q+1
24 NEXT:A=W-R:FORY=1TOR:F$=MID$(L$,Y,1):IFY>UANDF$=" "THEN27
25 IFM=PANDG=0 ORF=1THENGOSUB52
26 PRINTF$;:NEXT:GOTO28
27 H=INT(A/Q):X=H+1:FORZ=1TOX:PRINT" ";:NEXT:A=A-H:Q=Q-1:NEXT
28 G=0;GOTO39
29 L$=L$+W$:IFBTHENL$=L$+" ":GOTO31
30 IFLEN(L$)<WTHENL$=L$+" "
31 W$="":IFN>=LEN(T$)THEN5
32 C$=MID$(T$,N,1):N=N+1:IFC$=" "THEN32
33 IFC$=""THENC$=" ":GOTO5
34 IFC$="#"ORASC(C$)=127ORASC(C$)=64THEN5
35 W$=W$+C$:C$=MID$(T$,N,1):IFC$=" "THEN5
36 N=N+1:IFC$="#"ORASC(C$)=127ORASC(C$)=64THEN5
37 GOTO35
38 G=0:GOSUB48:L$=" "
39 PRINT:F=F+1:W=Q-P+J
40 IFS=0 ORF=1THEN42
41 FORZ=1TOS:PRINT:F=F+1:NEXT
42 IFF<=PLTHENRETURN
43 IFLEFT$(H$,1)="N"THEN45
```

Continued

```
44 POKE517,0:PRINT:INPUT"*Hold next page ";
   H$:PRINT:POKE517,1:GOTO46
45 PRINTCHR$(12)
46 F=1:IFTTHENPRINTTAB(J+INT(O-3)/2)T:T=T+1:PRINT:F=F+2
47 RETURN
48 IFLEN(L$)+LEN(W$)>WTHENGOSUB17:L$=""
49 GOSUB52:PRINTL$;:L$=""
50 IFTITHENPRINTW$;:TI=0:RETURN
51 PRINTW$:F=F+1:RETURN
52 IFITHEN55
53 IFE=27THENE=0:RETURN
54 M=0:PRINTTAB(P):RETURN
55 IFD=0THENPRINTTAB(I):GOTO59
56 IFV>I-PTHENV=0:X=I-V:W=W-P:GOTO58
57 X=I-V-P:IFV=I-PTHEN59
58 FORM=1TOX:PRINT".";:NEXT
59 M=0:I=0:D=0:RETURN
60 GOSUB48:GOSUB45:RETURN
61 C$=MID$(T$,N,1):N=N+1:FORZ=1TOL:E$=MID$(A$,Z,1):
   IFC$<>E$THENNEXT
62 ONINT((Z+1)/2)GOSUB38,60,63,65,66,69,77,78,84:GOTO31
63 GOSUB48:GOSUB40:P=J+VAL(MID$(T$,N,1)):N=N+1:W=O-P+J:G=0
64 RETURN
65 D=-1
66 GOSUB86:I=10*VAL(MID$(T$,N,1)):N=N+1:IFI<V+PTHENGOSUB39
67 W=O+J-I:G=0
68 RETURN
69 IFGTHENE=27
70 IFLEN(W$)>=WTHENGOSUB39:GOSUB54
71 GOSUB86:CH=10*VAL(MID$(T$,N,1))+VAL(MID$(T$,N+1,1)):W=W-V
72 IFW=0THENGOSUB39:GOSUB54
73 Q=CH:PRINTCHR$(Q);:IFQ=44THENPRINT" ";:W=W-1
74 IFQ<>27THENG=-1
75 IFQ=58ORQ=32ORQ=44ORQ=35ORQ=64ORQ=93ORQ=94THENW=W-1
76 E=Q:N=N+2:RETURN
77 GOSUB48:S=VAL(MID$(T$,N,1)):N=N+1:RETURN
78 GOSUB86:POKE517,0:PRINT:
   INPUTE$:C$=RIGHT$(T$,LEN(T$)-N+1):PRINT
79 IFMID$(E$,LEN(E$),1)<>" "THENE$=E$+" "
80 IFC$<>" "THENT$=E$+C$:GOTO82
81 T$=E$
82 W=W-V:G=-1:N=1:IFMID$(T$,1,1)="#"THENE=27
83 POKE517,1:RETURN
84 GOSUB48:GOSUB54:FORZ=1TOO-P+J:PRINTC$;:NEXT:PRINT:F=F+1
85 GOSUB39:GOSUB54:RETURN
86 TI=-1:Z=LEN(L$)+LEN(W$):IFZ>WTHANZ=LEN(W$)
87 V=Z:GOSUB48:RETURN
88 INPUT" Copies ";C:INPUT" Line Spacing ";
   S:PRINT:PRINT"PAGE:":PRINT
89 INPUT" Margin ";P:INPUT" Width ";W:
   INPUT" Length ";PL:J=P:O=W
90 INPUT" Number ";T:INPUT" Hold ";H$:
   PRINT:S=S-1:TT=T:SS=S:GOTO2
91 PRINT:C=C-1:S=SS:T=TT:P=J:W=O:H$=HH$
92 IFCTHENPRINTCHR$(12):GOTO2
93 PRINTCHR$(12):POKE517,0:END
```

COMMENTARY ON WP6502

Entry points:

```
ZAP   086D.   READ   0480.
S/Ed  09CF.   WRITE  087D.
MOVE  0979.   VIEW   0651.
G/Ed  050E.   OS     0AE5.
TYPE  0386.   BLOCK  0AC6.
DEL   0943.
```

USEFUL ADDRESSES

```
0222 4C F202. JUMP TO $02F2.
0225    00D7.
0227 4C EEFF. CHARACTER OUT.
022A 4C EDFE. GET A KEY (FD00).
022D E10A     START OF TEXT POINTER, IE. 0AE1.
022F 02       LINE FEEDS BEFORE START OF PRINTOUT.
0230 03       DEVICE.
0231 40       END OF FILE CHARACTER.
0232 1A       LOWEST CHARACTER ALLOWED.
0233 80       HIGHEST CHARACTER ALLOWED.
0234 01       'VIEW' LINEFEEDS.
```

Continued 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

Continued from page 14

```
0235 18       LINES SCROLLED IN 'VIEW' SCREEN.
0236 00       'VIEW' SCREEN INDENT.
0237 3C
0238 01       PAGE NUMBERING.
0239 01       NUMBER OF COPIES.
023A 00       NEW PARAGRAPH/NEW SCREEN CONTROL.
023B 2D       MESSAGES BEGIN...AND END AT 026A
026B 01       LINEFEEDS TO PRINTER.
026C 42       PAGE LENGTH.
026D 0A       PRINTER MARGIN.
026E 3C       PAGE WIDTH.
026F 0D       CARRIAGE RETURN.
0270 0A       LINE FEED.
0271 7F       CHR$(127) FOR LINEFEEDS.
0272 1D 5D    TEXT MARKERS.
0274 23       CHR$(23).
0275 C0       'DELETE' AND MEMORY FILL CHARACTER.
0276 5D       'INSERT' CHARACTER.
0277 5E       'SHIFT-N' CHARACTER FOR 'MOVE'/'DELETE'.
0278 1B       'ESC'.
0279 3C 3E    'GLOBAL' EDIT MARKERS.
027B 5F       SHIFT-O, BACKSPACE.
027C          MORE MESSAGES.
02A4          PROGRAM BEGINS HERE AND ENDS AT 0AE1.
0341 14       'VIEW AND SCREEN/ED' SCREEN WIDTH.
0436          FILL MEMORY....BYTES FREE CHECK.

For a more readable MENU try:  .023B/2D 54  79
70  E5  0A  56  69  65  F7  0A  47  2F  45  E4
0A  53  2F  45  E4  0A  44  65  EC  0A  4D  6F
F6  0A  42  6C  EB  0A  5A  61  F0  0A  4F  D3
0A  52  2E  54  E1  0A  D7  0A
and the next byte should be A0 at 026A.
```

factor of 16. A$ in line 2 lists the commands supported. Variable K in line 3 contains the first RAM address from which text will be read. To renumber the program in tens merely add 0 onto the end of each line number. Contact the author if you require more information on the program.

★ ★

A BEXEC* in 1 TRACK

Courtesy of King County O.S.I. Users Group

```
10 DEFFNA(X)=10*INT(X/16)+X-16*INT(X/16)
12 DEFFNB(X)=16*INT(X/10)+X-10*INT(X/10)
14 POKE8993;2:POKE8994.2:POKE741,76:POKE750,78:F$="FILENAME"
16 POKE2073,96:POKE2893,28:POKE2894,11:E$="EXISTING NAME:
18 T=39:DIMN$(T),F(T),Q(T),U(T):C=0:K=0:P=11897
20 DISK?"CA 2E799=12,1":GOSUB96:DISK!"CA 2E79=12,2":GOSUB96
22 PRINT:PRINTK"Files;"C"Tracks":PRINT
24 PRINT"List/Print/Free/Rename/Delete/Create/Exit/X-ON"
26 DISK!"GO 252B":R$=CHR$(PEEK(9059))
28 V=2:IFR$="X"ANDV=2THENV=1:GOTO58
30 IFR$="L"THEN50
32 IFR$="P"THEN V=1:GOTO50
34 IFR$="F"THEN58
36 IFR$="R"ORR$="D"THEN68
38 IFR$="C"THEN82
40 IFR$<>"E"THEN22
42 GOSUB108:DISK!"SA 12,1=2E79/1":GOSUB108:DISK!"SA 12,2=E791"
44 POKE2073,173:END
46 FORA=1TO30:PRINT#V,"-";:NEXTA:PRINT#V:RETURN
48 PRINT#V,N$(B)TAB(12)F(B)TAB(16)"-"Q(B)
   TAB(24)((F(B)-Q(B))+1):RETURN
50 PRINT#V:PRINT#V,"OS-65D V3.X DIRECTORY"
52 PRINT#V,F$"   TRACK RANGE SIZE":GOSUB46
54 FORA=0TOT:FORB=1TOK:IFA=F(B)THENGOSUB48
56 NEXTB:NEXTA:GOTO22
58 PRINT#V:PRINT#V,TAB(10)"--FREE TRACKS--":GOSUB46:J=0
60 FORI-JTOT:IFU(I)=0THEN64
62 NEXTI:GOTO22
64 PRINT#V,TAB(3)I;TAB(7)"-";:FORJ=ITOT:IFU(J)=0THENNEXTJ
66 PRINT#V,J-1TAB(19)"#OF "(J-I):I=J;GOTO62
```

Continued

Continued from page 16

```
68 GOSUB118:FORI=1TOK:IFN$(I)=O$THEN72
70 NEXTI:PRINTF$O$"NOT FOUND":GOTO22
72 IFR$="R"THEN78
74 FORJ+F(I)TOQ(I):U(J)=0:C=C-1:NEXTJ:N$(I)=N$(K)
76 F(I)=F(K):Q(K):K=K-1:GOTO22
78 GOSUB120:N$=0$:FORJ+1TOK:IFI<>JANDN$(J)=N$THENPRINTE$E:
   GOTO22
80 NEXTJ:N$(I)=N$:GOTO22
82 IFK=TTHENPRINT"NO SPACE":GOTO22
84 GOSUB120:N$=O$:FORI=1TOK:IFN$(I)=N$THENPRINTE$:GOTO22
86 IF (S>Q)OR(Q>T)THENPRINT"ERROR":GOTO22
90 FORI=STOQ:IFU(I)=1THENPRINT"TRACKS USED":GOTO22
92 NEXTI:FORI=STOQ:U(I)=1:C=C+1:NEXTI:K=K+1
94 N$(K)=N$:F(K)=S:Q(K)=Q:GOTO22
96 FORI=PTOP+248STEP8:IFPEEK(I)=35THEN106
98 IFK=TTHENPRINT">"T"FILES":RETURN
100 K=K+1:N$(K)="":FORJ=ITOI+5:N$(K)=N$(K)+CHR$(PEEK(J)):NEXTJ
102 F(K)=FNA(PEEK(I+6)):Q(K)=FNA(PEEK(I+7))
104 FORJ=F(K)TOQ(K):U(J)=1:C=C+1:NEXTJ
106 NEXTI:RETURN
108 FORI=PTOP+248STEP8:IFK=OTHEN114
110 L=0:FORJ=ITOI+5:L=L+1:POKEJ,ASC(MID$(N$(K),L,1)):NEXTJ
112 POKEI+6,FNB(F(K)):POKEI+7,FNB(Q(K)):K=K-1:GOTO116
114 FORJ=ITOI+5:POKEJ,35:NEXTJ:POKEI+6,0:POKEI+7,0
116 NEXTI:RETURN
118 PRINT"OLD ";:GOTO122
120 PRINT"NEW";
122 PRINTF$:INPUTO$:I$=O$+"       ":O$=LEFT$(I$,6):RETURN
130 L$="12345678901234567890"
134 O$="12345678901234567890"
136 K$="12345678901234567890"
138 P$="123456789012345"
```

LINES 130 TO 138 ARE EXTRA AND ARE UNDER 2 K TRACKS
YOU MAY ADD OWN CODE TO FILL ANY REQUIREMENTS FOR SPECIAL BEXEC NEEDS, LIKE POKE'S. OR DISPLAYS.

★ ★

THE ULTIMATE USR FUNCTION

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

This month I will show you some ways you can put together some features of the C1P Basic interpreter covered in previous articles to make what I call the ultimate USR function. Within certain limitations, this USR function allows you to add any number of machine language functions to a Basic program, calling each of them by name, passing as many parameters (values, strings, or even arrays) to and from the functions, and getting around the usual problems of allocating space and protecting it from Basic. You can actually incorporate the machine code right into your Basic program, without the space- and time- consuming DATA statements usually associated with machine language extentions to Basic. This article, like its predecessors, will assume that you are familiar with machine language programming for at least simple programs.

When Basic attempts to execute the USR function, the interpreter executes a machine language jump (JMP) to lo-

cation $000A. This location and the following two locations normally contain a JMP to some other location - the actual location of the machine language implementing the desired function. During a cold start, Basic sets up this JMP to point to the FC ERROR (function call error) routine, so that if you call USR without first initializing it, you will get the appropriate error message. HEXDOS reinitializes this pointer to its own set of routines implementing its various features accessed through USR.

I will be referring to memory locations which change if you are using HEXDOS rather than straight ROM Basic. If you are using Pico-DOS, the locations should all remain as for ROM Basic. Whatever variation of ROM Basic you are using, the beginning of the program space is pointed to by $0079 (low byte) and $007A (high byte). Under HEXDOS, these pointers contain $0B01. Also, I will be using the HEXDOS pointer at $00F0-$00F1 for its USR(-7) function. ROM Basic users may use the normal pointer at $000B-$000C for USR(.) . The decimal point inside the parenthesis is not a typo - it substitutes for zero and runs faster.

The first problem we usually encounter with a machine language program is just where in RAM to put it. Two common solutions are to use the unused area in page 2 or to use an area at the top of RAM and change Basic's end-of-memory pointer to reserve the area against the interpreter. These both cause severe incompatibilities with various systems:

1. Since the original OSI monitor ROM had some serious deficiencies, many have replaced it with ROMs offered by other vendors, most of which use a portion of the page 2 free space.

2. Adding memory to the system or using several of these machine language routines together causes them to move around in memory.

To avoid these and other pitfalls, we can include the machine language code directly within the program and execute it there. This example will use a very short segment of code which will implement a sort of "PRINT AT" function, in the format

```
USR(-7) X,Y
```

where X is the screen location, varying from 0 (upper left) through 1023 (lower right), and Y is the ASCII value of the character to be placed there. It will check for out-of-range values and return an FC ERROR if appropriate, and by calling Basic routines, it will accept any Basic arithmetic expression for X and Y. That is, it could be used in ways like

```
USR(-7) 32*R+C,128+V
```

Though the function may be useful in its own right, I will concentrate on using it to illustrate the concepts involved in implementing it within the Basic program.

First, type in a line 0 consisting of REM and 19 spaces (the routine to be loaded is 19 bytes long). Then jump to the monitor via USR(-6) (under HEXDOS) or a reset. Now type in the following code starting at $0B06 ($0306 for non-HEXDOS systems).

```
20 FC B3 A5 12 C9 04 90 03 4C
   88 AE 69 D0 85 12 4C 2C B4
```

This is the complete machine language routine needed to implement this function, thanks to the routines in ROM. Also place the starting address ($0B06) into the USR pointer ($00F0) ($0306 into $000B for non-HEXDOS systems). Then return to Basic via a reset and warm start (or GO at $0000). You may now experiment with the function as described above.

If you LIST the program at this time, you will see nonsense for line 0 but any lines you later add will list and execute normally. Using 0 for the line number ensures that you will not insert a line ahead of the machine language and move it in memory. Note that there are no zeroes in the machine language. If there were, Basic would interpret the first 0 as the end of the REMark and line, causing serious problems during listings and later Basic editing. If the machine language you wish to include contains a zero, you can almost always get around it by coding it somewhat differently. For instance, to load X with an immediate value of 0, try laoding it with 1 and decrementing it. To JSR to routines in page zero (notably $00BC) try to find a place in ROM which includes such a jump ($A617 for the aforementioned routine). True, such tricks

may add one or two bytes to the machine language, but this method saves many bytes which would be used by the normal methods for mixing machine language with Basic.

Now for the routine which makes all the magic I promised above possible. Start over with NEW, type in a new line 0 with a REMark of 21 spaces, and enter this program as above:

```
20 C1 AA 20 B2 AA A0 01 B1 AE
   85 A2 C8 B1 AE 85 A3 68 68
   4C A1
```

For those who are figuring this out as you go, it will appear that a byte was dropped from the end of the list. Trust me for now.

This routine will accept the contents of a Basic string as machine language and execute it in place (wherever it lies in memory). The calling format will be

```
USR(-7) A$ <parameters>
```

Notice that this allows you to name the string with a name related to the function to be carried out by that code and then call it by name. You may use any valid Basic variable name for the name of the string, but remember that Basic only considers the first two letters of the name in determining uniqueness. Thus, if your program uses a string CL$, you could not name a machine language string CLS$ without creating a conflict. You also may not use names with imbedded reserved words, such as CLEAR or STORE (imbedded TO).

You can use any method Basic allows to build the string containing the machine code. For instance, you could put the code in DATA statements and use CHR$ to concatenate characters onto the string, but this would be even worse than the normal ways of setting up machine language, except that memory allocation is automated. A better way is to assign them as literal strings within the program. For disk users, such strings can be saved and reloaded with no special considerations. For tape-based systems or ASCII disk files you will not in general be able to save the strings with your programs and load them back, since the Basic interpreter only accepts the 96 ASCII characters (rather than 255) when loading those types of files.

To enter the string in a Basic program, first type a line such as

```
CL$="                "
```

with one space in the string for each byte of code. Then jump to the monitor and find the beginning of the actual string as it is stored in memory. It will be most obvious as a series of repeated $20's (ASCII blanks). Just prior to the string should be the Basic statement assigning it to the string variable. The variable name will be stored as ASCII characters, including the dollar sign. The equals sign is stored in its tokenized form, as a $AB. The opening quote for the string is stored as $22. Another way to find the line is by searching for a particular line number just as Basic does. Start at the address pointed to by the contents of $0079. This will be the beginning of the first line of the program. Each line starts with a two-byte pointer to the succeeding line, followed by a two-byte line number. If the first line is not the desired line, look where its pointer indicates for the beginning of the next line, and on. When you find the line containing the string, use the monitor to enter the machine language into it. While entering your machine language program, be sure to keep in mind the restriction I mentioned above about not having any zero bytes in it.

An added bonus of having the machine language functions in strings is that you can edit and splice them using the usual Basic string functions. For instance, to make the above "print at" function check for a comma between the dollar sign and the first number, if it were set up in a string called AT$ we could write

```
BT$=CHR$(32)+CHR$(1)+
    CHR$(172)+AT$
```

and we would then have a new function which would be called like

```
T$ = USR(-7) BT$,X,Y
```

By now, I hope you're itching to try out some of this. Here are the addresses of some routines in ROM which are useful in setting up these functions. When I refer to the "interpretation stream" or just "stream" I mean the characters coming after the right parenthesis of the USR call. Unless I say otherwise, each of these routines moves the interpretation pointer to the first character which was not accepted by it. Also, anything you type which is not contained in quotes is fair game for Basic's tokenizer, so you may find that some things you type will be converted to Basic tokens.

$00BC GETCHAR
Returns the next character in the input stream and advances the pointer. The Z flag set if the character was a null or a colon; the C flag is set if the character was not an ASCII digit.

$00C2 REGETCHAR
Same as GETCHAR, except that it gets the last character returned by GETCHAR and does not advance the pointer.

$AC03 CHECKCHAR
Checks to see that the next character matches the character in the A register. Trips a SN ERROR if the character doesn't match; discards the character and returns the next character if they do.

$ABFB CKLPAREN
Uses CHECKCHAR to check for a left parenthesis.

$ABFF CKRPAREN
Ditto for right paren.

$AC01 CKCOMMA
Ditto for comma.

$ABF5 SINGPARAM
Accepts a single parameter from the stream in the form of a Basic arithmetic or string expression enclosed in parenthesis. Returned values are as in EXPRESSION below.

$AAC1 EXPRESSION
Evaluates a Basic arithmetic or string expression. The expression may use any features of Basic; i.e., arithmetic operators string concatenation, functions, etc. If the expression returns a numerical value, $5F will be set to zero and the value will be returned in the floating point accumulator at $00AC-$00B0. The Z flag will indicate if the result was zero. If the expression evalutes to a string, a pointer to the string descriptor will be returned at $AE-$AF. The location pointed to by that pointer will contain the length of the string; the succeeding two bytes will contain a pointer to the beginning of the string.

$AAB0 CKNUMERIC
Checks the most recently returned value for a numeric indicator in $5F without dis-

turbing any processor registers.

$AAB2 CKSTRING
Ditto for strings.

$AD0B FINDVAR
Accepts a variable from the interpretation stream and returns the address of its value field. Sets $5F to indicate string or numeric variable as above. If the variable is an array element (such as A(3)), this routine will return the address of that element. The address is returned in Y and A (high byte in Y) and also in $95-$96.

$AE05 INTEGER
Converts a floating point number in the floating point accumulator to a two-byte integer at $AE-$AF, with the high byte in $AE. Accepts values in the range 0-65535.

$AFC1 FLOATTWO
Converts a two-byte integer in A and Y (high byte in A) to a floating point number in the floating point accumulator.

$AFD0 FLOATONE
Converts a one-byte integer in Y to floating point as above.

$B774 STORENUM
Stores a numeric value from the floating point accumulator to the variable whose value field is indicated by $95-$96.

$A7D5 STORESTR
Stores a string returned by EXPRESSION to the string variable whose value field is indicated by $95-$96.

$B831 INTEGER2
Same as integer except that the number may be of any value and only the low-order 16 bits will be returned.

$B3AE GETINT
Accepts an expression for the interpretation stream as in EXPRESSION and converts it to a one byte value in $AF and the X register. Values range from 0 - 255.

$B3FC GET2VALS
Accepts two expressions separated by a comma as used by POKE. Returns the first as a two-byte integer at $11-$12 and the second as a one byte integer in $AF and the X register.

There are many other routines in the Basic ROMs which might be of use for machine language programming, but these are the ones which are of the most use in passing parameters back and forth between Basic and machine language routines. You can make up a set of strings containing the calls to these routines and then concatenate them to make one routine which accepts exactly the sequence of parameters you desire from the interpretation stream and returns results to your Basic program. Who says a USR function can have only one parameter, anyway?

One last note before closing for this month. Since I have added RAM beyond 32K to my system, I ran across one very frustrating bug in one of my programs. It seems that the VAL function does not work correctly on strings stored above $8000. I haven't had the time to trace this and figure out just what the problem is or if there is a quick fix. At the present rate, I'll get around to that about 1998.

LETTERS

ED:

I have an OSI C2-4PMF with monitor. I purchased a word processor WP-6502 (DOW QUONG FOK LOK SOW) that when used with my C.Itoh Printer works fine. No poblems. When I first purchased the WP-6502, I only had a converted Selectric 72. In order to make the word processor work, it was necessary to enter the "INSTALL" mode and make certain changes. First, in the printer code, it was necessary to use 00 code and in the "Line Feed" also a 00 because of automatic carriage return. However, after making these changes, it was not possible to properly view the text as it was truncated.

I have found a way around this problem. With the WP6502 there is a built in protection against an accidental "BREAK" so that if the Break Key is accidentally hit or depressed all you have to do to keep the text is to reboot (D) then on Menu hit (V) and the matter will be viewable. This makes it possible to first write the material using the disk for one of the other printers (one that will allow viewing without problems) and then print the text on the unusual printer using a disk in which the changes have been made by use of the install mode.

Leonard F. Watkins, Jr.
Wichita, KS 67203

* * * * *

ED:

ADAPTER 6809 - 6800

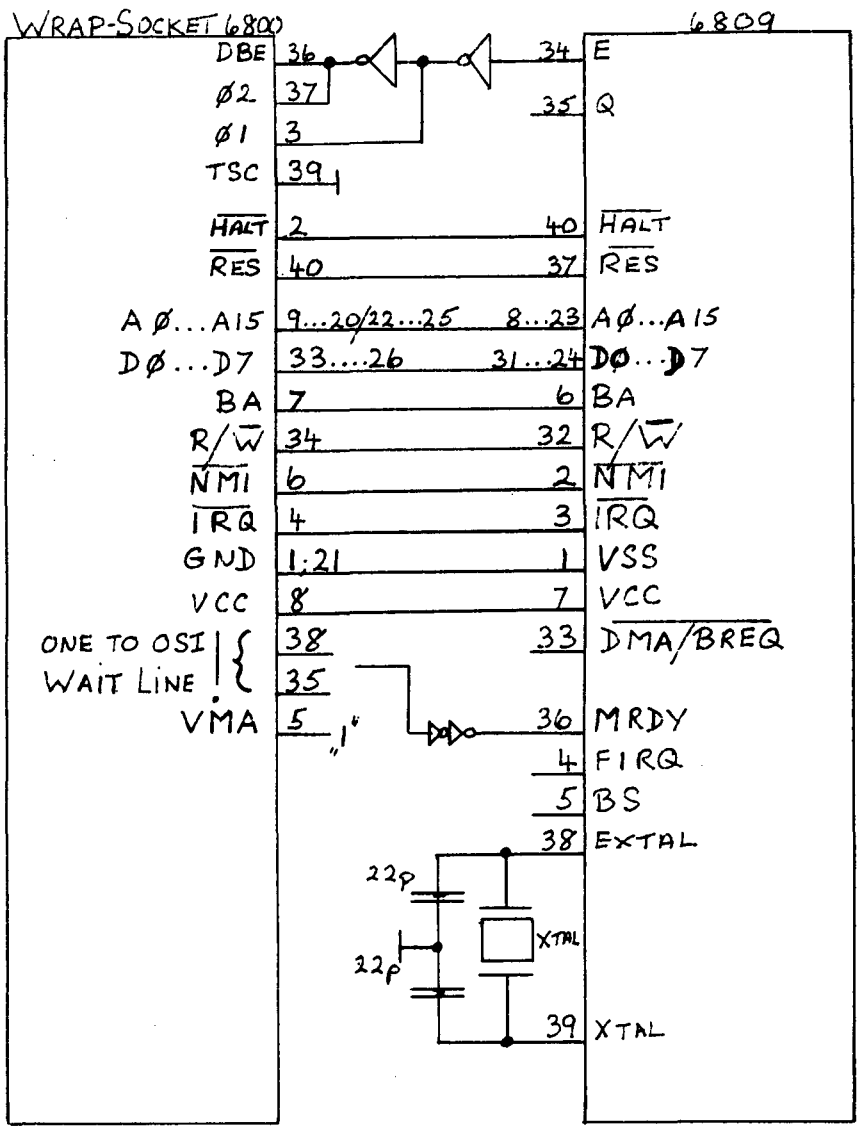
My letter covers two items.

Firstly, a remark to the letter of Jeff Easton in PEEK(65), Vol. 4 No. 2. I am planning to run a 6809 on my OSI, too. But I do not believe designing a new PC board for this purpose would be the best solution. Why not use the benefits OS65D offers (DOS with disk I/O, I/O distributor, etc.)? The simplest way to achieve this is to get a 510 board (SPACE COM offers it, PEEK(65), Vol. 4 No. 2 page 21). This three processor board contains a 6502, a Z80, and last but not least, a 6800. You can replace this CPU by a small piggypack board containing a 6809, a crystal, and 1 TTL chip. The only drawback would be that not all features of the 6809 could be realized. Using a 510 board this way, you can run 65D, load a 6809 program treated as a hexa- decimal file, and then switch to the 6809 by some keystrokes (CHALLENGER III utilities diskette). A different way would be to select the 6809 before reset via an external switch. On the 510 Rev A board (caution, bad layout!) you can replace the 6800 bootstrap 1702A by a self designed 6809 bootstrap 1702A (256 bytes!) or replace it by a second piggypack with a better prom type. On the 510 Rev C board, you can replace the SYNMON monitor prom with a modified one. The plus is you can use existing software, the minus is that you must have serial I/O (video terminal, or a second computer). By the way, modifying my C2-8P DF from a video system to a serial system is my present activity.

Further, I enclose a patch to the Extended Monitor's Dump command. First, it appends an ASCII line to each line (except the last one). 'ASCII line' means that each byte's ASCII representation is shown, if possible. If not, a period is printed out. This feature is very useful in investigating unknown software, e.g. the XMON itself. The second modification is something like 'pagination' for my printer. You can find enough place for patches between ASM (end at $1583) and XMON (start at $1700). However, I started my patch at $1F60, immediately behind the XMON.

Uwe Pitz
Wolfsburg, West Germany

* * * * *



REMARK:This circuit has been adopted from EDN, but is not yet tested!

```
:01F60

                                    @ 1C7A:  JMP 1F60

PTCH 1  1F60 CA       DEX
        1F61 F003     BEQ $1F66     OSI DUMP READY?
        1F63 4C6F1C   JMP $1C6F     NO
        1F66 20AF1B   JSR $1BAF     OUTPUT 2 SPACES
        1F69 20AF1B   JSR $1BAF
        1F6C A210     LDX #$10      PARSE THE CURRENT LINE
        1F6E A5CC     LDA $CC       ONCE MORE
        1F70 38       SEC
        1F71 E910     SBC #$10
        1F73 85CC     STA $CC       DUMP ADDR LO
        1F75 B004     BCS $1F7B
        1F77 C6CD     DEC $CD       DUMP ADDR HI
        1F79 EA       NOP
        1F7A EA       NOP
        1F7B B1CC     LDA ($CC),Y
        1F7D 20891F   JSR $1F89     DUMP READY?
        1F80 20851A   JSR $1A85     NO
        1F83 CA       DEX
        1F84 D0F5     BNE $1F7B     LINE READY?
        1F86 4C5D1C   JMP $1C5D     YES
```

```
          1F89  297F    AND  #$7F      MASK ASCII
          1F8B  C97F    CMP  #$7F      RUBOUT?
          1F8D  F004    BEQ  $1F93
          1F8F  C920    CMP  #$20      SPACE?
          1F91  B002    BCS  $1F95
          1F93  A92E    LDA  #$2E      PERIOD
          1F95  4C4323  JMP  $2343
          1F98  00      BRK
          1F99  00      BRK
          1F9A  00      BRK
          1F9B  00      BRK
          1F9C  00      BRK
          1F9D  00      BRK
          1F9E  00      BRK                 @1C3D:  JMP 1FA0
          1F9F  00      BRK
PTCH 2    1FA0  A947    LDA  #$47      $47 LINES PER PAGE
          1FA2  EA      NOP
          1FA3  EA      NOP
          1FA4  EA      NOP
          1FA5  85FE    STA  $FE
          1FA7  85FF    STA  $FF
          1FA9  206B1A  JSR  $1A6B
          1FAC  4C401C  JMP  $1C40           @1C5D:  JMP 1FB0
          1FAF  00      BRK
          1FB0  C6FF    DEC  $FF       DECREMENT LINE COUNTER
          1FB2  F006    BEQ  $1FBA     PAGE READY?
          1FB4  20561A  JSR  $1A56     NO. CRLF
          1FB7  4C601C  JMP  $1C60
          1FBA  A91A    LDA  #$1A      PAGE READY: NMBR OF CRLF
          1FBC  85FF    STA  $FF
          1FBE  20561A  JSR  $1A56
          1FC1  C6FF    DEC  $FF
          1FC3  10F9    BPL  $1FBE
          1FC5  EA      NOP
          1FC6  EA      NOP
          1FC7  EA      NOP
          1FC8  A5FE    LDA  $FE       LOAD LINE COUNTER
          1FCA  85FF    STA  $FF
          1FCC  4C431C  JMP  $1C43     START  AGAIN  WITH  HEADER
          1FCF  00      BRK
          1FD0  00      BRK
          1FD1  00      BRK
          1FD2  00      BRK
          1FD3  00      BRK
          1FD4  00      BRK
          1FD5  00      BRK
          1FD6  00      BRK
          1FD7  00      BRK
          1FD8  00      BRK
          1FD9  00      BR
```

SAMPLE RUN

```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
3000  A9 2E 85 E2 A9 1E 85 E1 20 73 20 0D 0A 0A 2D 20  ).b)..a s-...-
3010  44 49 53 4B 45 54 54 45 20 55 54 49 4C 49 54 49  DISKETTE UTILITI
3020  45 53 20 2D 0D 0A 0A 53 45 4C 45 43 54 20 4F 4E  ES -..SELECT ON
3030  45 3A 0D 0A 31 29 20 43 4F 50 49 45 52 0D 0A 32  E:..1) COPIER..2
3040  29 20 54 52 41 43 4B 20 30 20 52 45 41 44 2F 57  ) TRACK 0 READ/W
3050  52 49 54 45 0D 0A 3F 20 00 20 98 2C A9 00 8D E5  RITE..? . .,)...e
3060  2C 20 E4 2C C9 31 F0 0A C9 32 D0 03 4C 00 05 4C  , d,I1p.I2P.L..L
3070  51 2A 20 73 2D 0D 0A 0A 2D 20 44 49 53 4B 45 54  Q* s-...- DISKET
3080  54 45 20 43 4F 50 49 45 52 20 2D 0D 0A 0A 46 52  TE COPIER -...FR
3090  4F 4D 20 44 52 49 56 45 20 28 41 2F 42 2F 43 2F  OM DRIVE (A/B/C/
30A0  44 29 3F 20 00 20 6A 04 85 00 20 73 20 0D 0A 54  D)? . j... s-..T
3080  4F 20 44 52 49 56 45 20 28 41 2F 42 2F 43 2F 44  O DRIVE (A/B/C/D
30C0  29 3F 20 00 20 6A 04 85 01 20 73 20 0D 0A 0A 53  )? . j... s-...S
30D0  54 41 52 54 49 4E 47 20 54 52 41 43 4B 3F 20 00  TARTING TRACK? .
30E0  20 A4 04 85 02 20 73 20 0D 0A 45 4E 44 49 4E 47   $... s-..ENDING
30F0  20 54 52 41 43 4B 20 28 49 4E 43 4C 55 53 49 56   TRACK (INCLUSIV
3100  45 29 3F 20 00 20 A4 04 85 03 C5 02 10 20 20 73  E)? . $...E.. s
3110  2D 0D 0A 2A 2A 20 53 54 41 52 54 20 3E 20 45 4E  -..** START * EN
3120  44 20 54 52 41 43 4B 20 2A 2A 00 4C C9 02 20 73  D TRACK **.LI. s
3130  2D 0D 0A 52 45 41 44 59 20 28 59 2F 4E 29 3F 20  -..READY (Y/N)?
3140  00 20 98 2C A9 00 8D E5 2C 20 E4 2C C9 59 F0 03  . .,)...e, d,IYp.
3150  4C 72 02 A5 01 20 C6 29 20 63 26 A5 02 20 BC 26  Lr.%. F) c&%. u&
3160  A5 00 20 C6 29 20 63 26 A5 02 20 BC 26 A5 02 D0  %. F) c&%. u&%.P
3170  30 48 20 6A 20 68 20 92 20 E6 02 A9 06 85 FF A9  0H j-h .-f.)...)
3180  AF 85 FE 20 6E 06 A5 07 8D 5F 26 20 8A 26 A5 01  /.@ n.%.. & .&%.
3190  20 C6 29 A9 06 85 FF A9 AF 85 FE 20 1F 06 20 83   F))...)/-@ .. .
31A0  2C A5 03 85 E5 C5 02 10 39 20 61 27 20 63 26 20  ,%..eE..9 a' c&
31B0  73 20 0D 0A 0A 41 4E 4F 54 48 45 52 20 28 59 2F  s-...ANOTHER (Y/
31C0  4E 29 3F 20 00 20 9B 2C A9 00 8D E5 2C 20 E4 2C  N)? . .,)...e, d,
3100  C9 59 D0 03 4C 72 02 A5 00 20 C6 29 20 63 26 4C  IYP.Lr.%. F) c&L
31E0  51 2A A5 00 20 C6 29 20 6A 20 A5 02 85 EE 20 92  Q*%. F) j-%..n .
31F0  20 20 54 27 A9 01 8D 5E 26 A9 FF 85 08 20 C4 28
:
```

ED:

A while back I purchased a Hayes Chronograph with the intention of date/timing my DMS reports. I could just as easily have purchased another OSI board (clock board) but that would have cost much more and I did not wish to sacrifice another slot in the motherboard particularly since there is a problem with the 8th (front most) slot.

The other day, I decided it would be nice, for business reasons, to be able to determine run-time when I do word-processing (using WP6502 by DQFLS, v1.3a), and I devised a program that does this for me. Of course, it has its limitations, but I can live with these.

Remember, a while back I asked you why it was that I could not select the correct port to run an input from an external device under OSU 1.42? Well, after much testing I determined that as long as I don't ask the clock for more than one input, it WILL work. In the WPEXEC program, I added only one gosub to the input#8 line (clock works under RS232C) and that works! If I were to insert another gosub to the input line (let's say for the date) then the whole thing hangs. Most curious, particularly since I do not have that problem under OSI v1.2!! I don't own OSU v1.42, but the DQFLS program uses it and came with it, but it's locked up. I did purchase a manual, an EXCELLENT manual from OSI just recently, for OSU v.1.43, in the hope that I might learn something from it that could be adaptable to V1.2.

Since I am not at all convinced that OSU (the newest versions) are so easy to interface with DMS, particularly because of the filesize boundaries that need be adhered to, I am sticking to v 1.2. It works, and it works well now. So why upset the applecart?

Now I have a really interesting question -- how can I program (remember, I know next to nothing about assembly language) these run-time things under CP/M?? I know how to change the PRINTER PORT (from a data sheet received from Lifeboat when I purchased CP/M 2.24a (regarding IOBYTE) but how can it be changed let's say from inside a BASIC program (OBASIC or whatever they call it)...is there a poke for it and if so what is the poke for the CP/M system? Also, the change has to be made for this one instance, then changed back to the equivalent of Dev#5. (LPT). Dev#8 is UL1, but to temporarily change this you have to use the STAT program (I could put this in a SUBMIT statement I suppose); I'm a bit shaky on this subject.

You will note that in the Hayes program, I used the pokes normally reserved for the DATE (DT$) for time, since in the DQFLS program we do not need the date. I would love to know another set of 3 poke locations that are unused though, for this purpose, particularly under OSI's 65U1.2. I did put a runtime subroutine in EDMAFL (DMS) <u>not</u> using pokes at all (since I am already using the date pokes for the DATE, and this works equally well). The reason for the variables (e.g. T9) being one letter and one number is that I am running out of variables, but I would much prefer to have used all alpha-numerical variables as they are easier to read.

Hope this information is useful - I suppose it's only useful to other OSI users who also use Hayes chronographs.

Fred S. Schaeffer
Jamaica, NY 11435

Fred:

CP/M stores a number which determines where output goes in memory location 3. Make the change you desire with STAT, go into BASIC and type PRINT PEEK(3) to see what STAT has done. Then, any time you want to change to that same device, POKE 3,XX will do it!

Al

* * * * *

ED:

First of all, in reply to A. J. Smith's letter in May's PEEK concerning the Keyboard scan for OS65D3.3. The keyboard scan now begins at $2336 and the ASCII value of the key pressed is stored in 9059 (dec).

I also have a question. Does anyone know how to make a VALPTR command for OS65U1.42? I am going to write a word processor one of these days and need the command for a machine code justify routine.

Daniel J. McDonald
Portland, OR 97222

* * * * *

ED:

This letter is in response to the request for a regression program which appeared on page 23 of the May issue of PEEK(65).

I wrote and use a stepwise multiple correlation and regression program. The program will accept an unlimited number of variables and data sets and will select the four most significant independent variables based on the coefficient of determination. The output includes a correlation matrix, range of data, means, medians, standard deviation, coefficient and index of determination, the intercept point, the slope and the beta coefficients.

I have never sold the program nor do I have a manual for it, but I have used it for about four years and I believe that it is free of error.

William K. Groover
Lewisburg, PA 17837

AD$

FOR SALE: OSI C8P-DF, 2Mhz, 48K, dual floppies single-sided. EPSON MX-80 printer w/stand. COMREX monitor. RF modulator. Disk caddy & head cleaning kit. OS65D V3.2 & V3.3 & Home Control. OS65U V1.3 & V1.42. OSDMS Nucleus, A/R, A/P, G/L. WP6502 V1.2 (word processor). SARGON II chess game. OS65D STOS (Smart Terminal Operating System). OS65U terminal software. All manuals, OSI tech notes & PEEK(65) since May/1981. Lots of home-written software (which works!). EVERYTHING IN PERFECT CONDITION. $3000 or best offer. Gary L. Levine, 2731 S. Roslyn St., Denver. Colorado 80231. (303) 750-9660 'EMAIL' on Compuserve: 72115.173

* * * * *

OSI 230E System, 10 M Byte Winchester hard disc drive. Used exclusively for demonstration purposes. Retails for $8500.00. asking $4995.00. Contact MCD Corp., Detroit. (313) 924-1020.

* * * * *

FOR SALE: OS 65D 3.3 AND 65D 3.2 operating systems diskettes and manuals plus WP6502 word processor and 2 games diskettes for C4P MF. $30 for all (15 disks). Sold my OSI. Contact S. Schlehuser, 184 Hickory Corner Rd., East Windsor, NJ 08520. Phone 609-443-6380.

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:

06:8

GOODIES for OSI Users!

PEEK (65)
The Unofficial OSI Users Journal

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

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

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

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

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

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

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

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

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

BOOKS AND MANUALS (while quantities last)

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

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

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

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

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

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

( ) Cash enclosed ( ) Master Charge ( ) VISA

Account No. _____ Expiration Date _____

Signature _____

Name _____

Street _____

City _____ State _____ Zip _____

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

POSTAGE MAY VARY FOR OVERSEAS