PEEK (65)

The Unofficial OSI Users Journal

$1.75

JULY 1983
Vol.4, No.7

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

INSIDE

Column One

Last month, we requested that all you brilliant programmers tell us how you REALLY write programs. The response has not yet started coming in, but to get the ball rolling, I will tell all about my highly structured, top-down technique for producing flawless programs.

The problem was simple. Write a program which would copy files from a hard disk to multiple floppy disks, using Microsoft Basic and CP/M, then restore the same files to the same hard disk in case of a disaster.

Step one. Define the problem. No sweat, I knew what I wanted to do. Skip step one.

Step two. Design the program.. Also easy. Open the file on the hard disk as a random file, open a file by the same name on the floppy disk, also random. Then copy as much of the file as would fit onto one floppy. Then close the file on the floppy, prompt the user to change to the next floppy, do a RESET instruction to let CP/M know the disk had been changed, and copy another floppy full, checking after each read from the hard disk to see if we have reached the end of the file. If so, close all files and print "DONE!" If not, do some more. No real design needed. Skip step two.

Step three. Write the code. No need to write structured style with lots of GOSUBs and REMs. It was all too easy. I should be finished by noon.

Step four. Wonder why the files copied didn't look the same. Spend about 4 hours figuring out which of the FOR-NEXT loops had to be changed from FOR X=1 TO NP to FOR X = 1 TO NP-1 and so forth.

Step five. Flowchart the whole process, including very specific references to the names of the variables and arrays I would use, walk through the process, keeping written notes of specifically which records would be read and written each time through, find 3 more miscounted or misplaced loops.

Step six. Rewrite the code to follow the flowchart like I should have done in the first place.

Lesson: even if it seems like a simple process, it is usually worth flow charting or outlining and certainly worth walking through, with pencil and paper in hand, pretending to be the computer and deciding on the basis of the flow chart just what you would really do if you were blindly following instructions like those darn computers always do. Write down all the variables. When the instruction says "X=X+1" cross out the 44 under the heading "X" and write in a 45. You may be surprised where you wind up!

In any case, you may save yourself the many hours I spent doing steps one through seven in this "simple" example.

The call for articles still stands. We are getting lots of neat information from our dedicated hackers on the C1 and C2 machines, but could use more. However, we are getting virtually nothing on the new multiprocessor machines, on OSU 1.42 and 1.43, on Keybasic and Turbodos.

Yes, I realize all this stuff is very new, and you are all real busy getting it to do what you want it to ... but think of the glory (and money) you will get if you become the recognized OSI hardware Turbodos Keybasic Guru! The way to do it is to be the first guy (or one of the first few guys at least) to write an article for PEEK(65) telling us how you solved your problems, created a multistate payroll program, designed a new database manager, or whatever else you are working on.

OSI has announced two new software products: M80, the standard Macro Assembler, and Keysort, a high-speed sort routine for Keybasic.

Also on things new, OSI plans to show a 16-bit product at Comdex in November. In all probability, this will be a multiprocessor computer like the 300 series of 8-bit computers, using the 16-bit version of Turbodos which has just been completed. OSI will probably not join the UNIX parade, since it would essentially involve time-sharing rather than giving each user his own separate CPU.

There will be a super sale on 74-Mbyte hard disks in July and August. The drives sold are brand new, and include the 592 board (the one which is mounted on the drive). You need only have the CA59 board set which mounts in the computer. If you have a 74-Meg and want to add another, or a C3A which you would like to upgrade to a 74-Meg hard disk machine, this is quite an opportunity.

We are also promised by OSI "three or four major announcements within the next 30 days or so." Nope, we don't know what they are either -- but you will know as soon as we do!

al.

THE WORKINGS OF THE C1P DISK BOOT ROUTINE

By: Jim McConkey
7304 Centennial Rd.
Rockville, MD 20855

When you answer the D/C/W/M? prompt with a "D" the processor jumps to the disk boot routine at $FC00. The JSR at $FC00 goes to the main disk boot routine at $FC0C which reads a starting address from the disk and places this address in $FD,$FE. The number of 256 byte sectors is then fetched and placed in $FF. The code on track 0 is then loaded starting at the address specified by $FD, $FE and is then run by the indirect jump at $FC03.

Data communication to and from the disk is made through a 6850 asynchronous communications interface adapter (ACIA) which is located at $C010-$C011. $C010 functions as both a write-only control register and a read-only status register. Data is passed via $C011.

Control signals are sent to and received from the disk drive via a 6821 peripheral interface adapter (PIA) located at $C000-$C003. The PIA provides two 8-bit I/O ports and actually contains 6 registers. How can 6 registers be accessed when the device only occupies 4 memory locations you ask? Each port has associated with it 3 registers; a control register, a data direction register and a data register. When bit 2 of the control register ($C001,$C003) is set, $C000 and $C002 act as data direction registers. When this bit is reset, they act as data registers. By setting bits in the data direction registers to 1, the corresponding data bits are configured as outputs. Likewise, by reseting data direction register bits, the corresponding data bits are

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.

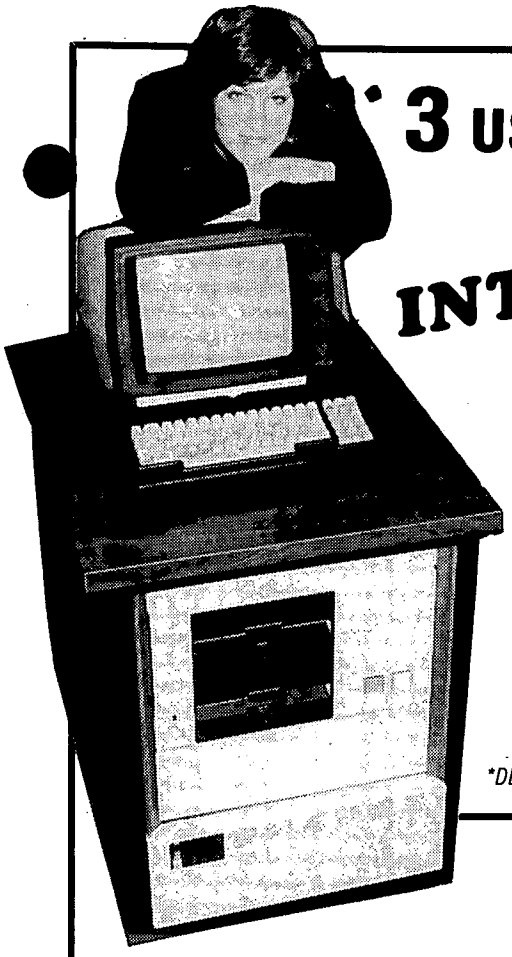
For back issues, subscriptions, change of address or other information, write to:

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

LISTING 1
C1P DISK BOOT

```
$FC00  20 0C FC   D(ISK)     JSR BOOT     ;load track 0
$FC03  6C FD 00              JMP ($FD)    ;run
$FC06  20 0C FC              JSR BOOT     ;reload track 0
$FC09  4C 00 FE              JMP VM       ;goto ML monitor
$FC0C  A0 00      BOOT       LDY #$00
$FC0E  8C 01 C0              STY CR.A     ;address DDR A
$FC11  8C 00 C0              STY DDR.A    ;PA0-PA7 all inputs
$FC14  A2 04                 LDX #$04
$FC16  8E 01 C0              STX CR.A     ;address DR A
$FC19  8C 03 C0              STY CR.B     ;address DDR B
$FC1C  88                    DEY          ;Y=11111111
$FC1D  8C 02 C0              STY DDR.B    ;PB0-PB7 all outputs
$FC20  8E 03 C0              STX CR.B     ;address DR B
$FC23  8C 02 C0              STY DR.B     ;reset all control lines
$FC26  A9 FB                 LDA #$FB     ;mask for step-in
$FC28  D0 09                 BNE MOVE.HD  ;goto MOVE.HD
$FC2A  A9 02      CK.TRK     LDA #$02     ;mask for TRACK 0
$FC2C  2C 00 C0              BIT DR.A     ;check TRACK 0
$FC2F  F0 1C                 BEQ TRK0     ;track 0
$FC31  A9 FF      SEEK.TK0   LDA #$FF     ;another track
$FC33  8D 02 C0   MOVE.HD    STA DR.B     ;set direction controls
$FC36  20 A5 FC              JSR RTS      ;waste 12usec
$FC39  29 F7                 AND #$F7     ;set step flag
$FC3B  8D 02 C0              STA DR.B     ;step
$FC3E  20 A5 FC              JSR RTS      ;waste 12usec
$FC41  09 08                 ORA #$08
$FC43  8D 02 C0              STA DR.B     ;reset all controls
$FC46  A2 18                 LDX #$18     ;set delay time
$FC48  20 91 FC              JSR DELAY    ;waste 30msec to allow
                                          ;head to settle
$FC4B  F0 DD                 BEQ CK.TRK   ;goto CK.TRK
$FC4D  A2 7F      TRK0       LDX #$7F
$FC4F  8E 02 C0              STX DR.B     ;load head
$FC52  20 91 FC              JSR DELAY    ;let head settle
$FC55  AD 00 C0   INDEX      LDA DR.A     ;check for index hole
$FC58  30 FB                 BMI INDEX    ;if not found, keep looking
$FC5A  AD 00 C0   HOLE       LDA DR.A
$FC5D  10 FB                 BPL HOLE     ;wait till end of hole
$FC5F  A9 03                 LDA #$03
$FC61  8D 10 C0              STA ACR      ;reset ACIA
$FC64  A9 58                 LDA #$58     ;configure ACIA
$FC66  8D 10 C0              STA ACR
$FC69  20 9C FC              JSR FETCH    ;get  high byte
$FC6C  85 FE                 STA $FE      ;and store
$FC6E  AA                    TAX          ;save also in X
$FC6F  20 9C FC              JSR FETCH    ;get low byte
$FC72  85 FD                 STA $FD      ;and save
$FC74  20 9C FC              JSR FETCH    ;get number of sectors
$FC77  85 FF                 STA $FF      ;and save
$FC79  A0 00                 LDY #$00     ;init byte counter
```

Continued



3 USERS-80 Mega Bytes — $8990.00*

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

INTRODUCTORY SPECIAL

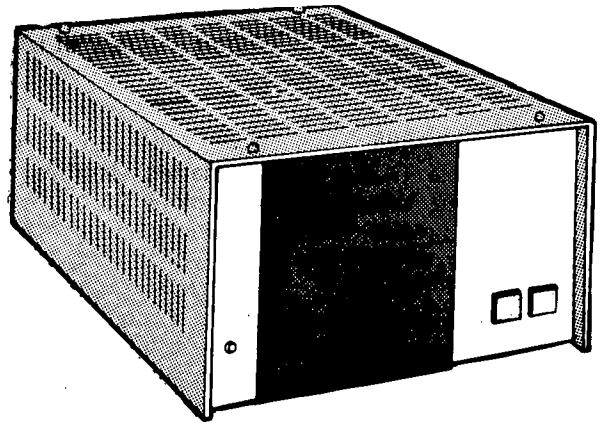
- 90 Days on Power Supply, Floppy Drives — Circuit Boards.
- Configured for Time-Share @ 2 MHZ
- Includes: 2 Serial Printer Ports with Handshake, Improved Cooling, and Ball Bearing Roller Chassis Rails

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

configured as inputs. An I/O word thus programmed need not be all inputs or all outputs but may be mixed.

The code from $FC0C to $FC19 configures PA0-PA7 of the PIA to be all inputs and PB0-PB7 to be all outputs. The STY at $FC23 resets all the control lines, after which a jump is made to $FC33. The code from here to $FC4C moves the head, in this case toward the spindle. The (unconditional) branch at $FC4B goes back to $FC2A which examines the TRACK 0 signal from the drive. If it is not zero, meaning that the head is not at track 0, the direction is reversed and the head is moved outward toward track 0.

If the head is at track 0, control is passed to the code at $FC4D, where the signal is given for the head to be loaded. (Yes, you heard it right! Why the disk boot loads the head but OS-65D doesn't is beyond me.) The JSR at $FC52 executes a subroutine at $FC91-$FC9B, which is a simple time delay whose time can be set via the X register. In this case, the delay is set for about 158msec which allows the head to settle after being loaded.

Upon returning to $FC55 the INDEX signal from the drive is looked for, marking the beginning of the index hole. Once the hole is found, the code from $FC5A to $FC5E waits for it to pass.

The code from $FC61 to $FC68 then rests the ACIA and configures it for 8 bits + 1 stop bit with even parity.

The JSR at $FC69 goes to a subroutine at $FC9C-$FCA5 which waits till the ACIA data register is full (as indicated by the status register) and then gets the byte of data from the data register and returns with the data in A.

The first two bytes of data fetched from track 0 indicate the address at which the data fetched from the disk should be stored. The address is stored in $FD,$FE. The third byte fetched is the number of 256 byte sectors of data stored on track 0. This number is stored in $FF.

The LDY at $FC79 initializes the byte counter. A byte is fetched and stored by the indirect indexed STA at $FC7E. The BNE at $FC81 checks if all 256 bytes of the current sector have been fetched. If not, the code loops back to $FC7B to get another byte. When the sector fetching is complete, control passes to $FC83 which increments the sector pointer and checks if all sectors have been completely fetched. Again, the program loops until all sectors have been completely fetched.

Finally, the starting address is restored by the STX at $FC89 and the ACIA is reset by $FC8B-$FC8F. The RTS at $FC90 returns to $FC03 where the code that was just loaded from track 0 is executed by the indirect JMP. In the case of HEXDOS, the code from track 0 comprises the entire operating system. For OS-65D I imagine the code is a more extensive disk boot which allows for a multiple track load.

LISTING 1
(cont'd)

```
$FC7B  20 9C FC  ANOTHER   JSR FETCH    ;get a byte
$FC7E  91 FD               STA ($FD),Y  ;store
$FC80  C8                  INY          ;done yet?
$FC81  D0 F8               BNE ANOTHER  ;no,get somemore
$FC83  E6 FE               INC $FE      ;yes,point to next sector
$FC85  C6 FF               DEC $FF      ;done yet?
$FC87  D0 F2               BNE ANOTHER  ;no,get next sector
$FC89  86 FE               STX $FE      ;yes,restore address
$FC8B  A9 FF               LDA #$FF
$FC8D  8D 02 C0            STA DR.A     ;reset all controls
$FC90  60                  RTS
$FC91  A0 F8       DELAY   LDY #$F8     ;init inner loop count
$FC93  88          LOOP    DEY          ;inner loop done yet?
$FC94  D0 FD               BNE LOOP     ;no, keep looping
$FC96  55 FF               EOR $FF,X    ;yes
$FC98  CA                  DEX          ;outer loop done yet?
$FC99  D0 F6               BNE DELAY    ;no, wait some_more
$FC9B  60                  RTS          ;yes, done.

$FC9C  AD 10 C0  FETCH     LDA ASR      ;look at ACIA status
$FC9F  4A                  LSR          ;C=data buffer full
$FCA0  90 FA               BCC FETCH    ;not full yet, try again
$FCA2  AD 11 C0            LDA ADR      ;ready, get byte
$FCA5  60          RTS     RTS          ;return
```

NOTES:

CR.A	control register for PA0-PA7
DDR.A	data direction register for PA0-PA7
DR.A	data register for PA0-PA7
CR.B	control register for PB0-PB7
DDR.B	data direction register for PB0-PB7
DR.B	data register for PB0-PB7
ACR	ACIA control register
ASR	ACIA status register
ADR	ACIA data register

★ ★

CRE8, A SEMI-AUTOMATIC FILE CREATION UTILITY FOR OS-65D

by: Willis H. Cook
1298 Renee Drive
Lilburn, GA 30247

The normal way to create a file under the 65D operating system is first to run DIR to find out where all the existing files reside on the disk, then run CREATE to allocate and clear the new file. This program, CRE8, is designed to replace the CREATE utility and will save one step by automatically allocating the number of tracks you specify in the first free space it finds on the disk. A completely automatic file creation utility would tell you how much space your file requires and then create the file for you (Yep, some systems do that), so this one is only semi-automatic, but it is more convenient than the standard OSI utility.

Actually, the program does a lot more than just create a file. It searches the directory to make sure the file name you specify is not already in use on the current disk, it checks to see if there is room in the directory for a new entry and it determines if there are enough free tracks adjacent to each other to hold the size file you have specified. If there is enough space in both the directory and on the disk for the file, it adds the name and track numbers to the directory and initializes the tracks the file will occupy. That is a lot of activity for an 80-line program.

Listing 1. Version for 8" Disks

```
10 REM  ************************************************
20 REM  *                                              *
30 REM  *                  C R E 8                     *
40 REM  *                                              *
50 REM  ************************************************
60 :
70 FOR I=11902 TO 11919 : READ D : POKE I,D : NEXT I
80 J$="3A7E" : IF PEEK(8999)=49 THEN POKE 11909,49 : J$="317E"
90 POKE 8955,126 : POKE 8956,46 : REM Set USR address.
100 X=USR(X) : POKE 8955,212 : POKE 8956,34 : REM  Reset USR ad-
110 DIM T%(76) : BUF=11889                                    dress.
120 PRINT : INPUT "Filename";F$
130 IF LEN(F$)<6 THEN F$=F$+" " : GOTO 130
140 IF LEN(F$)>6 THEN F$=LEFT$(F$,6)
150 INPUT "Number of tracks";TR
160 :
170 REM *  Read existing file names and track numbers,  *
180 REM *  find free tracks and check existing names.   *
190 :
200 FOR I=1 TO 76 : T%(I)=0 : NEXT
210 DISK!"CALL 2E79=08,1"
220 FLAG=0
230 FOR I=1 TO 32
240 : ST=PEEK(I*8+BUF+6) : ET=PEEK(I*8+BUF+7)
250 : IF ET=0 THEN 350
260 : ST=10*INT(ST/16)+ST-16*INT(ST/16)
270 : ET=10*INT(ET/16)+ET-16*INT(ET/16)
280 : FOR J=ST TO ET : T%(J)=1 : NEXT J : N$=""
290 : FOR J=I*8+BUF TO I*8+BUF+5
300 :   N$=N$+CHR$(PEEK(J))
310 : NEXT J
320 : IF N$<>F$ THEN 350
330 : PRINT : PRINT "File name '";F$;"' is already in use." :
340 : GOTO 120                                          PRINT
350 NEXT I
360 IF FLAG=1 THEN 420
370 DISK!"CALL 2E79=08,2"
380 FLAG=1 : GOTO 230
390 :
400 REM *  Find enough contiguous free tracks for file. *
410 :
420 FOR I=1 TO 76
430 : IF T%(I)=1 THEN 480
440 : FOR J=I TO 76
450 :   IF T%(J)=1 THEN I=J : GOTO 480
460 :   TF=J-I+1 : IF TF=TR THEN 550
470 : NEXT J
480 NEXT I
```

Continued on page 6

My machine is a C2-4P with a single 8" drive and I have used the program with both versions 3.2 and 3.3 of the 65D operating system. A version for 5 1/4" disks is given in listing 2, but I have not actually run that version. If you run this program on a machine with a 5 1/4" drive, please first try it on an expendable disk, so you won't lose anything if it doesn't work. The program modifies the directory, so an error could cause any files on a disk to become inaccessible.

HOW TO LOAD AND RUN THE PROGRAM

The program requires a disk buffer large enough to hold one disk track: 3072 bytes for an 8" disk and 2048 bytes on a 5 1/4" disk. (Create a buffer either by means of the BUFFER utility that comes with 65D version 3.3 or by loading CREATE then typing NEW before typing in this program.) On an 8" system, the program with the buffer space fits onto two tracks, the same as the standard CREATE utility. On a 5 1/4" system, it may spill over into three tracks, since it is somewhat longer than CREATE. Will someone let the other PEEK(65) readers know how it works and how much space it requires?

When running the program it is only necessary to give the file name and the number of tracks desired. If there is enough space the program reports that the file has been created and informs the user of the tracks used. If there is insufficient space the program informs the user of that fact and terminates.

HOW THE PROGRAM WORKS

The first thing the progam does is read a machine-language routine into memory from a couple of DATA statements and then execute the routine by means of a USR function. This routine fills the buffer space with nulls, and the buffer is subsequently read into each track of the new file to clear it. This is identical to the standard CREATE program. The user enters the file name and number of tracks required and the program reads the disk directory into the directory buffer area, which is at the same location ($2E79) in both 8" and 5 1/4" machines. Next, an array T% is filled with 1's representing tracks in use and 0's for empty tracks. This tells the program which tracks are available. At the same time, the existing directory names are compared to the input file name to make sure that the new name is unique to that disk.

If the file name is not currently in the directory, the program examines the T% array to find enough adjacent free tracks to hold the new file. When it does, it has to re-examine the directory to find the first free directory location in which to store the new file name and track addresses. It saves the directory back to the disk, initializes the tracks for the new file and finally tells the user it is finished.

LISTING 1. VERSION FOR 8" DISKS
Continued from page 5

```
490 D$="s" : IF TR=1 THEN D$=""
500 PRINT : PRINT "I can't find";TR;"free track";D$;" on the
510 END                                                    disk."
520 :
530 REM *  Store file name in free directory location.  *
540 :
550 ST=I : ET=J : FLAG=0
560 FLAG=0 : DISK!"CALL 2E79=08,1"
570 FOR I=1 TO 32
580 : IF PEEK(I*8+BUF)=ASC("#") THEN 630
590 NEXT I
600 IF FLAG=1 THEN 620
610 FLAG=1 : DISK!"CALL 2E79=08,2" : GOTO 570
620 PRINT : PRINT "No room in the directory." : END
630 FOR J=1 TO 6
640 : POKE I*8+BUF+J-1,ASC(MID$(F$,J,1))
650 NEXT J
660 POKE I*8+BUF+6,16*INT(ST/10)+ST-10*INT(ST/10)
670 POKE I*8+BUF+7,16*INT(ET/10)+ET-10*INT(ET/10)
680 IF FLAG=1 THEN DISK!"SAVE 08,2=2E79/1" : GOTO 700
690 DISK!"SAVE 08,1=2E79/1"
700 FOR I=ST TO ET : T$=RIGHT$(STR$(100+I),2)
710 : DISK!"INIT "+T$ : DISK!"SAVE "+T$+",1="+J$+"/B"
720 NEXT I
730 D$="s" : IF ET=ST THEN D$=""
740 PRINT : PRINT TAB(5)"File ";F$;",";ET-ST+1;" track";D$;"
                                                           long,"
750 PRINT TAB(5)"has been created on track";D$;ST;
760 IF ET>ST THEN PRINT "through";ET;
770 PRINT : END
780 :
790 DATA 160,0,162,12,152,153,126,58,200
800 DATA 208,250,238,133,46,202,208,244,96
```

Listing 2. Program Changes for 5 1/4" Disks

```
80 J$="3A7E" : IF PEEK(8999)=50 THEN POKE 11909,50 : J$="327E"
110 DIM T%(39) : BUF=11889
200 FOR I=1 TO 39 : T%(I)=0 : NEXT
210 DISK!"CALL 2E79=12,1"
370 DISK!"CALL 2E79=12,2"
420 FOR I=1 TO 39
440 : FOR J=I TO 39
560 FLAG=0 : DISK!"CALL 2E79=12,1"
610 FLAG=1 : DISK!"CALL 2E79=12,2" : GOTO 570
680 IF FLAG=1 THEN DISK!"SAVE 12,2=2E79/1" : GOTO 700
690 DISK!"SAVE 12,1=2E79/1"
710 : DISK!"INIT "+T$ : DISK!"SAVE "+T$+",1="+J$+"/B"
790 DATA 160,0,162,8,152,153,126,58,200
```

Eight inch disks hold 77 tracks versus 40 for the smaller disks and the directory is on different tracks. These differences are shown in listing #2 which contains the lines that are different for the 5 1/4" version. All other lines are identical for the two disk sizes.

One point to be especially careful of when typing in this program is to be sure to leave a space after the command INIT in line 710. INIT followed by a space and then a number initializes that track number. INIT without a space initializes the entire disk,---- obviously to be avoided.

★ ★

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

ADDRESS/MAILING LIST FOR OSI C1P

By: Charles Stewart
3033 Marvin Dr.
Adrian, MI 49221

GENERAL DESCRIPTION:

This program will perform the following functions.

A. Receive user input from keyboard or program data tape.

B. Print out labels suitable for mailing.

C. Generate data tape which may be inserted into host program for changes, deletions, etc.

D. Generate label printing program utilizing user inputs as data statements.

DESCRIPTION OF OPERATION:

When run the program will prompt the user as follows:

1. Generate program.

2. Input data for correction.

Type in your choice and return (CR)

If the user chooses the generate label program, he is prompted as follows:

PLEASE SPECIFY THE NUMBER OF INPUTS REQUIRED...

This program will generate a program containing data statements which contain the user inputed data, or will generate labels suitable for mailing purposes. The user has the option of generating a data tape which may be inputed into this program for correction or revision. If you make a mistake during the input section type 'HELP' in response to the prompt!

After response to the number of inputs prompt the OSI prompts....'FULL NAME OF PARTY' and continue with address, city, state, and zip code until the number of labels requested is fulfilled. Typing 'HELP' to any prompt will delete that input and return you to the 'FULL NAME OF PARTY' prompt.

When complete, your computer will print out 'YOU HAVE COMPLETED (NUMBER OF INPUTS) DO YOU REQUIRE ADDITIONAL SPACE'. Upon a yes (y) answer 'HOW MANY', enter number of additional inputs required. This procedure will continue until you answer NO (n). At

LISTING

```
1 REM CHARLES STEWART
10 REM MAILING LIST REV 2.0
30 CLEAR
41 POKE11,34:POKE12,2:POKE574,96
42 FORX=0TO27:R3=PEEK(65036+X):POKE546+X,R3:NEXT
45 X=USR(X):DIMC$(3*13+2),D$(3*13+2),E$(3*13+2),F$(3*13+2)
46 DIMG$(3*13+2):A$="DATA"
60 A=61440:B=A+1:DIMA(15):X=USR(X)
70 PRINTTAB(10)"MENU":PRINT:PRINT:PRINT
80 PRINT"(1)GENERATE PROGRAM
85 PRINT:PRINT"(2)INPUT DATA FOR":PRINT"CORRECTION":PRINT
90 INPUTY:ONYGOTO100,415
100 X=USR(X):INPUT"NUMBER OF INPUTS";Z:FORQ=1TOZ
140 GOSUB555 :INPUTC$(Q):IFC$(Q)="HELP"THEN205
150 GOSUB560 :INPUTD$(Q):IFD$(Q)="HELP"THEN200
160 GOSUB565 :INPUTE$(Q):IFE$(Q)="HELP"THEN200
170 GOSUB570 :INPUTF$(Q):IFF$(Q)="HELP"THEN200
180 GOSUB575 :INPUTG$(Q):IFG$(Q)="HELP"THEN200
190 X=USR(X):NEXTQ:GOTO220
200 GOTO140
205 Q=Q-1:GOTO140
220 GOTO305
225 INPUT"HOW MANY";H:R=Q:Z=Z+H:FORQ=RTOZ
240 GOSUB555 :INPUTC$(Q):IFC$(Q)="HELP"THEN300
250 GOSUB560 :INPUTD$(Q):IFD$(Q)="HELP"THEN240
260 GOSUB565 :INPUTE$(Q):IFE$(Q)="HELP"THEN240
270 GOSUB570 :INPUTF$(Q):IFF$(Q)="HELP"THEN240
280 GOSUB575:INPUTG$(Q):IFG$(Q)="HELP"THEN240
290 NEXTQ:PRINT:GOTO375
300 Q=Q-1:GOTO240
305 PRINT:PRINT"DO YOU REQUIRE A PRINT":PRINT"OUT TO CHECK YOUR"
310 INPUT"WORK";R$
315 IFLEFT$(R$,1)="N"THEN375
320 X=USR(X):FORQ=1TOZ
330 PRINTC$(Q):PRINTD$(Q):PRINTE$(Q),F$(Q):PRINTG$(Q)
345 INPUT"THIS SET CORRECT";P$:IFASC(P$)=78THEN360
350 X=USR(X):NEXTQ:GOTO375
360 GOSUB555 :INPUTC$(Q):GOSUB560 :INPUTD$(Q):GOSUB565 :INPUTE$
365 GOSUB570 :INPUTF$(Q):GOSUB575 :INPUTG$(Q)                    (Q):
370 PRINT:GOTO330
375 X=USR(X):PRINT" MENU":FORX=0TO5:PRINT:NEXT
390 PRINT" (1 GENERATE PROGRAM TAPE":PRINT" (2 PRINT OUT LABELS"
395 PRINT" (3 INPUT DATA FOR CORR-":PRINT"ECTION"
400 PRINT:PRINT" (4 CREATE DATA TAPE"
402 PRINT" (5 EDIT":PRINT" (6 SEARCH":PRINT" (7 ADDITIONAL INPUTS
405 PRINT:INPUT"YOUR SELECTION";K
410 ONKGOTO580,440,415,510,700,2000,225
415 X=USR(X):INPUT"FILE NAME TO LOAD";V$:R(7)=ASC(V$)+100
417 WAITA,1:C=PEEK(B):IFC()R(7)THEN417
418 WAITA,1:IFPEEK(B)()13THEN418
420 LOAD:INPUTZ:FORQ=1TOZ:INPUTC$(Q):INPUTD$(Q):INPUTE$(Q):
                                                    INPUTF$(Q):
425 INPUTG$(Q):NEXTQ:POKE515,0:GOTO375
440 PRINT"NUMBER OF SPACES BET-":INPUT"WEEN LABELS";J
443 IFJ(=0THENPRINT:PRINT"INVALID ENTRY":PRINT:GOTO440
444 IFJ)=6THENPRINT:PRINT"INVALID ENTRY":PRINT:GOTO440
445 INPUT"ARE YOU READY";Y$
450 IFLEFT$(Y$,1)="N"THEN445
455 POKE517,255
460 FORT=0TO1500:NEXT:FORQ=1TOZ:IFC$(Q)="X"THENNEXTQ
470 PRINTTAB(5)C$(Q):PRINTTAB(5)D$(Q):PRINTTAB(5)E$(Q),F$(Q)
475 PRINTTAB(5)G$(Q):FORW=1TOJ:PRINT:NEXT
480 NEXTQ:POKE517,0
505 GOTO375
510 X=USR(X):INPUT"FILE NAME";V$
512 REM KICK NAME OUT OF ASCII RANGE
514 R(7)=ASC(V$)+100
516 INPUT"IS RECORDER READY";V$
518 SAVE:PRINT:PRINTCHR$(R(7))
520 PRINTZ:FORQ=1TOZ
530 PRINTC$(Q):PRINTD$(Q):PRINTE$(Q):PRINTF$(Q):PRINTG$(Q)
535 NEXTQ:POKE517,0:GOTO375
550 GOTO580
555 PRINT:PRINT"FULL NAME OF PARTY":RETURN
560 PRINT"ADDRESS":RETURN
565 PRINT"CITY":RETURN
570 PRINT"STATE":RETURN
575 PRINT"ZIP CODE":RETURN
```

Continued on page 9

```
580 PRINT"NUMBER OF SPACES REQ-":INPUT"UIRED BETWEEN LABELS";I
585 PRINT:INPUT"OUTPUT PROGRAM--TAPE READY";Z$:POKE517,255:X=0:
590 FORT=0TO1000:NEXT                                    Y=10
595 X=X+Y:PRINTX;"POKE517,255:?:?"
600 X=X+Y:PRINTX;"READA$,B$,C$,D$,E$"
605 X=X+Y:PRINTX;"?A$:?B$:?C$,D$:?E$"
610 X=X+Y:PRINTX;:FORS=1TOI:PRINT"?:";:NEXT:PRINT" "
615 X=X+Y:PRINTX"GOTO20"
620 FORQ=1TOZ:IFC$(Q)="X"THENNEXTQ
625 X=X+Y:PRINTX;A$;C$(Q);B$;D$(Q);B$;E$(Q);B$;F$(Q);B$;G$(Q):
630 PRINT"POKE515,1:RUN":POKE517,0:GOTO375                NEXT
700 REM-EDIT SECTION
710 X=USR(X):FORQ=1TOZ:IFC$(Q)="X"THENNEXTQ
720 PRINTC$(Q):PRINTD$(Q):PRINTE$(Q),F$(Q)
725 PRINTG$(Q):PRINT:PRINT
730 PRINT" EDIT FUNCTIONS":PRINT:PRINT"(1 CHANGE NAME
735 PRINT"(2 ADDRESS":PRINT"(3 CITY":PRINT"(4 STATE"
740 PRINT"(5 ZIP CODE":PRINT"(6 SCRATCH":PRINT"(7 EXIT"
745 PRINT"(8 NEXT ENTRY":PRINT:INPUTR1
750 ONR1GOTO800,900,1000,1100,1200,1300,1400,1500
800 GOSUB555:INPUTC$(Q):X=USR(X):GOTO720
900 GOSUB560:INPUTD$(Q):X=USR(X):GOTO720
1000 GOSUB565:INPUTE$(Q):X=USR(X):GOTO720
1100 GOSUB570:INPUTF$(Q):X=USR(X):GOTO720
1200 GOSUB575:INPUTG$(Q):X=USR(X):GOTO720
1300 C$(Q)="X":D$(Q)="X":E$(Q)="X":F$(Q)="X":G$(Q)="X"
1310 NEXTQ
1400 GOTO375
1500 NEXTQ:GOTO710
2000 REM--SEARCH ROUTINE
2005 X=USR(X)
2010 PRINT:PRINT"SEARCH FUNCTIONS":PRINT:PRINT:PRINT
2020 PRINT"(1 BY NAME":PRINT"(2 BY CITY":PRINT"(3 BY STATE"
2030 PRINT"(4 BY ZIP CODE":PRINT"(5 EXIT"
2040 PRINT:PRINT:INPUTI:ON IGOTO2100,2200,2300,2400,375
2100 X=USR(X):INPUT"NAME OF PARTY";T$
2110 FORQ=1TOZ:IFT$=C$(Q)THEN2150
2120 NEXTQ:GOTO2500
2150 GOSUB3000:GOTO2010
2200 X=USR(X):INPUT"NAMQOF CITY";T$
2210 FORQ=1 TOZ:IFT$=E$(Q)THEN2250
```

Continued on page 10

```
2230 GOTO2290
2250 GOSUB3000:NEXTQ:GOTO2010
2290 NEXTQ:GOTO2500
2295 GOTO2010
2300 X=USR(X):INPUT"NAME OF STATE";T$
2310 FORQ=1TOZ:IFT$=F$(Q)THEN2350
2320 GOTO2390
2350 GOSUB3000:NEXTQ:GOTO2010
2390 NEXTQ:GOTO2500
2395 GOTO2010
2400 X=USR(X):INPUT"ZIP CODE NUMBER";T$
2410 FORQ=1TOZ:IFT$=G$(Q)THEN2450
2420 GOTO2490
2450 GOSUB3000:NEXTQ:GOTO2010
2490 NEXTQ
2500 X=USR(X):GOTO2010
3000 POKE517,255:PRINT:PRINTC$(Q):PRINTD$(Q):PRINTE$(Q)
3010 PRINTF$(Q):PRINTG$(Q):POKE517,0:RETURN
```

this time, you are asked 'DO YOU REQUIRE A PRINTOUT TO CHECK YOUR WORK'. A yes answer will print on the screen each label and gives the opportunity to correct any mistake. Following this section, you have the following options.

MENU

1. Generate program tape.
2. Print out labels.
3. Input data for correction
4. Create data tape
5. Edit
6. Seach
7. Additional inputs

★ ★ ★

A BEXEC* in 1 TRACK

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

Here is a corrected and slightly modified copy of the "BEXEC* in 1 track" that appeared in the June issue of PEEK(65). A number of errors and omissions somehow got into the code between the time I wrote it, Craig Lombard modified it, and it was published. This program was written for the C1PMF but Craig found that it works on a C4 as well. Users with 8" disks will need to modify line 14 so that T=76 and lines 16 and 36 to refer to the correct track and sectors for an 8" directory. The program combines several useful functions: List directory in sorted order; Print directory to device #1 (serial printer); list Free tracks; Create, Delete, or Rename files; and Save updated directory. Exit terminates the program. The length of the program is 1753 bytes.

Jim Hays, Seattle, WA 98116

```
2 REM BEXEC* 06/15/83
4 DEFFNA(X)=10*INT(X/16)+X-16*INT(X/16)
6 DEFFNB(X)=16*INT(X/10)+X-10*INT(X/10)
8 D=8994:POKED-1,2:POKE741,76:POKE750,78:F$="FILENAME"
10 POKE2073,96:POKE2893,28:POKE2894,11:E$=" ALREADY EXISTS"
14 T=39:DIMN$(T),F(T),Q(T),U(T):C=0:K=0:P=11897
16 DISK!"CA 2E79=12,1":GOSUB88:DISK!"CA 2E79=12,2":GOSUB88
18 POKED,2:NULL0:PRINT:PRINTK"Files";"C"Tracks"
20 INPUT"List/Print/Free/Rename/Delete/Create/Save/Exit";R$
22 R$=LEFT$(R$,1):IFR$="L"THEN40
24 IFR$="P"THENNULL8:POKED,3:GOTO40
26 IFR$="F"THEN50
28 IFR$="R"ORR$="D"THEN60
30 IFR$="C"THEN74
32 IFR$="E"THENPOKE2073,173:END
34 IFR$<>"S"THEN18
36 GOSUB100:DISK!"SA 12,1=2E79/1":GOSUB100:DISK!"SA 12,2=2E79/1"
38 GOTO18
40 PRINT:PRINT"OS-65D V3.X DIRECTORY"
42 PRINTF$"   TRACK RANGE":GOSUB116
44 FORA=0TOT:FORB=1TOK
```

Continued on page 11

```
46 IFA=F(B)THENPRINTN$(B)TAB(12)F(B)TAB(16)"-"Q(B)
48 NEXTB,A:GOTO18
50 PRINT:PRINT"    --FREE TRACKS--":GOSUB116:J=0
52 FORI=JTOT:IFU(I)=0THEN56
54 NEXTI:GOTO18
56 PRINTTAB(6)I;TAB(7)"-";:FORJ=ITOT:IFU(J)=0THENNEXTJ
58 PRINTJ-1:I=J:GOTO54
60 GOSUB110:FORI=1TOK:IFN$(I)=N$THEN64
62 NEXTI:PRINTN$" NOT FOUND":GOTO18
64 IFR$="R"THEN70
66 FORJ=F(I)TOQ(I):U(J)=0:C=C-1:NEXTJ:N$(I)=N$(K)
68 F(I)=F(K):Q(I)=Q(K):K=K-1:GOTO18
70 GOSUB112:FORJ=1TOK:IFI<>JANDN$(J)=N$THENPRINTN$;E$:GOTO18
72 NEXTJ:N$(I)=N$:GOTO18
74 IFK=TTHENPRINT"NO SPACE":GOTO18
76 GOSUB112:FORI=1TOK:IFN$(I)=N$THENPRINTN$;E$:GOTO18
78 NEXTI:INPUT"1st track,# tracks";F,N:Q=F+N-1
80 IF(F>Q)OR(Q>T)THENPRINT"ERROR":GOTO18
82 FORI=FTOQ:IFU(I)=1THENPRINT"TRACK"I"USED":GOTO18
84 NEXTI:FORI=FTOQ:U(I)=1:C=C+1:NEXTI:K=K+1
86 N$(K)=N$:F(K)=F:Q(K)=Q:GOTO18
88 FORI=PTOP+248STEP8:IFPEEK(I)=35THEN98
90 IFK=TTHENPRINT">"T"FILES":RETURN
92 K=K+1:N$(K)="":FORJ=ITOI+5:N$(K)=N$(K)+CHR$(PEEK(J)):NEXTJ
94 F(K)=FNA(PEEK(I+6)):Q(K)=FNA(PEEK(I+7))
96 FORJ=F(K)TOQ(K):U(J)=1:C=C+1:NEXTJ
98 NEXTI:RETURN
100 FORI=PTOP+248STEP8:IFK=0THEN106
102 L=0:FORJ=ITOI+5:L=L+1:POKEJ,ASC(MID$(N$(K),L,1)):NEXTJ
104 POKEI+6,FNB(F(K)):POKEI+7,FNB(Q(K)):K=K-1:GOTO108
106 FORJ=ITOI+5:POKEJ,35:NEXTJ:POKEI+6,0:POKEI+7,0
108 NEXTI:RETURN
110 PRINT"OLD ";:GOTO114
112 PRINT"NEW ";
114 PRINTF$;:INPUTN$:N$=LEFT$(N$+"        ",6):RETURN
116 FORI=1TO22:PRINT"-";:NEXTI:PRINT:RETURN
```

★ ★

DEVEL3 - OLD FILE TO NEW FILE LOADER FOR OS-DMS MASTER FILES

Program written by Colin Law

Reviewed by:
Fred S. Schaeffer
84-55 Daniels St. #4f
Jamaica, NY 11435

Some of you may recall that sometime ago, I submitted one of many 'Letters to the Editor' asking for some ideas how to pre-load a field in a master file without having to manually pump in data. Little did I realize that someone would send me an entire program to do it. That individual was Colin Law who works as a programmer for New Zealand Television, in Auckland, N.Z. It is not only the first solid program that was useful that anyone sent me, but it is also the first program that with minor modifications worked from the very beginning.

Let me try to explain it: Let's suppose (as in the sample audit trail) I have two master files, the OLD file is EBMEM0 and the NEW file is EBCNL0. EBMEM0 is a master membership listing of an organization and every year so many members drop by the wayside. For the purpose of soliciting those ex-members to rejoin, I like to keep them in a cancelled (EBCNL0) file rather than packing them out of EBMEM0 (which is done eventually). At present, there is NO WAY with the standard OS-DMS utilities to do that. You cannot take a portion of the files and move them to another MASTER file without removing them also from the original file. But what about that sequential or random merge utility - well, it does all this in some way, but by far not as fast or as nicely. DEVEL3 (assume this was the third version of a developmental program and I didn't bother to give it another name) gives me two options, I can fill the NEW file from record number 1 (thereby relacing previous contents) OR I can append the incoming records to the end of the current records already residing in the NEW file. Furthermore, I can, with DEVEL3, load data from the OLD file to another field in the NEW file, because the NEW file's fields need not be the same as those in the OLD file, provided that those used in the transfer are the same or greater field length than those whence they came.

Here's some explanation what's actually happening in the program: First of all, most set-up param questions allow X for escape. I have deleted the flags (as Colin programmed this under OSU 1.42 and I needed it for 1.2)....line 100 clears screen; line 120, 130 gets file 1 and 2 info. Lines 200-270 is accessed twice, once for each file and checks for master file name, length, etc. Lines 300-400 verifies the files (disk I/O), lines 500-600 sets DIMS and gets field labels and lengths. Line 590 EQ is a flag to tell us that files have identical field lengths. 800-890 sets parameters for OFIELD (old field) to NFIELD (new field) and you can put in many combinations of fields (not just one). 1000-1210 is a review of the params you set. 2000-2160 searches the first file to assure the selections chosen match with the fields and lengths, and also that the word FIELD is not part of a portion of the field contents. 2300-2350 does similar for file 2. 2525-2620 does padding when field length of 2nd file is longer than that of first file. The rest is sort of self-explanatory. The line of flags for version 1.42 which I omitted was:

```
30 FLAG2:FLAG5:FLAG9:FLAG11:
   FLAG15:FLAG18:FLAG21:
   FLAG23:FLAG27
```

If any of you have any questions, suggest you write me. If I cannot answer the questions (in PEEK(65)), I'll try to get the data from Colin, with whom I have quite regular correspondence since we are both in different ways using highly customized OS-DMS and, therefore, have a lot in common.

```
SAMPLE AUDIT TRAIL
------------------

FROM FILE:EBMEM0                        TO FILE:EBCNL0
WHERE (T) IS 00))
FROM FIELD NO: 1  TO FIELD NO: 1
FROM FIELD NO: 2  TO FIELD NO: 2

FROM RECORD NUMBER:        TO RECORD NUMBER:
#1     1                          1
#2     7                          2
#3    16                          3


etc
```

```
10 REM OLD FILE TO NEW FILE LOADER
20 REM BY COLIN LAW/NEW ZEALAND TV/FEB83
30 REM SAVE FOR FLAGS
40 BB$="                ":BB$=BB$+BB$+BB$+BB$+BB$
50 UU$="**********":UU$=LEFT$(BB$,15)+UU$+UU$+UU$+UU$
60 HH$=LEFT$(BB$,15)+"***  OLD FILE TO NEW FILE - LOADER ***"
70 REM
100 FOR I=1TO24:PRINT:NEXT
105 PRINTUU$:PRINTHH$:PRINTUU$:PRINT:PRINT
110 REM GET FILE INFO
120 CH=1:PRINT"FROM FILE:":PRINT:GOSUB 210
130 CH=2:PRINT"TO FILE:":PRINT:GOSUB 210
140 REM
150 PD=PEEK(11686):PRINT"AUDIT PRINTOUT: (P OR C) ";:INPUT PD$
155 IF PD$="X" THEN 50200
160 IF PD$="P" THEN PD=5:GOTO310
170 ID PD$()"C" THEN GOSUB 60110:GOTO 150
180 PRINT:GOTO 310
200 REM GET FILE NAMES
210 PRINT"DEVICE: ";:INPUTDV$(CH):PRINTTAB(15);
215 IF DV$(CH)="X" THEN 50200
220 PRINT"FILE NAME: ";:INPUT F$(CH):PRINTTAB(40);
225 IF F$(CH)="X" THEN 50200
230 IF RIGHT$(F$(CH),1)()"0" THEN ER=1:GOTO30000
240 IF LEN(F$(CH))()6 THEN ER=1:GOTO30000
250 PRINT"PASSWORD: ";:INPUTP$(CH):PRINT
255 IF P$(CH)="X" THEN 50200
260 IF LEN(P$(CH))()4 THEN ER=2:GOTO 30000
270 RETURN
280 REM
300 REM CHECK FILES & GET LABELS ETC
310 CLOSE:FOR CH=1TO2:DEV DV$(CH):OPEN F$(CH),P$(CH),CH
320 PRINT:PRINT:PRINT "))))CHECKING "; F$(CH);" ";
330 INPUT%CH, T$:IF T$()LEFT$(F$(CH),5)THEN ER=3:GOTO30000
340 INDEX(CH)=6:INPUT%CH,TY:IF TY()10 THEN ER=3:GOTO30000
350 INDEX(CH)=9:INPUT%CH,EODF(CH):INDEX(CH)=20:INPUT%CH,BODF(CH)
360 INDEX(CH)=31:INPUT%CH,RL(CH):INDEX(CH)=42:INPUT%CH,RN(CH)
370 INDEX(CH)=53:N(CH)=1:REM COUNT THE FIELDS
380 INPUT%CH,T$:INPUT%CH,T:PRINT"*";
390 IF INDEX(CH)(BODF(CH)THEN N(CH)=N(CH)+1:GOTO380
400 PRINT:NEXT CH:N=N(1):IF N(2))N(2))N(1) THEN N=N(2)
410 REM
500 DIM FDLB$(N,2),FL(N,2),FC$(N,2),FP(N+1,2)
510 FOR CH=1TO2:INDEX(CH)=53
520 FORI=1TON(CH):INPUT%CH,FDLB$(I,CH):INPUT%CH,FL(I,CH)
530 FL(I,CH)=FL(I,CH)-1:NEXT I
540 REM
550 FOR I=1TON(CH):FP(I+1,CH)=FL(I,CH)+FP(I,CH)+1:NEXT I
560 NEXT CH:CLOSE
570 EQ=0:IF N(1)()N(2) THEN 700
580 FOR I=1TON(1):IF FL(I,1)()FL(I,2) THEN 700
590 NEXT:EQ=1
600 REM
700 REM GET TRANSFER INSTRUX
```

```
705 PRINT:PRINT:PRINTUU$:PRINTHH$:PRINTUU$:PRINT:PRINT
710 PRINT"ENTER X IF COMPLETED - FIELD NUMBER TO EXAMINE IN ";
715 PRINTF$(1);" ";:INPUT OFIELD$:IF OFIELD$="X" THEN 50200
720 PRINT:OFIELD=VAL(OFIELD$)
725 IF OFIELD(1OROFIELD)N(1) THEN GOSUB 60100:GOTO710
730 PRINT:PRINT"CONTENTS TO SEEK IN ";FDLB$(OFIELD,1);":";
740 INPUT CT$:IF CT$="X" THEN 50200
750 IF EQ=0 THEN 805
760 PRINT:PRINT"TRANSFER ENTIRE RECORD (Y OR N) ";:INPUT Q$
765 IF Q$="X" THEN 50200
770 IF LEFT$(Q$,1)="N" THEN 805
780 IF LEFT$(Q$,1)()"Y" THEN GOSUB 60110:GOTO760
790 NFIELD(1)=99:GOTO850
800 PRINT:PRINT"??????????? TRY AGAIN": REM GET TRANSFER FIELD
805 NK=1:PRINT:PRINT"--TRANSFER FIELDS:ENTER 0 WHEN COMPL--"
810 PRINT:PRINT"TRANSFER FROM ";F$(1);:INPUT"WHICH FIELD No:";FIELD$
815 PRINT"TO ";F$(2);" WHICH FIELD No:";:INPUT NFIELD$
820 FIELD(NK)=VAL(FIELD$):IF FIELD(NK)=0 THEN 845
825 NFIELD(NK)=VAL(NFIELD$):IF NFIELD(NK)=0 THEN 50000
830 IF FIELD(NK)(1 OR FIELD(NK))N(1) THEN GOSUB60100:GOTO800
835 IF NFIELD(NK)(1OR NFIELD(NK))N(2) THEN GOSUB 60100:GOTO800
840 NK=NK+1:GOTO810
845 NK=NK-1:IF NK=0 THEN 800
850 PRINT:PRINT"TRANSFER TO MATCHING(SAME) RECORD NUMBERS"
860 PRINT"OR CONSECUTIVELY FROM END OF EXISTING DATA (M OR C)";
870 INPUT MC$:MC$=LEFT$(MC$,1)
875 IF MC$="X" THEN 50200
880 IF MC$()"M" AND MC$()"C" THEN GOSUB60110:GOTO850
890 REM
900 FORI=1TO24:PRINT:NEXT
1000 REM CHECK IT ALL
1010 PRINT"I SHALL SEARCH THROUGH ";F$(1);" ON DEVICE ";DV$(1):PRINT
1020 PRINT"FINDING ALL RECORDS IN WHICH";FDLB$(OFIELD,1);" = ";CT$
1030 PRINT:PRINT" AND SHALL THEN":PRINT
1040 IF NFIELD(1)=99 THEN 1070
1045 FOR I=1TONK
1050 PRINT"TRANSFER ";FDLB$(FIELD(I),1);" OF ";F$(1);" TO ";
1060 PRINTFDLB$(NFIELD(I),2);" OF ";F$(2);" ON DEVICE ";DV$(2):NEXT
1065 GOTO1090
1070 PRINT"TRANSFER ALL FIELDS OF THAT RECORD IN ";F$(1);" TO ";
1080 PRINTF$(2);" ON DEVICE ";DV$(2)
1090 PRINT:PRINT"TRANSFER TO BE DONE TO ";
1100 IF MC$="M" THEN PRINT"SAME RECORD NUMBERS IN ";F$(2)
1110 IF MC$="C" THEN PRINT"NEXT EMPTY RECORD IN ";F$(2)
1120 PRINT:PRINT:PRINT
1200 REM
1210 PRINT:PRINTTAB(25);"IS THAT CORRECT ";:INPUT Q$
1220 IF LEFT$(Q$,1)="Y" THEN 2010
1230 CLOSE:GOTO50200
1240 REM
2000 REM SEARCH FILE 1
2010 CLOSE:FOR CH=1TO2:DEV DV$(CH):OPEN F$(CH),P$(CH),CH:NEXT
2020 INDEX(1)=BODF(1):TK=0
2030 REM
2100 FIND CT$,1:IF INDEX(1))=1E8 THEN CLOSE:GOTO3510
2110 RPTR=(INDEX(1)-BODF(1))/RL(1)+1:RPTR=INT(RPTR)
2120 RX=BODF(1)+(RPTR-1)*RL(1)
2130 INDEX(1)=RX+FP(OFIELD,1):INPUT%1,FC$(OFIELD,1)
2140 TL=LEN(CT$):IF LEFT$(FC$(OFIELD,1),TL)()CT$ THEN 2100
2150 REM
2160 TK=TK+1
2170 REM
2180 REM
2200 REM GET OLD RECORD
2210 FOR I=1TON(1):INDEX(1)=RX+FP(I,1)
2220 INPUT%1,FC$(I,1)
2230 IF LEN(FC$(I,1))(FL(I,2)THEN FC$(I,1)=" " +FC$(I,1):GOTO 2230
2240 NEXT
2250 REM
2300 REM DETERMINE RECORD IN FILE 2
2310 IF MC$="M" THEN NPTR=RPTR:GOTO 2510
2320 IF MC$()"C" THEN 50000
2330 NPTR=(EODF(2)-BODF(2))/RL(2)+1
2340 IF NPTR()INT(NPTR) OR NPTR)RN(2) THEN ER=4:GOTO 30000
2350 REM
2500 REM PRINT TO FILE 2
2510 NX=BODF(2)+(NPTR-1)*RL(2)
2515 IF NFIELD(1)=99 THEN 2700
2520 REM
2525 FOR I=1TO NK
2530 IF LEN(FC$(FIELD(I),1))=FL(NFIELD(I),2) THEN 2610
2540 IF LEN(FC$(FIELD(I),1))(FL(NFIELD(I),2) THEN 2590
```

Continued

```
2550 L=LEN(FC$(FIELD(I),1))
2560 IF RIGHT$(FC$(FIELD(I),1),1)<>" "THEN ER=5:GOTO 30000
2570 FC$(FIELD(I),1)=LEFT$(FC$(FIELD(I),1),L-1):GOTO 2530
2580 REM
2590 FC$(FIELD(I),1)=" "+FC$(FIELD(I),1):GOTO2530
2600 REM
2610 INDEX(2)=NX+FP(NFIELD(I),2)
2620 PRINT%2,FC$(FIELD(I),1)
2630 NEXT I
2640 GOTO2800
2650 REM
2700 FOR I=1TON(2)
2710 INDEX(2)=NX+FP(I,2)
2720 PRINT%2,FC$(I,1):NEXT
2730 REM
2800 IF INDEX(2)<=EODF(2)THEN 3010
2810 EODF(2)=NPTR*RL(2)+BODF(2)
2820 INDEX(2)=9:PRINT%2,EODF(2)
2830 REM
3000 REM PRINT CHECK AND GO BACK FOR MORE
3010 IF TK>1 THEN 3060
3015 PRINT#PD,"------------------------------------------------------------"
3020 PRINT#PD, TAB(5);"FROM FILE: ";F$(1); TAB(35);"TO FILE: "; F$(2)
3025 PRINT#PD, TAB(5);"WHERE ("; FDLB$(OFIELD,1); ") IS "; CT$; ")) ";
3030 IF NFIELD(1) = 99 THEN PRINT#PD,"ALL FIELDS TRANSFERRED" :GOTO3050
3035 PRINT#PD:FORI=1TO NK:PRINT#PD,TAB(10);
3040 PRINT#PD, "FROM FIELD No:";FIELD(I); " TO FIELD No:"; NFIELD(I)
3045 NEXTI
3050 PRINT#PD,TAB(5);"FROM RECORD NUMBER:"; TAB(35);"TO RECORD NUMBER:"
3060 PRINT#PD,"#"; MID$(STR$(TK),2); TAB(10); RPTR; TAB(40); NPTR
3070 IF INDEX(1)<EODF(1) THEN 2100
3080 PRINT#PD:IF INDEX(1)<EODF(1)THEN 2100
3090 REM
3500 REM FINISHED
3510 PRINT:PRINT:IF TK>0 THEN 3550
3520 PRINT"??? COULD NOT FIND ";CT$; " IN FIELD";FIELD; "OF ";F$(1)
3530 PRINT:PRINT
3540 REM
3550 PRINT"JOB COMPLETED":PRINT
3560 PRINT#PD,"------------------------------------------------------------"
3570 CLOSE:GOTO700
3580 REM
30000 REM DISC ERROR
30010 GOSUB 60100
30020 ON ER GOTO 30030,30040,30050,30060,30070
30030 PRINT:PRINT"NOT MASTER FILE NAME":PRINT:GOTO 50100
30040 PRINT:PRINT"INVALID PASSWORD":PRINT:GOTO 50100
30050 PRINT:PRINT"FILE HEADER ERROR":PRINT:GOTO 50100
30060 PRINT:PRINT"EODF WRONG (OR FILE FULL)-FILE 2":PRINT:GOTO 50100
30070 PRINT:PRINT"FROM FIELD LARGER THAN TO FIELD":PRINT:GOTO 50100
30080 REM
50000 REM ERROR
50010 EL=256*PEEK(11775)+PEEK(11774):ER=PEEK(10226)
50040 PRINT:PRINT"ERROR ";ER;" IN LINE ";EL;" DEVICE ";DV$
50050 REM
50100 CLOSE:PRINT:PRINT"*** AN ERROR HAS OCCURRED ***":PRINT
50110 PRINT"PLEASE CHECK PROGRAM AND DISCS":PRINT
50200 PRINT:PRINT:INPUT "CONTINUE? Y=AGAIN N=MAIN MENU ";Q$
50205 IFQ$="STOP"THEN STOP
50210 IFLEFT$(Q$,1)="N" THEN DEV"A":RUN"DBMSYS","PASS"
50220 RUN
50250 REM
60100 REM ERROR MESSAGE
60110 FORT=1TO3:PRINT"**** DOES NOT COMPUTE ****":NEXT:RETURN
60120 REM
63999 DEV"A":SAVE"DEVEL3","PASS"
```

★ ★ ★

USING A C1P IN A VERY SMALL BUSINESS

By: Bruce Showalter
857 Cedar
Abilene, TX 79601

Bruce's Repair Service is a sole proprietorship that works on small kitchen appliances, power tools, and other small electrical items. Invoices are printed on an IBM Selectric mechanism, interfaced to the C1's serial port.

In previous articles, I've described the hardware. The original machine was a Superboard II. It now has a 32/64 character video and 16K of RAM. Since time is a surplus commodity at Bruce's, there has been no need to upgrade from the cassette I/O to disk. However, the cassette I/O baud rate has been raised to 600.

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

The first major project on the computer was the building of an alphabetical customer file. This was a rather long project because I've been in business since '76, and I didn't get the computer till '80. So a lot of names had accumulated. They were in chronological order only. I began by creating cassette files. Each file had about 600 customers (determined by free RAM), each with an I.D. number, name, telephone no. or address and date of first invoice. These facts I stored as strings in DATA statements.

Each customer is alloted ten lines. The first line contains the I.D. number, and the line number always ends in '9'. I.D. numbers are allocated at the rate of 99 per letter of the alphabet, beginning with 100. So, I.D. numbers 100-199 are for 'A', 200-299 are for 'B', etc. The line numbers are equal to (10 x I.D. number) - 1. Thus I.D. number 100 is found in line number 999. I designated line number 26999 as I.D. number '2700'. This customer's name is 'END OF SECTION X'. The 'X' represents the section indicator number. The first line in the file reads -

```
10 REM CUSTOMER FILE SECTION X
```

After this line and before the DATA statements is the following program:

```
20 PRINT:INPUT"PRINTER (Y/N)";
   A$:PRINT
30 INPUT"FROM, TO";X,Y:PRINT
40 IFA$="Y"THENSAVE
60 RESTORE
70 READD$:IFVAL(D$)=XTHEN90
80 GOTO70
90 FORX=1TOLEN(D$):H=ASC(MID$
   (D$,X,1))
91 IFH<480RH>57THEN93
92 NEXT:PRINT:READD$:GOTO100
93 PRINTD$:READD$
100 IFVAL(D$)=YTHEN130
110 GOTO90
130 PRINT:PRINT:POKE517,0:
    GOTO20
```

The purpose of this program is to allow me to list any part of the file, such as all the K's. I can list the entire file as well. Line 20 gives me the option of printing as well as displaying on the screen. It causes a SAVE command to be executed in line 40 if A$='Y'. Line 30 asks for the beginning and ending I.D. numbers I want. The DATA statement pointer is initialized in line 60, then line 70 begins the search phase. Since all my data is in string format, I compare the VAL of the string to the starting I.D. number entered in line 30 (variable X). If the string's VALue is not equal to X, then the next DATA statement is read and comparison continues.

Finally, if the I.D. number is found, a FOR loop is begun in line 90. The purpose of this loop is to omit the I.D. number when I list the customer data. I.D. numbers are unique in that they have only numeric characters (ASCII values 48-57), no alpha or special characters. If a non-I.D. number is found, the program jumps to line 93. There, the data is listed.

In either case, the program now arrives at line 100 where a test is made for the end of the desired range of I.D. numbers (variable Y, input at line 30). If the end has not been reached, the program repeats from line 90. Otherwise, two blank lines are printed, the SAVE command is cancelled, and the program jumps back to line 20.

At this point, my file contains a list of customers grouped alphabetically by last name. Each section has a collection of A's, B's, C's, etc. I now wanted to sort these sections to produce a file that was truly alphabetized.

To do this, I first took all the A's out of every section and created a single 'A' section. These customers were grouped according to the second letter in their last name. For example, the I.D. numbers for all the Adamses were in the 400-499 range. Allens fell in the range of 1200-1299. Now all I had to do was alphabetize this new 'A' section. The same process was used for the B's, C's, etc.

To accomplish the alphabetical sort, I wrote the following program:

```
199 REM TO EXECUTE SORT, TYPE
    'RUN200'
200 PRINT:INPUT"STARTING STMT.
    #";J
210 X=1:Z=1:DIMF$(160),N(40)
215 PRINT:PRINT"ENTER I.D.#S
    IN SEQUENCE, FOLLOWED
216 PRINT"BY 1st # OF NEXT
    SECTION, THEN 0
220 INPUTN(X)
230 IFN(X)=0THEN 260
240 X=X+1:GOTO220
260 FORY=1TO X-1:RESTORE
270 READD$
280 IFVAL(D$)=N(X-1)THEN330
290 IFVAL(D$)<>N(Y)THEN270
300 F$(Z)=D$:Z=Z+1
310 READD$
315 IFLEN(D$)>LEN(STR$(N(Y)))
    THEN325
320 IFVAL(D$)>N(Y)ANDVAL(D$)
    <=N(X-1)THENNEXTY
325 F$(Z)=D$:Z=Z+1:GOTO310
330 SAVE
340 PRINT:INPUT"READY TO TAPE.
    ENTER <SPACE>";A$
345 PRINT:PRINT
350 FORV=1TOZ-1
352 FORT=1TOLEN(F$(V)):H=ASC
    (MID$(F$(V),T,1))
354 IFH<480RH>57THEN360
356 NEXTT:GOTO400
360 PRINT" ";J;"DATA";CHR$(34)
    ;F$(V):J=J+1
370 NEXTV:END
400 IFRIGHT$(STR$(J),1)="9"
    THEN420
410 J=J+1:GOTO400
420 F$(V)=STR$((J+1)/10)
425 F$(V)=RIGHT$(F$(V),LEN
    (F$(V))-1)
430 GOTO360
```

I LOADed in lines 10-130, then lines 199-430, and lastly the DATA statements 999-27000. Then I started a scratch tape section by SAVEing lines 10-130, then doing a RUN200. My very first statement number (variable J) was 999, of course. Prior to this run, I had manually written a list of customer I.D. numbers in the sequence that would alphabetize them. This sequence was next entered in the computer at line 220. When I'd entered all the 'Aa' numbers, I then entered 200 and 0. If there were no 'Aa' customers, or if group 'Aa' had only one customer, or the names were alphabetized by coincidence, then I simply LISTed statements 999-1998.

Now I have a list of I.D. numbers, variable N(X), which is the desired sequence of customers in this group. A FOR loop begins at line 260. Starting at the first DATA statement, the program looks for a match to the first I.D. number, N(1). When a match occurs, a new list (F$) is begun. F$(1) contains the first I.D. number I want. The next DATA statement is read. If its VALue is that of another I.D. number, the program goes back to the top of the loop. If not, READing continues in line 310. The program will now find the customer data belonging to the current I.D. number. This is stored in subsequent F$ locations. This process repeats until the first I.D. number of the next group, N(X-1), is reached.

I now have a list, F$(Z), containing one alphabetized group of customers. Turning on the tape recorder, I SAVE this group by program lines 330-430. Line 340 is simply a pause in the program that

gives me time to turn on the recorder. The loop beginning in line 350 controls listing of the F$ items. Another loop, starting at line 352, tests to see if the data is an I.D. number. If not, the desired line number (J) is recorded, followed by the word DATA, CHR$(34) - a quote mark - , and the F$ item. The line number (J) is incremented and the loop started at 350 repeats.

But suppose an I.D. number is found in line 354. The program then jumps to line 400. Here, a routine is performed on the line number J so that the ten-line-per-customer allocation is preserved.

The program ends when the list of F$ items is exhausted. Now I switch off the recorder and RUN200 again. This time, J equals the starting line number of the next group, 'Ab' (1999). I repeat these group sortings till I reach the end of section A. The scratch tape I have is now a recording of section A, preceeded by the printing or listing program, lines 10-130. There are pauses on the tape between groups where I switched the recorder on and off. When reLOADed into the computer, I find a cohesive program with DATA statements containing alphabetized customers. This is now SAVEd onto a permanent tape as CUSTOMER FILE SECTION A.

As you can imagine, this was a rather inefficient process. Most of the data was keyed into the computer at least twice. With more memory and disk I/O, a lot of manual work could have been bypassed. But I have more time than money, so I made do with what I have. The net result was still the same - a neatly printed alphabetical customer list. The program could also be used to print mailing labels, if the file contained addresses.

I found that reading invoices one at a time and keying in the data was too tedious. So I first made an audio tape, reading each invoice aloud. Then I keyed in the data while listening to this tape being played back. I used the Pause switch on the recorder microphone to stop the tape between items.

Now we come to the Invoice Program which I use nearly every day. Although first written in '81, it has undergone some changes to make it more efficient.

INVOICE PROGRAM

```
1 REM  INVOICE  FEB. 1981  LAST STMT. # 1040
10 X=1:T=0:POKE15,80
20 PRINTCHR$(127):PRINT"NOTE - 1 appliance per invoice"
30 INPUT"INV. #";D$:PRINT
40 INPUT"CUST. NAME";C$
50 INPUT"PHONE/ADDR. 1";P1$
60 INPUT"PHONE/ADDR. 2";P2$:PRINT:PRINT
70 INPUT"BRAND NAME CODE";BN:IFBN=0THEN100
90 ONBNGOTO700,710,715,720,730,740,750,760
100 INPUT"APPLIANCE NAME CODE";AN:PRINT:IFAN=0THEN130
110 IFAN>12THENON(AN-12)GOTO870,880,890
120 ONANGOTO770,780,790,800,810,820,830,835,840,850,855,860
130 INPUT"DESCRIPTION CODE";D(X)
140 OND(X)GOTO900,910,920,930,935,940,945
150 INPUT"PARTS CODE";P(X):IFP(X)=0THENP$(X)=" ":GOTO180
170 ONP(X)GOTO949,950,960,970,980,990,1000,1010,1020,1030,1040
180 INPUT"AMOUNT (XXXX)";A(X)
181 AC$(X)=RIGHT$(STR$(A(X)),2)
182 AD$(X)=RIGHT$(STR$(INT(A(X)/100)),2)
190 X=X+1:PRINT:INPUT"IS THAT ALL (Y/N)";E$
200 IFE$<>"Y"THEN130
210 Y=X-1:F=0:T=0:PRINTCHR$(127)
220 PRINT"Bruce's Repair Service
230 PRINT"857 Cedar":PRINT"Abilene, TX 677-4189
250 PRINT"INV. # ";D$;
260 IFPOS(L)<42THENPRINT" ";:GOTO260
280 PRINT"# ";D$:PRINT:PRINTC$;
300 IFPOS(L)<42THENPRINT" ";:GOTO300
320 PRINTC$:IFLEN(P1$)>1THENPRINTP1$;:GOTO330
325 GOTO350
330 IFPOS(L)<42THENPRINT" ";:GOTO330
340 PRINTP1$
350 IFLEN(P2$)>1THENPRINTP2$;:GOTO360
355 GOTO390
360 IFPOS(L)<42THENPRINT" ";:GOTO360
380 PRINTP2$
390 PRINT:PRINT:PRINTBN$;AN$;
400 IFPOS(L)<42THENPRINT" ";:GOTO400
420 PRINTBN$;AN$:PRINT
425 FORX=1TOY
430 PRINTDE$(X);P$(X);
440 IFPOS(L)<31THENPRINT" ";:GOTO440
460 PRINTAD$(X);".";AC$(X);
470 IFPOS(L)<42THENPRINT" ";:GOTO470
490 PRINTDE$(X);P$(X);
500 IFPOS(L)<72THENPRINT" ";:GOTO500
520 PRINTAD$(X);".";AC$(X)
530 T=T+A(X):NEXT:PRINT
551 TC$=RIGHT$(STR$(T),2):TD$=RIGHT$(STR$(INT(T/100)),2)
560 PRINTSPC(30);"------";SPC(35);"------"
570 PRINTSPC(21);"TOTAL     $";TD$;".";TC$;
580 IFPOS(L)<62THENPRINT" ";:GOTO580
600 PRINT"TOTAL     $";TD$;".";TC$
604 PRINT:PRINT:PRINT"'Your recommendation is"
605 PRINT"my best advertisement.'"
610 IFF=1THENPOKE517,0:POKE15,72:END
620 PRINT:PRINT:INPUT"ENTER <SPACE> TO PRINT";PR$
640 PRINTCHR$(127):SAVE:F=1:T=0:GOTO220
700 BN$="Conair ":GOTO100
710 BN$="Gen. Elec. ":GOTO100
715 BN$="HAM.-BEACH ":GOTO100
720 BN$="Mr. Coffee ":GOTO100
730 BN$="Norelco ":GOTO100
740 BN$="Sunbeam ":GOTO100
750 BN$="Toastmaster ":GOTO10
760 INPUT"WHAT BRAND";BN$
765 BN$=BN$+" ":GOTO100
770 AN$="Blender":GOTO130
780 AN$="Blow Dryer":GOTO130
790 AN$="Broiler-Oven":GOTO130
800 AN$="Clock-Radio":GOTO130
810 AN$="Coffeemaker":GOTO130
820 AN$="Elec. Fan":GOTO130
830 AN$="Elec. Heater":GOTO130
835 AN$="Food Proc.":GOTO130
840 AN$="Iron":GOTO130
850 AN$="Desk/Table Lamp":GOTO130
855 AN$="Floor/Hang. Lamp":GOTO130
860 AN$="Mixer":GOTO130
870 AN$="Percolator":GOTO130
```

Continued

```
880 AN$="Shaver":GOTO130
890 IPUT"WHAT APPL.";AN$:GOTO130
900 DE$(X)="Install ":GOTO150
910 DE$(X)="New ":GOTO150
920 DE$(X)="Repair ":GOTO150
930 DE$(X)="Labor":GOTO150
935 DE$(X)="Lubricate ":GOTO150
940 D$(X)="REPLACE ":GOTO150
945 INPUT"WHAT DESC.";DE$(X)
947 DE$(X)=DE$(X)+" ":GOTO150
949 P$(X)="Brushes":GOTO180
950 P$(X)="Connection":GOTO180
960 P$(X)="Cord":GOTO180
970 P$(X)="Cord and Plug":GOTO180
980 P$(X)="Fuse":GOTO180
990 P$(X)="Plug":GOTO180
1000 P$(X)="Socket":GOTO180
1010 P$(X)="Switch":GOTO180
1020 P$(X)="Thermostat":GOTO180
1030 P$(X)="Warming Element":GOTO180
1040 INPUT"WHAT PART";P$(X):GOTO180
```

```
Bruce's Repair Service
857 Cedar
Abilene, TX 677-4189
INV. # 830524

SAMPLE, Just A.
654-3210

Toastmaster Elec. Fan

New Switch              1.50
Labor                   7.00
                       ------
          TOTAL       $ 8.50

'Your recommendation is
my best advertisement.'
```

```
# 830524

SAMPLE, Just A.
654-3210

Toastmaster Elec. Fan

New Switch              1.50
Labor                   7.00
                       ------
          TOTAL       $ 8.50
```

In a nutshell, the program asks for the information, then formats and prints it. Rather than having carbon copies, the program prints a duplicate invoice alongside the original on a blank piece of 8 1/2 by 5 1/2 typing paper. When cut in half, I get two 4 1/4 by 5 1/2 copies. The original has a heading printed by lines 220-230. There's also a slogan printed at the bottom of the original, as shown in lines 604-605.

Line 10 initializes two variables and sets the terminal line length at 80. Line 20 clears the screen and displays a reminder. Next comes the invoice number (D$), the customer name (C$), and phone number or address (P1$, P2$). If I don't have an entry for P1$ or P2$, I simply type <SPACE> <RETURN>.

There are seven brand names which are most common in my business. To select 'Gen. Elec.' for example, I enter 2 in line 70 (BN). If I want to skip the brand name, I enter 0. An 8 will put me in line 760 where the program allows me to key in any other brand name.

A similar procedure is used for the name of the appliance.

If I enter 14 for AN in line 100, then AN$ = 'Shaver'. A 15 allows me to key in any other name in line 890. Then there's the description, D(X), in line 130 and the parts, P(X), in line 150. One appliance may have more than one description and/or parts. So these two variables are subscripted.

At last we get to the money. The amount for a part or description is entered as an integer in line 180. Lines 181-182 reformat this amount as two strings: AC$ for cents and AD$ for dollars. Again, these variables are subscripted to allow for multiple amounts.

Line 190 tests to see if all the invoice data has been entered. If not, the program goes back to line 130. A proof run begins at line 210. Three variables are initialized and the screen is cleared. Since my duplicate invoice will appear on the second half of the page, the print position must be correctly placed. This is done by a GOTO loop, such as in line 260. When the print position equals 42, the program continues to line 280. Otherwise, a SPACE will occur. I use this procedure because my Selectric doesn't have a TAB function which the computer can control. Line 425 begins a FOR loop to display the subscripted variables. Also, in line 530, the various amounts are added into variable T. Line 551 converts this variable into strings for dollars and cents as was done with amounts before. Total lines are drawn in line 560. Then the totals are displayed, followed by the slogan.

Line 610 tests variable F which is set to 1 when the invoice has actually been printed. Since this is only a proof run, F=0. So line 620 asks me to enter a <SPACE> if I'm ready for a print run. Then in line 640, the screen is cleared, the SAVE mode is engaged, the variable F is set to 1, and the variable T is cleared. Lines 220-610 are then executed once more. But this time, since F=1, the SAVE mode is cancelled in line 610 and terminal width is restored to 72. Enclosed is a sample invoice so you can see the finished product.

I've enjoyed developing this software to assist me in my business. Although somewhat crude in comparison to many business systems, it has proved to be highly cost-effective and well-suited to my needs.

LETTERS

ED:

Many thanks to Mr. Hendrix for his excellent articles on Microsoft BASIC. I am concerned, however, with the example given towards the end of the May article (Vol. 4 No. 5: Internal Format used by Microsoft BASIC). I tried the example on both Superboard II and 230E (65U v.1.42) and on both of them it functioned in the opposite way to his detailed description!

The reasoning seems fine until the paragraph commencing, "Now consider the task....." where he goes on to say "Since the step size was zero, BV being exactly equal to the limit of zero will cause the loop to repeat". But the function of this loop is to continue until BV is equal to zero, then to end! In the example given by Mr. Hendrix, listing 1 below, the loop will continue only if the INPUT string is "QUIT". For any other INPUT string BV

is set to zero in line 130, thus is equal to the loop limit and the loop terminates.

I believe the answer to the problem is found earlier in the article, "there is a byte set according to +ve, -ve or zero step, this is also used to determine if the loop variable has passed its limit". It would seem that with step +ve the system tests whether the loop variable is greater than the limit, with step -ve it tests whether the variable is less than the limit and with step 0 it tests whether the loop variable is equal to the limit. Thus a loop FORX=0TO0STEP0:NEXT will not loop forever, it will loop only once. The loop FORX=1 TO0STEP0:NEXT will loop forever since X remains at its initial value 1 and never reaches zero.

Listing 2 shows how the example may be made to operate as apparently intended. The logic for A$<>"QUIT" may be any non-zero value: the -ABS ensures that BV is either -ve or zero. If BV is set to -1 as with our usual Microsoft BASIC then normal STEP 1 will increment BV to 0 and the loop enters what would appear to be the final circuit (since BV is not yet greater than the loop limit). Whenever A$<>"QUIT" this state of affairs exists and the loop is always just one step below the limit until A$ is entered as "QUIT". Then BV is logic 0 (the -ABS does not affect this) and line 240 (NEXT) adds 1 to BV and since 1 is greater than zero the loop terminates.

Listing 3 shows an alternative method and uses the "forever" loop FORBV=1TO0STEP0. The logic for A$<>"QUIT may be any non-zero value and thus BV is set to a non-zero value, adding step 0 leaves BV at the same value and the loop continues since the condition for termination is BV=0. When "QUIT" is entered then logic for A$<>"QUIT" is zero and adding step 0 leaves BV at 0 and the loop terminates.

This method can be useful for leaving a loop when a required value is reached but extra care is sometimes necessary - for example when you are searching through an array for a matching value. You cannot rely on the loop value as a counter since it is always being reset to -1 or 0.

```
100 REM  *** LISTING 1 ***
110 FOR BV = 0 TO 0 STEP 0
120 INPUT A$
130 BV = (A$ = "QUIT")
140 NEXT
150 STOP

200 REM *** LISTING 2 ***
210 FOR BV = 0 TO 0
220 INPUT A$
230 BV = -ABS (A$<>"QUIT")
240 NEXT
250 STOP

300 REM *** LISTING 3 ***
310 FOR BV = 1 TO 0 STEP 0
320 INPUT A$
330 BV = (A$<>"QUIT")
340 NEXT
350 STOP
```

Colin Law
Auckland, New Zealand

* * * * *

ED:

PEEK(65) always has lots to interest both business and hobby users. David L. Kuhn's letter (May issue) about line feeds in OS65-D and how they affect a Radio Shack printer struck a responsive chord. I have a C2-OEM, H19 Heathkit terminal (now masquerading as a VT100 thanks to a new PROM), and a Radio Shack Daisy Wheel II printer. Since the terminal is smart, I can make it behave like the printer (auto line feed upon receipt of carriage return code) with the proper escape sequence. Then I get rid of the line feed in either 65-D or 65-U with a POKE 2683,0.

I managed to find the right spot in WP-1 and got rid of the line feed in its print routine, and changed the MEMORY SIZE? query to MEM SIZE? so I could scavenge three characters for my escape sequence. WP6502, Version 1.2 (1979), has proved completely recalcitrant -- it does not use the OSI print driver routine, and scrolling through the code with the monitor has yet to reveal it to me (I was not able to achieve this revelation from the dealer from whom I bought it or from Dwo, either, despite several attempts).

Has anyone solved this problem? I have Version 1.3 also, which allows changing the value of the line feed as part of the configuration routine. This doesn't work so well, however, because the same print routine is used to VIEW a file on the terminal as to print it on the printer; thus, if you set the line feed to zero, when you go to view a file on the screen the terminal simply writes over the same line -- fun to watch, but not very useful. Maybe

there's no solution to the problem, but if anyone has any ideas, please share them!

William E. Shawcross
Cambridge, MA 02138

* * * * *

ED:

I am having a problem in getting the enclosed program to run. I hope that your readers may help and find the problem. Line 10 dimensions A$ array. Line 20 turns off the CTRL/C function so that line 30 and line 40 can wait for the space bar to be pressed. Line 50 turns on the CTRL/C function. Line 60 sets the load flag, and lines 70 - 90 input from the tape to the A$ array.

```
 10 DIMA$(100)
 20 POKE530,1
 30 POKE57088,253
 40 WAIT57088,17,255
 50 POKE530,0
 60 POKE515,128
 70 FORX=1TO100
 80 INPUTA$(X)
 90 NEXT
100 POKE515,0
```

The program runs till the input ? appears but does not read the tape. If I delete lines 20 thru 50 the tape is read. And lastly, line 100 turns off the load flag. My system is an early C1P with 16K memory and a cassette. This letter is being typed on a word processor program that I hope to have use the above routine.

Jack Kramer
N. Massapequa, NY 11758

* * * * *

ED:

I always read PEEK(65) from cover to cover and always find it extremely interesting. Of course, having just got my 8 inch disk system working, I must go back and read all the articles I didn't pay much attention to before.

I wanted to comment on indirect files (read article by Charles Stewart, May 83). Being new to the disk system, I found myself writing a program and not able to make it fit into the track space alloted, then noting that I had typed my program over the directory program. Well, it was either type my program over or eliminate the directory program. Both alternatives are very tedious and so the thought of the indirect file came to mind. So I typed list- (the last item of my program), SHIFT-K (REPEAT-K on my machine) and RETURN. Then a SHIFT-M and a NEW cleared the the memory. When I typed CTRL-X I got back my program plus a few lines which were not overwritten by my program. What a saving.

Gerald Van Horn
Junction City, OR 97448

* * * * *

ED:

HELP! My system is a 1979 vintage C1P-MF with an MPI model B51 5-1/4" floppy drive. A couple months ago the infrared sensor on the track zero sensing assembly went bad (disk head would not move toward track zero). The sensor is a small infrared detector (I think it's a diode -- it has two leads).

Now, my problem. I have been unable to find out what this part is, and where to obtain one. A letter to MPI requesting information produced the name of a local distributor. The local distributor (who has a $100 minimum order requirement) supplied me with a part number for the entire track zero sensing assembly (which costs about $6), and suggested I order it from PRIORITY ONE ELECTRONICS (who has a $15 minimum order requirement).

If nothing else, I refuse to go this route on principle! Why should I shell out $15+ for a $.50 part? Incidentally, I have tried to use another larger sensor (which cost 3/$1), but haven't been able to mount it in place. Does anyone know what this part is and where it can be obtained?

Thanks, and keep up the good work at PEEK(65).

LeRoy R. Sorem
New Brighton, MN 55112

LeRoy:

As far as we know, one MUST buy the subassembly.

Al.

* * * * *

ED:

Here is a change in the last Aardvark issue to allow use on C2-4P computers and others with the 540 board. If the line below the input line is not needed change the 55359 of line 60050 to 55295 it might be necessary to save values of X,Y and Z when used with an existing program. This was a

good program and needs to be shared. We'll miss Aardvark.

```
60000 DIM(64.32):C=53248:
      D=53311:X=0:Y=1
60010 FORA=CTOD:X=X+1
60020 REM
60030 Z=PEEK(A):S(X,Y)=Z
      IFX=64THEN60050
60040 NEXTA
60050 Y=Y+1:C=C+64:D=D+64
      X=0:IFD<55295GOTO60020
60060 SAVE:FORY=1TO32:
      FORX=1TO64
60070 PRINTCHR$(S(X,Y));:
      IFX=64THENPRINT
60080 NEXTX:NEXTY:POKE517,0:
      END
60090 REM FROM AARDVARK 4/82 &
      8/82 PAGE 8
60100 REM MOD FOR C2/4P BY
      B. GROOME 1/5/83
```

Bob Groome
Bethlehem, PA 18018

* * * * *

ED:

In the April issue, Donn Baker presented a pair of programs designed to insert and extract a volume I.D. from OS65D diskettes. His approach required that the volume I.D. be the first entry in the directory. The 'DIR' program he modified was the unsorted version.

A similar modification to the 'sorted by track' version, extracting element NM$(0), will accomplish the same purpose no matter where in the directory track the volume I.D. entry is located. If lines 20090 and 20290 are deleted from the 'CREATE' utility, the volume I.D. can be easily entered.

Cyrus N. Wells
Las Vegas, NV 89101

* * * * *

ED:

I have just finished reading the May 83 issue of PEEK(65) and am still amazed that there are so many OSI users. I have owned my C4P MF for two years and knew of only two others.

Anyway, I want to thank you for the much needed information in your articles. Keep it up!

Also in the issue, a letter from A. J. Smith asked for information on the keyboard routine in OS65D v3.3. I use it often in my programming. The location of the routine is the same as v3.2 (252B hex), only the storage location has changed. In v3.2, the character was stored at 9815 decimal; in v3.3, it is stored at 9059 decimal. I hope that takes care of his problem.

Since I just found out that there are people with OSI equipment out there, I only need names and addresses of hardware/software houses now.

Norman Thorsen
Poulsbo, WA 98370

* * * * *

ED:

I was very excited to learn about the OSI users' net which meets on 7.229 MHz on Sunday afternoons (Ham radio and computers are my major hobbies.). I do have one question, however: where the heck are you guys???!? For the past 4 weeks I've tuned to the net frequency, but haven't heard a thing. I don't think I am the only one left out, so please post the proper time and frequency.

Paul Elder
N2CUY

* * * * *

ED:

Anyone know why programs of more than 4K in size have difficulty in loading when system is cold? I experience no problem with 6K/8K size programs after system has run for ten minutes.

Also, does anyone know why plot BASIC "PLOT 7" instruction does not work for C4PMF system? I get ERROR #9 when "PLOT 7" instruction is executed. All other plot BASIC program demo's work except when "PLOT 7" is executed.

C.D. Lombard
Olympia, WA 98506

C.D.:

1. Check power supply output.

2. Error #9 usually indicates a bad disk. Make a new copy of your master disk or, if master is also bad, return to your dealer for a "refresh".

Al.

* * * * *

ED:

I have recently become aware of a modification to the C1P which will increase the baud rate from 300 to 600. However, this is all I know about it and I was wondering if you or a fellow subscriber could tell me either how to do it, or where to find out more about it.

No complaints about PEEK (65), but I would like to see more written on the C1P.

Don Bruechert
Manitowoc, WI 54220

* * * * *

ED:

I own a C1P hoping to program in machine code but have had problems locating an assembler/disassembler. If you could suggest a program, I would apreciate it.

Robert Crane
Vista, CA 92083

Robert:

Check with your dealer, it should be available. See letter from Mr. Cobb in this issue.

Al.

* * * * *

ED:

The following is an experience I had with the OSI Assembler that I thought readers might be interested in.

Since I expanded my C1P to 16K, my OSI Assembler can handle much larger source programs. One of my source programs was in two pieces and I wanted to put them back together. However, I had one major problem. Line numbers of both pieces started with 10 and incremented by 10.

Just to see what would happen, I loaded the second part as a basic program. Next, I ran a re-number program (which was intended to re-number basic programs) against it starting with 3010 and incrementing by 10. Then, I saved the re-numbered version on tape. I noticed that the horizontal spacing was messed up making it a little hard to read.

After re-loading the assembler, I loaded the first part and then the re-numbered second part. The Al assembly only had one error which was easily fixed (a space between the OP code and OPERAND had been dropped).

This procedure saved me from re-typing 127 lines of source code. It could also be used to combine two or more source programs into one.

David D. Cobb
Midland, MI 48640 * * * * *

ED:

I have an OSI-C4P. Is there any way I can learn COBOL on my computer by modifying it some way?

T. E. Shoemaker
Beltsville, MD 20705

CALENDAR ITEM

CP/M'83 EAST

BOSTON

SEPTEMBER 29-OCTOBER 1, 1983

CP/M'83 East will be held Thursday - Saturday, September 29 - October 1, 1983 at Boston's Hynes Auditorium. Show Hours are 10.30 am to 5.30 pm daily. CP/M'83 is an International Conference and Exposition for the CP/M industry and CP/M users featuring manufacturers, independent software developers, OEMs, venture capitalists, software publishers, distributors and dealers. The Exposition portion of the event is the largest presentation of CP/M based hardware and software ever assembled. The Conference Program, with nearly one hundred sessions, will include noted leaders from the software industry, including Garly Kildall, Sol Libes, Stewart Alsop, Benjamin Rosen and dozens of others. Seminars will begin at 11.00 am and conclude at 6:00 pm each day. Admission is $10.00 for a one-day Exhibitits-only ticket, or $25.00 for a three-day exhibits and Conference ticket. For more information, call or write Northeast Expositions, 822 Boylston St. Chestnut Hill, MA 02167. Telephone: 800-841-7000 or 617-729-2000 (within Mass.).

ERRATA!

June 1983, page 12. The article on the Alaskan Pipeline Problem. by: Robert Van Clampitt.

List #1, line 2300 should be the same as line 2300 in List #2, i.e. B(Z) = 15 NOT B(Z) = 150.

PRODUCT INFORMATION

A company called Grafix has designed an 80 character wide video board/floppy controller for the OSI bus. The board used a 6545 video controller chip and can replace the OSI 540 board. I have a working prototype of the Grafix SEB-3 video board and can verify it works well.

Grafix has since discontinued the manufacturing of OSI compatible boards. Production of the video boards was started, but halted before completion. If there is enough interest in this board, they will be completed and sold bare with manual for $59. Anyone interested please contact me.

Earl Morris
3200 Washington
Midland, MI 48640

AD$

FOR SALE: Superboard II, custom enclosure, 8K RAM, 8K memory/PIA board, Mother Board, C1S ROM, 2mhz clock, other mods, extensive documentation, manuals, over 30 programs. A real BARGAIN! For complete details, send S.A.S.E. to: B. Garland, 205 Maplebrook Drive, R.D.3, Ebensburg, PA 15931.

* * * * *

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

* * * * *

FOR SALE: 24K OSI C4P-MF w/D&N disk switch, BMC green phosphor monitor, CAT modem, 65D 3.3 & 3.2 w/all manuals, smart terminal program, MDMS, plus other useful software, books, manuals, etc. Everything in beautiful condition and working perfectly. $1150 or best offer. Paul Elder, 273 High St.,Nutley, NJ 07110. Call (201) 667-1331 or (201) 667-0123.

* * * * *

FOR SALE: 48K Challenger with terminal. System in excellent condition and is available for $1,200.00. For more information, contact Mike Guidry (318) 988-1300 during office hours.

* * * * *

FOR SALE: Shugart SA-400 SS/SD 35 Track bare disk drive with documentation. Works fine. Asking $175 or will trade for a similar 40 track drive. Jim McConkey, 7304 Centennial Rd., Rockville, MD 20855, (301) 926-6059.

* * * * *

OSI C8P Dual 8", 48k, no crt or printer, $1,000.00. John K. McDonald, 10116 SE Stanley Aveue, Portland, OR 97222, Phone 503-774-0077.

* * * * *

FOR SALE: OSI 48K C3-OEM 1/2 mhz with a Hazeltine 1500 terminal. Machine is in mint condition (used as a home computer). Comes with original OSI 65U & 65D diskettes. Also, over 30 diskettes, most contain software. Price includes all PEEK(65) & OSIO Newsletters along with miscellaneous BASIC books. First $2800.00 takes all. Please call Pat Clusman at 414-923-1526 after 5 PM CST.

* * * * *

C2-4P in new C4 case, with 32K Ram, OSI Controller Board, 2 MP151 disks in matching case as C4, OS65D 3.2 and 3.3, D&N 24K RAM board, bare, and 8 slot backplane.

ROM Basic has been retained and read/write to cassette is selectable at 300, 600, and 1200 baud. I am asking $1200 firm. Charles F. Merica, Rt. 3, Box 450, Covington, VA 24426, (703) 747-3193.

* * * * *

FOR SALE--I have too much computer equipment and will sell any of the following four items:

1. C2-4P cassette based computer, which includes: 16K on model 525 bd., model 502 CPU bd. with RS-232 output, CEGMON monitor ROM and sockets to add 8K more of RAM, model 540 Video bd., model 542 Polled Keyboard, Sams schematics, printer cable, wooden case. $225

2. Sanyo 15 in. black & white monitor. Was $250 now $100.

3. Complex Sound Generator bd. for OSI bus with software on tape. Software includes programs for designing thousands of different sounds, music, and sound effects. Was $199, now $40.

4. Assembler/Editor and Extended Monitor on tape (for OSI C2 computers). Was $50, now $25.

Add 10% to the items for shipping. All the above items are with documentation. Irvin Johnson, 1214 Ravine St., Janesville, WI 53545, phone (608) 752-9178.

* * * * *

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:

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