PEEK [65]
The Unofficial OSI Journal

October/November 1986
Volume 7 Nos. 10 & 11

Column One

Yes, PEEK[65] is late again. There's a good reason for it. Several things have been happenning which are so important that I held up publication as long as possible in order to get the story out to you as quickly, completely, and accurately as I could. As I did with the Summer issue, this issue has been expanded in length to make up for the delays.

Here we go again, boys and girls. ISOTRON is in some kind of financial difficulty. The company headquarters in Connecticut isn't talking, so all I can do is pass on what I've heard. Apparently, one of the major European backers has withdrawn his support from the company. In October, the factory was completely shut down for some time and most employees were laid off. However, enough dealers placed pre-paid orders to re-open some of the assembly lines. I don't know what the current situation is.

This news came as a big shock to me. All year, the company has been saying that the new 700 series systems were selling extremely well and that the Portland board was making significant sales in the 8-bit line. They hired a west coast sales representative not too long ago in order to expand the dealer base. None of these things sound like a company struggling to keep afloat.

What does this mean for current owners? I don't know. We've always been orphans, even in the best of times. One dealer has speculated that this could well spell the ultimate demise of the 6502-based systems from the factory. I have some doubts about that. Throughout its history, the only reason Ohio Scientific has survived is that there is a core group of dealers out there who have written their own vertical market applications who just keep pumping out C3's (or whatever incarnation of that machine happens to be available). Whenever the company has gone astray, it has been this core of business that has kept them afloat long enough to seek out new ownership and new money.

The bright spot on the horizon is DBI, Inc. As I write this they are finalizing their 65816-based CPU board. The Denver Boards have long been held in high esteem for their speed, versatility, and reliability. They'll be advertising this board in the near future. Just before the dam broke in Connecticut, DBI signed an agreement with ISOTRON for distributing OS-65U. I don't have the details on that agreement, but I do know that DBI has completely disassembled 65U and am confident that they can and will support whatever version goes out their door.

Unfortunately, that's not the end of the bad news. CompuServe has decided not to continue OSI SIG, our bulletin board on that system. We've never been exactly overwhelming the system with our presence there, although we have usually been able to hold our own with a lot of the smaller SIGs. After 5 years, its going to be very hard to say good-bye to OSI SIG. I always tried to make it as easy as possible for OSI owners to use the system and contribute to the SIG.

We will not be totally abandoned on CompuServe. We will be given section number 8 of the Computer Club Forum which caters to systems whose followings don't rate a separate SIG. All of our files in OSI SIG will continue to be available in the Computer Club Forum. I will be an assistant SYSOP in the Computer Club Forum and except for some organizational differences, our activities on the system will go on exactly as before. To reach the Computer Club Forum, just enter "GO CLUB" at any prompt on CompuServe.

Speaking of subscriptions, PLEASE send in your renewals. Renewal notices have been shoved to the bottom of my "THINGS TO DO LIST" once again and so many of you will miss issues unless I hear from you soon. I have given a lot of folks a month's grace because I haven't sent the notices out, but I can't do that forever, OK? Thanks, people.

Rich

OS-65U Machine Code Directory

by Richard L. Trethewey

Hard disk owners are painfully aware of how long it can take for the DIR program to display the contents of the directory. I'm sure that floppy-based system owners have suffered through the same experience as well. This program replaces the DIR program on your OS-65U system diskette with a very fast machine code program that both emulates the original and adds some searching abilities.

As with any machine code routine for OS-65U, there are two components to NDIR - the assembly language source code and the support program in BASIC.

There are a number of things within the program worth examining. First of all, there is the interface to BASIC where the machine code calculates what command you've issued from the main menu. Naturally, the USR(X) vectors pointing to the machine code (ie. locations 8778 and 8779) have been set up. I have mentioned this before, but it bears repeating. Whenever you alter the USR(X) vector to your own code, you should always retain a copy of the initial contents of these locations and restore the vector when your program is finished because other programs may assume that the vectors are untouched since they were installed. Thus, if you alter these locations without restoring them afterward, you can get hit with some mysterious crashes.

```
10 .PAGE 'DIRECTORY UTILITY'
20; WRITTEN BY RICHARD L. TRETHEWEY
30; COPYRIGHT 11/08/85  ALL RIGHTS RESERVED
40;
50; BASIC EXTERNALS
60;
70 STRFLG =$000E  STRING FLAG
80 INTFLG =$000F  INTEGER FLAG
90 POSCNT =$0016  CURSOR POSITION
100 POKER  =$0019  UTILITY POINTER
110 BUF    =$001B  BASIC 2-PAGE BUFFER (71 CHARS.)
120 INDEX  =$006F  UTLITY POINTER
130 MEMSIZ =$0084  END OF BASIC MEMORY
140 VARNAM =$0092  VARIABLE NAME STORAGE
150 VARPNT =$0094  POINTER TO VARIABLE STORAGE
160 FORPNT =$0096  PTR. TO VAR. FOR STORING
170 VARPTR =$00AC  VARIABLE POINTER
180 FACEXP =$00AE  F.P. ACC. EXPONENT
190 FACHI  =$00AF  F.P. ACC. MSB
200 FACMHI =$00B0  F.P. ACC. MMSB
210 FACMLO =$00B1  F.P. ACC. MLSB
220 FACLO  =$00B2  F.P. ACC. LSB
230 FACSGN =$00B3  F.P. ACC. SIGN (+/-)
240 FACGRD =$00BD  F.P. ACC. EXPONENT GUARD BYTE
250 CHRGET =$00C0  FETCH NEXT CHARACTER
260 CHRGOT =$00C6  RETRIEVE LAST CHAR. SEEN
270 TXTPTR =$00C7  PTR. TO PROGRAM FOR CHRGET/GOT
280 CRDO   =$0A73  OUTPUT CR/LF PAIR
290 OUTSTR =$0ACC  OUTPUT STRING POINTED TO BY A/Y
300 OUTDO  =$0AEE  OUTPUT CHARACTER IN ACC.
310 CHKTYP =$0CBC  MAKE SURE NUMERIC TYPE EXPRESSION
320 CHKSTR =$0CBE  MAKE SURE STRING EXPRESSION
330 FRMEVL =$0CCD  FORMULA EVALUATOR
340 CHKCLS =$0E0D  INSURE ")", EXIT THROUGH CHRGET
350 CHKOPN =$0E10  INSURE "(", EXIT THROUGH CHRGET
360 CHKCOM =$0E13  INSURE ",", EXIT THROUGH CHRGET
370 SNERR  =$0E1E  SYNTAX ERROR
380 PTRGET =$0F2E  FIND VARIABLE IN STORAGE TABLE
390 FCERR  =$10D0  FUNCTION CALL ERROR
400 GIVAYF =$1218  GIVE A/Y PAIR TO F.P. ACC.
410 FREFAC =$1520  FIND STRING LOCATION & LENGTH
420 GETBYT =$1618  EVALUATE EXPRESSION<256 --> X REG.
430 GETVAR =$1A9D  PUT VARIABLE IN F.P. ACC.
440 FLOAT  =$1B44  CONVERT INTEGER TO F.P. TYPE
450 QUINT  =$1B96  CONVERT F.P. TO INTEGER
460 ASCFP  =$1BEE  CONVERT ASCII AT 'TXTPTR' TO FP

470 ASCII  =$1CEC  CONVERT F.P. ACC. TO ASCII STRING
480;
490; OS-65U EXTERNALS
500;
510 CCLOC  =15006  CONTROL C SAVE LOCATION
520 DISCH  =$2660  CURRENT DRIVE
530 DUN    =$26A1  DISK UNIT CONTROL BLOCK
540 DIRADR =$26AB  DIRECTORY DISK ADDR. STORAGE
550 DIRSIZ =$26AE  DIRECTORY SIZE STORAGE
560 DIRBUF =$26F2  DIRECTORY BUFFER
570 OUTCH  =$2800  OUTPUT CHARACTER IN ACC.
580 GET    =$28E8  READ DISK
590 PUT    =$28F3  WRITE TO DISK
600 FLUSH  =$2C23  FLUSH SYSTEM DISK BUFFER/CLOSE
```

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

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

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

Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes endorsement of the product or products by this magazine or the publisher.

```
610 OUFLAG =$2DA6  CURRENT OUTPUT DEVICE #
620 SWBUFF =$4700  PAGE 0/1 SWAP BUFFER
630 SWAP   =$4907  SWAP 0/1 WITH SWAP BUFFER
640;
650; OS-65U DISK CONTROL BLOCK DEFINITION
660;
670; DUN     = DISC UNIT NUMBER TO READ/WRITE
680; DUN+1   = DISK ADDRESS LSB
690; DUN+2   = DISK ADDRESS MLSB
700; DUN+3   = DISK ADDRESS MMSB
710; DUN+4   = DISK ADDRESS MSB
720; DUN+5   = NUMBER OF BYTES LSB
730; DUN+6   = NUMBER OF BYTES MSB
740; DUN+7   = MEMORY ADDRESS LSB
750; DUN+8   = MEMORY ADDRESS MSB
760;
770; ASSEMBLY CONSTANTS
780;
790 LF     =$0A
800 CR     =$0D
810 SP     =$20
820 SKIP2  =$2C
830 STACK  =$100
840;
850        *=$6000
860;
870        LDA FORPNT        FETCH ENTRY FORPNT
880        STA OLDFOR        SAVE FOR RESTORE ON EXIT
890        LDA FORPNT+1
900        STA OLDFOR+1
910        JSR $1047         MAKE CMD# AN INTEGER
920        LDA FACLO         PICK UP CMD#
930        CMP #TYPE-CMDTBL/2
940        BCS BADCMD
950        STA CMD           SAVE COMMAND #
960        ASL A             *2!
970        TAX
980        LDA CMDTBL,X
990        STA DOCMD+1
1000        LDA CMDTBL+1,X
1010        STA DOCMD+2
1020 DOCMD  JMP $FFFF        MODIFIED CODE!!!!
1030 BADCMD JMP FCERR
1040;
1050 USRDIR JSR CRDO
1060        LDA #$00
1070        STA CCLOC        CLEAR ^C LOCATION
1080        JSR DIRSU
1090        JSR HEADER
1100        JMP D2           GO TO DISPLAY
1110;
1120 DIRSU  LDA DISCN        GET DEVICE NUMBER
1130        STA DUN          GIVE TO 65U CONTROL BLOCK
1140        LDA #$00         INIZ
1150        STA DUN+1        CLEAR DISK ADDR. LSB
1160        STA DUN+3
1170        STA DUN+4
1180        STA DUN+5        CLEAR # BYTES LSB
1190        LDA #$01
1200        STA DUN+6        SET R/W FOR 1 PAGE
```

Two things happen when BASIC processes the statement X=USR(??). First of all, BASIC knows it's processing an equation as soon as it sees a variable name at the start of the statement. It then insures the inclusion of the "=" and then begins to decipher the right hand side of the equation. In our case, the only thing there is the USR(??) function. BASIC handles USR by evaluating the contents of the parenthesis and then jumps to the machine code pointed to by locations 8778 and 8779 (low/hight byte format, of course).

The first thing my machine code does when it gets control is to save the location of the storage for the variable "X" that BASIC found when it began to process the left hand side of the equation. The reason I do this is because I will be passing values back to BASIC and in the interim, I will likely have overwritten the pointer labeled "FORPNT" at $96 several times. Next I make sure the contents of the parenthesis is not a string and change its numeric value from floating point into an integer so I can handle it easily in machine code at the byte level. Based on the value found here, the command number, I use a look-up table to jump to the code that corresponds to the desired command.

The directory printer will probably interest a lot of people for a couple of reasons. First of all, it's fast. I mean **REALLY FAST!** Have your fingers ready on <CTRL>'S' when you use this baby. Second, the code used does several interesting things. It expands the normal format of the USR(??) function, and it demonstrates how to access the disk drives and the directory under OS-65U. It also demonstrates several useful techniques for calling routines in BASIC from your own machine code.

The vanilla directory printer is fairly straightforward. It calls sectors of the directory into the 65U directory buffer one page at a time and proceeds to count the entries by file type and size. When it hits the end of the directory, a summary is displayed and several parameters are passed back to the BASIC program. The routine will also display only selected file types, depending on the command number passed to it by the BASIC program. Note that the routine counts any data file whose name ends with "0" as an OS-DMS Master File and any data file that ends with a number from "1" to "7" is considered an OS-DMS Key File. All other data files are denoted as "Scratch". You can stop the program at any time by entering a <CTRL>'C' and either continue or quit by responding to the prompt.

When the directory routine(s) finish, they pass four values back to the BASIC program. Look at the BASIC program where USR(??) is called. Note that instead of simply ending with the closing parenthesis, the statement continues with four variables, separated by commas. It is these four variables that recieve the values of the counters within the directory code. They are; the number of program or BASIC files, the number of Master files, the number of Key files, and the number of scratch files. In addition, the original variable "X" from the left hand side of the equation is passed the number of bytes currently in use.

The passing of the values is done by the routine labeled "SAVVAL". SAVVAL calls the routine "SAVNUM" to pass the values to the individual variables. SAVVAL gives the values to BASIC by storing them in BASIC's floating point accumulator (a buzzword for 5 bytes in page zero where BASIC does a lot of its math. BASIC actually has 3 such internal accumulators) via the BASIC routine "GIVAYF". Note that GIVAYF expects to find a 16-bit signed integer in the Y register (LSB) and the Accumulator (MSB) and so it can only yield values from -32768 to 32767.

SAVNUM begins by looking for the separating comma in the program text by calling the BASIC routine CHKCOM. Then it looks for/at the variable name following the comma using the BASIC routine "PTRGET". PTRGET looks for the variable in BASIC's variable table. If it doesn't find the variable, it creates a new entry in the table. In any event, the memory address of the location within the table is stored in

```
1210        LDA #DIRBUF
1220        STA DUN+7         SET RAM ADDRESS LSB
1230        LDA #DIRBUF/256
1240        STA DUN+8         SET RAM ADDRESS MSB
1250        LDA #25000/256
1260        STA DUN+2         POINT TO DIREC*
1270        JSR GETDSK        READ IT
1280        LDY #$00          INIZ
1290 D1     LDA DIRBUF+$C,Y   LOOK AT DIREC*'S SIZE
1300        STA SIZE,Y        SAVE IT LOCALLY
1310        INY
1320        CPY #$03
1330        BNE D1            LOOP 'TIL DONE
1340        LDY #$00
1350        TYA
1360 D6     STA INUSE,Y
1370        INY
1380        CPY #TABTO-INUSE
1390        BNE D6
1400        LDA #98
1410        STA INUSE+1          SHOW DIR OFFSET
1420        JSR SETLIN           SET UP PAGE DATA
1430        RTS
1440;

1450; MAIN LOOP
1460;
1470 D2     JSR GETDSK            READ IN DIR PAGE
1480        LDA #DIRBUF           LOAD DIRBUF LSB
1490        STA POKER             GIVE TO POKER
1500        LDA #DIRBUF/256       LOAD MSB
1510        STA POKER+1           SET IT UP TOO
1520        LDA #$00
1530        STA EC
1540        LDA COUNT             BUMP COUNTER LSB
1550        BNE D3                WATCH FOR PAGING
1560        LDA COUNT+1           BUMP NMSB ON PAGING
1570        BNE D3                AND WATCH AGAIN
1580        LDA COUNT+2           BUMP MSB ON PAGING
1590        BNE D3
1600        INC EC
1610        LDA #$10
1620        CLC
1630        ADC POKER
1640        STA POKER
1650        BCC D3
1660        INC POKER+1
1670 D3     JSR DIROUT            DISPLAY CONTENTS
1680        INC COUNT             BUMP COUNTER LSB
1690        BNE D4                WATCH FOR PAGING
1700        INC COUNT+1           BUMP NMSB ON PAGING
1710        BNE D4                AND WATCH AGAIN
1720        INC COUNT+2           BUMP MSB ON PAGING
1730 D4     LDA COUNT             FETHC LSB
1740        CMP SIZE              READ ENTIRE DIR?
1750        BNE D5                NO! --> D5
1760        LDA COUNT+1           MAYBE, CHECK NMSB
1770        CMP SIZE+1            SAME?
1780        BNE D5                NO! --> D5
1790        LDA COUNT+2           FETCH MSB
1800        CMP SIZE+2            SAME?
```

```
1810        BEQ DIRQT          YES! END!
1820 D5     JSR DBUMP          BUMP DIRECTORY PTRS
1830        JMP D2             AND LOOP!
1840 DIRQT  JSR SAVVAL         SAVE FILE COUNTS
1850        LDA INUSE
1860        STA FACLO
1870        LDA INUSE+1
1880        STA FACMLO
1890        LDA INUSE+2
1900        STA FACMHI
1910        LDA INUSE+3
1920        STA FACHI
1930        LDA OLDFOR         GET X= FORPNT
1940        STA FORPNT         RESTORE IT FOR BASIC
1950        LDA OLDFOR+1       GET MSB TOO
1960        STA FORPNT+1
1970        LDA #$00
1980        STA CCLOC          SUPRESS ANY ^C'S LEFT
1990        JMP NORMAL         EXIT VIA NORMAL
2000;
2010 SAVVAL LDA NUMPRG+1       GET # OF PRG. FILES MSB
2020        LDY NUMPRG         GET # OF PRG. FILES LSB
2030        JSR GIVAYF         GIVE TO FPACC.
2040        JSR SAVNUM         GIVE TO "NP" VARIABLE
2050        LDA NUMMF+1        GET # OF MASTER MSB
2060        LDY NUMMF          AND LSB
2070        JSR GIVAYF         GIVE TO FPACC.
2080        JSR SAVNUM         GIVE FPACC. TO "NM" VAR.
2090        LDA NUMKF+1        GET # OF KEY FILES MSB
2100        LDY NUMKF          GET # OF KEY FILES LSB
2110        JSR GIVAYF         GIVE TO FPACC.
2120        JSR SAVNUM         GIVE TO "NK" VARIABLE
2130        LDA NUMSCR+1       GET # OF SCRATCH FILES
2140        LDY NUMSCR
2150        JSR GIVAYF         GIVE TO FPACC. AND FALL THRU
2160;
2170 SAVNUM JSR CHKCOM       FIND OUR FRIEND AGAIN
2180        JSR PTRGET       FIND THE VARIABLE
2190        STA FORPNT       SAVE PTR TO VARIABLE
2200        STY FORPNT+1
2210        LDA STRFLG
2220        BNE SAVNU2
2230        LDA INTFLG
2240        BPL SAVNU1
2250        JMP $09C5       GIVE F.P. TO % VAR
2260 SAVNU1 JMP $1ACB       GIVE FACC. TO F.P. VAR
2270 SAVNU2 JMP FCERR       CAN'T USE STRINGS!
2280;
2290 DBUMP  INC DUN+2
2300        BNE DBUM1
2310        INC DUN+3
2320        BNE DBUM1
2330        INC DUN+4
2340 DBUM1  RTS
2350;
2360 CMDTBL .WORD USRDIR        DISPLAY ALL DIRECTORY
2370        .WORD USRDIR        DISPLAY DATA FILES ONLY
2380        .WORD USRDIR        DISPLAY PROGRAM FILES ONLY
2390        .WORD USRFIL        FIND FILE'S DISK ADDR.
2400        .WORD WILD          WILD CARD DIR SEARCH
```

the page zero vector labeled "FORPNT". PTRGET also determines the type of variable it has processed and sets either the STRFLG for strings or INTFLG for integer variables as appropriate. SAVNUM's last task is to see what kind of variable it's dealing with and call the proper BASIC routine to store the contents of the floating point accumulator in the variable's entry in memory.

SAVNUM's technique is fine for smaller values, but when dealing with hard disks, as we so often do with OS-65U, the numbers GIVAYF can handle are too limited. For example, when the directory utility cleans up and prepares to go back to BASIC, it must pass the number of bytes in use on the diskette in question to the variable on the left hand side of the USR(??) statement. It does this by directly transferring the four bytes needed to track this value into the floating point accumulator. However, one last job remains before BASIC will understand this number. It must be "normalized".

The floating point accumulator is always assumed by BASIC to be normalized. That is, the mantissa (ie. the four bytes) is shifted to the left until the most significant bit is 1. For each shift, the exponent is decreased to reflect the increase (ie. the left shift) in the mantissa. The exponent (FACEXP) is a signed integer. We start out with FACEXP equal to $80 (it's negative - a very large mantissa times 2 raised to a negative power) plus the number of bits in the mantissa (for four bytes, that's 32 bits).

There are three other functions built into this routine. The first is a file lookup function. Given a file name, the function returns the disk address of that file on the current disk DEVice. If the file isn't found, the function returns a -1. The second function is much like the directory printer, but it accepts a string argument to be used to search the directory with wildcards. That is, given "A?????", this function will display all files that begin with the letter "A". The search in this routine considers "?" to match all characters. Any other character in the string argument must be matched by position within the name, although it is upper/lower case blind. Finally, the third function waits for a keypress on the console and passes the ASCII value of the keypress back to BASIC.

One thing you have to watch out for when dealing with strings using the USR(??) function. When control is passed back to BASIC, the interpreter checks the STRFLG to insure the value in the parenthesis is numeric. Be sure you store $00 in STRFLG when your routine is done if it uses BASIC to deal with strings.

This program also demonstrates the essentials of dealing with the disk drive from machine code under OS-65U. The key routines are labeled "GETDSK" and "PUTDSK" for reading and writing, respectively. These routines are virtual copies of those found inside 65U's code that is intended to be called from BASIC. Like 65D, you must switch contexts from BASIC to DOS by calling the SWAP routine which calls in a separate set of page zero and page one (stack) values when using the disk. Another thing worth noting is that the routines can point to any disk parameter table you like, not just the normal one inside OS-65U. I haven't tried using my own table because I wasn't sure 65U could really handle it, but it is interesting.

One last parting note. All of the labels for BASIC are also valid for OS-65D. The disk support code is all 65U specific, of course, but hopefully some of these routines will give you some new ideas of your own. If so, I hope you'll share them with the rest of us.

HAVE YOU RENEWED YOUR SUBSCRIPTION?

Don't Miss an Issue! Renew Now!

```
2410        .WORD KEYGET        GET KEYPRESS
2420;
2430 TYPE   .BYTE 'DATA '
2440        .BYTE 'BASIC'
2450        .BYTE 'OTHER'
2460 AR     .BYTE 'NONE '
2470        .BYTE 'READ '
2480        .BYTE 'WRITE'
2490        .BYTE 'R/W  '
2500 DELTYP .BYTE '[----] Deleted File',$00
2510 MFTYP  .BYTE 'Master',$00
2520 KFTYP  .BYTE 'Key',$00
2530 SCRTYP .BYTE 'Scratch',$00
2540 FBUFF  .BYTE 'XXXXXX'
2550 HEAD   .BYTE 'Name       '
2560        .BYTE 'Type   Access  '
2570        .BYTE 'Address     '
2580        .BYTE 'Size        Special'
2590        .BYTE CR,LF,$00
2600 CNT$   .BYTE CR,LF,'Continue (Y/N) ? ',$00
2610 TMPTYP .BYTE $00          TEMPORARY TYPE STORAGE
2620 CMD    .BYTE $00          COMMAND
2630 OLDFOR .WORD $FFFF
2640 SIZE   .BYTE $00,$00,$00
2650 INUSE  .BYTE $00,$00,$00,$00
2660 RECOV  .BYTE $00,$00,$00,$00
2670 COUNT  .BYTE $00,$00,$00
2680 NUMMF  .WORD $0000
2690 NUMKF  .WORD $0000
2700 NUMSCR .WORD $0000
2710 NUMPRG .WORD $0000
2720 TEMP   .BYTE $00
2730 EC     .WORD $0000
2740 PW     .BYTE $00,$00,$00,$00
2750 LC     .BYTE $00
2760 LPPG   .BYTE $15          LINES PER PAGE
2770 PL     .BYTE $18          PAGE LENGTH
2780 OUTTMP .BYTE $FF          OUTPUT DV# TEMP.
2790 TABTO  .BYTE $00
2800;
2810 TABER  LDA POSCNT
2820        CMP TABTO
2830        BCS TABER1
2840        LDA #SP
2850        JSR OUTDO
2860        JMP TABER
2870 TABER1 RTS
2880;
2890 HEADER LDA #HEAD
2900        LDY #HEAD/256
2910        JSR OUTSTR

2920        LDY #$00
2930        LDA #'-
2940 HEADE1 JSR OUTDO
2950        INY
2960        CPY #52
2970        BNE HEADE1
2980        LDA #$02
2990        STA LC             INIT LINE COUNT
3000        JMP CRDO
```

```
3010;
3020 TYPCHK LDY #$00
3030        LDA (POKER),Y
3040        AND #%11100
3050        LSR A
3060        LSR A
3070        PHA
3080        STA TYPCH1+1
3090        ASL A
3100        ASL A
3110 TYPCH1 ADC #$FF        *5!
3120        STA TMPTYP      SAVE FOR LATER
3130        PLA
3140        TAX
3150        INX             +1!
3160        LDY CMD         CHECK COMMAND #
3170        BEQ TYPCH2      CMD 0? --> PASS
3180        CPX CMD         CMD = TYPE?
3190        BNE TYPCH3      NO! -->
3200 TYPCH2 SEC
3210        RTS
3220 TYPCH3 LDY #$0C
3230        LDA (POKER),Y
3240        CLC
3250        ADC INUSE+1
3260        STA INUSE+1
3270        INY
3280        LDA (POKER),Y
3290        ADC INUSE+2
3300        STA INUSE+2
3310        INY
3320        LDA (POKER),Y
3330        ADC INUSE+3
3340        STA INUSE+3
3350        CLC
3360        RTS
3370;
3380 DIRDUN JSR CRDO
3390        PLA
3400        PLA
3410        JMP DIRQT       AND RETURN TO CALLER
3420;
3430 DIRNX0 LDA #DELTYP
3440        LDY #DELTYP/256
3450        JSR OUTSTR
3460        JSR TYPE4
3470        JSR CRDO
3480        JMP DIRNXT
3490;
3500 DIROUT JSR TYPCHK      CHECK ENTRY TYPE
3510        BCS DIR01
3520        JMP DIRNXT      NOT WANTED! SKIP!
3530 DIR01  LDY #$00        INIZ
3540        LDA (POKER),Y   FETCH ENTRY CHARACTER
3550        BEQ DIRDUN      0? YES! END OF DIR! -->
3560        CMP #$01        DELETED ENTRY?
3570        BEQ DIRNX0      YES! SKIP TO NEXT ENTRY
3580        JSR PNAME       PRINT FILE NAME/PW
3590        JSR TYPER       PRINT FILE TYPE & RIGHTS
3600        JSR FTYPE
```

Status Report on the Public Digital Radio Service

by Donald L. Stoner, W6TNS
6014 East Mercer Way
Mercer Island, WA 98040

(Editor's Note: The following letter was forwarded for publication by Earl Morris, a frequent and tireless contributor to PEEK[65]. Thanks Earl!)

You may recall contacting me some time ago regarding my petition to the Federal Communications Commission requesting a Public Digital Radio Service. To refresh your memory, I have petitioned the FCC to allocate a section of the radio spectrum so that computer owners can "talk" to each other via radio waves, in addition to the telephone network.

Most petitions to the FCC are deemed not in the public interest and almost immediately refused for consideration. My idea must have met with a certain amount of interest as it was given an RM-number (RM-5241). It was submitted in October of 1985 yet, a year later, it is still being considered by the FCC.

PDRS is Coming!! Actually, the petition has gone a bit further than evaluation. I have received some important information from a reporter who spoke to the Commission about my petition. It is his understanding that the FCC will be issuing a "Notice of Proposed Rule Making" this November. If this is correct, it means that the general public may, at last, have a radio service for computers.

An NPRM is a document issued by the FCC which says, in effect, here are the rules which we propose for this new service if no one objects. When the NPRM is issued, the public is invited to submit their comments and to say whether they are for or against the service, suggest changes or improvements and give indications of why such a service would be of value to the writer.

JOIN OSI SIG!

PDRS Opposition - One would think that no one could possibly object to the idea of a public digital radio service. One would think wrong! A service such as I have proposed is going to be strongly opposed by radio amateurs in general, and the American Radio Relay League in particular. Why? First, I am asking for a portion of the amateur radio six meter band. Most amateurs couldn't care less about this forlorn piece of radio spectrum. Less than one percent of the radio amateur population have ever operated on the six meter band.

More important, the American Radio Relay League does not want you to have access to the radio spectrum for another reason. Membership in the amateur radio fraternity has been virtually stagnant for the past couple of decades. The average age of radio amateurs keeps climbing. The League feels that access to the radio spectrum by hobbiests should only be possible for those with a ham license (i.e. more members for the fraternity and more members for the ARRL).

You can also expect opposition from other commercial groups who see the PDRS as a threat to their interests, in one way or another.

So, what am I leading up to? You cannot assume that the FCC will do what is right and fair for the computer hobbiests. The FCC will do what it perceives the majority desires or needs. If thousands of hams write in to oppose PDRS and if computer hobbiests don't bother to tell the FCC they want PDRS, it will die!

I Need Your Help! - So here is what I have "up my sleeve". I need you to publicize the coming of PDRS with as many people as you can. It is imperative that news about PDRS be on every BBS in the country. If any of you have access to ARPANET, the UNIX NET, or FIDO net, make sure you funnel information to these groups. I will handle CompuServe and you can usually find me on HAMNET (the ham radio special interest group). The documents I have submitted to the FCC are in their DL4 Data Library and also on the Commodore SIG. If you are

```
3610         JSR BUMPLN       ONLY BUMP ON PRINTS!
3620 DIRNXT  LDA CCLOC
3630         CMP #$03
3640         BNE DIRNX1
3650         JSR CNTCHK       CONTINUE?
3660         BEQ DIRNX1       YES! -->
3670         JMP DIRDUN       NO! STOP!
3680 DIRNX1  LDA POKER
3690         CLC
3700         ADC #$10
3710         STA POKER
3720         BCC DIRNX2
3730         INC POKER+1
3740 DIRNX2  INC EC
3750         LDA EC
3760         CMP #256/16
3770         BNE DIROUT
3780         RTS
3790;
3800 CNTCHK  LDA 11686       GET CURRENT OUTPUT DV
3810         STA OUTTMP      SAVE IT FOR A MOMENT
3820         LDA 11668       GET CONSOLE DV NUMBER
3830         STA 11686       MAKE IT CURRENT
3840 CNTCH1  LDA #CNT$
3850         LDY #CNT$/256
3860         JSR OUTSTR      DISPLAY "CONTINUE?"
3870 CNTCH2  JSR $0587       WAIT FOR KEYPRESS
3880         CMP #CR         DID USER HIT <CR>?
3890         BEQ CNTCH3      YES! --> MAKE IT "Y"
3900         CMP #$20        NO, IS CHAR. VALID?
3910         BCC CNTCH2      KILL ANY TRAILING ^S
3920         BCS CNTCH4      OK -->
3930 CNTCH3  LDA #'Y         <CR> = "Y"
3940 CNTCH4  JSR CASECK      FIX LOWER CASE
3950         CMP #'Y
3960         BEQ CNTCH5
3970         CMP #'N
3980         BEQ CNTCH5
3990         JSR CRDO
4000         JMP CNTCH1      FORCE VALID RESPONSE!
4010 CNTCH5  PHA             SAVE RESPONSE ON STACK
4020         JSR OUTCH       DISPLAY IT TOO!
4030         JSR CRDO        CLEAN UP LINE
4040         JSR CRDO        DO ANOTHER FOR DRILL
4050         LDA OUTTMP      RETRIEVE ORIGINAL DV
4060         STA 11686       MAKE IT CURRENT AGAIN
4070         LDA #$00
4080         STA CCLOC       CLEAR WHATEVER'S THERE
4090         PLA             RETRIEVE RESPONSE
4100         CMP #'Y
4110         RTS
4120;
4130 PNAME   LDY #$00         INIZ
4140 PNAME1  LDA (POKER),Y    FETCH CHARACTER
4150         JSR OUTDO        PRINT IT
4160         INY              BUMP IT
4170         CPY #$06         PRINTED WHOLE NAME?
4180         BNE PNAME1        NO! LOOP!
4190         RTS
4200;
```

```
4210 BUMPLN INC LC          BUMP LINE COUNT
4220        LDA LC          FETCH NEW
4230        CMP LPPG        AT PAGE END?
4240        BEQ BUMPL2      YES! -->
4250 BUMPL1 RTS             NO! RETURN
4260 BUMPL2 JSR CRDO        DO CR-LF
4270        INC LC          BUMP LINE COUNT
4280        LDA LC
4290        CMP PL          AT PAGE END?
4300        BNE BUMPL2      NO! LOOP! -->
4310        LDA 11686
4320        CMP. 11668
4330        BNE BUMPL3      ALWAYS CONTINUE W/PRINTER
4340        JSR CNTCHK      ASK USER
4350        BNE BUMPL4      NO! STOP HERE!
4360 BUMPL3 JMP HEADER      OUTPUT PAGE HEADER & RTS
4370 BUMPL4 PLA
4380        PLA             PULL BUMPLN OFF STACK
4390        JMP DIRDUN      EXIT THROUGH DIRDUN!
4400;
4410 SETLIN LDA 11686       GET REQ. OUTPUT DV
4420        CMP 11668       SAME AS CONSOLE?
4430        BEQ SETCON      YES! -->
4440        LDA #63
4450        STA LPPG        SET 63 LINES PER PAGE
4460        LDA #66
4470        STA PL          SET 66 LINES PAGE LENGTH
4480        RTS
4490 SETCON LDA #19
4500        STA LPPG        SHOW 15 LINES ON CONSOLE
4510        LDA #21
4520        STA PL
4530        RTS
4540;
4550 GETDSK JSR SWAP
4560        LDA #GETD1-1/256
4570        PHA
4580        LDA #GETD1-1
4590        PHA
4600        JMP GET
4610 GETD1  .WORD DUN
4620        JSR SWAP
4630        TAY
4640        BNE GETD2
4650        RTS
4660;
4670 GETD2  JMP FCERR      ABORT ON DISK ERROR
4680;
4690 TYPER  LDA #0
4700        STA TABTO
4710        JSR TABER
4720        LDX #$00
4730        LDY TMPTYP
4740 TYPE1  STX TEMP
4750        LDA TYPE,Y
4760        JSR OUTDO
4770        LDX TEMP
4780        INX
4790        INY
4800        CPX #$05
```

not on CompuServe, have someone download the files for you. I will be happy to send anyone copies of the documents for a buck apiece to cover postage and reproduction costs. Here are the files on CompuServe:

FCC1.DOC - First portion of petition which became RM-5241.
FCC2.DOC - Second and technical portion of my petition.
FCC.DOC - My rebuttal to those who opposed the petition.
FCCADD.DOC - A six meter bandplan showing PDRS and displacement of current activities.

Public Domain - Any material in these files, in the original paper documents or in these letters I send you is public domain. You may reproduce it, extract it, or whatever, as you desire.

Complete Your PEEK[65] Library with backissues!

ATTENTION: DEALERS!

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

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

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

```
4810        BNE TYPE1
4820        LDA #15
4830        STA TABTO
4840        JSR TABER
4850        LDY #$00
4860        LDA (POKER),Y
4870        AND #$03
4880        STA TYPE2+1
4890        ASL A
4900        ROL A
4910 TYPE2  ADC #$FF
4920        TAY
4930        LDX #$00
4940 TYPE3  STX TEMP
4950        LDA AR,Y
4960        JSR OUTDO
4970        LDX TEMP
4980        INY
4990        INX
5000        CPX #$05
5010        BNE TYPE3
5020 TYPE4  LDA #22
5030        STA TABTO
5040        JSR TABER
5050        LDA #$00
5060        STA FACLO
5070        LDY #$09
5080        LDA (POKER),Y
5090        STA FACMLO
5100        INY
5110        LDA (POKER),Y
5120        STA FACMHI
5130        INY
5140        LDA (POKER),Y
5150        STA FACHI
5160        JSR NORMAL
5170        JSR ASCII
5180        LDA #STACK
5190        LDY #STACK/256
5200        JSR OUTSTA
5210        LDA #33
5220        STA TABTO
5230        JSR TABER
5240        LDA #$00
5250        STA FACLO
5260        LDY #$0C
5270        LDA (POKER),Y
5280        STA FACMLO
5290        CLC
5300        ADC INUSE+1
5310        STA INUSE+1
5320        INY
5330        LDA (POKER),Y
5340        STA FACMHI
5350        ADC INUSE+2
5360        STA INUSE+2
5370        INY
5380        LDA (POKER),Y
5390        STA FACHI
5400        ADC INUSE+3
```

```
5410        STA INUSE+3
5420        JSR NORMAL
5430        JSR ASCII
5440        LDA #STACK
5450        LDY #STACK/256
5460        JSR OUTSTA
5470        RTS
5480;
5490 FTYPE  LDY #$00
5500        LDA (POKER),Y
5510        AND #%11100
5520        BNE FTYPE6       NOT DATA! CHK FOR PRG!
5530        LDA #45
5540        STA TABTO
5550        JSR TABER
5560        LDY #$05
5570        LDA (POKER),Y
5580        CMP #'0
5590        BEQ FTYPE4     MASTER
5600        CMP #'1
5610        BCC FTYPE2     SCRATCH
5620        CMP #'8
5630        BCS FTYPE2     SCRATCH
5640        LDA #KFTYP
5650        LDY #KFTYP/256
5660        JSR OUTSTA
5670        INC NUMKF
5680        BNE FTYPE1
5690        INC NUMKF+1
5700 FTYPE1 JMP CRDO
5710 FTYPE2 LDA #SCRTYP
5720        LDY #SCRTYP/256
5730        JSR OUTSTA
5740        INC NUMSCR
5750        BNE FTYPE3
5760        INC NUMSCR+1
5770 FTYPE3 JMP CRDO
5780 FTYPE4 LDA #MFTYP
5790        LDY #MFTYP/256
5800        JSR OUTSTA
5810        INC NUMMF
5820        BNE FTYPE5
5830        INC NUMMF+1
5840 FTYPE5 JMP CRDO
5850 FTYPE6 CMP #%100
5860        BNE FTYPE7
5870        INC NUMPRG
5880        BNE FTYPE7
5890        INC NUMPRG+1
5900 FTYPE7 JMP CRDO
5910;
5920 CASECK CMP #'a
5930        BCC CASE1
5940        CMP #'z+1
5950        BCS CASE1
5960        EOR #$20
5970 CASE1  RTS
5980;
5990 USRFIL JSR GTFNAM       GET FILE NAME
6000        JSR DIRSU        SET UP FOR DIR READ
```

```
6010 USRF5  JSR GETDSK
6020        LDA #DIRBUF
6030        STA POKER
6040        LDA #DIRBUF/256
6050        STA POKER+1
6060 USRF6  LDY #$00
6070 USRF7  LDX #$00
6080 USRF8  LDA (POKER),Y
6090        BEQ USRFC        END OF DIR! -->
6100        CMP FBUFF,X
6110        BNE USRF9
6120        INY
6130        INX
6140        CPX #$06
6150        BNE USRF8
6160        JMP USRFD        FOUND IT!
6170 USRF9  TYA
6180        AND #$F0
6190        CLC
6200        ADC #$10
6210        TAY
6220        BNE USRF7
6230        LDA COUNT
6240        CLC
6250        ADC #$01
6260        STA COUNT
6270        BCC USRFA
6280        INC COUNT+1
6290        BNE USRFA
6300        INC COUNT+2
6310 USRFA  LDA COUNT+2
6320        CMP SIZE+2
6330        BNE USRFB
6340        LDA COUNT+1

6350        CMP SIZE+1
6360        BNE USRFB
6370        LDA COUNT
6380        CMP SIZE
6390        BEQ USRFC
6400 USRFB  JSR DBUMP
6410        JMP USRF5
6420 USRFC  JSR CHKCOM        FIND THE COMMA
6430        JSR PTRGET        MOVE BASIC PAST VARIABLE
6440        LDA #$FF
6450        TAY
6460        LDX OLDFOR
6470        STX FORPNT
6480        LDX OLDFOR+1
6490        STX FORPNT+1
6500        JMP GIVAYF        SHOW NO MATCH!
6510;
6520 USRFD  TYA
6530        AND #$F0
6540        CLC
6550        ADC POKER
6560        STA POKER
6570        BCC USRFE
6580        INC POKER+1
6590 USRFE  LDY #$0C
6600        LDA (POKER),Y
6610        STA FACMLO
6620        INY
6630        LDA (POKER),Y
6640        STA FACMHI
6650        INY
6660        LDA (POKER),Y
6670        STA FACHI
6680        LDA #$00
6690        STA FACLO
6700        JSR NORMAL
6710        JSR SAVNUM
6720;
6730        LDY #$09
6740        LDA (POKER),Y
6750        STA FACMLO
6760        INY
6770        LDA (POKER),Y
6780        STA FACMHI
6790        INY
6800        LDA (POKER),Y
6810        STA FACHI
6820        LDA #$00
6830        STA FACLO

6840        LDA OLDFOR
6850        STA FORPNT
6860        LDA OLDFOR+1
6870        STA FORPNT+1
6880        JMP NORMAL
6890;
6900; NORMALIZE FLOATING POINT ACCUMULATOR
6910;
6920 NORMAL LDA #32+$80
6930        STA FACEXP       32 BITS!
6940        LDA FACHI        MAKE SURE VALUE IS
6950        BMI NORMA2       NON-ZERO
6960        BNE NORMA1
6970        LDA FACMHI
6980        BNE NORMA1
6990        LDA FACMLO
7000        BNE NORMA1
7010        LDA FACLO
7020        BEQ NORMA3       0! -->
7030 NORMA1 DEC FACEXP
7040        ASL FACLO
7050        ROL FACMLO
7060        ROL FACMHI
7070        ROL FACHI
7080        BPL NORMA1
7090 NORMA2 RTS
7100 NORMA3 STA FACEXP
7110        RTS
7120;
7130 KEYGET JSR $0587
7140        TAY
7150        LDA #$00
7160        JMP GIVAYF
7170;
7180 GTFNAM JSR CHKCOM      FIND THE COMMA
7190        JSR FRMEVL      EVALUATE EXPRESSION
7200        JSR FREFAC-3    CHKSTR & LOCATE IT
```

```
7210        CMP *$07       CHECK LENGTH
7220        BCC GTFN1      O.K. --> CONTINUE
7230        JMP SNERR      BAD! ERROR!
7240 GTFN1  STX GTFN2+1    SAVE ADDR LSB
7250        STY GTFN2+2    SAVE ADDR MSB
7260        STA GTFN3+1    SAVE LENGTH
7270        LDY *$00       INIZ
7280        STY STRFLG     CLEAR STRFLG EARLY
7290 GTFN2  LDA $FFFF,Y    FETCH A CHARACTER
7300        JSR CASECK     MAKE IT ALL CAPS
7310        STA FBUFF,Y    SAVE IT
7320        INY

7330 GTFN3  CPY *$FF
7340        BNE GTFN2
7350        LDA *SP
7360 GTFN4  CPY *$06
7370        BEQ GTFN5
7380        STA FBUFF,Y
7390        INY
7400        BNE GTFN4
7410 GTFN5  RTS
7420;
7430 WILD   JSR GTFNAM      GET FILE NAME
7440        JSR DIRSU       SET UP FOR DIR READ
7450        JSR HEADER
7460        LDY *$00
7470 WILD0  LDA FBUFF,Y
7480        CMP *SP
7490        BNE WILD1
7500        LDA *'?
7510        STA FBUFF,Y
7520 WILD1  INY
7530        CPY *$06
7540        BNE WILD0
7550 WILD2  JSR GETDSK
7560        LDA *DIRBUF
7570        STA POKER
7580        LDA *DIRBUF/256
7590        STA POKER+1
7600        LDA *$00
7610        STA EC
7620 WILD3  LDY *$00
7630 WILD4  LDA (POKER),Y
7640        BEQ WILDC       END OF DIR! -->
7650        CMP *$01
7660        BEQ WILD9       SKIP DELETED'S
7670        LDA FBUFF,Y
7680        CMP *'?
7690        BEQ WILD8
7700        CMP *'*         LOOK FOR *?
7710        BNE WILD6       NO -->
7720        LDA (POKER),Y   YES! FETCH CHAR.
7730        CMP *'0
7740        BCC WILD9
7750        CMP *'9+1
7760        BCS WILD9       NOT A NUMBER? -->
7770        BCC WILD8       OK! -->
7780 WILD6  LDA FBUFF,Y     DO NORMAL CHECK
7790        CMP (POKER),Y
7800        BNE WILD9
```

```
7810 WILD8  INY

7820        CPY *$06
7830        BNE WILD4
7840        JSR TYPCHK
7850        JSR PNAME
7860        JSR TYPER
7870        JSR FTYPE
7880 WILD9  LDA POKER
7890        CLC
7900        ADC *$10
7910        STA POKER
7920        LDA POKER+1
7930        ADC *$00
7940        STA POKER+1
7950        INC EC
7960        LDA EC
7970        CMP *256/16
7980        BNE WILD3
7990        INC COUNT
8000        BNE WILDA
8010        INC COUNT+1
8020        BNE WILDA
8030        INC COUNT+2
8040 WILDA  LDA COUNT+2
8050        CMP SIZE+2
8060        BNE WILDB
8070        LDA COUNT+1
8080        CMP SIZE+1
8090        BNE WILDB
8100        LDA COUNT
8110        CMP SIZE
8120        BEQ WILDC
8130 WILDB  JSR DBUMP
8140        JMP WILD2
8150 WILDC  LDA *$00
8160        TAY
8170        LDX OLDFOR        POINT BACK TO "X"
8180        STX FORPNT
8190        LDX OLDFOR+1
8200        STX FORPNT+1
8210        JMP GIVAYF        SHOW NO MATCH!
8220;
8230        .END DIR
```

Dealers: Have you signed up for PEEK's CO-OP Ad program?

```
10 REM- Directory Utility
20 REM- Written by Richard L. Trethewey
30 REM- Copyright 11/18/85   All Rights Reserved
40 K0=0:K1=1:K2=2:K3=3:K4=4:K5=5:K6=6:K7=7:K8=8:K9=9:KT=10
50 U1SEA=PEEK(8778):U2SEA=PEEK(8779): AA=ASC("A"): AZ=ASC("Z")
60 T=PEEK(9832): IF T>127 THEN T=T-124: IF T>63 THEN T=T-58
70 POKE 8778,0: POKE 8779,96: FLAG 9: FLAG 27: OD$=CHR$(T+65)
80 PRINT CHR$(27);CHR$(28);:PD=K5:IFPEEK(11686)=K2THENPD=K1
90 PRINT "Directory Utility": PRINT
100 INPUT"DEVice ";DV$: D=ASC(DV$+" "): IF DV$="" THEN D=T+65
110 PRINT: IF D>AZ THEN D=D-32
120 DV$=CHR$(D): IF D<ASC("A") OR D>ASC("E") THEN 100
130 IF OD$<>"" THEN 150
140 INPUT "Console or Printer Output (C or P) ";OD$
150 PRINT: DV=PEEK(11686): OD$=LEFT$(OD$+" ",K1)
160 IF OD$=" " OR OD$="C" OR OD$="c" THEN 190
170 IF OD$="P" OR OD$="p" THEN DV=PD: GOTO 190
180 PRINTCHR$(K7) : GOTO 140
190 PRINT "(1) Display ALL Files"
200 PRINT "(2) Display Only DATA Files
210 PRINT "(3) Display Only PROGRAM Files"
220 PRINT "(4) Find a File's DISK ADDRESS"
230 PRINT "(5) Wildcard Directory Search"
240 PRINT: INPUT "    Your Choice "; CMD$: CMD=VAL(CMD$)
250 PRINT: IF CMD$="" THEN CMD=K1
260 IF CMD<K1 OR CMD>K5 THEN 190
270 CMD=CMD-K1: DEV DV$: IF CMD>K2 THEN 600
280 POKE 11686,2^(DV-K1)
290 X=USR(CMD),NP,NM,NK,NS: PRINT X; " Bytes in use
300 GOSUB 800: GOSUB 900: PRINT: ON CMD+K1 GOSUB 450,490,520
310 :
400 CD=PEEK(11668): POKE 11686,CD
410 POKE 8778,U1SEA: POKE 8779,U2SEA
420 DEV DD$: GOSUB 63000: IF AP$="" THEN END
430 RUN AP$
440 :
450 PRINT NP; " Program Files"; TAB(30); NS; " Scratch Files"
460 PRINT NM; " MASTER Files"; TAB(30); NK; " KEY Files"
470 PRINT: RETURN
480 :
490 PRINT NM; " MASTER Files"; TAB(30); NK; " KEY Files"
500 PRINT NS; " Scratch Files": PRINT: RETURN
510 :
520 PRINT NP; " Program Files": PRINT: RETURN
530 :
600 IF CMD = K4 THEN PRINT "WILD CARD ";
610 INPUT "File Name "; F$: L=LEN(F$)

620 PRINT: IF L<K1 OR L>K6 THEN 610
630 IF CMD=K4 THEN 670
640 X=USR(CMD),F$,SZ: IF X=-K1THEN PRINT"Not Found": GOTO 400
650 PRINT F$; " is at DISK ADDRESS";X;" and is";SZ;" bytes long."
660 PRINT: GOTO 400
670 X=USR(CMD),F$: PRINT: GOTO 400
680 :
800 REM- Get system floppy and hard disk size
810 FS=275968:HS=72898560
820 IF PEEK(13316)=129 THEN HS=7340032
830 IF PEEK(13316)=1   THEN HS=23166976
840 IF PEEK(13316)=193 THEN HS=29369128
850 IF PEEK(13316)=16  THEN HS=36449280
```

Speed Hints for 65U BASIC

by Roger Clegg
Data Products Maintenance

Microsoft BASIC for the 6502 is so fast that computation time is usually insignificant compared to printing, waiting for keystrokes, and even waiting for the disk drive.If you need to speed up a program, rethinking your whole approach to the problem or reducing the number of disk drive accesses will probably do more than any amount of fiddling with the code. Nevertheless there are some computation-intensive programs which need speeding up, and the following hints may be useful.If you get really serious, you should prepare a program profile by inserting counting lines like C5=C5+1 in a copy of the program, at enough places to determine the busiest parts of the code, and then concentrate your efforts on those lines.

If a condition is false then add 250+19*N cycles, where N is the number of characters (tokenised) after the THEN. Ascii to floating-point conversions run at an average of 1500 cycles per digit before the decimal point and 4200 cycles after. Garbage collections take about N*N/25300 seconds, where N is the number of strings.Null strings reduce the time. Table 1 contains a breakdown of many BASIC functions in terms of CPU cycles.

1. The first two places in the variable table are best reserved for general-purpose variables X and X$, which should be used for any temporary value which will be kept for only a line or two, and can also be used to pass values to and from subroutines.Thus most programs should start X=0: X$="", and should then list the most-used variables and constants. Use a cross-reference utility to print out a variable cross-reference and use it to help decide the best order for the variable table. Any variable referenced inside a busy loop deserves a higher place for that reason, but note that NEXT I doesn't count as a reference to I,

```
851 IF PEEK(13316)=131 THEN HS=23166976
860 RETURN
870 :
900 REM- Display Remaining Space on Disk
910 IF PEEK(9832)<K5 THEN PRINT FS-X;: GOTO 930
920 PRINT HS-X;
930 PRINT " bytes available": RETURN
940 :
50000 REM- LINE 50000
50010 EL=PEEK(11774)+PEEK(11775)*256:EN=PEEK(10226)
50020 ED=PEEK(9832): IF ED>127THENED=ED-124:IFED>63THENED=ED-58
50030 ED$=CHR$(65+ED): IF EN<>128 THEN 50060
50040 PRINTCLS$;"Can't find ";F$;" on DEVICE "; ED$: GOTO 50080
50050 :
50060 PRINTCLS$;"DEVICE ";ED$;" ";
50070 PRINT "Error #";EN;" on line ";EL
50080 PRINT: GOSUB 63000: GOTO 63020
50090 :
60000 RP$="BEXEC*": OD$="C": GOTO 40
60010 RP$="BEXEC*": OD$="P": GOTO 40
60020 :
63000 INPUT "Press <RETURN> to continue "; Y$
63010 IF Y$<>"STOP" AND Y$<>"stop" THEN RETURN
63020 POKE 8778,U1SER:POKE8779,U2SER
63030 CD=PEEK(11668): POKE 11686,CD
63040 DEV OD$: END
```

because BASIC doesn't search the variable table.

2. Avoid Ascii constants of more than one digit, particularly within loops. Either assign the value to X or Y just before use in a loop, or name it as a constant early in the program. Single digits should not generally be named, as only the first 12 variables in the table are faster than Ascii conversion, and they are generally too valuable for such use. (This hint doesn't apply to more recent Microsoft BASICs, such as IBM PC BASIC, which store constants in internal format.)

3. Avoid working out any formula more often than necessary. E.g.

IF MID$(NAME$,II+1,2)="JR" OR MID$(NAME$,II+1,2)="SR" THEN NEXT

should be restated

X$=MID$(NAME$,II+1,2): IF X$="JR" OR X$="SR" THEN NEXT

This example substitutes three variable table lookups for the second formula evaluation, which includes two lookups, two ASCII conversions, and a floating-point addition, besides the MID$ function itself. Being constantly aware of such considerations is perhaps the most important thing of all in writing fast programs. Even looking up an array element is a significant cost, so

IF A(I)=P1 OR A(I)=P2 OR A(I)=P3 OR A(I)=P4 THEN A(I)=0

should be restated

X=A(I): IF X=P1 OR X=P2 OR X=P3 OR X=P4 THEN A(I)=0

These considerations are particularly important within loops.

4. When executing a GOTO or GOSUB, BASIC first compares the high bytes of the current line C and the required line R. If INT(R/256) > INT(C/256) then BASIC starts the search from the current line, otherwise from the beginning of the program. So search time is minimized if R follows C closely, but is numbered 300 higher. A critical subroutine called from widely different places is best placed at line 2 or 3, with line 1 jumping around it. (Microsoft BASIC has made two major improvements since the 6502 version: it stores the line number in binary to avoid ASCII conversion, and the first time the GOTO or GOSUB is executed, it substitutes the actual address. Before an edit or save, it whips through the program and replaces the line numbers.)

5. To handle repeated backward GOTOs, recent versions of Microsoft BASIC have provided WHILE and WEND for looping while a condition is true, as in this fragment of Quicksort:

```
   640 WHILE I<J
   650  IF A(I)>A(J) THEN SWAP
A(I),A(J): K=1-K
   660  I=I+K: J=J+K-1
   670 WEND
```

In OSI BASIC this would usually be coded;

```
  650 IF A(I)>A(J) THEN SWAP
A(I);A(J): K=1-K
  660 I=I+K: J=J+K-1: IF I<J
GOTO 650
```

However, loops like this can be greatly speeded up by setting up an infinite loop and using NEXT instead of GOTO, to avoid GOTO's line search:

```
   650 FOR X=0 TO 1 STEP 0: IF
A(I)>A(J) THEN SWAP A(I);A(J):
K=1-K
   660 I=I+K: J=J+K-1: IF I<J
THEN NEXT
```

6. OS-65U BASIC has the unusual ability to call subroutines by name, which saves interpreting the Ascii line number as well as making the program clearer. Name them early in the program: e.g.

```
   20 MENU=100: SEARCH=600:
LINE=700: GET=800: PUT=900
```

I have a personal convention of putting the named subroutines before line 1000, so that I can safely renumber the program from 1000 on up. Critical GOTOs can also be speeded up by naming the line number as a constant, but this is a dangerous practice. If you really need the extra speed, put a warning at the top of the program:

```
    1 REM   INVOIC
    2 :
    3 REM   Don't renumber this
program!  See line 30.
    4 REM   Version of 8/14/85.
    5 :
    10 X=0: Y=0: X$="": Y$="": D=0:
T=0:   REM Most-used variables
    20 C1=.5: C2=32: C3=100:
CR$=CHR$(13): REM Constants
    30 L1210=1210: L2440=2440:
M2500=2500: REM Line numbers
```

Note that naming the lines like this improves readability, but line 2500 has to be named M2500 instead of L2500 to prevent BASIC from confusing it with L2440.

7. Recent versions of Microsoft BASIC allow only one NEXT in a loop, but in OSI BASIC loops can often be speeded up by having more than one NEXT. All but the final NEXT must be followed by a GOTO (out of loop) or a RETURN to avoid a NF ERROR. For example, the following subroutine from a dump in IBM BASIC

```
    500 FOR I=J TO J+CPL-1:
C=PEEK(I)
    510  IF C=0 THEN
PRINT#D,"-";: GOTO 550
    520  IF C>L AND C<U THEN
PRINT#D,CHR$(C);: GOTO 550
    530  IF C=CR THEN
PRINT#D,"#";: GOTO 550
    540  PRINT#D,"@";
    550 NEXT
    560 RETURN
```

can be greatly speeded up in OSI BASIC as follows

```
    500 FOR I=J TO J+CPL-1:
C=PEEK(I)
    510 IF C=0 THEN
PRINT#D,"-";: NEXT: RETURN
    520 IF C>L AND C<U THEN
PRINT#D,CHR$(C);: NEXT: RETURN
    530 IF C=CR THEN
PRINT#D,"#";: NEXT: RETURN
    540 PRINT#D,"@";: NEXT:
RETURN
```

Figures (for OS-65U at 2 Mhz)	Cycles	Per Second
Ignore space in program	17	117000
Ignore character of remark	19	105000
Ignore character following false condition	19	105000
Get next character of program	38	53000
Ignore extra character in long variable name	69	29000
Search of variable table, per variable	31	65500
Search of array tables, per array	56	36700
Search for line number, per line	57	36200
FIND, per character of CD-74 disk file	47	43900
FIND, per character of floppy disk file	287	7130
New statement	171	12000
New line number and statement	249	8220
Blank line (one space)	266	7600
Blank line (one colon)	420	4800
:REM (Add 19 cycles per character)	296	6920
:NEXT (Typical step 1 loop)	1400	1500
:X=X (If X is first variable in table)	1260	1620
:X=9	1600	1280
:X=1	1690	1210
:X=-1	2100	975
:X=90	2610	785
:X=Y (If Y is 50th variable in table)	2830	720
:X=71	3410	600
:X=100	3710	550
:X=717	5210	393
:X=.5	3670	557
:X=.1	4550	449
:X=3.1415926535	45500	45
:POKE 12345,123	11600	176
:POKE X,Y (If X and Y are 1st & 2nd vars.)	2360	865
:IF X THEN (If true, and X is 1st variable)	850	2400
:IF NOT X THEN	1760	1160
:IF X=1 THEN	2730	750

Table 1

8. Examine all operations and functions inside busy loops to see if they can be moved outside. For example,

```
    FOR I=1 TO 32:
A(I)=FNA(I)+FNA(TT): NEXT
```

should be restated

```
    X=FNA(TT): FOR I=1 TO 32:
A(I)=FNA(I)+X: NEXT
```

9. Loops will execute a little faster if contained in one line; searches can usually be stated with the <> sign to achieve this:

```
    650 FOR I=1 TO N: IF A$(I)<>X$
THEN NEXT: STOP
```

If the search can fail then the STOP must of course be replaced by GOTO. In any event, when the search succeeds it drops to the next line with the correct I.

10.When there is a series of IF statements, only one of which will be true, as in 7 above, the likeliest should be tested first. It may be worthwhile to do test runs with temporary extra lines for counting occurences, and rearrange the IFs accordingly.

11.In complex IF statements like IF A>0 AND A<6 AND A<>3, BASIC always evaluates the whole expression. If the condition is usually false, then time can be saved by first testing the part likeliest to be false: IF A<6 THEN IF A>0 AND A<>3 ...

12.Most Basics include an ELSE option:

```
   IF D>2 THEN X=10 ELSE X=4
```

In OSI BASIC this should be expressed not by two IFs, but by assigning the more likely option and testing for the other:

```
   X=10: IF D<=2 THEN X=4
```

13.In working with numbers IFs can often be avoided altogether by using the fact that BASIC evaluates True as -1 and False as 0. For example,

```
   750 IF D>2 THEN PRINT#D,
TAB(10)
   760 IF D<=2 THEN PRINT#D,
TAB(4)
```

can be replaced by

```
   750 PRINT#D, TAB(4-6*(D>2))
```

14. If there are several IF statements with the same condition, the condition should be replaced by a flag, which is tested faster:

```
   740 PR=0: IF D>2 THEN PR=-1
```

Then branch by IF PR THEN ... or by IF NOT PR THEN ... Even though it may waste a millisecond, flags should always be turned off before being turned on, as in 740 above, to make certain they are set correctly if the program loops. The program is a little clearer if YES=-1 and NO=0 are defined early on, so that 740 reads

```
        740 PR=NO: IF D>2 THEN
PR=YES
```

15. Elements of multi-dimensional arrays take particularly long to look up, because a multiplication is required. Sometimes two or more simple arrays can be substituted. For example, Quicksort needs to save the lower and upper bounds of each segment on a stack, but implementing this as two arrays L(20) and U(20) will be much faster than a single stack S(20,2).

```
10 FORI=1TO26:PRINT:NEXTI
20 PRINT"                          ARITHMETIC PRACTICE":PRINT
30 PRINT"You may select add, subtract, multiply, or divide, either on
printed
40 PRINT"sheets (with an answer sheet) or on the screen.  In the
immediate"
50 PRINT"(screen) mode, your problems are timed and wrong answers are"
60 PRINT"repeated later. A wide-paper printed summary records the
session."
70 FLAG25:FLAG27
80 FORI-1TO8:PRINT:NEXTI
90 POKE16141,10:CU$-CHR$(126)+CHR$(12):CC-15006:A$-CHR$(0)
100 CD$-CHR$(126)+CHR$(11)
110 POKE15006,0:PRINT"Input title: ";
120 FORI-1TO1E8
130 PRINTCHR$(0);
140 X-PEEK(15006)
150 IFX-0THEN180
160 IFX-13THEN:Z1--1*I:Z1-RND(Z1):I-1E9:GOTO180
170 C$-C$+CHR$(X):PRINTCHR$(X);:POKE15006,0
180 NEXTI
190 PRINT
200 T$-C$
210 INPUT"Printer # (3,5,6,8) <5>";DV:IFDV-0THENDV-5
220 FLAG26:DIMA(15,300)
230 PRINT#1,CHR$(12);:INPUT"(I)mmediate mode or (P)rinted <P>";Q$
240 IFQ$<>"I"THEN250
250 INPUT"(A)dd, (S)ubtract, (M)ultiply, (D)ivide";OP$
260 INPUT"Maximum number <999>";A:IFA-0THENA-999
270 INPUT"Minimum number <0>";B
280 C-2:IFOP$-"A"THENINPUT"How many addends";C
290 IFOP$-"D"THENINPUT"Maximum divisor";Q
300 IFQ$-"I"THEN1250
310 T$-"      "+T$
320 FORI-1TO80
330 FORX-1TOC
340 A(X,I)-INT(RND(21)*A):IFA(X,I)<BTHEN340
350 IFOP$-"S"ANDX-1ANDA(X,I)<1THEN340
360 IFOP$-"S"ANDX-2ANDA(X,I)>A(X-1,I)THEN340
370 IFOP$-"D"ANDX-2THENA(X,I)-INT(RND(21)*Q):IFA(X,I)<BTHEN370
380 IFOP$-"D"ANDX-2ANDA(X,I)>A(1,I)THEN370
390 NEXTX:IFOPTHENRETURN
400 NEXTI
410 Z-4
420 IFOP$-"A"THEN460
430 IFOP$-"S"THEN900
440 IFOP$-"M"THEN940
450 IFOP$-"D"THEN980

460 FORI-1TO80
470 FORX=1TOC
480 A(C+1,I)-A(C+1,I)+A(X,I)
490 NEXTX:IFOPTHENGOTO1340
500 NEXTI:X$-"+"
510 W-INT(60/(C+Z))
520 PRINT#DV,T$:PRINT#DV
530 FORX-0TOW-1
540 FORY-1TOC
550 FORI-1TO7
560 IFY<>CTHEN590
570 PRINT#DV,TAB(10*I-LEN(STR$(A(Y,X*7+I)))-1)X$;A(Y,X*7+I);
```

```
580 GOTO600
590 PRINT*DU,TAB(10*I-LEN(STR$(A(Y,X*7+I))))A(Y,X*7+I);
600 NEXTI:PRINT*DU
610 NEXTY
620 FORI=1TO7
630 PRINT*DU,TAB(10*I-8)"--------";
640 NEXTI:PRINT*DU
650 IFX$="X"ANDAP=0THENGOSUB730
660 IFX$="X"ANDAP=1THENGOSUB780
670 FORI=1TO7
680 IFAP=1THENPRINT*DU,TAB(10*I-LEN(STR$(A(C+1,X*7+I))))A(C+1,X*7+I);
690 NEXTI:PRINT*DU:PRINT*DU:PRINT*DU
700 NEXTX
710 IFAP=0THENAP=1:PRINT*DU,CHR$(12):GOTO520
720 GOTO1870
730 FORI=1TOLEN(STR$(A))-1:PRINT*DU:NEXTI
740 FORI=1TO7
750 PRINT*DU,TAB(10*I-8)"--------";
760 NEXTI:PRINT*DU
770 RETURN
780 FORL=1TOLEN(STR$(A))-1
790 FORI=1TO7
800 J=A(1,X*7+I):K=A(2,X*7+I)
810 M=INT(K/(10^L))*(10^L):N=K-M
820 N=K-M:IFL>1THENN=INT(N/(10^(L-1)))
830 IFM=0ANDN=0THEN870
840 O=J*N:O$=MID$(STR$(O),2)
850 IFO=0THENP=LEN(STR$(M))-1:O$=LEFT$("0000000000",P)
860 PRINT*DU,TAB(10*I-LEN(O$)-(L-1));O$;
870 NEXTI:PRINT*DU
880 NEXTL
890 GOTO740
900 FORI=1TO80
910 A(3,I)=A(1,I)-A(2,I):IFOPTHENGOTO1340
920 NEXTI
930 X$="-":GOTO510
940 FORI=1TO80

950 A(3,I)=A(1,I)*A(2,I):IFOPTHENGOTO1340
960 NEXTI
970 X$="X":Z=4+LEN(STR$(A)):GOTO510
980 FORI=1TO80
990 A(3,I)=A(1,I)/A(2,I):A(3,I)=INT(A(3,I))
1000 A(4,I)=A(1,I)-A(3,I)*A(2,I):IFOPTHENGOTO1340
1010 NEXTI
1020 Z=1+2*INT(LEN(STR$(A))):W=INT(60/(C+2))
1030 PRINT*DU,"    "T$:PRINT*DU
1040 FORX=0TOW-1
1050 IFAP=1THENGOSUB1190
1060 PRINT*DU
1070 FORI=1TO5
1080 PRINT*DU,TAB(16*I-9)"--------";
1090 NEXTI:PRINT*DU
1100 FORI=1TO5
1110 PRINT*DU,TAB(16*I-(LEN(STR$(A(2,X*5+I)))+10));
1120 PRINT*DU,A(2,X*5+I)")"A(1,X*5+I);
1130 NEXTI:PRINT*DU
1140 IFAP<2THENFORI=1TOZ-3:PRINT*DU:NEXTI
1150 REM IFAP=1THENGOSUB35300
1160 NEXTX
1170 IFAP=0THENAP=1:PRINT*DU,CHR$(12):GOTO1030
```

16. Combine statements by concatenation or nesting. E.g., replace

PC=INT(100*A/T+.5):
PC$=MID$(STR$(PC),2)

by

PC$=MID$(STR$(INT(100*A/T+.5)),2)

This saves about 1200+62*P cycles (or 1/1600 seconds on up), where P is the position in the variable table of the eliminated variable, in this case PC. But do this only in critical sections of code, because it is precisely this kind of concatenation that makes APL and Lisp so unreadable. Readability is worth more than 1/1600 seconds. If PC is, say, fiftieth in the variable table, then 70% of the potential time savings can be realised by keeping two statements but replacing PC by X.

17. Functions are faster than subroutines. The cost of a function call over in-line code is 1810+31*P cycles, where P is the function's position in the variable table.

OSI-CALC: SPREADSHEET PROGRAM

OSI-CALC has been a smash hit here at PEEK[65]. Written entirely in BASIC by Paul Chidley of TOSIE, the program gives you a 26 column by 36 row spreadsheet with many features. Don't let the fact that it's written in BASIC fool you. It's VERY FAST.

Each cell can contain text (left or right justified) or numeric data (in floating point or dollar format) or a formula which computes its results based on the contents of the other cells. Formulas can perform addition, subtraction, multiplication or division using cell contents and/or numeric constants. Spreadsheets can be stored on disk, and the program does very nice printing too.

OSI-CALC requires 48K of memory and OS-65D V3.3. Specify video or serial system and mini-floppy or 8" disks. Price $10.00 plus $3.70 shipping ($13.70 total).

18. If the space is available, replace GOSUBs in critical sections of code with a copy of the subroutine. Eliminating a GOSUB 2 saves 3200 cycles. More typically, eliminating a GOSUB 10000, where 10000 is the hundredth line searched, saves 12400 cycles.

19. Remove REMs from a critical section and substitute a block of comment lines before the section.

20. Multiplication is faster than division, so that

```
    FOR I=1 TO 100: A(I)=A(I)/7:
NEXT
```

should be restated

```
    X=1/7: FOR I=1 TO 100:
A(I)=A(I)*X: NEXT
```

21. X+X is faster than 2*X, and X*X is much faster than X^2. Exponentiation and the mathematical functions are very slow.

22. A little time is saved, and the appearance of the program is improved, by deleting the leading colons on blank lines or indented lines. Insert these temporary lines, and RUN 6.

```
    6 FOR I=PEEK(120) + 256 *
PEEK(121) - 1 TO PEEK(122) +
256*PEEK(123)
    7 IF PEEK(I) THEN NEXT
    8 I=I+5: IF PEEK(I)=58 THEN
POKE I,32
    9 NEXT: STOP
```

23. Avoid integer variables; they save no space and execute slower because OSI BASIC has no integer routines. Even A%=NOT A% takes longer than A=NOT A because the NOT routine expects floating-point, resulting in four unnecessary conversions! (This has been fixed in recent Microsoft Basics.) Integer arrays suffer from the same disadvantage but are often worthwhile because they do save space, and if the program suffers from garbage-collection delays, saving space means saving time.

```
1180 GOTO1870
1190 FORI=1TO5
1200 AN$=STR$(A(1,X*5+I)):AO$=STR$(A(3,X*5+I)):AN=LEN(AN$)
1210 AO$=RIGHT$("       "+AO$,AN)
1220 PRINT*DV,TAB(16*I-(8))AO$"r"A(4,X*5+I);
1230 NEXTI
1240 RETURN
1250 INPUT"Number of problems (50 to 200) <50>";N
1260 IFN=0ORN>200THENN=50
1270 FLAG25
1280 OP=INT(1/ASC(OP$)*1000):IFOP$="M"THENOP=OP+1
1290 OP=OP-11
1300 FORI=1TON
1310 IFA>0AND(A+S)/3=INT((A+S)/3)THENGOSUB2190
1320 GOSUB330
1330 ONOPGOTO910,950,990,470
1340 PRINTCHR$(126)CHR$(28)
1350 PRINTT$"'s Average time:"U" Last problem:"T" Total time:"V
1360 T=0
1370 PRINTTAB(LEN(T$)+3)"Right:"A" Wrong:"S
1380 FORX=1TO8-INT(C/2):PRINT:NEXTX
1390 X$="+":IFOP=1THENX$="-"
1400 IFOP=2THENX$="X"
1410 IFOP=3THENX$=")"
1420 ONOPGOTO1430,1430,2260,1430
1430 FORX=1TOC

1440 W$=STR$(A(X,I))
1450 IFX=CTHENW$=X$+W$
1460 PRINTTAB(38-LEN(W$))W$
1470 NEXTX
1480 PRINTTAB(30)"--------"
1490 IFOP=2THENGOSUB1930
1500 POKE15006,0:PRINTTAB(37);
1510 FORF=1TO1E9
1520 IFF=1ANDT>0THENF=T:T=0
1530 PRINTCHR$(0);
1540 E=PEEK(15006)
1550 IFE=3THENFLAG26:STOP
1560 IFE=127THENE$="  ":D=D-2:GOTO1600
1570 IFE=13THENT=F:F=1E9:GOTO1610
1580 IFE<48ORE>57THEN1610
1590 E$=CHR$(E):AN$(D)=E$
1600 PRINTE$;:POKE15006,0:D=D+1:PRINTCHR$(13)TAB(37-D);
1610 NEXTF
1620 FORX=DTO1STEP-1:AN$=AN$+AN$(X-1):NEXTX
1630 IFOP=3THENAN$="":FORX=1TOD:AN$=AN$+AN$(X-1):NEXTX
1640 D=0:AN=VAL(AN$)
1650 IFOP=3ANDAM<>A(4,I)THEN1670
1660 IFAN=A(C+1,I)THENA=A+1:GOSUB2100:GOTO1680
1670 S=S+1:A(15,I)=VAL(AN$):GOSUB2150
1680 V=V+T:U=INT(V/(A+S))
1690 A(14,I)=U:A(13,I)=T:A(12,I)=V:A(11,I)=A(11,I)+1
1700 A(10,I)=A(10,I)+T
1710 AN$="":AM$=""
1720 IFI4THENRETURN
1730 NEXTI
1740 FLAG26
1750 PO$="ADDITION":IFOP=1THENPO$="SUBTRACTION":GOTO1780
1760 IFOP=2THENPO$="MULTIPLICATION"
1770 IFOP=3THENPO$="DIVISION"
```

```
1780 PRINT#DV,TAB(40-INT(LEN(PO$)/2))PO$
1790 PRINT#DV,T$,"Average time:"U,"Total time:"V,"Right:"R,"Wrong:"S
1800 FORJ=1TOC:PRINT#DV,"   PART",:NEXTJ
1810 PRINT#DV,"  ANSWER","    TIME"," AVERAGE","   WRONG","TTL TIME",
1820 PRINT#DV,"ELAPSED","USED"
1830 FORX=1TON
1840 FORJ=1TOC+1:PRINT#DV,A(J,X),:NEXTJ
1850 PRINT#DV,A(13,X),A(14,X),A(15,X),A(10,X),A(12,X),A(11,X)
1860 NEXTX
1870 IFDV>2THENPRINT#DV!
1880 INPUT"Do you want to do more";QS$
1890 IFLEFT$(QS$,1)="Y"ORLEFT$(QS$,1)="y"THENRUN
1900 PRINTCHR$(126)CHR$(28)
1910 PRINTTAB(35)"GOODBY!"
1920 RUN"BEXEC*","~~~~~"

1930 POKE15006,0:PRINTTAB(37);
1940 G=LEN(STR$(A(2,I)))-1:H=37-G
1950 IFG<2THENRETURN
1960 FORF=1TO1E9
1970 PRINTCHR$(0);
1980 E=PEEK(15006)
1990 IFE=3THENFLAG26:STOP
2000 IFE=127THENE$="  ":D=D-2:GOTO2040
2010 IFE=13THENGOTO2060
2020 IFE<48ORE>57THEN2050
2030 E$=CHR$(E)
2040 PRINTE$;:POKE15006,0:D=D+1:PRINTCHR$(13)TAB((H+G)-D);
2050 NEXTF:RETURN
2060 POKE15006,0:PRINT:D=0:G=G-1
2070 IFG>0THENPRINTTAB(H+G);:GOTO2050
2080 T=F:F=1E9:PRINTTAB(30)"--------"
2090 GOTO2050
2100 PRINT:PRINT:FORX=1TO5
2110 PRINTTAB(25)"ABSOLUTELY RIGHT!!!"
2120 :NEXTX:FORX=1TO400:
2130 NEXTX
2140 RETURN
2150 PRINT:FORX=1TO300
2160 IFX/20=INT(X/20)THENPRINTCHR$(7)TAB(20)"* * * !!!WRONG!!! * * *"
2170 NEXTX
2180 RETURN
2190 I1=I:I2=0
2200 FORI3=1TOI
2210 IFA(13,I3)>I2THENI2=A(13,I3):I4=I3
2220 IFA(15,I3)>0THENI4=I3:A(15,I3)=0:I3=I
2230 NEXTI3
2240 I=I4:GOSUB1340
2250 I=I1:I4=0:RETURN
2260 W$=STR$(A(1,I)):Y$=STR$(A(2,I)):DT=1
2270 PRINTTAB(30)"--------"
2280 PRINTTAB(29-LEN(Y$))Y$" "X$W$
2290 PRINTCU$CU$CU$CU$:PRINTTAB(32);
2300 G=LEN(STR$(A(1,I)))-1:H=37-G
2310 POKECC,0
2320 FORF=1TO1E9
2330 IFF=1ANDT>0THENF=T:T=0
2340 PRINTR$;
2350 E=PEEK(CC)
2360 IFE=3THENFLAG26:STOP
2370 IFE=32THENPRINT" ";:AN$(D)=" ":D=D+1:POKECC,0:NEXTF
```

This not only reduces the time taken for one garbage collection, but releases more string space on each collection, so that the time between collections is increased. If FRE(X) is less than a few thousand then the most urgent improvement is to reduce the frequency of collections by making more string space available. Some options are removing spaces and/or comments, giving off some functions to a separate program, and splitting the program in two. If FRE(X) drops to less than 256, the effect on speed is disastrous as BASIC does a garbage collection before every string storage. The available string space is given (without doing a garbage collection) by

```
PRINT PEEK(130) +
256*PEEK(131) - (PEEK(126) +
256*PEEK(127))
```

Finally, a few hints about the really slow and recalcitrant factors: data entry, printing, and disk access.

25. Some time can usually be saved if the programmer talks to the data entry person after the program is running smoothly. Whenever there is a usual reply, particularly Y or N, it should be presented on the screen as the default so that the operator can just hit <RETURN>. Little-used options can be hidden instead of being asked about each time; for example, if the operator needs to change the bank account, he can enter ! in the date column.

26. Not much can be done about printing except persuading management to buy a buffer or a faster printer. If a table has many 0.00 entries, blanks may be acceptable. If there are many column markers |, omitting all or most of them may be acceptable.

Support Your Local OSI Dealer or Vendor

Math Trainer

by Richard E. Reed

Here at my company, we have an 8-processor C*-B using Denver Boards. Some of our terminals are free part of the time, and several of our workers have children who come here after school. A few years ago, I wrote a math program for the C1P which used screen formatting and timed the problems for the students. I decided to port it over to our serial system. It was a virtual re-write. For any of you interested souls out there, here it is.

The program can be operated in either the immediate mode (for on-screen work) or from printed exercise sheets. We will describe the latter mode first. The first input is a title. This usually consists of a person's name, but could be a message like "Sandra's First Quiz". Since the program uses the pseudo random number generator, it needs a random seed. Rather than require this as an input, I chose to use an internal counter to self-select a seed. The code is in lines 10 to 35.

Line 10 disables <CTRL>'C'. The POKE in line 12 enables a character read. Line 14 sets the <CTRL>'C' location to 0 and prints a prompt. The actual counter loop is in lines 15 to 35. In line 15, we allow huge time lapses by ending the count at 1E+8 (note that 65U extensions which disable the EXP function must first be removed). Line 16 lets the system look for a <CTRL>'C'. Line 17 looks for an input. If the <CTRL>'C' location is still zero then we jump to the NEXT increment to our counter. Line 25 checks for a <CR>. If is is present, we set the random seed, set NEXT beyond the loop value and continue with the program.

In line 30 we build the "title" string, print the character on the screen, and re-initialize the <CTRL>'C' location. The rudiments of this loop can be used to perform processing while waiting for a keyboard input. It is used several times throughout this program.

After selecting the Printer mode, the program asks whether you want addition, subtraction, multiplication, or division. It then requests the range of numbers you want to include. If special limitations to the range are required for certain operations, these are called for. The program automatically formats just enough problems to fill an 8-1/2 by 11" page (with work space) and it prints two sets of the page, one with answers. It then terminates with a STOP to the immediate mode.

When using the program in the immediate mode, division has not been implimented. All of the other functions work normally, and the cursor knows where it should be. The user must be careful to enter all answers from right to left, even if the problem is trivial, because that is the order the computer expects to accept digits.

It is assumed that the immediate mode is for the development of skills, therefore the user's time is kept track of which working the problems. Time is only relative since it is the time lapsed performing loops in the program. Every fourth problem is either one the user got wrong or the problem it took him the most time to solve. That way, if the range of numbers is limited to the basic facts, the user can be drilled on them with ever-increasing skill, and attention to those facts which are most difficult.

When the student has finished performing his problems, he will get a printout formatted for wide paper or condensed type which shows; (1) The parts of the problem, (2) The answer, (3) The time spent on the last solution, (4) The average time when the problem was last performed, (5) The last wrong answer given, (6) The total time spent on all repeats of the problem, (7) The total elapsed time when ther problem was last done.

```
2380 IFE=127THENPRINTCHR$(8);:POKECC,0:D=D-1:NEXTF
2390 IFE=13THENT=F:F=1E9:GOTO2410
2400 IFE>47ANDE<58THENAN$(D)=CHR$(E):GOSUB2490
2410 NEXTF

2420 PRINT"r";
2430 POKECC,0
2440 PRINTA$;
2450 EE=PEEK(CC)
2460 IFEE>47ANDEE<58THENAM$=AM$+CHR$(EE):PRINTCHR$(EE);:POKECC,0
2470 IFEE=13THENAM=VAL(AM$):POKECC,0::GOTO1620
2480 GOTO2440
2490 PRINTCHR$(E);:POKECC,0
2500 DS=0:FORLP=1TODT:PRINT:PRINT:PRINT:NEXTLP
2510 PRINTTAB(32+D)
2520 FORFF=1TO1E9
2530 PRINTA$;
2540 EE=PEEK(CC)
2550 IFEE=3THENFLAG26:STOP
2560 IFEE=127THENE$="  ":DD=DD-2:GOTO2620
2570 IFEE=13ANDDS=1THENF=F+FF:FF=1E9:GOTO2630
2580 IFEE=13THENDS=1:PRINT:PRINTTAB(32)LEFT$("--------",D+1)
2590 IFEE=13THENPRINTTAB(32+D);:EF$="":DD=0:POKECC,0:GOTO2630
2600 IFEE<48OREE>57THEN2630
2610 EE$=CHR$(EE):EF$=EE$+EF$
2620 PRINTEE$;:POKECC,0:DD=DD+1:PRINTCHR$(13)TAB(32+D-DD);
2630 NEXTFF
2640 :DD=0
2650 PRINTTAB(33+D-LEN(EF$))EF$;:EF$=""
2660 IFD+2<LEN(W$)THENPRINTMID$(W$,D+3,1);
2670 PRINT
2680 AN$="":FORLP=1TOD+1:AN$=AN$+AN$(LP-1):NEXTLP
2690 DT=DT+1:FORLP=1TODT:PRINTCU$CU$CU$;:NEXTLP:PRINTCU$
2700 PRINTTAB(32)AN$;:POKECC,0:D=D+1:AN$="":RETURN
2710 FORTA=1TO5000:NEXT
```

Book Bonanza!

Sam's Service Manuals
The hardware enthusiast's best friend. These are the only professional guides available for servicing and modifying your OSI equipment. They include full schematics, block diagrams, wave form tracings, parts lists, and diagnostic tips. They were written for the pre-1980 series of OSI systems, but since OSI never has changed that much they are still valuable no matter when your computer was made.

C1P Sam's	Regular: $7.95	Sale: $4.00
C4P Sam's	Regular: $15.00	Sale: $7.50
C2/C3	Regular: $30.00	Sale: $15.00

65V Primer
This is an introductory guide to machine code that shows you how to program your video system using the Monitor ROM. An excellent tutorial on the fundamentals of machine code.
Regular: $4.95 Sale Price: $2.50

Assembler/Editor - Extended Monitor Manual
Until recently, OSI included the Assembler/Editor and Extended Monitor software with all copies of OS-65D. However, even when it was free, there was little documentation accompanying the disks. If you've been looking for instructions on these two programs, this is the book for you!
Regular: $6.95 Sale Price: $3.50

How To Program Microcomputers
By William Barden, this book explains the instruction set of the 8080, 6500, and 6800 series of microprocessors. While not OSI-specific, this book contains many valuable algorithms for solving problems in machine code using the microprocessors available in OSI computers.
Regular: $7.95 Sale Price: $4.00

Professional Computers Set Up and Operations Manual
A valuable guide for installing and using OSI serial systems. Includes an overview of classic OSI software for these systems. The book also provides information on how to program the C3 series using the Z-80 and 6800 microprocessors.
Regular: $8.95 Sale Price: $4.50

User Guides
These are excellent books. They are complete tutorials on all of the standard hardware and software for video systems. Covers many topics not documented anywhere else. If you've been struggling along with just the big blue notebooks, don't wait! Order today!

C1P-MF	Regular Price: $8.95	Sale Price: $4.00
C4P-MF/DF	Regular Price: $8.95	Sale Price: $4.50
C8P-DF	Regular Price: $8.95	Sale Price: $4.50

C1P Programmer's Package
C1P Introductory Manual, C1P User's Guide, C1P Sam's Service Manual, How To Program Microcomputers, ASM/EM Manual, and 65V Primer.
Regular: $42.70 Sale:$20.00 (includes shipping)

C4-C8 Programmer's Package
Same as above, but for C4P-MF, C4P-DF, or C8P-DF (specify).
Regular: $49.75 Sale: $25.00 (includes shipping)

C2/C3/200/300 Programmer's Package
Includes C8P User's Guide, C3 Sam's, How To Program Microcomputers, ASM/EM Reference Manual, BASIC Reference Manual, Professional Computers Setup and Operations Manual.
Regular: $69.95 Sale: $35.00 (includes shipping)

Letters to the Editor

Dear Editor;

For several months I have been trying to get a TEAC 55B disk drive to work with my C1P. I am using the data separator (SASI) which Jim McConkey described in the May, 1983 issue of PEEK[65]. I am also using Steve Hendrix's HEXDOS. All I get when I try to boot the disk is a Venetian blind effect on the prompt and then back to the initial conditions.

Jim McConkey has tried the data separator and HEXDOS disk on his system and everything works fine. I connected the TEAC 55B on a friend's computer which uses the disk drive (NCR DMV) and it worked OK. I have gone through the procedures outlined in the Sams' manual several times and have not been able to determine anything that is not within the specs. Does anyone have any suggestions as to what is preventing my disk drive to boot up?

Robert L. Dingle
657 Dell Ridge Drive
Dayton, OH 45429

Dear Robert,

I'm afraid I can't be of much help beyond suggesting that you also examine Steve McGinnis' recent articles on adding a disk drive to the C1P which were published earlier this year. He suggests some novel methods for getting the drives aligned to the hardware. However, some of the symptoms you describe do not sound like they are necessarily related to the drive or the interface. If you have some non-standard hardware attatched to your system, remove it and try again. Some modifications I have seen don't always decode all 16-bits of memory addresses and this can lead to conflicts and stray writes to RAM. The apparent <RESET> you experience is very suspicious since the system should never return to the ROM due to a failure to boot. I hope others will write to you directly with some more experienced advice. Good luck!

Rick

Dear Editor,

Thanks for being so helpful on the phone today. I have enclosed my order for TRM65U, which I hope will get my C3B talking to my PC/AT.

I am using a null modem, and have tried both the 550 board (populated only for Port 1) and the D&N CA-10X board (only one port again), with no success. I am using TERMCC (TERMNL by Jim Sanders modified by Jim Versace of Community Computers), but everytime I go to the USR vector, it hangs-up and I have to re-boot.

I have used MITE (XMODEM & XON/OFF) and COM1 on the PC/AT but neither of these will respond to the C3B. I have used the D&N card successfully with a HEI punched card reader, so the hardware should not be the problem. This runs at 2400 baud. Is this too fast for TRM65U?

The BASIC program (& data) I am trying to transfer to my PC/AT is a real A/N sort. It used to take 7+ hours to build the three sort indeces. Now I have an OM problem because of the file size (disk I/O was also tremendous). I am hoping a 2.0 MB RAM disk on the PC/AT will give me ample working space. If I get it to work, would it be interesting for PEEK[65]?

A. E. Stark
1925 North Lynn Street Suite 804
Arlington, VA 22209

Dear Mr. Stark,

My first suggestion is to examine the null modem cable to be sure that the PC/AT is getting all of the signals it needs in addition to the data lines. Forgive me for not having the precise information here, but one of these two suggestions may be a big help. Both of these jumpers refer to the DB-25 connector on the PC/AT (the OSI software doesn't need or support these handshaking lines). First try jumping pins 20 and 7. If that doesn't work, remove it and jumper pin 8 to pin 7. Be sure that between the two DB-25's, pin 2 on one goes to pin 3 on the other.

Contemporary smart terminal programs often assume you're using a smart modem. Be sure you've set MITE properly so it isn't waiting for a response from a modem. If MITE has a null modem selection, use it. If not, I'm sure it has a setting something like "OTHER".

I have successfully used TRM65U on a serial system running at 2 MHz at 1200 baud, but I wouldn't guarantee it at anything faster for bi-directional communication. In this case though, since you only need it to go one way, it may very well be that you could transmit files at a much higher baud rate - although the actual throughput will be significantly slower than the baud rate would indicate.

As far as your application goes, I don't know what an "A/N" sort is (although I have a hunch you mean Alpha/Numeric). However, if the file you're using is an OS-DMS Master file, Sanders' SORT/MERGE is widely heralded as a speed demon on OSI systems. Have you ever tried it? Key files will also help you in this regard on the OSI. The large RAM disk on the PC/AT will certainly be a factor, but I would still like to see how the OSI fares in any such application. No matter what you end up doing, I hope you will write to PEEK and describe your experiences. Thanks and good luck!

Rick

Dear Editor,

I was pleased to recieve the latest issue of PEEK[65]. It's arrival reminded me of a couple of questions that I have been wanting to ask. They concern two programs my father has purchased from PEEK. My father, who has not spent much time figuring out these computers, failed to specify serial systems when he ordered OSI-Calc and Edit-Plus. I have OSI-Calc running fairly well, but haven't been able to figure out what to do with the messages that are supposed to be POKEd onto the screen. If you could send the mods that convert it from video to serial, I'd appreciate it. I know that I can make it work well on my system, since I can send the POKEd messages to the monitor scroll rather than the workspace scroll where the rest will have to be sent. Edit-Plus is a bit trickier to work with. I'm not very good with machine code, especially wihtout a good listing to work with. If you could tell me where the polling routine is, I think I'll be able to figure out how to switch that over to the other port. My system is a C0-S, an old C0P that Ron Fial converted to serial for me when I acquired a TEK 4027 terminal. I will be bringing the old polled keyboard with me when I return to school where my system is in storage, so I ought to be able to at least get an idea if the program works, but would much prefer to use my terminal's keyboard.

Daniel J. McDonald

Dear Daniel,

The serial version of OSI-Calc that I came up with from Paul Chidley's original version totally emulates the video version. It just uses the operating system and a small piece of machine code for console inputs instead of doing direct POKEs to the video display. In fact, the "serial" version would actually run on a video system transparently. In any event, the mods are rather extensive, so I'll simply send you a copy of the current software.

I'm surprised you're having trouble with Edit-Plus. It supports both serial and video systems without modification. However, it decides which kind of system it is running on based on the operating system current input flag - upon reflection, probably a poor method of doing this, but it has held me in good stead 'til now. If your system's montior ROM is not strapped as a serial system, Edit-Plus will decide its running on a video system. If this is the cause of your problems, then the solution is to boot up on an OS-65D V3.3 disk that properly handles your system and then manually execute Edit-Plus. To do this, boot up as you normally do and get to 65D's "A*" prompt. Insert the Edit-Plus diskette in the "A" drive. Then enter the following commands;

```
CA 0200-05,1
CA 0D00-06,1
CA 1000-07,1
GO 0200
```

This will load and execute Edit-Plus and since 65D's internal pointers will have been set up by your software, Edit-Plus should recognize your hardware properly.

Rick

Dear Editor;

I read with interest your discussion in PEEK[65] of a new OS-65D. The one overall thought I have is that continuing OS-65D is not moving the OSI world into the 80's but is perpetuating a 70's kludge. While I admit to being somewhat prejudiced, I think the answer is DOS/65.

DOS/65 at its beginning and as it stands today (Version 2.1) is a far superior product and solves all the problems that OS-65D has. To be specific;

(1) DOS/65 uses a fully user-modifyable software module (called SIM but analogous to the CP/M and MS-DOS BIOS) to define the hardware interface. While the standard OSI SIM supports the normal OSI hardware devices, it can be adapted to handle almost any hardware configuration desired. For example, if you wanted to use 40 or 80 track double-sided disk drives, all DOS/65 software would be able to use the full capacity of those drives with very few changes to the SIM. When the hardware changes, only SIM changes!

(2) DOS/65 does use a transparent, fully dynamic file allocation scheme for all disk formats. The user need never be concerned about or even know where on the disk a given piece of software or data is stored.

(3) DOS/65 comes with a fully-featured, semi-compiled BASIC (BASIC-E/65) that in many respects is much better than Microsoft BASIC. I have also adapted the OSI "9-digit" version of Microsoft BASIC to the DOS/65 environment. That adaptation was accomplished by completely disassembling BASIC and then restructuring it so that it is "patch-free". One problem is how I can get this product to existing OSI licensees without violating copyright laws - perhaps as a "patch" to OSI BASIC. Because of this problem, the DOS/65 version of Microsoft BASIC has not been distributed to anyone.

(4) DOS/65 comes with the following software: Assembler, Editor, Debugger, Sysgen, Disk Format, Disk Test, SIM source code, BASIC-E/65, XMODEM compatible Comm package, Disk Copy, File Move, Loader, example BASIC-E programs, OSI Transfer (to read OSI BASIC, ASM, or ASCII files), OSI Directory, (and more).

Again, I admit to being prejudiced, but in my mind DOS/65 is what OSI needed all along. I welcome the chance to discuss this with you. Drop me a note or leave me a message on CompuServe (74435,1213).

Richard A. Leary
Micro Systems Technology
450 Forrest Avenue #D312
Norristown, PA 19401

Dear Rich,

DOS/65 is indeed a powerful operating system. It brings a lot of the improvements that other systems have enjoyed for years now to the OSI community, and the tools you have developed to augment the system make it easy to move software from the 65D environment to it.

The new 65D project has languished lately for many reasons. There clearly is no massive outcry for it at this time. Very few readers have written with suggestions or offers to help. But when the new 65816 systems begin to filter through the community, I am sure this will change. Therefore, I think it may be wise to consider writing for that environment to make whatever we come up with that much more enticing. If this project is to get off the ground, it will take the support of the users. I also believe that for it to take hold, it will have to be free save for the price of a PEEK subscription, or a user group membership, or for the cost of downloading it from CompuServe.

Rick

Dear Editor,

There is one single item in OS-65U V1.2 that prevents us from expanding the use of OSI in our business. That item is the money mode. It defaults to a 14 character minimum width column, thus limiting use to 9 columns on standard width paper. This is inadequate for journal and ledger sheets. Does anyone have a fix to allow us to choose our own column width?

Stuart Hilborn
Fuel Injection Engineering Company
25891 Crown Valley Parkway
South Laguna, CA 92677

Dear Stuart,

I'm not sure what the real problem is that you're running into. If it is the accuracy of the money mode in 65U, that can be a problem just because of the way the BASIC interpreter works with floating point numbers. It is known to be inaccurate in many situations. Most of the solutions to this problem is to convert all values/variables which refer to dollars into pennies, thus removing the fractional part of all values into integers which can then be converted into strings and displayed accurately in dollars.

If, however, your problem is with the columnar positioning of the numbers when using $R and $L, then you're going to have to convert the numbers to strings or use some other programming method to give you discreet control over the output. Again, converting dollars into pennies seems to be the solution for you.

Rick

Editor;

Congrats on taking over PEEK[65]. If I can be of assistance, let me know. I have a C8 with Spinwriter.

Re: OS-65D - Serial input should be interrupt driven. It takes one wire to bring the IRQ line out from the 6850. I have done the code in FORTH. Perhaps you would care to adopt it? My version has a 256 key buffer.

I have been working with the Atari ST and find their file handlers excellent. For random access, the function LSEEK allows the programmer to seek a specific byte in the file - absolute or relative. I do any needed relative calculations in the application program.

Charles Curley
146 Lockwood Lane
Scotts Valley, CA 95066

Dear Charles,

Thanks for your offer to help with PEEK. I'd certainly like to see your code for IRQ inputs. I hope you'll write it up and send it in to PEEK. As I've said before, the only reason my terminal software hasn't used interrupts is because vanilla OSI hardware doesn't support them from the 6850 ACIAs. I agree that the modification is simple, although it gets a lot more complicated in serial systems where the ACIAs are not on the CPU board. If you get a chance, I hope you'll also include a diagram for modifying the 505 board for interrupts from the 6850.

As to the Atari's file handlers, I couldn't agree more. Like most contemporary operating systems, the Atari's file handlers are enhancements to our old friend CP/M. There's no doubt in my mind that any enhancement to OS-65D should include operating system level file access support so that any language, not just BASIC, can enjoy the benefits.

Rick

Dealers: Have you signed up for PEEK's CO-OP Ad program?

Practical Insights to Programming Sorts under OS-65U

by Roger Clegg
Data Products Maintenance

(Editor's Note: This didn't originate as an article, but merely a demonstrative program that Roger made available to us. I've left it largely intact to preserve Roger's thoughts which were sprinkled in the program listing. That's why the text is somewhat terse.)

All sorts in Listing 1 will run faster if the variables used are listed in the first line of the program.

For an alphabetic (ASCII) sort, just substitute A$( ) for A( ), and K$ for K. To sort on two fields at once, say DEPT$ and NAME$, set A$(I)=DEPT$+NAME$ for each I. To sort on two numeric fields at once, say CUST and INV, set A(I)=M*CUST+INV for each I, where M is bigger than any invoice number. The maximum A(I) must be less than 4,294,967,296 if the last digit is critical. To sort on an alphabetic field and a numeric field, say NAME$ and INV, set

```
A$(I)=NAME$+RIGHT$("      "+STR$(INV),6)
```

or a similar formula.

Over 3000 strings can be imperfectly sorted by storing the first seven letters in a numeric array as in Listing 2. This algorithm treats "A" and "a" identically, and all non-alphabetic characters identically.

CHOOSING AN ALGORITHM

(1) INDIRECT VS. DIRECT
Usually one needs to keep track of the original order, so an index or pointer array P( ) or P%( ) is needed as well as the main array A( ). Before sorting, set P(1)=1, P(2)=2, etc. You can carry the pointer array along passively (the direct sorting method) or use it to do the work and leave the main array unsorted (the indirect method). After an indirect sort, you read the main array as in line 80.

```
1 REM *********************    S O R T E R    **************************
2
10 I=0: J=0: K=0: G=0: T=0: L=0: U=0: S=0: CR$=CHR$(13)
20 N=1000: REM Number to sort
30 DIM P(N),A(N),L(20),U(20),RS(5)
40 FOR I=1 TO N: P(I)=I: A(I)=N*RND(1): NEXT
50 INPUT"QUICKSORT OR SHELL-DPM SORT";R$
60 IF R$="Q" THEN L=1: U=N: S=0: GOSUB 800: GOSUB 100: GOSUB 900
70 IF R$="S" THEN G=N: GOSUB 200
80 PRINT CHR$(7): FOR I=1 TO N: PRINT A(P(I)): NEXT: REM Indirect sort
90 END
97
98 REM   INDIRECT QUICKSORT  (1000 elements in 100 seconds at 2 Mhz)
99
100 PRINT L;CR$;: NULL P(U);P((L+U)/2): J=L-1: K=A(P(U))
110 FOR I=L TO U: IF A(P(I))<=K THEN J=J+1: NULL P(J);P(I)
120 NEXT: IF J+1<U THEN S=S+1: L(S)=J+1: U(S)=U
130 U=J-1: IF L<U GOTO 100
140 IF S THEN L=L(S): U=U(S): S=S-1: GOTO 100
150 RETURN
160
190 REM   INDIRECT SHELL-DPM SORT  (without SWAP, 140 seconds)
199
200 G=2*INT(G/7)+1: PRINT G;CR$;: FOR J=1 TO N-G: T=P(J+G): K=A(T)
210 FOR I=J TO 1 STEP -G: IF A(P(I))>K THEN P(I+G)=P(I): NEXT
220 P(I+G)=T: NEXT J: IF G>1 GOTO 200
230 RETURN
240
290 REM   DIRECT QUICKSORT  (110 seconds)
299
300 PRINT L;CR$;: J=(L+U)/2: NULL A(J);A(U);P%(J);P%(U): J=L-1: K=A(U)
310 FOR I=L TO U: IF A(I)<=K THEN J=J+1: NULL A(J);A(I);P%(J);P%(I)
320 NEXT: IF J+1<U THEN S=S+1: L(S)=J+1: U(S)=U
330 U=J-1: IF L<U GOTO 300
340 IF S THEN L=L(S): U=U(S): S=S-1: GOTO 300
350 RETURN
360
390 REM   DIRECT SHELL-DPM SORT  (using SWAP, 130 seconds)
399
400 G=2*INT(G*.22)+1: PRINT G;CR$;: FOR J=1 TO N-G: FOR I=J TO 1 STEP
-G
410 IF A(I)>A(I+G) THEN NULL A(I);A(I+G);P%(I);P%(I+G): NEXT
420 NEXT J: IF G>1 GOTO 400
430 RETURN
440
450
800 REM   ENABLE "SWAP" COMMAND

810                 RS          9029
820 FOR I=0 TO 3: AS(I)=PEEK(9028+I): NEXT: REM Save reserved word
830 RS(4)=PEEK(8738): RS(5)=PEEK(8739):     REM Save dispatch address
840 POKE 9025,83: POKE 9026,87: POKE 9027,65: POKE 9028,208: REM "SWAP"
850 POKE 8738,255: POKE 8739,95: RETURN:    REM SWAP code at 24576
860
900 REM   DISABLE "SWAP" COMMAND
910
920 FOR I=0 TO 3: POKE 9025+I, RS(I): NEXT
930 POKE 8738,RS(4): POKE 8739,RS(5): RETURN
940
```

Listing 1

You must choose an indirect sort if you are sorting strings and the SWAP verb is not available. This program contains code for the SWAP verb, enabled by the routine at 800. If SWAP is not available you can substitute T=A(I): A(I)=A(J): A(J)=T. But in a direct string sort this causes garbage-collection delays.

An indirect sort is also preferable if you have two or more related arrays, say accounts AC$(x) and amounts AM(x). A direct sort would rearrange one array but not the other.

A direct sort is preferable when you are sorting certain records from a file, as you can use the pointer array P%(x) for the record numbers, so that P%(1)=6, P%(2)=8, say.If the indirect method if necessary, then a third array R%( ) is needed for the record numbers, and after sorting they can be read in order as R%(P%(1)), R%(P%(2)), etc.

The indirect Shell-DPM runs 6% slower, and the indirect Quicksort 2% slower, if an integer array P%(x) is used for the pointers.But an integer array will save 3*N bytes of memory, and in string operations such as reading a file before a string sort, it will save time by making garbage collections less frequent.

(2) QUICKSORT VS. SHELL-DPM
The ideal situation for Quicksort is when the array is randomly arranged and has few or no duplicates.If you are sure the array is random you can speed up the sort 7% to 10% by eliminating the SWAP in lines 100 and 300, which chooses the middle element as the "pivot" in case the array is partially sorted.

A Shell-DPM sort is a much safer choice if there may be a number of identical elements. Zeros and null strings are particularly disastrous: if there is a block of zeros in the middle of the array, for example, the above version of Quicksort will make almost no progress for several minutes. You may also need to consider whether the array is sometimes zeroed out. For example, if you are sorting a customer file by sales year-to-date, the

```
30 DIM A(N),P%(N): K=27: L=32: M=64
40 FOR I=1 TO N: INDEX<1>=120*I: INPUT%1,NAME$: Y=0
41  IF LEFT$(NAME$,9)="AMERICAN " THEN NAME$="AME"+MID$(NAME$,10)
50  FOR J=1 TO 7: X=ASC(MID$(NAME$,J))-M: IF X>L THEN X=X-L
60  IF X<0 OR X>=K THEN X=0
70  Y=Y*K+X: NEXT J
80  A(I)=Y: P%(I)=I
90 NEXT I
```

Listing 2

```
1 REM *********************  M I S C  ********************************
2
3 REM   Miscellaneous useful routines
4 REM   For sorting routines see SORTER
5 REM   This program is sometimes useful in the direct mode for finding
6 REM   a day of the week: LOAD"MISC": X$="6/6/44": GOSUB 800: ? DAY$
7
100 REM   A GENERALISED MONEY FORMATTING ROUTINE
101
102 REM   "GOSUB 110" returns X$ with two decimal places and no leading
103 REM   blank,  unless padded to length W.
104 REM   "GOSUB 120" prints X, left justified with either leading
105 REM   blank or minus, then $ sign, then number with 2 places.
106 REM   "GOSUB 130" and "GOSUB 140" print X right justified,  with
107 REM   rightmost digit at TA-1, with or without | following.
108
110 W%=1
120 L%=1
130 C%=1
140 X$=STR$(INT(X*100+.5)/100): IF W% THEN L%=0: C%=0
150 IF ASC(RIGHT$(X$,2))=46 THEN X$=X$+"0"
160 IF ASC(RIGHT$(X$,3))<>46 THEN X$=X$+".00"
170 IF L% THEN PRINT#D,LEFT$(X$,1)"$"MID$(X$,2);: L%=0: C%=0: RETURN
180 IF ASC(X$)=32 THEN X$=MID$(X$,2)
190 IF W% AND LEN(X$)<W THEN X$=RIGHT$("              "+X$,W)
200 IF W% THEN W%=0: RETURN
210 PRINT#D,TAB(TA-LEN(X$))X$;: IF C% THEN PRINT#D,"|";: C%=0
220 RETURN
230
240 D  is the output device number
250 W  is the minimum width, if wanted, for GOSUB 110.
260 TA is the tab for the right-justification in GOSUB 130 or 140.
270
280
300 REM   ACCEPTS X, RETURNS X$ WITH TWO DECIMAL PLACES
310
320 X$=STR$(INT(X*100+.5)/100): IF ASC(X$)=32 THEN X$=MID$(X$,2)
330 IF ASC(RIGHT$(X$,2))=46 THEN X$=X$+"0": RETURN
340 IF ASC(RIGHT$(X$,3))=46 THEN RETURN
350 X$=X$+".00": RETURN
360
370
400 REM   PRINTS X WITH TWO DECIMAL PLACES, RIGHT-JUSTIFIED AT TA-1
410
```

Quicksort will be fine in December, but hopelessly slow in January. You may be able to avoid the problem by sorting on two fields at once, or by saying:

```
IF A(I)=0 THEN A(I)=RN/100000
```

where RN = record number. This may be desirable anyway, to keep the duplicates in a fixed order.

```
420 X$=STR$(INT(X*100+.5)/100): IF ASC(X$)=32 THEN X$=MID$(X$,2)
430 IF ASC(RIGHT$(X$,2))=46 THEN X$=X$+"0"
440 IF ASC(RIGHT$(X$,3))<>46 THEN X$=X$+".00"
450 IF TA THEN PRINT#D,TAB(TA-LEN(X$)) X$;
460 RETURN
470
480 If the column width is adequate then the second half of line 420,
485 which strips the leading blank,  can be omitted.
490
495
500 REM   ACCEPTS X$ IN BASE 2, 8, 10 or 16,  RETURNS DECIMAL X
501
502 REM   Prefix $ or H denotes Hexadecimal
503 REM   Prefix & or O denotes Octal
504 REM   Prefix % or B denotes Binary
505 REM   No prefix denotes Decimal
506
510 X=ASC(X$): IF X=36 OR X=72 THEN BASE=16: GOTO 550
520 IF X=38 OR X=79 THEN BASE=8: GOTO 550
530 IF X=37 OR X=66 THEN BASE=2: GOTO 550
540 X=VAL(X$): RETURN
550 X=0: FOR II=2 TO LEN(X$)
560 Y=ASC(MID$(X$,II))-48: X=X*BASE+Y+7*(Y>9): NEXT: RETURN
570
580
600 REM   ACCEPTS DECIMAL X AND BASE,  RETURNS X$
610
620 X$=""
630 Y=INT(X/BASE): Z=X-Y*BASE: X$=MID$("0123456789ABCDEF",Z+1,1)+X$
640 IF Y THEN X=Y: GOTO 630
650 RETURN
660
670 The above two routines handle the natural numbers only.
680
690
700 REM ACCEPTS NAME, RETURNS SURNAME (FLG on) OR 2 INITIALS & SURNAME
710
720 FOR II=LEN(NAME$)-2 TO 1 STEP-1: IF MID$(NAME$,II,1)<>" " THEN NEXT
730 X$=MID$(NAME$,II+1,2): IF X$="JR" OR X$="SR" OR X$="II" THEN NEXT
740 K=II: X$=MID$(NAME$,K+1): IF FLG THEN RETURN
750 FOR II=2 TO K
760 IF MID$(NAME$,II,1)<>" " OR MID$(NAME$,II+1,1)=" " THEN NEXT
770 Y$=MID$(NAME$,II+1,1)+". ": IF II>=K THEN Y$="   "
780 X$=LEFT$(NAME$,1)+"."+Y$+X$: RETURN
790
795
800 REM   CONVERTS DATE TO NUMBER SINCE 1/1/1900, AND GIVES DAY OF WEEK
810
820 DIM M(12),D$(6): C4=365.25: M$=" 31283130313031313031303l"
825 X=0:FOR I=1 TO 12: M(I)=X+.01*I-.025: X=X+VAL(MID$(M$,I*2,2)): NEXT
830 D$(0)="Sunday": D$(1)="Monday": D$(2)="Tuesday": D$(3)="Wednesday"
835 D$(4)="Thursday": D$(5)="Friday": D$(6)="Saturday"
840 REM  The above lines should be in initialization
845
850 X=3: IF MID$(X$,3,1)="/" THEN X=4
860 X=INT(VAL(MID$(X$,X))+M(VAL(X$))+C4*VAL(RIGHT$(X$,2)))
870 DAY$=D$(X-7*INT(X/7))
880 RETURN
890
895
```

transmit selected fields in records from OS-DMS Master files with sorts. Includes utilities to print captured text files or to convert them into WP-3/Edit-Plus or BASIC files. $50.00

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

Edit-Plus
Styled after WP-3-1, although not quite as powerful, Edit-Plus allows composing and editing WP-3 compatible files and to have those files printed as formatted text. Edit-Plus uses line-oriented editing, as opposed to the screen editing of WP-3, and also allows global search and replace. Edit-Plus fixes problems in WP-3 including pagination, inputs from the console, and file merging(selectable line numbers from the merged file). Edit-Plus can perform a trivial right-justification, but it does not support true proportional spacing. Requires OS-65D V3.3. or OS-65U V1.44 (specify) $40.00

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

Data-Plus Nucleus
Data-Plus Nucleus is a replacement package to the OS-DMS Nucleus from OSI. All of the programs from the original except SORT have been duplicated and enchanced and new software, the MC-DMS Interface, has been added. The name "MC-DMS"

```
900 REM   CONVERTS DAY NUMBER SINCE 1/1/1900 INTO DATE
901
902 REM   Initialization of M( ) and C4 are required as above.
910
920 Y=INT(X/C4): Z=X-Y*C4: FOR M=1 TO 11: IF M(M+1)<Z THEN NEXT
930 D=INT(Z-M(M))+1
940 DATE$=MID$(STR$(M),2)+"/"+MID$(STR$(D),2)+"/"+RIGHT$(STR$(Y+100),2)
950 RETURN
960
970
1000 REM   ACCEPTS DATE WITH SLASHES, RETURNS DATE IN WORDS
1010
1020 INPUT"Date (M/D/Y) ";DATE$
1030 X=VAL(DATE$): IF X<1 OR X>12 THEN PRINT CHR$(7);: GOTO 1020
1040 DATA January,February,March,April,May,June,July,August,September
1050 DATA October,November,December:RESTORE: FOR II=1 TO X:READ X$:NEXT
1060 X=3: IF MID$(DATE$,3,1)="/" THEN X=4
1070 DATE$=X$+STR$(VAL(MID$(DATE$,X)))+", 19"+RIGHT$(DATE$,2): RETURN
1080
1090
2000 REM   ROUTINES FOR HANDLING MONEY ACCURRATELY UP TO $32 BILLION
2001
2002 REM   These routines split a dollar amount X$ into two components:
2003 REM   X% for the millions, and X for the remainder.  This avoids
2004 REM   inventing new variable names.  The routines assume that
2005 REM   M=1000000  has already been defined.
2006
2010 REM   ACCEPTS X$, RETURNS X% AND X
2020
2030 IF ASC(X$)=32 THEN X$=MID$(X$,2): GOTO 2030
2040 Z=1: IF LEFT$(X$,1)="-" THEN Z=-1: X$=MID$(X$,2)
2050 FOR II=1 TO LEN(X$): IF MID$(X$,II,1)<>"." THEN NEXT
2060 IF II<8 THEN X=Z*VAL(X$): X%=0: RETURN
2070 X=Z*VAL(MID$(X$,II-6)): X%=Z*VAL(LEFT$(X$,II-7)): RETURN
2080
2100 REM   ACCEPTS X% AND X,  RETURNS X$, AND PRINTS X$ IF TA<>0
2110
2120 Z=INT(ABS(X/M))*SGN(X): IF Z THEN X=X-M*Z: X%=X%+Z
2130 IF X% THEN Z=SGN(X%): IF SGN(X)=-Z THEN X=X+M*Z: X%=X%-Z
2140 X$=STR$(INT(X*100+.5)/100): IF ASC(RIGHT$(X$,2))=46 THEN X$=X$+"0"
2150 IF ASC(RIGHT$(X$,3))<>46 THEN X$=X$+".00"
2160 IF X% THEN X$=STR$(X%)+RIGHT$("000000"+MID$(X$,2),9)
2170 IF ASC(X$)=32 THEN X$=MID$(X$,2)
2180 IF TA THEN PRINT*D,TAB(TA-LEN(X$)) X$;
2190 RETURN
2195
2200 REM   PERFORMS  T$=A$+B$
2210
2220 X$=A$: GOSUB 2000: A=X: A%=X%
2230 X$=B$: GOSUB 2000: X=X+A: X%=X%+A%: GOSUB 2100: T$=X$
2240
2300 REM   PERFORMS  T$=0: FOR I=1 TO N: T$=T$+A$(I): NEXT
2310
2320 T=0: T%=0: FOR I=1 TO N: X$=A$(I): GOSUB 2000
2330 T=T+X: Z=INT(ABS(T/M))*SGN(T): T=T-Z*M: T%=T%+X%+Z
2340 NEXT I: X=T: X%=T%: GOSUB 2100: T$=X$
2350
2360
2370
3000 REM   ALTERNATIVE FOR MONEY UP TO $42,949,672.95
3001
```

stems from the extensive use of machine code support built into the utilities to replace slower, BASIC code. Features include; (1) MC-DMS Interface code supports up to 8 Master files simultaneously without requiring OPEN/CLOSE commands under Level 3 at every file access. The only 65U software support needed for Level 3 file access is semiphores, and it does not conflict with any software transients like COMKIL. This produces a significant increase in speed. READ, WRITE, and FIND commands operate on the field level. FIND skips over embedded garbage between fields, and automatically stops on the last record in the file. (2) Machine code DIR utility. Ultra-fast. Automatic paging. ^C interrupt. Can selectively list by file type or can search for file name matches with wildcards. (3) Machine code file manager. Creates, deletes, or renames files in a flash. The file manager is linked to the Master/Key file creation utility. (4) Machine code file transfer/merge. Grabs up to 30 records per pass. Single/dual drive. Fully selectable field specifications. Also allows searching for matches in source and destination files for linked merges. (5) Machine code single/dual drive floppy diskette copier. Moves up to 7 tracks per pass. (6) Disk-based mailing label printer. Stores printing format designs on disk. Selectable fields and record range, Key file access, searches, and more. (7) Disk-based report writer. Stores report format designs on disk. Same features as above, but with formatted columns by type and width. (8) Edit-Plus 65U. Most of the same features as the 65D version, but with a smaller workspace. Suitable for correspondence and form letters. (9) Data-Plus Mail Merge. Complete documentation allows implimenting the MC-DMS Interface into your own applications. $150.00

HAVE YOU RENEWED YOUR SUBSCRIPTION?

Don't Miss an Issue! Renew Now!

```
3002 REM  If money is kept in cents, addition and subtraction will be
3003 REM  accurate to the above figure, but a special output routine is
3004 REM  required.  FLAG 30 can be set to catch overflows.
3005
3010 Y=X: Z=INT(ABS(X/1E9))*SGN(X): IF Z THEN Y=Y-Z*1E9
3020 X$=STR$(Y/100): IF ASC(RIGHT$(X$,2))=46 THEN X$=X$+"0"
3030 IF ASC(RIGHT$(X$,3))<>46 THEN X$=X$+".00"
3040 IF Z THEN X$=STR$(Z)+RIGHT$("0000000"+MID$(X$,2),10)
3050 IF ASC(X$)=32 THEN X$=MID$(X$,2)
3060 IF TA THEN PRINT#D,TAB(TA-LEN(X$)) X$;
3070 RETURN
3080
3090
3100
4000 REM   ACCEPTS X (DOLLAR AMOUNT), RETURNS X$ IN WORDS
4010
4020 DIM W$(27): GOSUB 4300: REM This should be done at initialization
4030 X=INT(X*100+.5)/100: IF X<=0 OR X>1E7 THEN X$="** VOID **": RETURN
4040 X$=""
4050 Y=INT(X/1E6):  IF Y THEN X=X-Y*1E6:  GOSUB 4200: X$=X$+"MILLION "
4060 Y=INT(X/1000): IF Y THEN X=X-Y*1000: GOSUB 4200: X$=X$+"THOUSAND "
4070 Y=INT(X): IF Y THEN X=X-Y: GOSUB 4200
4080 IF X$="" THEN X$="ZERO "
4090 IF X$="ONE " THEN X$="ONE DOLLAR ": GOTO 4110
4100 X$=X$+"DOLLARS "
4110 IF X<.005 THEN X$=X$+"AND NO CENTS": RETURN
4120 Y=INT(X*100+.5): GOSUB 4200
4130 X$=X$+"CENT": IF X>.015 THEN X$=X$+"S"
4140 RETURN
4150
4200 Z=INT(Y/100): IF Z THEN X$=X$+W$(Z)+" HUNDRED ": Y=Y-Z*100
4210 IF Y=0 THEN RETURN
4220 IF X$<>"" THEN X$=X$+"AND "
4230 IF Y<21 THEN X$=X$+W$(Y)+" ": RETURN
4240 Z=INT(Y/10): X$=X$+W$(18+Z)
4250 Y=Y-Z*10: IF Y THEN X$=X$+"-"+W$(Y)
4260 X$=X$+" ": RETURN
4270
4300 DATA ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,EIGHT,NINE,TEN,ELEVEN
4310 DATA TWELVE,THIRTEEN,FOURTEEN,FIFTEEN,SIXTEEN,SEVENTEEN,EIGHTEEN
4320 DATA NINETEEN,TWENTY,THIRTY,FORTY,FIFTY,SIXTY,SEVENTY
4325 DATA EIGHTY, NINETY
4330 RESTORE: FOR I=1 TO 1000: READ W$(1): IF W$(1)<>"ONE" THEN NEXT
4340 FOR I=2 TO 27: READ W$(I): NEXT: RETURN
4350
4360
5000 REM   PRIME NUMBER GENERATOR
5001
5002 REM   Finds all primes less than 16K, using sieve of Eratosthenes
5010
5020 I=0: K=0: U=1: P=0: S=SQR(16384): N=16384: DIM F%(N/2): PRINT 2
5030 FOR I=1 TO N/2: IF F%(I) THEN NEXT: END
5040 P=I+I+1: PRINT P: IF P>S THEN NEXT: END
5050 FOR K=(P*P-1)/2 TO N/2 STEP P: F%(K)=U: NEXT: NEXT: END
```

Software Spectacular!

C1P/Superboard Cassettes

OSI Invaders	Hangman	Star Trek
Biorhythm	Zulu 9	Racer
SpaceWar	Add Game	Advertisement
Basic Math	High Noon	Tiger Tank
Hectic	Annuity I	Math Intro.
Cryptography	Sampler	

Assortment of 10 for just $20.00!

Specify your preferences, but due to limited quantities, some substitutions will be made.

C4P/C8P Cassettes

Statistics I	Frustration	Space War	Battleship
Annuity II	Mastermind	Trig. Tutor	Powers
Bomber	Loan Finance	Star Trek	Zulu 9
Stock Market	Annuity I	Math Intro	Mathink
Metric Tutor	A.C. Control	Blackjack	High Noon
Electronics Equ.	Star Wars	Math Blitz	Calendar
Prgmble. Calc.	Checking Acct.		

Sargon II Chess Software

Disk version for C8, C4, or C1 (specify)
Regular $34.95 Sale Price $15.00

Cassette version for C8, C4, or C1 (specify)
Regular $29.95 Sale Price $10.00

Extended Monitor

Cassette version for all systems
Regular $50.00

Sale Price $15.00

Classified AD$

Maxell 8" DSDD diskettes. Not $49.50. Not $39.50. But $35.00 per box. Limit 10 boxes per customer. Send check or money order to Artiface Software, Box 8284, Pittsburgh, PA 15218

IOMEGA 10MB Cartridges. Not $95.00.Not $85.00. Not even $75.00, but just $66.50 each (minimum 3). Limit 10. We prepay postage to 48 states. Send check or money order to Artiface Software, Box 8284, Pittsburgh, PA 15218

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

Have you got something to sell? Make a big splash with a classified ad in PEEK[65]. Ads cost 35 cents per word, not counting price words.

Watch This
Space Grow!

PEEK (65)
The Unofficial OSI Users Journal
P.O. Box 586
Pacifica, CA 94044
415-993-6029

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

DELIVER TO: 1/87

GOODIES for OSI Users!

PEEK (65)
The Unofficial OSI Users Journal

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

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

POSTAGE MAY VARY FOR OVERSEAS

Name ______________________
Street ______________________
City ____________ State ________ Zip ________