PEEK[65]

March-May, 1987
Volume 8, No. 3-5

The Unofficial OSI Journal

Column One

Things have been moving once more back in the East. This is most welcome news. Herewith are the highlights from a letter recently received;

AN OPEN LETTER TO DISTRIBUTORS AND DEALERS OF OSI AND DBI COMPUTER SYSTEMS

We are pleased to announce that, as of April 1987, the complete technology of Ohio Scientific (OSI), including trade names, copyrights, and manufacturing rights, has been purchased by DevTech Corporation (DBI) of Denver, Colrado. As you know, OSI and DBI have long supported a common market.

Our primary purpose in this acquisition is to restore OSI products to the marketplace as quickly as possible. We are acutely aware of the difficulties caused by policies of former OSI corporate owners, and we know that the prolonged interruption in supply must be resolved on an urgent basis. DBI's excellent professional team is now meeting this challenge.

Although the Aurora, OH facilities have been permanently closed, Jim Cross will be maintaining general sales offices (in) Chagrin Falls, OH. In addition, Thomas Jablonski has joined the new effort as applications and support specialist for the 700 series.

Those of you who have been purchasing DBI products already know Mike Ammon as general manager of the Denver manufacturing operation.

Perhaps the worst problem historically with OSI has been poor communications, both at the corporate and market levels. I wish to emphasize that our policy is precisely the opposite. Open communication channels are mandatory in any good business activity.

Meanwhile, we will keep you informed by bulleting concerning resumed production schedules, new products, and other developments of interest. We appreciate your patience, tolerance, and dedication to the OSI and DBI products, and we hope you will join with us in a brighter future.

Cordially,
F. Mark Bojarzin
President
DevTech Corporation

I sincerely congratulate everyone involved in this development. It's the best thing that could have happened to the user community. PEEK[65] hasn't had a chance to develop lines of communication with the new company yet, but I certainly intend to do so, and I apologize

Continued on page 14

Inside This Month

Livesay's 65816 page 2
Tiny Compiler page 6
New DOS & Stuff page 15
Simple 65D Accounting page 19
Terminal for CIS page 34

The New 65816 CPU with Double-Density Disk Controller Boards

by David Livesay

As some of you may know, I have in the past few years worked on a system which adapts a 68000 system to the OSI as an attatched processor. During this time, I have also been working on two boards which should be of particular interest to the hobbiest and business users of OSI systems. The new boards are first of all, a new CPU board using the 65816 microprocessor and secondly, a combined SCSI controller and double-density floppy disk controller. Both boards have been designed to offer both high-quality and a reasonable cost to the OSI user. All of the IC's are on sockets and ribbon cable headers are provided for all I/O ports. The configuration of the board connectors allows the use of readily available printer and serial cables.

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.

The 65816 CPU Board

The processor card is a 48-pin OSI-compatible board with the following features:

(1) 65816 microprocessor running at 3 or 4 MHz.
(2) 256K of RAM
(3) 128K of ROM
(4) 8 or 16K Monitor PROM
(5) Parallel printer port
(6) Standard OSI serial port
(7) Spare serial port
(8) OSI type disk controller with OSI real-time controller.
(9) Connector for small plug-in board with math chip, clock, and software interrupt controller.
(10) New expansion bus
(11) Interface for DTACK 680XX system
(12) Usable in any single-user 48-pin system
(13) Use as stand-alone serial system by adding power supply and disk drive.

The CPU

The CPU is a 65816 running at 2, 3, or 4 MHz. A power-on reset circuit is built in and is also connected to the 48-pin bus and the new expansion bus. The clock circuit includes a wait state controller to slow down the clock for slow devices.

48-pin OSI Bus

This is the standard OSI 48-pin bus except that the power-on reset circuit is connected to pin 13 and a disable circuit (described below) is connected to pin 12. All of the address and data lines are fully buffered.

New Bus

A new bus has been built into the board which facilitates adding on such items as more memory or a new display board. Signals are provided that would allow the use of DMA. All of the address and data signals are fully buffered and separate from the 48-pin bus and the board bus.

OSI Disk Controller and Real Time Clock

The disk controller is a standard OSI type disk controller except that no adjustments are required. This is accomplished by using two PALs to replace the two one-shots, capacitors, and potentiometers used on the standard OSI controller. Switching between 8" and 5-1/4" data rates is accomplished with a jumper. The real-time clock is just a standard OSI circuit. The PALs were used to save board space and save the time required to adjust the disk controller. As it turns out, the hardware comes out to be about the same in cost.

Serial Ports

There are two serial ports on the board. One of them is the same as a standard OSI serial port (except for an improved interface driver) with baud rates of 300 to 19.2K. The other serial port uses a 6551 and the baud rate is software selectable from 50 to 19.2K. This second serial port provides, in addition to the Receive Data and Transmit Data signals, lines for Data Carrier Detect, Data Set

Ready, Data Terminal Ready, Ready To Send, and Clear To Send. The standard serial port was retained to allow full compatibility with the existing OSI software for serial systems. The serial ports have RS-232 signal compatibility and a 10-pin connector is provided on the board for each port. This allows a ribbon cable with a 10-pin header on one end and a DB-9 connector on the other. The DB-9 pin-out is the same as the IBM PC-AT and will allow the use of AT-series cables.

Parallel Port

The parallel port is a standard OSI Centronics port. A connector is provided which will allow connection of a ribbon cable with a DB-25 connector on one end and a 26-pin connector on the other. The pin-out of the DB-25 connector is compatible with the IBM parallel port and allows the use of any IBM compatible cable for connecting the printer.

Hardware Interrupt

A hardware interrupt vector controller has been built into the board. This allows 8 separate hardware interrupts to be generated. When the hardware interrupt is detected, the interrupt vector is changed by switching which part of the Monitor PROM is being addressed. Interrupt 0 will always be the same as the normal OSI interrupt. The three lines defining the 8 hardware interrupt vectors are available on the new bus.

Monitor PROM

The Monitor PROM allows up to 16 computer configurations to be defined with the use of a 4 position dip-switch. The standard selections will allow selecting (1) standard OSI serial system, (2) video system, (3) new video system with IBM keyboard, (4) booting from a standard disk, (5) booting from SCSI hard disk, and (6) booting from a new high-density disk and several combinations of the above. There will, in fact, be more possibilities than will be used. This will allow the user of the system to add new functions and still retain the standard monitor.

DTACK Interface

This is a parallel interface which allows the connection of one of Digital Accoustics' 68000 systems. This was described in a prior issue of PEEK[65], so I won't go into it here other than to say that this parallel interface could also be used as a high speed interface to other devices.

New Math Chip

Included on the board is a small interface connector for a small add-on board which will hold a math chip, a date and time clock, and a software interrupt vector generator. The software interrupt generator will detect the "BRK" code, decode the next byte, and generate on of 256 software interrupt vector. Interrupt 0 will be the standard OSI interrupt. The software interrupt generator will operate in native mode only to avoid problems with existing software (Native mode is the 65816 mode, while 6502 mode is called emulation mode).

Memory and Hardware Address Mapping

All of the on-board OSI hardware is decoded to the standard OSI locations. This is accomplished with the use of a custom-programmed logic device, a PAL, and can be changed to allow more memory in bank 0 by locating the hardware at $F000 and up. The standard configuration is such that any address space which is not used by hardware on the board or on another OSI board is used by the RAM. All of the hardware is decoded to use no more than 128 bytes. The 256K of RAM is decoded to be in bank 0 (as described above), bank 1, bank 2, and bank 3. The PROMs are decoded to be in bank 254 and bank 255. Again, since the decoder is programmable, this can be changed.

Interface To Standard OSI Computers

One of the problems with installing a 65816 processor on the 48-pin bus is that the existing boards don't know that the upper 8 bits of the 24-bit addresses that the 65816 used are multiplexed on the data bus. This means that to a standard OSI board, address $00489A (bank 0, $489A) looks the same as address $FF489A (bank 255, $489A). This, of course, leads to data bus conflict on a read, and writing to more than one location on a write. This would never do. Since I have already designed and sold two other boards which were designed to anticipate a 65816 board installed on the 48-pin bus, I had to configure the board so that the upper 8 address lines which are multiplexed on the data bus and the address lines are always enabled. Since there won't be too many boards used on the 48-pin bus when you use one of these 65816 CPU boards (you can throw out all memory boards, OSI disk controller boards, etc.) I decided that the easiest thing to do was to provide a disable signal to the other boards that will become active whenever any BANK other than bank 0 is addressed. This means that other boards in the system will need to be modified. I have looked at all of the common boards that might be in the system, such as the 540 video display board or the various hard disk controller boards, and have found that by cutting one trace and adding one jumper, you can use the disable signal to tell the board that it should ignore the addresses, For those who will never use a board on the 48-pin bus that is designed to work with the 65816, I have provided another way to avoid memory conflict. Any time that we don't address bank 0, the address lines on the 48-pin bus are tri-stated, and the pull-up resistors on the bus will force them high. This will generate an address of $FFFF, which is in the Monitor ROM and won't be recognized by any of the boards on the 48-pin bus.

Use As A Stand-Alone Computer

The CPU board has a separate power connector for use in a stand-alone system. All that you need to supply to use this boards as a complete serial system is a power supply, the disk drives, and the terminal.

65816 Software

In order to use the 65816, we will need to have some software that will take advantage of the expanded memory space and new instructions of the 65816. This means that we will need to have an assembler for the 65816. Since I wrote an assembler for the 68000 a few years ago, I already have the core of an assembler that could be adapted for the 65816. The most difficult part is to make it work for 6502, 65C02, and 65816's all at once. The work is at this time about 80% complete. I feel that all of our efforts to write a new DOS should be oriented towards the 65816.

Converting OSI Systems to 65816 CPU

The conversion of OSI computers is relatively straightforward. For example, if you have a C4 or C8, you will need to remove the CPU board and any memory boards that are installed. The 540 board will need to be modified to add a wait diode and two jumper wires. This can be done in about ten minutes. The reset wires need to be connected to a new two-pin connector (supplied) and the new CPU board can now be installed. As stated earlier, the serial ports can be connected with DB-9 connectors on a ribbon cable. If you remove the A-15 board from a C4 or C8, you can add a small adapter plate that will accept the connectors. The C8's also have several unused mounting holes that will accept DB-25 connectors. If you have a C3 serial system without a hard disk, you can take everything out of it except for the NEC Spinwriter interface board (a modified 470) and simply installed the I/O port cables and the reset cable.

The SCSI And Double-Density Floppy Disk Controller

This board contains two major sections. The SCSI controller consists of principally an NCR 5380 SCSI controller chip along with the necessary interface and glue chips. The interface is through a standard 50-pin connector as defined by the SCSI standard. The controller will support up to 8 devices. At this time, more and more hard disk manufacturers are coming out with disk drives which include embedded SCSI controllers. Several companies also package hard disk systems with SCSI interfaces. For example, any of the units sold for use with the Apple Macintosh can be used. The SCSI interface should perhaps be described in another article.

The second major section of the board is the floppy disk controller. This controller is built around a Western Digital 2793 controller chip which can be used in single or double density formats with 3-1/2", 5-1/4", or 8" disk drives. This controller will allow reading and writing almost any disk format used today. All of the clock rates and density selection are controlled with software and the board will therefore support several types of disk drives at one time. Four disk drives can be controlled at one time. Two connectors are provided for the disk drives. These are a 34-pin connector for the 3-1/2" and 5-1/4" and a 50-pin for 8" drives. Another feature of the disk controller is the inclusion of a disk controller bus switch. This bus switch allows the disk drives, under software control, to be connected to either the new controller or the standard OSI controller. The board interfaces to the OSI controller with the use of a short ribbon cable which plugs into the new controller board and into the data separator board, which in turn plugs into the OSI controller. This makes for a very neat and compact package. The board also includes an 8K RAM which can be selected in two banks of 2K, plus two banks of 2K, which are bank-switched to the same memory location. This is for use in standard OSI systems where the standard OSI CPU board is used. When the board is used in a system with the 65816 CPU, the RAM will not be used.

This board was designed to decode the 65816 addresses on the data bus and can, therefore, be used with the new CPU board without any modifications.

Software

This board has, in fact, been finished for about 9 months, but the software has been slow in coming. The reason for this is that I didn't want to simply mimic the standard OSI disk format. The intention was to provide a new DOS for use with the new controller when used under OS-65D. I have set up this DOS for use with the new high-density 5-1/4" or 3-1/2" disks which hold 1.2 megabytes. This DOS will also work with double-sided 8" disks or, with some modification, single-sided 8" drives. Other configurations could be supported, but I don't intend to do it. I will, however, write a program which will allow reading IBM PC and PC-AT text files.

Obviously, a new DOS was required to support the 1.2 megabyte drives. The major changes are a new Catalog and the support of dynamic file sizes. The catalog support files with names of 11 characters plus extension of 3 characters. The catalog stores the type of file and the time and date.

The new DOS for the new disk controller doesn't replace the existing DOS, but instead is used whenever the new drives are selected. I have called the new floppy drives "F" through "I" and the first hard disk connected to the SCSI interface is device "J". This new DOS support all of the current 65D functions, such as LOAD and SAVE. BLOAD and BSAVE functions have also been added. The directory program is also resident. New functions can be added as external DOS functions by writing a new function and placing it in a file with an extension of "CMD". When DOS searches the dispatch table and doesn't find the command you issued, it searches the current drive for a file with the same name as the command (with the extension ".CMD". If the command file is found, it is loaded and executed.

As previously mentioned, the files are dynamic, which means that if the file grows in length, then the DOS will allocate more sectors to the file. In order to maintain the maximum speed, I have decided that I won't allow the files to become fragmented in such a way that the disk drive head has to step in and out to find all of the sectors. Instead, any time a file needs to be lengthened, we look for the next available sector which is towards the inside of the disk. This also has the disadvantage of forcing us to not reuse sectors from files that have been deleted. This is not too bad in that repacking the disk will eliminate the problem.

I intend to write software to use the new controller with OS-65U, but haven't started. This software would not make any changes to OS-65U other than to provide the drivers for the controller board and the interface to OS-65U. If anyone is interested in writing the software for using this board with OS-65U, feel free to contact me via PEEK[65].

Cost

Now we come to the hard questions like "How much does it cost?" The CPU board will range in cost from about $320 for a minimum configuration with only 64K of memory installed, to a little over $400 for a board with 256K of RAM. The SCSI with floppy controller will be about $300 with software. These are preliminary prices and could be reduced by about 10% if the boards could be built in quantities of at least 25 at a time.

Tiny Compiler

by David E. Pitts

(**Editor's Note:** The following is a summary of the documentation for Pitts' Tiny Compiler for OS-65D BASIC as of 1981. David released the OSI version into the public domain with the proviso that it only be used on OSI systems. I ask that you abide by that request. I modified David's original code to impliment the DISK! keyword.)

The Tiny Compiler can produce relocatable object code and the USR(X) routine allows linkage of these object codes such that large routines can be created.

Both the object code location and the variable table location are chosen by the user, thus allowing multiple machine code routines to use the same variable table or a different variable tables. The object code is stand alone and does not address ROMs or OS-65D. It uses a 16 bit arithmetic stored in standard LSB-MSB format and uses the Accumulator to hold the LSB and the X register to hold the MSB. Only positive intergers are used, but the user can utilize two's compliment to create dummy negative integers. No page zero locations are used and the only working locations are the zero and first locations in the variable table, which is 54

```
TINY COMPILER COMMANDS

Legal variables: A - Z

A = nnn (where 0 <= nnn <= 65535)
A = B
A = B + C     A = B + nnn    A = B OR C    A = B OR nnn
A = A + C     A = A + nnn    A = B AND C   A = B AND nnn
A = B - C     A = B - nnn
A = PEEK(B)   A = PEEK(nnn)
POKE A, B     POKE A, nnn
GOSUB nnn     GOTO nnn
A = D * B     A = D * nnn    A = B/C       A = B/nnn
(where 0 <= D <= C <= 255, 0 <= B <= nnn <= 65535 above)

IF A=B THEN GOTO nnn               RETURN
IF A=B THEN GOSUB nnn              STOP
IF A<>B THEN GOTO nnn              REM
IF A<>B THEN GOSUB nnn
IF A<B THEN GOTO nnn
IF A<B THEN GOSUB nnn
IF A=nnn THEN ....
IF A<>nnn THEN ...

PRINT A
PRINT nnn A   (prints at $D000+n)

INPUT A       Retrieves one keypress. Returns 0 - 9 or
              ASCII value of Alpha key.

CLEAR         machine code screen clear

DISK!         accepts OS-65D DISK! commands. Does not
              support DISK PUT, DISK GET, DISK OPEN,
              or DISK CLOSE.

FOR I = A TO B
FOR I = nnn TO B
FOR I = A TO B STEP nnn   (nnn can be + or -)
FOR I = A TO D STEP nnn
FOR I = PEEK(nnn) TO B STEP mmm
FOR I = PEEK(C) TO B STEP nnn
NEXTX   (X optional)

Multiple statements per line are allowed except for
IF A= ..... THEN GOSUBnnn which must be at the end of the
line.
```

Table 1

bytes long. Self modifying code is used for the PEEK, USR, and POKE compilations. During the first pass, the line numbers for GOTOs and GOSUBs are stored as addresses for the JMP and JSR. Later this is replaced by the absolute address using vectors contained in the string variables L$ and L3$. The arithmetic routines used are from William Barden's book "How to Program Microcomputers", Howard Sams publishers.

The code generated by the compiler is not as efficient as an experienced programmer can write using assembly language, however, it is much easier to have the compiler do the dirty work. The speed of the object code has been compared to the interpreter using nested FOR...NEXT loops and found to be some 40 times faster. This means that some game program routines may require delay loops.

GENERAL LAYOUT

Line #	Description
8000	POKE object code
8005-8015	PEEK Source code
8020-8055	Set up integer, error check
8060-8250	POKE instruction codes
9000-9050	Initialization
9055-9150	MAIN LOOP
9155-9190	JUMP calculations
9195-9215	Run machine code & stop
9220-	Macro codes

MACRO CODES

9220-9290	A = #, check for +, -, *, / etc.
9295-9330	PEEK
9335-9380	Multiplication
9385-9430	Division
9435-9515	IF... THEN
9520-9535	USR(X)
9540-9545	GOSUB, GOTO
9550-9565	POKE
9570-9595	Self-modifying code for PEEK, POKE, USR(X)
9600-9625	FOR
9630-9665	NEXT
10000-10180	DISK

Table 2

```
10 CLEAR
20 DISK!"CA D200=08,1
30 H=1:L=2000
40 FORK=HTOL
50 PRINT 1024 K
60 NEXT
7999 END
8000 POKEM,P:PRINTP:M=M+1:RETURN:POKE OBJ CODE
8005 P=PEEK(Q):PRINTTAB(20)"TOKEN=";P;"LOC=";Q:Q=Q+1:IFP=32THEN8005

8010 IFP=0THENC=2
8015 RETURN
8020 IFP<65ORP>90THEN8030:REMCHECK ALPHA
8025 RETURN
8030 PRINT:PRINT"ERROR LINE# ";L$(L):END
8035 IFP<48ORP>57THENRETURN: NOT 0-9
8040 C$=C$+CHR$(P):GOSUB8005:GOTO8035
8045 C$="":GOSUB8035:IFC$=""THENF=-1:RETURN
8050 F=VAL(C$)
8055 MB=INT(F/256):LB=F-MB*256:RETURN
8060 GOSUB8235:GOSUB8085:GOSUB8185:GOSUB8225:GOSUB8090
8065 GOSUB8185:RETURN:REMLDX XXZZ,Y+1 LDA XXZZ,Y
8070 GOSUB8210:P=LB:GOSUB8000:GOSUB8215:P=MB:GOSUB8000:RETURN:LOAD
A&X
8075 P=160:GOSUB8000:RETURN:LDY#
8080
GOSUB8235:GOSUB8170:GOSUB8085:GOSUB8220:GOSUB8170:RETURN:REMSTA,Y
8085 P=200:GOSUB8000:RETURN:INY
8090 P=136:GOSUB8000:RETURN:DEY
8095 P=121:IFS=164THENP=249:GOTO8110:BEG OF SBC,ADC,AND,ORA
8100 IFS=168THENP=57:GOTO8110:AND
8105 IFS=169THENP=25
8110 GOSUB8245:RETURN
8115 P=24:IFS=164THENP=56:REMCLC OR SEC
8120 GOSUB8000:RETURN
8125 P=96:GOSUB8000:RETURN:RTS
8130 P=16:GOSUB8000:RETURN:BPL
8135 F=(V4-64)*2+ZZ+256*XX:GOSUB8055:GOSUB8140:RETURN:ROL
8140 P=46:GOSUB8240:RETURN:ROL
8145 P=10:GOSUB8000:RETURN:ASL A
8150 P=72:GOSUB8000:RETURN:PHA
8155 P=104:GOSUB8000:RETURN:PLA
8160 P=202:GOSUB8000:RETURN:DEX
8165 P=153:GOSUB8000:RETURN:STA QQPP,Y
8170 P=153:GOSUB8245:RETURN:STA XXZZ,Y
```

```
8175 P=240:GOSUB8000:RETURN:BEQ
8180 P=LB:GOSUB8000:P=MB:GOSUB8000:RETURN
8185 P=185:GOSUB8245:RETURN:LDA XXZZ,Y
8190 P=185:GOSUB8000:RETURN:LDA QQPP,Y
8195 P=144:GOSUB8000:RETURN:BCC
8200 P=176:GOSUB8000:RETURN:BCS
8205 P=208:GOSUB8000:RETURN:BNE
8210 P=169:GOSUB8000:RETURN:LDA#
8215 P=162:GOSUB8000:RETURN:LDX#
8220 P=138:GOSUB8000:RETURN:TXA
8225 P=170:GOSUB8000:RETURN:TAX
8230 P=217:GOSUB8245:RETURN:CMP XXZZ,Y
8235 GOSUB8075:P=(V1-64)*2:GOSUB8000:RETURN:VAR TABLE LOC
8240 GOSUB8000:P=LB:GOSUB8000:P=MB:GOSUB8000:RETURN
8245 GOSUB8000:P=ZZ:GOSUB8000:P=XX:GOSUB8000:RETURN:VAR TABLE ADDR
8250 P=0:GOSUB8000:GOSUB8000:RETURN
8425 REM- LINE REFERENCED BY OLD LINE #9427
8527 F=8955:GOSUB8055:GOSUB8180
9000 DIML$(50),L3$(50):POKE2888,0
9005 PRINT:PRINT:PRINTTAB(20);"TINY COMPILER  1.1":PRINT:PRINT
9010 X=PEEK(122)+256*PEEK(123)-5
9011 PRINT"TOP OF BASIC PROGRAM= ";X:PRINT
9015 Q=PEEK(120)+256*PEEK(121):L=1
9016 PRINT"FOR DEFAULT ENTER '0'"
9020 INPUT"DESIRED LOC(DECIMAL) OF OBJ CODE(32768 DEFAULT)";M
9025 IFM<XTHENM=32768
9030 MM=M:INPUT"LOC OF VARIABLE TABLE (33792 DEFAULT)";VT
9035 J=0:N=1:L=0:L3$(1)="0":R=0:IFVT<XTHENVT=33792
9040 INPUT"RELOCATE OBJECT CODE";C$:C$=LEFT$(C$+" ",1)
9041 IFC$<>"Y"THEN9050
9045 INPUT"DECIMAL ADDRESS";R:R=R-M
9050 F=VT:GOSUB8055:XX=MB:ZZ=LB:REM MSB & LSB-VAR STOR
9055 M1=PEEK(Q)+256*PEEK(Q+1):X=PEEK(Q+2)+PEEK(Q+3)*256
9060 PRINT:PRINT"LINE ";X;"LOC= ";M:L=L+1:Q=Q+4
9064 C$=STR$(X):Y=LEN(C$)-1
9065 L$(L)=RIGHT$(C$,Y)+STR$(M):IFX>7999THEN9155
9070 C=0:GOSUB8005:IFC=2THEN9070
9075 IFP>64ANDP<91THENGOSUB9220:GOTO9145:A=
9080 IFP=135THENGOSUB8005:GOSUB9220:GOTO9145:LET
9090 IFP=136THENX=76:GOSUB9540:REM GOTO
9095 IFP=138THENGOSUB9435:REM IF
9100 IFP=140THENX=32:GOSUB9540:REM GOSUB
9105 IFP=141THENGOSUB8125:REM RETURN
9110 IFP=143THENGOSUB8125:REM STOP
9115 IFP=129THENJ=J+1:GOSUB9600:REM FOR
```

```
9120 IFP=130THENGOSUB9630:J=J-1:REM NEXT
9125 IFP=150THENGOSUB9550:REM POKE
9130 IFP=128THENGOSUB8125:GOTO9155:REM END
9135 IFP=142THENQ=M1:GOTO9055:"REM"
9136 IFP=151THEN9700
9137 IFP=154THEN9900
9138 IFP=132THEN9950
9139 IFP=148THEN10000:DISK
9140 GOSUB8005
9145 PRINT:PRINT"P1= ";PEEK(Q-1)
9146 IFPEEK(Q-1)=58THEN9070:REM CHECK FOR COLON
9150 Q=M1:PRINT:GOTO9055
9155 C=VAL(L3$(1)):PRINT:PRINT"JUMP VECTORS":IFC<1THEN9190
9160 N=N-1:FORY=1TON:C=VAL(L3$(Y)):XX=PEEK(C)+256*PEEK(C+1):ZZ=0
9165 FORX=1TOL:S=LEN(L$(X)):FORJ=1TOS:IFMID$(L$(X),J,1)<>"
"THEN9168
9166 V2=VAL(RIGHT$(L$(X),S-J)):V1=VAL(LEFT$(L$(X),J-1)):GOTO9170
9168 NEXT:GOTO9175
9170 IFXX=V1THENZZ=V2+R:PRINT"JUMPTO";V1;"ADDR=";ZZ
9175 NEXT:IFZZ=0THENPRINT"NO ADDR FOR ";XX:GOTO9185
9180 MB=INT(ZZ/256):LB=ZZ-MB*256:POKEC,LB:POKEC+1,MB
9185 NEXT
9190 PRINT(M-MM)/256;"PAGES,TOP=";M:PRINT
9191 PRINT:PRINT"(1) EXECUTE PROGRAM":PRINT"(2) EXIT":PRINT
9192 INPUT"    YOUR CHOICE ";Y$:K=VAL(Y$):IFK=1THEN9200
9193 IFK<>2THEN9191
9195 END
9200 FORX=VTTOVT+54:POKEX,0:NEXT
9205 PRINT"RUNNING":X=INT(MM/256):Y=MM-X*256:POKE575,X:POKE574,Y
9210 X=USR(X):PRINT:INPUT"PRINT VARIABLE TABLE ";Y$:Y$=LEFT$(Y$+"
",1)
9211 IFY$<>"Y"THENEND
9214 FORX=2TO54STEP2:M=VT+X:Y=PEEK(M):Q=PEEK(M+1)
9215 PRINTCHR$(X/2+64);Y+256*Q:NEXT:STOP
9220 GOSUB8020:V1=P:GOSUB8005:IFP<>171THEN8030:REM "="
9225 GOSUB8005:IFP=187THEN9295:PEEK
9230 IFP=176THEN9520:USR
9235 GOSUB8045:IFF=-1THEN9245:REM F=-1 IF NOT INTEGER
9240 GOSUB8070:GOSUB8080:RETURN:A=#
9245 V2=P:V4=V1:GOSUB8005:IFP<163ORP>172THENQ=Q-1:GOTO9290:A=B
9250 S=P:GOSUB8005:GOSUB8045:V3=P:IFS=165THEN9335:REM *
9255 IFS=166THEN9385:REM /
9260 IFF=-1THENV8=P:GOTO9270:A=B+NNN
9265 V8=64:GOSUB8070:V1=V8:GOSUB8080:Q=Q-1
9270 V1=V2:GOSUB8060:V1=V8:V2=V8:GOSUB8235:GOSUB8115:GOSUB8095
```

```
9275 V1=V4:GOSUB8235:GOSUB8170:GOSUB8220:REM STOR LSB:TXA
9280 V1=V2:GOSUB8235:GOSUB8085:GOSUB8095:REM ADD MSB
9285 V1=V4:GOSUB8235:GOSUB8085:GOSUB8170:GOSUB8005:RETURN
9290 V1=V2:GOSUB8060:V1=V4:GOSUB8080:GOSUB8005:RETURN: A=B
9295 GOSUB8005:IFP<>40THEN8030: "("
9300 GOSUB8005:GOSUB8045:V4=V1:V1=P:IFF=-1THEN9315
9305 GOSUB8075:P=0:GOSUB8000:GOSUB8190:GOSUB8180:GOSUB8215
9310 P=0:GOSUB8000:V1=V4:GOSUB8080:GOSUB8005:RETURN
9315 GOSUB8005:IFP<>41THEN8030: ")"
9320 X=10:GOSUB9580:REM ABOVE MODS CODE
9325 GOSUB8075:P=0:GOSUB8000:GOSUB8190:GOSUB8250
9330 GOSUB8215:P=0:GOSUB8000:V1=V4:GOSUB8080:GOSUB8005:RETURN
9335 S=163:V1=V2:GOSUB8060:GOSUB8150
9340 IFF=-1THENV1=V3:GOSUB8060:V1=64:GOSUB8080:GOSUB8005:GOTO9350
9345 GOSUB8070:V1=64:GOSUB8080
9350 F=0:GOSUB8055:GOSUB8070:V1=V4:GOSUB8080
9353 GOSUB8155:GOSUB8215:P=8:GOSUB8000
9355 P=24:GOSUB8000:GOSUB8135
9360 F=F+1:GOSUB8055:GOSUB8140:GOSUB8145:GOSUB8195:P=33:GOSUB8000
9365 GOSUB8150:V1=V4:GOSUB8235:P=185:GOSUB8245:P=24:GOSUB8000
9370 GOSUB8075:P=0:GOSUB8000:GOSUB8095:V1=V4:GOSUB8235:GOSUB8170
9375 GOSUB8085:GOSUB8185:GOSUB8075:P=1:GOSUB8000
9377 GOSUB8095:V1=4:GOSUB8235
9380 GOSUB8085:GOSUB8170:GOSUB8155:GOSUB8160
9383 GOSUB8205:P=210:GOSUB8000:RETURN
9385 S=164:IFF=-1THENV1=V3:GOSUB8060:GOSUB8005:GOTO9395
9390 GOSUB8070
9395 GOSUB8225:GOSUB8210:P=0:GOSUB8000:V1=64:GOSUB8080
9400 V1=V2:GOSUB8060:V1=V4:GOSUB8080:GOSUB8215:P=17:GOSUB8000
9405 F=M+R+15:GOSUB8055:P=76:GOSUB8240:GOSUB8075:P=0:GOSUB8000
9410
GOSUB8185:GOSUB8115:GOSUB8085:GOSUB8095:GOSUB8130:P=4:GOSUB8000
9415 P=24:GOSUB8000:F=M+R+9:GOSUB8055:P=76:GOSUB8240:GOSUB8075
9420 P=0:GOSUB8000:GOSUB8170:GOSUB8115:GOSUB8135
9423 F=F+1:GOSUB8055:GOSUB8140
9425 GOSUB8160:GOSUB8175:P=6:GOSUB8000:P=46
9427 GOSUB8425:F=M+R-34:GOSUB8055
9430 P=76:GOSUB8240:RETURN
9435
GOSUB8005:GOSUB8020:V1=P:GOSUB8005:IFP>172ORP<171THEN8030:REM=<
9440 V4=P:IFP=172THENGOSUB8005:IFP<>170THEN9480:REM<>,<
9445 V2=V1:GOSUB8005:V1=P:GOSUB8045:IFF<>-1THEN9476
9446 GOSUB8060
9447 V1=V2:GOSUB8235:GOSUB8230:GOSUB8205
9450 GOSUB9500
```

```
9455 P=7:IFV4=171THENP=10
9460 GOSUB8000:GOSUB8220:GOSUB8085:GOSUB8230
9465 IFV4=172THENGOSUB8175:GOTO9475
9470 GOSUB8205
9475 P=3:GOSUB8000:GOTO9540:IF =,<>THEN
9476 GOSUB8055:P=169:GOSUB8000:P=LB:GOSUB8000:P=162:GOSUB8000
9477 P=MB:GOSUB8000:Q=Q-1:GOTO9447
9480 GOSUB8020:V2=P:GOSUB8235:GOSUB8185:GOSUB8225:GOSUB8085:REMIF<
9485 GOSUB9500:GOSUB8185:V1=V2:GOSUB8235:GOSUB8085
9487 GOSUB8230:GOSUB8195:P=11
9490
GOSUB8000:GOSUB8205:P=12:GOSUB8000:GOSUB8220:GOSUB8090:GOSUB8230
9495 GOSUB8175:P=5:GOSUB8000:GOSUB8200:P=3:GOSUB8000:GOTO9540
9500 GOSUB8005:IFP<>160THEN8030:REMTHEN
9505 GOSUB8005:IFP<>136ANDP<>140THEN8030
9510 X=76:IFP=140THENX=32:REMGOTO OR GOSUB
9515 RETURN
9520 GOSUB8005:GOSUB8005:GOSUB8005:IFP<>41THEN8030:REMUSR
9525 GOSUB8005:GOSUB8075:P=1:GOSUB8000:GOSUB8190
9530 GOSUB8225:GOSUB8090:GOSUB8190:GOSUB8180
9532 X=8:GOSUB9585:P=32:GOSUB8000
9535 GOSUB8250:RETURN
9540 GOSUB8005:GOSUB8045:IFF<1ORF>7999THEN8030
9545 P=X:GOSUB8000:L3$(N)=STR$(M):N=N+1:GOSUB8180:Q=Q-1:RETURN
9550 GOSUB8005:GOSUB8020:V1=P:GOSUB8005:IFP<>44THEN8030:REM ","
9555 GOSUB8005:GOSUB8045:IFF=-1THEN9570:REM F=-1 IF NOT INTEGER
9560 V4=LB:X=14:GOSUB9580:GOSUB8075:P=0:GOSUB8000
9565 LB=V4:MB=0:GOSUB8070:GOSUB8165:GOSUB8250:Q=Q-1:RETURN
9570 X=21:V2=P:GOSUB9580:V1=V2:GOSUB8060
9575 GOTO8250
9580 GOSUB8060
9585 GOSUB8075:P=0:GOSUB8000:GOSUB8165:F=M+X+R:GOSUB8055
9590 GOSUB8180:GOSUB8085:GOSUB8220:GOSUB8165:GOSUB8180
9595 RETURN:SELF MOD CODEFOR 3BYTE IND ADDR,Y
9600 GOSUB8005:V7(J)=P:GOSUB9220:Q=Q-1:GOSUB8005:IFP<>157THEN8030
9605 V6(J)=M-1:GOSUB8005:V5(J)=P:GOSUB8005:T(J)=1:V4=163
9610 IFP<>162THENQ=Q-1:RETURN
9615 GOSUB8005:IFP=164THENV4=P:GOSUB8005
9620 GOSUB8045:T(J)=F:Q=Q-1:IFV4=164THENT(J)=65536-T(J)
9625 RETURN
9630 GOSUB8005:IFP<65ORP>90THENQ=Q-1:REM NEXT
9635 V1=V7(J):GOSUB8060:V1=V5(J):GOSUB8235:GOSUB8230:GOSUB8205
9640 P=10:GOSUB8000:GOSUB8220:GOSUB8085:GOSUB8230
9645 GOSUB8205:P=3:GOSUB8000:P=76:GOSUB8000:F=M+26+R:GOSUB8055
9650 GOSUB8180:F=T(J):GOSUB8055:GOSUB8070:S=163:V1=V7(J)
```

```
9655 GOSUB8235:GOSUB8115:GOSUB8095:GOSUB8170:GOSUB8220:GOSUB8085
9660
GOSUB8095:GOSUB8170:P=76:GOSUB8000:F=V6(J)+1+R:GOSUB8055:GOSUB8180
9665 RETURN
9670 REM L$( )=DECIMAL # OF COMPILED LINE+DECIMAL ADDR OF OBJ LINE
9680 REM L3$( )=DEC LOC OF LOW BYTE OF JMP OR JSR, N=# OF L3
9685 REM Q=LOCATION IN BASIC TO BE PEEKED
9690 REM L=LINE BEING COMPILED
9695 REM VT BEGINNING ADDR OF VARIABLE TABLE
9700 P=76:GOSUB8000:GOSUB8250
9705 REM M1=NEXT BASIC LINE TO BE COMPILED
9710 TA=M:GOSUB8005:GOSUB8045:PP=53440+ABS(F)
9720 IFP<>34THEN9800
9730 P=PEEK(Q):Q=Q+1:IFP=34ORP=0THEN9750
9740 GOSUB8000:GOTO9730
9750 P=0:GOSUB8000:F=M:GOSUB8055
9760 POKETA-2,LB:POKETA-1,MB
9770 GOSUB8075:P=255:GOSUB8000:GOSUB8085
9780 GOSUB8190:F=TA:GOSUB8055:GOSUB8180
9785 GOSUB8175:P=5:GOSUB8000
9790 GOSUB8165:F=PP:GOSUB8055:GOSUB8180
9795 GOSUB8205:P=245:GOSUB8000
9797 IFPEEK(Q-1)=34THEN9140
9798 GOTO9150
9800 GOSUB8020:M=M-3:V1=P:GOSUB8060
9810 P=134:GOSUB8000:P=33:GOSUB8000:P=133:GOSUB8000
9820 P=34:GOSUB8000:RESTORE
9850 FORI=0TO58:READP:GOSUB8000:NEXTI
9852 DATA 160,4,169,0,133,32,162,17,208,7
9854 DATA 165,32,56,233,10,16,3,24,144,3
9856 DATA 133,32,56,38,34,38,33,202,240,5
9858 DATA 38,32,24,144,231,165,32,24,105,48
9860 DATA 153,0,212,165,34,208,9,165,33,208
9862 DATA 5,169,32,136,16,240,136,16,199
9870 F=PP:GOSUB8055:POKEM-18,LB:POKEM-17,MB
9890 GOTO9140
9900 RESTORE:FORI=0TO58:READP:NEXT
9910 FORI=1TO31:READP:GOSUB8000:NEXT:GOTO9140
9920 DATA 160,0,169,32,153,0,215,153,0,214,153,0,213
9930 DATA 153,0,212,153,0,211,153
9940 DATA 0,210,153,0,209,153,0,208,200,208,229
9950 GOSUB8210:P=63:GOSUB8000:P=141:GOSUB8000
9960 F=61440:GOSUB8055:GOSUB8180:P=32:GOSUB8000:REM - $F000 ?
9965 F=9014:GOSUB8055:GOSUB8180:REM- CHANGED TO JSR $2340
9970 P=201:GOSUB8000:P=58:GOSUB8000:GOSUB8130:P=3:GOSUB8000
```

```
9972 P=56:GOSUB8000:P=233:GOSUB8000:P=48:GOSUB8000
9974 GOSUB8215:P=0:GOSUB8000
9975 GOSUB8005-GOSUB8020:V1=P:GOSUB8080
9980 GOSUB8210:P=32:GOSUB8000:P=141:GOSUB8000:F=53509
9985 GOSUB8055:GOSUB8180
9990 GOTO9140
10000 GOSUB8005:IFP=33THENGOSUB8005:IFP=34THEND$="":GOTO10020
10010 GOTO8030:REM NOT DISK!"
10020 GOSUB10160:IFP=34ORP=0THEN10040
10030 D$=D$+CHR$(P):GOTO10020
10040 D$=D$+CHR$(13):DL=LEN(D$):REM- ADD <CR> TO STRING
10045 IFP=0THENQ=Q-1:REM- BACK UP ON E.O.L.
10050 P=32:GOSUB8000:P=247:GOSUB8000:P=44:GOSUB8000
10060 F=M+22:SA=F:GOSUB8055
10065 GOSUB8210:P=LB:GOSUB8000:P=133:GOSUB8000
10070 P=225:GOSUB8000:GOSUB8210:P=MB:GOSUB8000:P=133:GOSUB8000
10080 P=226:GOSUB8000:GOSUB8210:P=DL:GOSUB8000
10090 P=141:GOSUB8000:F=11501:GOSUB8055:P=LB:GOSUB8000
10100 P=MB:GOSUB8000:P=32:GOSUB8000:F=10884:GOSUB8055
10110 P=LB:GOSUB8000:P=MB:GOSUB8000:P=32:GOSUB8000
10120 P=247:GOSUB8000:P=44:GOSUB8000:P=76:GOSUB8000
10130 F=SA+DL:GOSUB8055:P=LB:GOSUB8000:P=MB:GOSUB8000
10140 FORK=1TODL:P=ASC(MID$(D$,K,1)):GOSUB8000:NEXTK
10150 GOTO9140
10160 P=PEEK(Q):PRINTTAB(20)"TOKEN=";P;"LOC=";Q:Q=Q+1
10170 IFP=0THENC=2
10180 RETURN
```

Continued from Page 1

for not contacting you personally.

There's a lot of exciting things happenning in the OSI community as we begin to bridge the gaps between the 8 and 16/32-bit worlds. The new DB-II systems breathe new life into old applications and hold the promise for even better performance on systems that have long been the unsung champions in that department. As PEEK[65] readers are the beneficiaries/victims of many previous incarnations of a single company, we haven't seen as large a leap forward in potential since MA/COMM bought the company from the Cheiky's.

Thanks. We all wish DevTech the best of luck in all they do.

Back on the home front, Dave Livesay tells us all about his 65816 CPU board. Dave will soon be moving back to California, so if you need to contact him, you can do so via PEEK[65]'s post office box. I have shared some of my thoughts on a new operating system and other issues. Bob Best of the KAOS user group in Australia presents his simple OS-65D v3.3-based accounting system. And there are a couple of other treats here and there.

Keep those disks and letters coming, folks. PEEK[65] is still very much in need of new articles to publish. Don't forget, uploading on CompuServe is free of standard connect charges, so it's usually even cheaper than the cost of sending a diskette. And PEEK pays you for your help! So, please pitch in. Thanks!

Musings on a New Disk Operating System and the Future

by Richard L. Trethewey

This article isn't going to be very specific about many of the things it discusses. The reason for that is that I want to discuss a piece of code I haven't written yet - the imfamous "new operating system". We're at a real crossroads now because there are three, count 'em, three CPU boards available for the 65816 microprocessor: the DB-II Denver Board, Paul Chidley's CxP board, and David Livesay's new board described elsewhere in this issue. Since most of us who are interested in a new operating system will be using one of the latter two boards or opting for the 65802 and keeping our current hardware, I'll keep my discussions centered around those two boards.

The first thing you notice about the design of both of these boards is that in addition to the new microprocessor, they make use of a lot of hardware that isn't in a vanilla OSI system. For example, Paul's board uses a new ACIA chip and Dave has interfaces for many different disk drives. This points up a fundamental problem that needs to be addressed. If any operating system is to flourish, it must be able to support all of these options and make provisions for future improvements.

The history of OSI operating systems, OS-65U in particular, makes it clear that a stable interface to the operating system is imperative. If you begin to support POKEs and PEEKs that alter the operating system's behavior, the memory locations have to remain in one place or you obsolete any piece of software that depends on them each time you upgrade the operating system. In this manner, you end up with spaghetti code with JMPs to JMPs and JSRs to branches and your operating system ends up fragmented all over its allotted space and taking an inordinate amount of time just winding its way through all of the patches. Its certain that the first few efforts to write a new operating system will be plagued with occasional bugs. By starting off sensibly right now, we can prepare for this eventuality and avoid making the same old mistakes.

As I've mentioned many times before, I have been working with an Apple Macintosh computer for several years now. A key element in the Mac is the operating system. That operating system uses something called "device independent I/O". What that boils down to is that I/O functions are routed through pieces of software called "device drivers" which performs its task with specific hardware devices but based on a uniform operating system command or "function call".

OS-65D and OS-65U have a form of device independence for character I/O. By setting a bit in a particular byte, calls to the character input and output routine are routed to device-specific routines built into the operating system. However, these individual routines are hard-coded into the operating system and can only be altered by patching them. This has been acceptable for the most part because there have only been a limited number of peripheral devices that were supported by OSI. However, the new CPU boards are quickly changing that and I expect even more changes.

In the Macintosh operating system (and I'm sure many others as well), the device drivers are small modules of code with special headers which hold an offset from the start of the module to the locations of the starts of the various I/O commands the module supports. In a strictly 6502-based environment, we would almost certainly need to specify blocks of memory to be allocated to such code because the branching instructions of the 6502 are so limited in their ability to be position-independent. It would be possible to include a system akin to a linking loader when the device driver is installed, but that technique suffers from size and speed overheads that become restrictive. However, the 65816 instruction set includes a couple of commands that make writing independent code practical.

Most operating systems since CP/M have used a table-driven form of command interface, whereby the operating system is entered at a static location in memory and the program is then routed to the code to execute the desired function based on the contents of this table. My proposal is that we try this method for device drivers to whatever extent seems appropriate.

The Macintosh operating system is based on a handfull of basic commands for I/O: Open, Close, Status, Control, Read, and Write. When combined with a table of command parameters, these 6 calls can handle all of the needs of input and output. Thus the headers of the device drivers would consist of a table of six two-byte values which are the offsets to the above commands within the software module. However, applications software still needs to be routed at the operating system level and not work by directly accessing device drivers in memory since the application can't know where the driver may be in memory, nor should it.

Thusfar, we have outlined some of the goals of the operating system. Implimentation takes extra planning. We have decide to use a table-based command interface. Note that this does not preclude a string-based interface being a part of the operating system so that a user can execute operating system commands from his keyboard or via a text-generating application. At the lowest level, the operating system still depends on the same six commands described above. However, variations on those commands to perform more complicated tasks - largely those dealing with disks. A key question is how many commands are we going to allow the operating system to support?

No matter how many slots we allocate for our table, it is almost inevitable that we'll want more. 16 is always a nice number to use in micros (OS-65D supports 19 text-based commands), but that seems skimpy so let's double that and plan for 32 command slots. However, the last slot is going to be reserved for a dispatch vector to additional commands, allowing for endless (albeit potentially cumbersome) expansion.

One of the other benefits of the 65816 is that because the accumulator and the X and Y registers can hold a full 16-bits of data. Thus we can specify that on entry to the operating system, a particular register must hold a pointer to a command parameter list. This technique further facilitates position independence since it releases the operating system from being responsible for allocating a specific amount of memory at a particular location for these parameter lists. Similarly, it helps allow for device independence. All we need are two or three bytes to hold this pointer. Such a parameter list might well look like the following:

Offset	Meaning
$00	Driver ID #
$01	Result code
$02	I/O Reference #
$03	RAM Address
$06	# of bytes
$0B	File Position Offset
$10	I/O Mode
$11	Current Position
$16	Logical EOF
$1A	Physical EOF
$1F	I/O Name Ptr
$22...	future use

Certainly the above list is incomplete and inaccurate in addressing the needs of all conceivable uses, but it does point us in the directions I think we should take. Not all operating system calls will need or use all of these parameters either, but by defining the list ahead of time, at least in part, it helps lay out the tasks the operating system will need to perform.

Meanwhile, back at the ranch, the operating system itself has some bookkeeping to do. The operating system has to be prepared to handle all I/O calls. Naturally, it has to maintain a list of open drivers and where the modules reside in memory (ah yes.... memory. We need to talk about that, too) so that calls asking for access to those drivers can be properly routed. It may be wise to automatically open the console input and output drivers on boot-up and leave them permanently open. A control call can be used to reset or initialize them each time an application starts up, but they'll always be needed as long as the system is running.

Of course, the most pressing need of OSI owners is a more efficient file manager. One of the biggest reasons that OSI systems run so fast is that disk files have always been physically contiguous. That is, we allocate a specific number of contiguous tracks on a diskette for each file. The files cannot grow or shrink based on operating system calls, but can only be created and deleted by utility programs. Most operating systems on other micros allocate additional space to disk files as they need it, but do so by using any available track or sector. This leads to a phenomenon known as "fragmentation", where parts of the file are scattered all over the diskette, in no particular order.

The operating system that Dave Livesay has written and described in his article limits fragmentation by only allocating successively higher tracks. This is a good idea to a certain extent, but I think it's probably best to only prioritize higher tracks, but if needed, the operating system should be allowed to allocate a lower track and use all available space on the diskette if need be. The primary cost of such a technique of allocating disk space is speed. It will simply take longer for files to be loaded, programs to be launched, and for many other operations because the disk drive will then have to read tracks in a non-sequential order.

Two big hurdles loom in my vision of writing the code to support this file system. First, we have to design a diskette directory that is capable of allocating space in this manner. While it is simple enough to set and clear flags that represent the individual sectors on a diskette, there is more information that also needs to be tracked. The operating system must have an efficient method for finding all of the sectors allocated to individual files, and to be able to determine their logical position within these files. Second, it must be able to keep track of how many bytes each file holds, the physical capacity of the sectors allocated to the file, an I/O pointer, and we'll probably want to track files by "type" - program, data, etc. Naturally, there will also have to be operating system commands which will let programs have access to this information in one form or another.

If you've been keeping track of the technical details of any of the new (i.e. post-1980) microcomputers, you'll know that they all have paid considerable attention to memory management. The most notable of these is the IBM PC. The INTEL microprocessors used in PC's and clones made it convenient to limit application and operating system to the lowest 640K of memory. Like the history of OSI with their original 4K and 8K models, memory costs made anything larger very expensive anyway, so it didn't seem like a bad idea at the time. Of course, programmers stubbornly found things to do with the computers that made even 640K seem restrictive. And, as these things usually go, people found ways around the limitation. However, even today, many of the memory expansion boards available for IBMs aren't compatible with each other. There are a couple of standards now, but not all of the problems have been resolved. You're probably going to be hearing a lot about problems with the 80386 used in the new IBMs, as well as limitations in the new OS/2 operating system.

At least with the 65816, we don't have many memory expansion problems. Since all of the various CPU boards are supporting their own methods of expansion, you'll be able to upgrade within a specific path and remain software compatible with everyone else in terms of memory addressing. But since this extra memory is a new frontier for OSI users, I think its a good idea to talk about memory management in terms of software. My idea is to have the operating system allocate memory to all programs by request. That is, when an application is started, the operating system will allocate memory to hold the program. From there, the application will be able to make calls to the operating system to reserve memory.

A simple 32-byte list stored in each bank will allow marking a page of memory in that bank as being "available" or "in use". This will allow multiple programs to reside in memory simultaneously without conflicts. Things like desk accessories or "Terminate and Stay Resident" programs and other things to co-exist. Naturally, this means fencing off a piece of memory in each bank. Fortunately, the 65816 can treat any memory as the 6502's "page 0" and has other features that reduces the impact you might expect from this restriction. However, I believe that it will be at least a goal of most of us to transport current OS-65D and OS-65U software to high banks in memory, thus it seems reasonable to consider roping off memory starting above $F000 in each bank for these and other purposes, since these locations are relatively unused in most programs.

Speaking of porting software, it is logical to assume that one of the first tasks is going to be to port our beloved Microsoft BASIC in one form or another. Doing this is not simple since Microsoft may take a dim view of any redistribution of BASIC. That's a major hurdle that must be addressed if any significant improvements are going to be made to the language itself. We can likely patch the current version to operate in any environment we come up with, but it will still have all of the little foibles we have come to know and hate.

Our best bet for an up-to-date language is FORTH. The people from FORTH Interest Group have (or soon will... I haven't looked) a 65816-based FORTH that will be available as source code for a reasonable fee. Once we've made it our own, we can expand and distribute it at will since FORTH has always been in the public domain (except for certain commercial implimentations). A lot of OSIers are FORTH enthusiasts and I'm sure we can count on them for support.

To wind things up here, I want to remind those of you who may be hesitant about investing in a new CPU board, with all that entails, that there is a lower-cost alternative. Don't forget that the 65802 chip is a plug-in replacement for your 6502 which will is also software-compatible with the 65816. It does have some restrictions, and since you'll still be running at 2 MHz, some of what we may do with the new operating system may not have as much speed as you'd like. But overall, it represents an opportunity to get your feet wet without going for broke.

As I get more familiar with the new microprocessors and the new CPU boards, I'll try to keep everyone informed. I'm very excited about this stuff and I think you will be too.

Sign Up for CompuServe!

CompuServe subscription kits with a $25.00 connect-time credit are now available directly from PEEK[65] for only $32.00 plus shipping. That's 20% off the regular price of $39.95. This kit includes the CompuServe User's Manual.

In addition to giving you access to the OSI-related files and bulletin board, a CompuServe account can be your gateway to a wealth of information and communications services such as MCI Mail, the Online Airline Guide, and the CompuServe Mall for shopping at home. Send for your kit now!

A Simple Personal Accounting System for the C1P

by Bob Best
courtesy of the KAOS Newsletter

(**Editor's Note:** This article and the accompanying programs ran over several months in the KAOS newsletter. I have edited the programs and text slightly where I saw something that needed clarifying, but most of the programs are untouched. The author used a value of 10 for routing output to the screen and the printer. I changed this to 3 to reflect what I believe is most commonly used here in the U.S. Note that the programs that can send output to the printer should be altered by adding the command "DV=2" at the very start so that output is still seen on the screen when the printer is not selected. While written for the C1P-MF, the software should also run as-is on the C4P-MF as written or with slight modification, on 8 inch systems as well.)

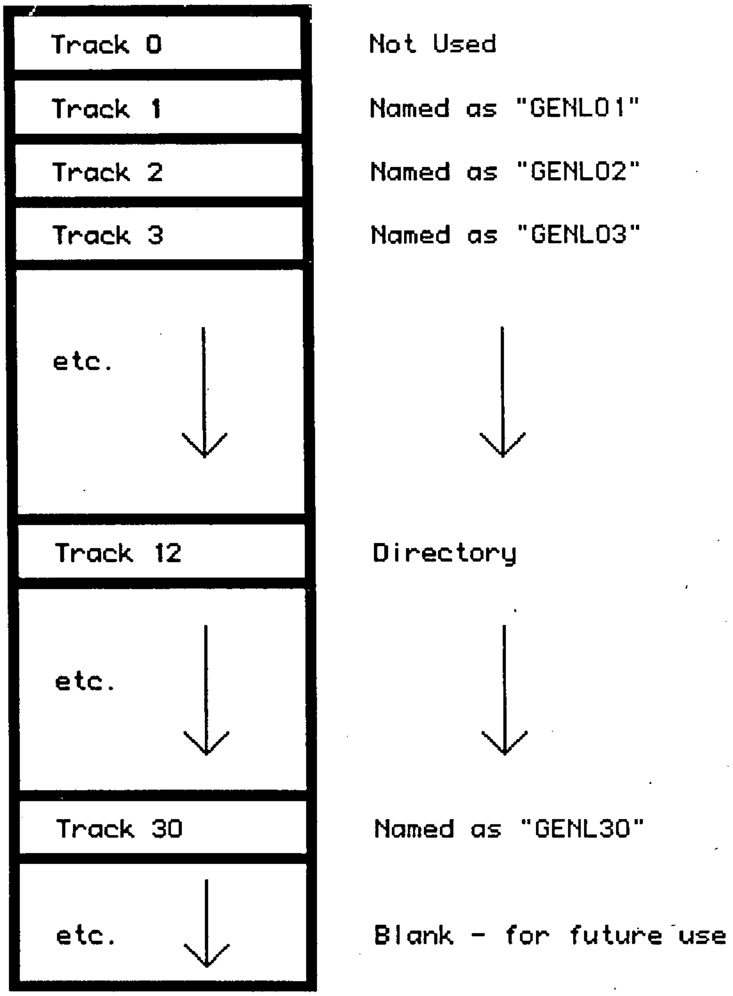


Figure 1

Before giving the history and explaination to the following accounting programs, I would like to thank Ed Richardson for the hours of help and also Graeme Reardon for the article in KAOS many months ago.

Through my association with the Scouting movement over the past few years, I was asked to help on the committee as treasurer. The job is not hard, but the state of the records showed a better system was possible.

The principle of the "double-entry system" of bookkeeping is kept alive on the two disks which are needed to run my accounts. Disk 1 holds OS-65D v3.3 with BEXEC*, Copier, Zero, Create, and the accounting programs. Disk 2 holds only the data files and the directory, To simplify this set-up, I have used the Tutorial Disk 5 of OS-65D v3.3, as it holds many of the utilities necessary to establish and back up the files. Figure 1 is a diagram of the data disk and describes it better than I can with words. Each track as shown is used as a separate "account", such as "Electricity", "Taxes", "Petty Cash", etc.

My files are detailed, a fact that might not be necessary for other groups/users. Larger files of 2 tracks or more might be necessary to handle the volume for a year's business.

Setting Up

(1) Decide on the number of accounts necessary to track money as it comes in and goes out.
(2) Decide on the maximum number of transactions per account per year.
(3) Initialize your data diskettes.

(4) Establish the directory on the data disk with the accounts named "GENL01", "GENL02", etc. The programs will be looking for these names.
(5) Run "ZERO" to write to files.
(6) Make a copy of your OS-65D v3.3 Tutorial Disk 5, and delete all of the files except the utilitiy programs mentioned above.

The program in Listing 1, called "NAME", allows you to establish the "0" record with the name each account is to be used for.

The program in Listing 2, called "INPUT2", starts "putting" transactions into accounts. Some points to watch: (1) The net total of your input for your transactions is transferred to "GENL30", which I call my Cash Account. Please alter the name in the program if you have changed from my set-up (see line 2860). A total of the postings to this account will give you the balance of your account at the bank (reversed) if you have no outstanding checks, etc. (2) This only applies to clubs. and users who are carrying forward a balance. An account will have to be started for the previous balance carried over. (3) The total of receipts put into the accounts can be verified to the total input if you "batch" your credits and debits separately. Note that the routines in lines 3125, 3170, 3210, etc. use a period (".") as a prompt and indicator of the field length (ie. maximum entry length).

Listing 3 is my amended "BEXEC*" program. It shows the overall system set-up and how the accounting programs and system utilities mesh.

Listing 4 shows the program "P STAT". With it you will start to see some results from the transactions input. This program prints all or some of the statements of accounts.

As part of the "double-entry" system of bookkeeping, it is necessary to check that the input balances for all accounts. The check of the accounts is made via a "Trial Balance". This audit is made when all the statements are printed. To put it in simple terms, it checks that all the +'s equal all of the -'s.

Listing 5 is the program named "ADJUST". It is used to correct those errors that have been input into files during the period. The reversal of the transaction has to be a complete one. For example, an original input of $1020.10 has to be reversed out in total to your "Cash Account" and the new amount put back in ($1020.13). If the reversal is not handled in this way, it will affect your true turnover.

Listing 6 is the program "F STAT". This program prints your final record of receipts and payments. Just because you have run this program, it is not necessarily the final act. In fact, this report could be used during the year to check if your budget is running to plan.

Finally, this small nucleus of programs gives you the very basic details of a cash book. There are many other programs that could access the files to give more detailed reports. Such reports could deal with profitability, cost control, and aged invoices, to name a few.

The programs were written with my limited knowledge of the "DOS", so there is much disk drive activity and wasted space on the tracks. Hopefully, there is a fix for these problems and it will be published in later copies of KAOS.

```
6000 REM- THIS PROGRAM IS SELF-EXPLANITORY
6002 PRINT "THIS PROGRAM IS TO RUN AFTER DATA FILES ARE READY"
6005 GOSUB 6350: PRINT!(28): REM- PROGRAM "NAME"
6010 PRINT:PRINT "Enter ";CHR$(34);"EXIT";CHR$(34)"; TO FINISH":PRINT:PRINT
6020 INPUT "What is the GENL number (XX) ";ZE$
6025 PRINT!(28): IF ZE$ = "EXIT" OR ZE$="exit" THEN 6200
6030 PRINT:PRINT"Please put the DATA disk in the drive now"
6040 PRINT:INPUT "What is the NAME/USAGE for this account ";RA$
6050 IF LEN(RA$)>20 THEN 6300
6060 Z$ = "GENL" + ZE$: Z$ = MID$(Z$,1,6)
6065 TRAP 6250
6070 DISK open,6, Z$
6090 DISK get,0
6100 HM$="00": HM=0
6110 PRINT#6, HM$;", ";RA$
6120 DISK PUT
6130 PRINT!(28)
6140 PRINT"DATA IS STORED": GOSUB 6350
6150 GOTO 6010
6200 PRINT"Please put the system disk in drive and press <RETURN>"
6210 INPUT "OK ";X$
6215 TRAP 6310
6220 IF X$="" THEN RUN"BEXEC*"
6230 GOTO 6200
6250 PRINT!(28);"You have the wrong disk in the drive":PRINT
6260 TRAP0: GOSUB 6350: PRINT!(28): GOTO 6010
6300 PRINT"Name is too long. 20 Characters Maximum": GOTO 6040
6310 PRINT"Are you ready": GOTO 6350: PRINT!(28): TRAP0: GOTO 6010
6350 FOR I = 1 TO 3000: NEXT I: RETURN
```

Listing 1

```
2000 PRINT!(28): REM- PROGRAM "INPUT2"
2010 PRINT"Before running this program, have you set up the disk"
2020 PRINT:PRINT"files properly by running the naming program?"
2030 PRINT:PRINT"Care must be taken as your files will be incomplete"
2040 PRINT:PRINT"if you do not follow the steps."
2050 PRINT:PRINT:INPUT "Enter 'C' if you wish to continue ";Y$
2060 IF Y$="C" OR Y$="c" THEN 2080
2070 RUN"BEXEC*"
2080 PRINT!(28): D=0: DV=2
2090 PRINT:PRINT"Make sure that you have the DATA DISK in the drive"
2100 PRINT" N O W !!!":PRINT:PRINT:PRINT: GOSUB 3330
```

```
2110 INPUT "Do you want this input printed"; X$: IF X$="Y" THEN DV=3
2120 PRINT:PRINT:INPUT"What is the date of the report ";W$
2130 PRINT!(28): REM- Screen Clear
2135 POKE 8994, DV: PRINTTAB(20);"Report Dated ";W$
2136 PRINT:PRINT:PRINT
2137 DISK!"IO ,02"
2150 REM- Works out the number of transactions in account
2160 REM  *************************************************
2165 TRAP 3340
2170 INPUT "What account is the transaction to go to ";ZE$
2175 PRINT!(28)
2180 Z$="GENL"+ZE$
2190 Z$=RIGHT$(Z$,6)
2210 DISK open,6,z$
2220 POKE 12042,32: POKE 12076,6: REM- Sets the record sizes
2230 DISK get,0
2240 INPUT#6, HM$,RA$
2250 HM=VAL(HM$): PRINT!(28)
2260 PRINT:PRINT
2270 REM **************************************************
2280 REM Input of Transactions
2300 PRINT!(28)
2310 DISK!"IO ,02": PRINT!(28)
2320 PRINT"There are";HM;" entries in the file ";RA$
2330 PRINT:PRINT
2340 PRINT"Enter ";CHR$(34);"EXIT";CHR$(34);" to finish entries"
2350 HM=HM+1: DISK GET, HM
2360 GOSUB 3120: PRINT
2370 GOSUB 3160: PRINT
2380 GOSUB 3200: PRINT
2390 GOSUB 3240: PRINT
2400 GOSUB 3290: PRINT
2410 PRINT"Length of field is too long": RETURN
2420 REM ***************************************************
2430 REM Routine for correction of mistakes
2440 PRINT!(28): PRINT"Are these details correct ???"
2450 PRINT:PRINT"Enter field number to be altered or (Y) if all OK"
2460 PRINT TAB(1);"1";TAB(13);"2";TAB(34);"3";
2470 PRINT TAB(44);"4";TAB(49);"$";"5"
2480 PRINT: PRINT TAB(1);N$; TAB(13);O$; TAB(34);P$;
2490 PRINT TAB(44);Q$; TAB(49);"$";T$
2500 PRINT:PRINT"Your Answer ";G$
2510 IF G$="1" THEN GOSUB 3120: GOTO 2440
2520 IF G$="2" THEN GOSUB 3160: GOTO 2440
2530 IF G$="3" THEN GOSUB 3200: GOTO 2440
```

```
2540 IF G$="4" THEN GOSUB 3240: GOTO 2440
2550 IF G$="5" THEN GOSUB 3290: GOTO 2440
2560 IF G$="Y" OR G$="y" THEN 2580
2570 GOTO 2440
2580 PRINT#6, N$;", ";O$;", ";P$;", ";Q$;", ";T$
2590 A = VAL(Q$+T$): D = D+A
2595 D = D*100/100
2600 DISK PUT
2610 PRINT!(28)
2620 POKE 8994,DV: REM- POKE Command to turn on printer
2630 PRINT N$; TAB(11);O$; TAB(31);P$; TAB(40);Q$;
2640 Z=LEN(T$): PRINT TAB(45);"$"; TAB(56-Z);T$;
2650 PRINT TAB(60);Z$
2660 GOTO 2310
2670 REM **************************************************
2680 REM Routine to put the new number of transactions up
2690 HM=HM-1: HM$=STR$(HM)
2700 DISK GET,0
2710 PRINT#6, HM$
2720 DISK PUT
2730 PRINT!(28)
2740 INPUT "Do you wish to input another account ";Y$
2750 IF Y$<>"Y" AND Y$<>"y" THEN 2770
2760 GOTO 2170
2770 POKE 8994,DV
2780 PRINT TAB(45);"--------------"
2790 X=LEN(D$)
2800 PRINT TAB(45);"$";: PRINT USING "######.##";D
2810 PRINT: PRINT TAB(45);"--------------"
2820 DISK!"IO ,02"
2830 D = D*-1
2840 REM ***************************************************
2850 REM Input total transactions to cash account
2860 DISK OPEN,6, "GENL30"
2870 POKE 12042,32: POKE 12076,6
2880 DISK GET,0
2890 INPUT#6, HM$,RA$
2900 HM = VAL(HM$)
2910 HM=HM+1
2920 DISK GET,HM
2930 M$="CASH ACCOUNT": N$="XXXXXX"
2940 D$=STR$(D): J$=LEFT$(D$,1)
2950 DD=LEN(D$): EE$=RIGHT$(D$,DD-1)
2960 PRINT#6,W$;", ";M$;", ";N$;", ";J$;", ";EE$
2970 DISK PUT
```

```
2980 HM$ = STR$(HM)
2990 DISK GET,0
3000 PRINT#6, HM$;", ";RA$
3010 DISK PUT
3020 PRINT!(17,25,10);"D A T A  S T O R E D"
3030 GOSUB 3330
3040 DISK CLOSE,6
3050 PRINT!(17,0,18);"Place System Disk in drive, and"
3060 INPUT "press <RETURN> to continue ";B$
3070 IF B$="" THEN RUN"BEXEC*"
3080 PRINT "Continued further input OK": GOSUB 3330: GOTO 2410
3090 REM The End
3100 REM **************************************************
3110 REM Subroutines for input of transactions
3120 PRINT"Date of Item (DD/MM/YY)"
3125 PRINT TAB(35);".........";CHR$(13);TAB(35);: INPUT N$
3130 IF N$="EXIT" OR N$="exit" THEN 2690
3140 IF LEN(N$)<9 THEN RETURN
3150 GOSUB 2410: GOTO 3120
3160 PRINT "Details of Payee or Favouree"
3170 PRINT TAB(35);"..................";CHR$(13);TAB(35);:INPUT O$
3180 IF LEN(O$)<18 THEN RETURN
3190 GOSUB 2410: GOTO 3160
3200 PRINT"Detail number of check or receipt"
3210 PRINT TAB(35);".......";CHR$(13);TAB(35);: INPUT P$
3220 IF LEN(P$)<7 THEN RETURN
3230 GOSUB 2410: GOTO 3200
3240 PRINT "Dr (-) or Cr (+)"
3245 PRINT TAB(35);"...";CHR$(13);TAB(35);: INPUT Q$
3250 IF Q$="+" OR Q$="-" THEN 3270
3260 GOTO 3240
3270 IF LEN(Q$)<2 THEN RETURN
3280 GOSUB 2410: GOTO 3240
3290 PRINT"Amount  XXXXXX.XX"
3295 PRINT TAB(35);"..........";CHR$(13);TAB(35);: INPUT T$
3300 M=VAL(T$): IF M=0 THEN 3290
3310 IF LEN(T$)<10 THEN 2440
3320 GOSUB 2410: GOTO 3290
3330 FOR I = 1 TO 1000: NEXT I
3335 RETURN
3340 PRINT"You have the wrong disk in the drive"
3350 TRAP 0
3360 GOTO 2000
```

Listing 2

```
1 REM POKE 133,126: DISK!"CA 7F00=12,5": DISK!"GO 7FC6"
5 POKE 133,126: CLEAR: POKE 14172,8: POKE 14170,16
10 POKE 2888,0: POKE 8722,0
20 X=PEEK(10950): POKE 8993,X: POKE 8994,X: DIM AL%(39)
30 IF PEEK(57088)=223 THEN POKE 9794,37
40 DEF FNA(X) = 10*INT(X/16)+X-16*INT(X/16)
50 DEF FNB(X) = 16*INT(X/10)+X-10*INT(X/10)
100 GOSUB 50000
105 PRINT: PRINT"OS-65D3.3 Accounting Disk"
110 PRINT"   by Bob Best  Nov 1985":PRINT
115 PRINT" 1 > Naming of accounts (or usage)"
120 PRINT" 2 > Input transactions to general ledger"
130 PRINT" 3 > Print Statements of Accounts"
140 PRINT" 4 > Adjustments of Accounts"
160 PRINT" 5 > Create Blank Data Diskette"
170 PRINT" 6 > Create Data Diskette with files"
180 PRINT" 7 > Final Statement of Receipts and Payments"
190 PRINT" 8 > Single or dual disk drive copier"
200 PRINT" 9 > Zero the Data Files"
890 PRINT:PRINT
900 PRINT "Type the number of your selection ";
910 INPUT "and depress <RETURN> ";S$: IF S$="PASS" THEN 60000
915 IF LEN(S$)<>1 THEN RUN
920 S=INT(VAL(S$)): IF S<1 OR S>9 THEN RUN
980 GOSUB 50010
989 PRINT"     ";
990 ON S GOSUB 1000,2000,3000,4000,5000,6000,7000,8000,9000
998 IF P$="PASS" THEN 60000
999 GOTO 100
1000 RUN "NAME"
2000 RUN "INPUT2"
2153 FORI=T1TOT2:T$=RIGHT$(STR$(I+100),2)
2155 PRINT"      Track ";T$
2160 DISK!"IN "+T$
2162 POKE 10304,169: POKE 10305,32: POKE 10549,201: POKE 10550,32
2164 DISK!"SA "+T$+",1=D000/"+P$
2166 POKE 10304,177: POKE 10305,254: POKE 10549,209: POKE 10550,254
2167 NEXT
2170 IF S=6 THEN RETURN
3000 RUN "P STAT"
3010 GOSUB 50000: PRINT "Type in the name of the file that you";
4000 RUN "ADJUST"
4232 PRINT "Data Disk Create Utility": PRINT: PRINT
5000 PRINT "Data Disk Create Utility": PRINT: PRINT
5010 PRINT"Be sure the Tutorial Disk is in Drive A": PRINT
```

```
5020 GOSUB 10200: DISK!"SE A": DISK!"CA 5C00=11,2"
5030 DISK!"CA 5D00=11,3": DISK!"CA 5E00=11,4":DISK!"CA 5F00=11.5"
5033 IF S=6 THEN DISK!"CA 5C00=11,6"
5035 GOSUB 5100: DISK!"GO 2768": DISK!"SA 12,1=5C00/1"
5040 DISK!"SA 12,2=5D00/1":DISK!"SA 12,3=5E00/1"
5050 DISK!"SA 12,4=5F00/1": IF S=6 THEN GOSUB 6000
5070 GOSUB 50010
5080 PRINT"Your diskette is now ready for data files":PRINT
5090 PRINT: GOTO 5505
5100 GOSUB 50010
5105 PRINT"Remove your Tutorial Disk from Drive A and":PRINT
5110 PRINT"replace it with your blank diskette":PRINT: GOTO 10200
5505 PRINT"Remove your blank diskette from Drive A and":PRINT
5510 PRINT"replace it with your Tutorial Disk": PRINT: GOTO 10200
6000 P$="8": T1=1: T2=10: GOSUB 2153
6010 T1=13: T2=27: GOTO 2153
7000 RUN"F STAT"
8000 X=PEEK(8960): POKE 133,X: RUN"COPIER"
9000 RUN "ZERO"
10200 INPUT "Press <RETURN> to continue ";P$: RETURN
50000 ST=11984:FORII=0TO36:READ SC:POKE ST+II,SC:NEXT:RESTORE
50010 IFPEEK(8999)=58THEN PRINTCHR$(27);CHR$(21):POKE56832,1:RETURN
50015 POKE 8955,208
50020 POKE 8956,346: X=USR(X): RETURN
50030 DATA 169,208,141,219,46,169,32,162,0,157,0,208,232
50040 DATA 208,250,172,219,46,200,140,219,46,192,232,240,10
50050 DATA 192,216,208,235,160,224,169,14,208,239,96
59000 POKE 741,76:POKE 750,78:POKE 2073,173:POKE 2893,55:POKE 2894,8
59010 POKE 2888,27: X=PEEK(8960): POKE 133,X
59020 RETURN
60000 GOSUB 59000
60010 GOSUB 50000: CLEAR
60020 PRINT"The system is now open for modification."
```

Listing 3

```
4000 REM This is the Print Statement Program
4010 PRINT "Please put the data disk in the disk drive now":PRINT
4020 INPUT "and press <RETURN> when ready ";L$
4030 IF L$="" THEN 4050
4040 RUN"BEXEC*"
4050 DIM E$(39),F(39),F$(39)
4060 DV=2
4070 PRINT:PRINT:PRINT"This allows statements to be printed"
```

```
4080 PRINT:PRINT
4090 PRINT!(28):INPUT "What is the statement date ";W$
4100 PRINT:PRINT
4110 INPUT "Do you want to print it ";Y$: IF Y$="Y" THEN DV=3
4120 PRINT:PRINT
4130 INPUT "Do you want them all (Y/N) ";V$
4140 IF V$="Y" OR V$="y" THEN 4490
4150 INPUT "What account number do you want ";ZE$
4160 Z$="GENL"+ZE$: Z$=RIGHT$(Z$,6)
4170 GOSUB 4240
4180 INPUT "Do you want another ";X$: IF X$="Y" THEN 4150
4190 POKE 8994,2: PRINT"Please put the system disk in drive and"
4200 INPUT"press <RETURN> to return to main menu ";C$
4210 RUN"BEXEC*"
4220 REM **************************************************
4230 REM Print Routine
4240 POKE 8994,DV: REM POKE for printer start
4250 PRINT TAB(23);"ACCOUNT ";Z$
4260 DISK OPEN,6,Z$
4270 POKE 12042,32: POKE 12076,6
4280 IF B=12 THEN GOSUB 4600: GOTO 4440
4285 TRAP 4615
4290 DISK GET,0
4300 INPUT #6, HM$,RA$
4310 HM=VAL(HM$): IF HM=0 THEN GOSUB4600: GOTO 4440
4320 PRINT TAB(23);RA$: PRINT: PRINT
4330 DISK!"IO ,02"
4340 HM=VAL(HM$): FOR NO=1 TO HM: DISK GET,NO
4350 INPUT#6, N$,O$,P$,Q$,T$
4360 POKE 8994,DV
4370 PRINT N$;TAB(11);O$;TAB(31);P$;TAB(40);Q$;
4380 Z=LEN(T$): PRINT TAB(45);"$";TAB(56-Z);T$
4390 A=VAL(Q$+T$)
4400 D=D+A: D=D*100/100: NEXT
4410 PRINT TAB(45);"--------------"
4420 X=LEN(D$): PRINT TAB(45);"$";: PRINT USING"XXXXXX.##";D
4430 PRINT:PRINT TAB(45);"--------------"
4440 PRINT:PRINT:PRINT:PRINT
4450 DISK!"IO ,02"
4460 E$(B)=RA$: F(B) = D
4470 DISK CLOSE,6
4480 D=0: RETURN
4490 REM ***************************************************
4500 REM Calculation of File Name on Complete Listing
4510 FOR B = 1 TO 38
```

```
4520 IF B<10 THEN 4580
4530 J$=STR$(B): J$=RIGHT$(J$,2)
4540 RC$="GENL": Z$ = RC$+J$
4550 GOSUB 4240
4560 NEXT B
4570 GOTO 4620
4580 J$=STR$(B): J$=RIGHT$(J$,1): J$="0"+J$
4590 GOTO 4540
4600 PRINT "There are NO TRANSACTIONS ON  ";Z$
4610 RA$="0": D=0: RETURN
4615 PRINT"!!! ERROR !!! IN TRACK HEADER": TRAP0: GOTO 4620
4620 REM ****************************************************
4630 REM Trial Balance Print
4640 INPUT "Do you want the Trial Balance ";G$
4650 IF G$="Y" OR G$="y" THEN 4670
4660 GOTO 4190
4670 POKE 8994,3
4680 PRINT:PRINT:PRINT:PRINT TAB(25);"Trial Balance ":PRINT:PRINT:PRINT
4690 FOR B = 1 TO 39
4700 F$(B)=STR$(F(B))
4710 IF LEFT$(F$(B),1)="-" THEN GOSUB 4830
4720 IF LEFT$(F$(B),1)=" " THEN GOSUB 4840
4730 J=0: NEXT B
4740 PRINT:PRINT:PRINT"Total  -'S = $";: PRINT USING"######.##";K
4750 PRINT:PRINT"Total  +'S = S";: PRINT USING"######.##";H: PRINT
4760 IF H=K*-1 THEN PRINT"Difference = $";:PRINT USING"######.##";J: GOTO
4190
4780 PRINT:PRINT"An ERROR has occurred in your records. Please check"
4790 PRINT "that each total input has been posted to your"
4800 PRINT "Cash Account correctly."
4810 PRINT
4820 PRINT: GOTO 4190
4830 F(B)=VAL(F$(B))/100: H=H+(F(B)*100): RETURN
```

Listing 4

```
5000 REM This program lets you adjust previously input transactions.
5005 PRINT!(28): REM PROGRAM "ADJUST"
5010 PRINT"Make sure Data Disk in now in drive!":PRINT:PRINT
5020 PRINT"This program lets you pass adjustments on accounts":PRINT
5030 INPUT "What is the date of the change ";Y$
5040 PRINT:PRINT
5050 Z1$="0"
5060 INPUT "What is the account number ";Z1$
```

```
5070 N$="0":O$="0":P$="0":Q$="0":R$="0":RA$="0":D$="0":V$="0":U$="0"
5080 RA$="0": X$="0"
5090 PRINT:PRINT
5100 A$="GENL"
5110 Z$=A$+Z1$: Z$=RIGHT$(Z$,6)
5120 PRINT"Details of the transaction to be reversed":PRINT
5130 INPUT "Detail Number ";N$
5140 PRINT
5150 INPUT "Amount $";M$
5160 DISK OPEN,6,Z$: GOSUB 5670
5170 D$=RA$
5180 IF HM=0 THEN PRINT"NO INFORMATION ON FILE": GOTO 5730
5190 FOR NO = 1 TO HM: DISK GET,NO
5200 INPUT#6, W$,O$,P$,Q$,T$
5210 IF P$=N$ AND T$=M$ THEN 5240
5220 NEXT NO
5230 PRINT"NO TRANSACTION FOUND": GOTO 5730
5240 PRINT!(28):PRINT:PRINT"Transaction to be reversed ???"
5250 PRINT:PRINT"Account ";Z$;" used for ";RA$
5260 PRINT:PRINT"Date";TAB(20);W$
5270 PRINT:PRINT"Detail";TAB(20);O$
5280 PRINT:PRINT"Detail No. ";TAB(20);P$
5290 PRINT:PRINT"Amount";TAB(20);"$";T$
5300 PRINT:INPUT "Is this the transaction ";B$
5310 IF B$="Y" OR B$="y" THEN 5330
5320 GOTO 5730
5330 RR$="Reversed "
5340 R$=RR$+Y$
5350 L$="0": PRINT#6,W$;", ";R$;", ";P$;", ";Q$;", ";L$
5360 DISK CLOSE,6
5370 HM=0: RA$="0"
5380 REM *************************************************
5390 REM Routine to put new transaction up on file
5400 INPUT "Account that the reversal is to go to ";U$
5410 X$=A$+U$: X$=RIGHT$(X$,6)
5420 DISK OPEN,6,X$: GOSUB 5670
5430 HM=HM+1: DISK GET, HM
5440 V$=RA$: M$=RR$+Z$
5450 PRINT#6, Y$;", ";M$;", ";P$;", ";Q$;", ";T$
5460 DISK PUT
5470 GOSUB 5700
5480 REM *************************************************
5490 REM Print Report - No Option
5500 POKE 8994,3
5510 PRINT TAB(15);"Reversal Report Dated ";Y$
```

```
5520 PRINT:PRINT
5530 PRINT"Original Transaction on account - ";D$
5540 PRINT W$; TAB(11);O$; TAB(31);P$; TAB(40);Q$;
5550 PRINT TAB(45);"$";: PRINT USING"######.##";T$: PRINT
5560 PRINT"What is now on file - ";D$
5570 PRINT W$; TAB(11);R$; TAB(31);P$; TAB(40);Q$;
5580 PRINT TAB(45);"$";: PRINT USING"######.##";L$:PRINT
5590 PRINT "New Transaction now on file - ";V$
5600 PRINT Y$; TAB(11);M$; TAB(31);P$; TAB(40);Q$;
5610 PRINT TAB(45);"$";: PRINT USING"######.##";T$
5620 POKE 8994,2: INPUT "Do you want to reverse another ";E$
5630 IF E$="Y" OR E$="y" THEN 5050
5640 PRINT:PRINT"Please put System Disk back in drive and"
5650 INPUT "press <RETURN> when ready ";F$
5660 IF F$="" THEN 5750
5670 POKE 12042,32: POKE 12076,6
5680 DISK GET,0: INPUT#6, HM$,RA$: HM=VAL(HM$)
5690 RETURN
5700 HM$=STR$(HM): DISK GET,0: PRINT#6, HM$;", ";RA$
5710 DISK PUT
5720 RETURN
5730 INPUT "Do you want to try again ";C$
5740 IF C$="Y" OR C$="y" THEN 5050
5750 RUN "BEXEC*"
```

Listing 5

```
8000 REM This is the Print Final Statement Program
8010 PRINT!(28): REM- PROGRAM "F STAT"
8020 PRINT"Please put the data disk in the drive now!": PRINT
8030 INPUT"Press <RETURN> when ready ";L$
8040 DV=2: IF L$="" THEN 8050
8050 DIM E$(39),F(39),F$(39),E(39)
8070 PRINT!(28): INPUT "What is the statement date ";W$
8080 PRINT:PRINT
8090 INPUT "Do you want to print it ";Y$: IF Y$="Y" THEN DV=3
8100 GOTO 8340
8110 INPUT "Put System Disk in drive and press <RETURN>"; CC$
8120 IF CC$="" THEN RUN"BEXEC*"
8130 REM ***********************************************
8140 REM Installation of totals into memory for later printing
8150 DISK OPEN,6,Z$
8160 POKE 12042,32: POKE 12076,6
8170 TRAP 8480
```

```
8180 DISK get,0
8190 TRAP 0
8200 INPUT #6,HM$,RA$
8210 HM=VAL(HM$): IF HM=0 THEN GOSUB 8460: GOTO 8420
8220 FOR NO = 1 TO HM
8230 DISK GET, NO
8240 INPUT #6, N$,O$,P$,Q$,T$
8250 A$=Q$+T$
8260 IF Q$="-" THEN 8280
8270 IF q$<>"-" THEN 8310
8280 A=VAL(A$): D=D+A: IF B=30 THEN 8330
8290 T=T+A
8300 NEXT NO: RETURN
8310 A=VAL(A$): K=K+A: IF B=30 THEN 8330
8320 U=U+A
8330 NEXT NO: RETURN
8340 REM ****************************************************
8350 REM Calculation of file name on complete listing
8360 FOR B = 1 TO 38
8370 IF B<10 THEN 8440
8380 J$=STR$(B): J$=RIGHT$(J$,2)
8390 RC$="GENL": Z$=RC$+J$
8400 GOSUB 8150
8410 E$(B)=RA$: F$(B)=RA$: E(B)=K: F(B)=D: K=0: D=0
8420 NEXT B
8430 GOTO 8500
8440 J$=STR$(B): J$=RIGHT$(J$,1): J$="0"+J$
8450 GOTO 8390
8460 PRINT "There are NO TRANSACTIONS ON ";Z$
8470 RA$="0": D=0: RETURN
8480 PRINT"!!! ERROR !!! IN TRACK HEADER": TRAP0: GOTO 8490
8490 REM ************************************************
8500 REM Statement of Receipts and Payments
8510 POKE 8994,DV
8520 PRINT:PRINT:PRINT TAB(21);"Statement of Receipts and Payments"
8530 PRINT:PRINT TAB(32);"for ";W$: PRINT
8540 PRINT TAB(15);"Receipts"; TAB(53);"Payments":PRINT
8550 FOR B=1TO29: REM CHANGE ****************************
8560 IF E(B)=0 AND F(B)=0 THEN 8630
8570 IF F(B)=0 THEN 8610
8580 IF E(B)=0 THEN 8620
8590 PRINT E$(B); TAB(25);: PRINT USING"######.##";E(B);
8600 PRINT TAB(37);F$(B); TAB(60);: PRINT USING"######.##";F(B): GOTO 8630
8610 PRINT E$(B); TAB(25);: PRINT USING"######.##";E(B): GOTO 8630
8620 PRINT TAB(37);F$(B); TAB(60);: PRINT USING"######.##";F(B)
```

```
8630 NEXT B
8640 BB=E(30)+F(30): IF E(30)+F(30) >1 THEN 8670
8650 EE=BB+T:PRINT TAB(37);F$(B);TAB(60);: PRINT USING"######.##";BB
8660 GOTO 8680
8670 DD=BB+U:PRINT E$(B); TAB(25);: PRINT USING"######.##";DD
8680 PRINT:PRINT TAB(25);"========   ; TAB(60);"=========="
8690 IF DD=0 THEN DD=U: IF EE=0 THEN EE=T
8700 PRINT TAB(25);: PRINT USING"######.##";DD;
8710 PRINT TAB(60);: PRINT USING"######.##";EE
8720 PRINT: PRINT TAB(25);"=========="; TAB(60);"=========="
8730 POKE 8994,2: INPUT "Do you want another copy "; AA$
8740 IF AA$="Y" OR AA$="y" THEN 8490
8750 GOTO 8110
8760 PRINT TAB(37);F$(B); TAB(60);: PRINT USING"######.##";F(B)
```

Listing 6

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

See Previous Issues for more Book Bargains! Please include reasonable postage

Software Spectacular!

C1P/Superboard Cassettes

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

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

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

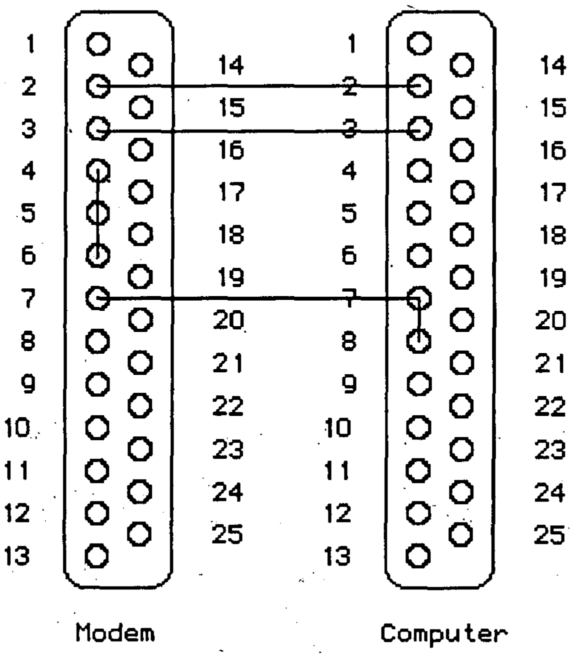
Sale Price $15.00

A Simple Terminal Program for CompuServe

The program presented here is very simple. It provides basic communication with a remote host. It also supports downloading files from CompuServe using their "A" protocol.

What is a protocol? It is a format for exchanging information via modem. The parties on each end of the connection agree to send special signals to one another to make sure that the information has been received as sent. The information is sent in small blocks which are often referred to as "packets". At the end of each packet, a special character (or sometimes characters) is sent that is a calculation based on the contents of the packet. If the receiver's calculations on the data received agrees with this special character, he signals the sender to send the next packet. If not, he sends a different signal to tell the sender to try again. If the effort fails a certain number of times, then both parties stop trying.

This program will run on any OSI disk-based system. On serial systems, it will support up to 1200 baud, but video systems will be limited to 300. Sorry, guys. To configure the program, make sure you change the value of "MODADR" to reflect the address of the serial port you've connected your modem to. In addition, you may need to change "CONFIG" to alter the speed. See Eddie Gieske's article on the 6850 ACIA chip in a previous PEEK[65] for details on changing this value. You should only change bits 0 and 1 of this byte, which control the baud rate. The other bits will give you 8 data bits, 1 stop bit, and no parity - a setting which will give you good results with just about any host system.



Directly wire pins 2, 3, and 7. On the connector to be attatched to the modem, jumper pins 4, 5, and 6. On the connector to be attatched to the computer, jumper pins 7 and 8.

Figure 1

Once assembled, the program resides in the transient language area of OS-65D, beginning at $0200. Note that if you want to move the program to a higher location (to attatch it to a BASIC program, for example), you'll have to add code to save BASIC's page zero contents. If you do end up running this program from BASIC, don't forget to adjust your point of entry to reflect whether OS-65D is in the operating system or language context. If you're unsure of those terms, keep it simple and don't try to move the program.

Note that this program cannot create files on its own like more sophisticated terminal programs. You'll have to prepare for downloading by creating the files you need ahead of time. Its best to do as much of this kind of thing off-line, since CompuServe starts counting as soon as you receive the greeting after you've entered your password.

The advantage of being able to use the A Protocol is twofold. First, you save money by getting error-free transfers.

Second, protocol transfers can accomodate full 8-bit bytes (another reason for the 8 data bits setting), so we aren't limited to simple program listings in ASCII. We can transfer exact images of program and data files. Further, machine code programs can also be exchange without need for source code or assembling.

Once you arrive in the Computer Club Forum on CompuServe ("GO CLUB"), leave me a message if you need help. Just address the message to "SYSOP" and leave it in section 8. You'll be prompted for all of this information when you leave your message.

There's a lot of on-line help available on CompuServe. In fact, you can enter "HELP" at any prompt on the system and there will always be some waiting for you. I've also posted a couple of files that you can read on-line that will help you with some OSI-specific problems.

The files for OSI are in section 8 of the Data Library in the Computer Club forum. To enter the Data Library, just enter "DL8". To examine the files available, enter "BRO", which stands for "BROWSE". You'll see all of the files, one by one, with a description of each. To download the file whose listing you're seeing, enter "D" at the "(R D M)" prompt. The system will ask you which protocol you want to use. Be sure to select the "A" protocol from the menu. The system will ask you to enter a filename for your computer.

```
10 ; SPECIAL TERMINAL EXECUTIVE
20 ; WRITTEN BY RICHARD L. TRETHEWEY
30 ; 11/1/83
40 ;
50 ; OS-65D EXTERNALS
60 ;
70       TMP2   =$FB
80       TMP    =$FD
90       MAXMEM =$2300
100      INFLAG =$2321
110      OUFLAG =$2322
120      INCH   =$2340
130      OUTCH  =$2343
140      DISC   =$265C
150      SECT   =$265E
160      PAGES  =$265F
170      ADRLX  =$2660
180      ADRHX  =$2661
190      TRAKX  =$2662
200      HOME0  =$2663
210      SEEKX  =$26A6
220      MRKT   =$267A
230      LOAD   =$2754
240      UNLOAD =$2761
250      SAVEX  =$27D7
260      CALLX  =$295D
270      SELECT =$29C6
280      ERROR  =$2A4B
290      OS65D3 =$2A51
300      ERRSU  =$2A7D
310      DEFAUL =$2AC5
320      SRCSIZ =$2BE9
330      SWAP   =$2CF7
340      CRLF   =$2D6A
350      STROUT =$2D73
360      PRBYTE =$2D92
370      DIRTRK =$2DC4
380      TXTBUF =$2E1E
390      DIRBUF =$2E79
400      CRSCHR =$32E2
430      HZLPRT =$33C0
440      KEYIN  =$3590
450      CASECK =$3A5F
460      SRCSTR =$3A79
470 ;
480 ; LOCAL EXTERNALS
490 ;
500      SAVADR =$01
510      SOH    =$01
520      ETX    =$03
530      INDEX  =$04
540      EOT    =$04
550      KEYNUM =$06
560      CHKS   =$0D
570      ORN    =$0E
580      SO     =$0E
590      NRN    =$0F
600      SI     =$0F
610      SLEN   =$10
620      EFFLAG =$11
680      DLE    =$10
690      ACK    =$2E
700      NAK    =$2F
710      INBUF  =$2280
720      MDCTRL =$F7D3
730      STATUS =$FC00
```

Since this information is passed in the signal that tells our terminal program a file transfer is beginning, I used it to designate both the file name and its drive location. Thus, when CompuServe asks you for a file name for your system, respond with the name of the file you created, followed by a slash ("/"), followed by the letter that corresponds to the drive which that file resides on (A through D). For example;

MYFILE/A

would tell the terminal program to save the incoming data in the file named "MYFILE" located on drive A. Its that simple. If the terminal program can't find the file you've named, you'll be asked to enter the drive and file name by hand.

There's a lot more that could easily be added to this program. For example, you may want to add the ability to capture incoming text in disk files or to send other files. Documentation on the A protocol is available in the Programmer's Forum on CompuServe.

Have fun! I hope we'll see you on-line soon!

```
740          MODEM  =$FC01
750 ;
760 ; ASSEMBLY CONSTANTS
770 ;
780          CTRLA  =$01
790          CTRLB  =$02
800          CTRLC  =$03
810          CTRLD  =$04
820          CTRLU  =$15
830          LF     =$0A
840          BS     =$08
850          CR     =$0D
860          SP     =$20
870          SKIP2  =$2C
880          ESC    =$1B
890          DEL    =$5F
900 ;
910          *=$0200
920          JMP START
930 ;
940 PNAME   JSR STROUT
950 CURFIL  .BYTE 'XXXXXX'
960         .BYTE $00
970         RTS
980 ESCBYT  .BYTE $1B
990 CLSBYT  .BYTE $1C
1020  RESLO  .BYTE $00
1030  RESHI  .BYTE $00
1040  FIFTH  .BYTE $00
1070  STTK   .BYTE $00
1080  ENDTK  .BYTE $00
1090  STKPTR .BYTE $00
1100  BFENPG .BYTE $00
1110  COUNT  .BYTE $00
1120  MODADR .WORD $FC00
1130  CONFIG .BYTE $16
1140  TOTAL  .BYTE $00
1150 ;
1160  DRSEL  JSR STROUT
1170         .BYTE CR,LF,'Drive (A/B/C/D) ? ',0
1180         JSR GETSTR
1190         LDA INBUF
1200         JSR CASECK
1210         CMP #'A
1220         BCC DRSEL
1230         CMP #'E
1240         BCS DRSEL
1250  DRS1   AND #$F
1260         CMP DISC
1270         BEQ DRS2-3
1280         STA TOTAL
1290         JSR SWAP
1300         LDA TOTAL
1310         JSR SELECT
1320         BCS DRS2
1330         JSR HOME0
1340         JSR SWAP
1350         JMP CRLF
1360  DRS2   LDA #$06
1370         JMP ERROR
1380 ;
1390  SCRCLR LDA ESCBYT
1400         JSR OUTCH
1410         LDA CLSBYT
1420         JMP OUTCH
1430 ;
```

```
1440  START  LDA #WARM
1450         LDY #WARM/256
1460         JSR ERRSU
1470         LDA DEFAUL+1
1480         STA INFLAG
1490         STA OUFLAG
1500         JSR SCRCLR
1510         LDA #$05
1520         STA $DE00
1530         LDX MRKT+1
1540         CPX #49
1550         BEQ STA1
1560         LDA #$08
1570         .BYTE SKIP2
1580  STA1   LDA #$04
1590         STA $363C
1600 ;
1610 ; WARM START RE-ENTRY POINT
1620 ;
1630  WARM   JSR SWAP
1640  WARMNS LDX #$FE
1650         TXS
1660  MENU   JSR STROUT
1670         .BYTE CR,LF
1680         .BYTE '    Terminal
Executive',CR,LF,LF
1690         .BYTE '(1) Exit to 65D',CR,LF
1700         .BYTE '(2) Enter Terminal
Mode',CR,LF,LF
1710         .BYTE '    Your Selection ? ',0
1720         JSR GETSTR
1730         JSR SCRCLR
1740         LDY #$00
1750         LDA INBUF,Y
1760         JSR CASECK
1770         CMP #'1
1780         BEQ EXIT
1790         CMP #'2
1800         BEQ TERM
1810  INERR  JSR STROUT
1820         .BYTE 'INVALID ENTRY'
1830         .BYTE CR,LF,LF,$00
1840         JMP MENU
1850 ;
1860  EXIT   JSR SWAP
1870         LDA #OS65D3
1880         LDY #OS65D3/256
1890         JSR ERRSU
1900         JMP OS65D3
1910 ;
1920  TERM   JSR GOTERM
1930         JMP MENU
1940 ;
1950 ; STRING INPUT ROUTINE
1960 ;
1970  GETSTR LDY #$00
1980  GETS1  JSR INCH
1990         STA INBUF,Y
2000         CMP #CR
2010         BEQ GETS2
2020         CMP #DEL
2030         BEQ BKSPC
2040         CMP #DEL+$20
2050         BEQ BKSPC
2060         INY
2070         BNE GETS1
2080  GETS2  STY TMP2
2090         JMP CRLF
2100 ;
2110  BKSPC  TYA
2120         BEQ GETS1
2130         PHA
2140         JSR STROUT
2150         .BYTE BS,BS,SP,SP,BS,BS,0
2160         PLA
2170         TAY
2180         DEY
2190         JMP GETS1
2200 ;
2210  GETANS JSR GETSTR
2220         LDA INBUF
2230         JSR CASECK
2240         CMP #'Y
2250         RTS
2260 ;
2270 ; INPUT FILE NAME AND FIND IT
2280 ;      IN THE DIRECTORY
2290 ;
2300  FNDFIL JSR STROUT
2310         .BYTE 'File Name ? ',0
2320         LDY #$00
2330         LDA #SP
2340  FNDF0  STA CURFIL,Y
2350         INY
2360         CPY #$06
2370         BNE FNDF0
2380         JSR GETSTR
2390         LDY #$00
2400  FNDF1  LDA INBUF,Y
2410         CMP #CR
2420         BEQ FNDF2
2430         STA CURFIL,Y
2440         INY
2450         CPY #$07
2460         BNE FNDF1
2470         JSR STROUT
2480         .BYTE CR,LF
2490         .BYTE 'TOO LONG',CR,LF,LF,0
2500         JMP FNDFIL
2510  FNDF2  TYA
2520         BEQ FNDFIL
2530  FNDF3  LDA #$01
2540         STA COUNT
2550  FNDF4  JSR SWAP
2560         JSR DIRIN
2570         JSR SWAP
2580         LDY #$00
2590         LDX #$00
2600  FNDF5  LDA CURFIL,X
2610         JSR CASECK
2620         STA TMP
2630         LDA DIRBUF,Y
2640         JSR CASECK
2650         CMP TMP
2660         BNE FNDF6
2670         INY
2680         INX
2690         CPX #$06
2700         BNE FNDF5
2710         BEQ FNDF8
2720  FNDF6  INY
2730         BEQ FNDF7
```

```
2740           INX
2750           CPX #$08
2760           BNE FNDF6
2770           LDX #$00
2780           BEQ FNDF5
2790  FNDF7    INC COUNT
2800           LDA COUNT
2810           CMP #$03
2820           BNE FNDF4
2830           SEC
2840           RTS
2850  FNDF8    LDA DIRBUF,Y
2860           JSR BCDH
2870           STA STTK
2880           INY
2890           LDA DIRBUF,Y
2900           JSR BCDH
2910           STA ENDTK
2920           CLC
2930           RTS
2940 ;
2950  GOTERM JSR SETPTR
2960           LDA SRCSIZ
2970           CMP #$08
2980           BEQ GOTRM1
2990           LDA #$0C
3000  GOTRM1 CLC
3010           ADC ADRHX
3020           STA BFENPG
3030           JSR SETUP
3040           LDA #ERRTRM
3050           LDY #ERRTRM/256
3060           JSR ERRSU
3070           LDA #$00
3080           TAY
3090  GOTRM2 STA SAVADR,Y
3100           INY
3110           BPL GOTRM2
3120           TSX
3130           STX STKPTR
3140           JMP P0
3150  ;
3160  ERRTRM JSR SWAP
3170           LDX STKPTR
3180           TXS
3190           LDA #CTRLU
3200           JSR XMIT
3210           JMP P0
3220  ;
3230  SETUP    LDA #$34
3240           STA MDCTRL
3250           LDX DEFAUL+1
3260           LDA INLO-1,X
3270           STA CNSLIN+1
3280           LDA INHI-1,X
3290           STA CNSLIN+2
3300           LDA OUTL2-1,X
3310           STA CNSLOU+1
3320           LDA OUTH2-1,X
3330           STA CNSLOU+2
3340           LDA MODADR
3350           STA XIN+1
3360           STA X1+1
3370           STA GTR3+1
3380           STA GTR4+1
3390           CLC
```

```
3400           ADC #$01
3410           STA X2+1
3420           STA X3+1
3430           LDA MODADR+1
3440           STA GTR3+2
3450           STA GTR4+2
3460           STA X1+2
3470           STA X2+2
3480           STA XIN+2
3490           STA X3+2
3500           LDA #$03
3510  GTR3     STA STATUS
3520           LDA CONFIG
3530  GTR4     STA STATUS
3540           RTS
3550 ;
3560  INLO     .BYTE TTYIN,KEYIN
3570  INHI     .BYTE TTYIN/256,KEYIN/256
3580  OUTL2    .BYTE TTYOUT, HZLPRT
3590  OUTH2    .BYTE TTYOUT/256,HZLPRT/256
3600 ;
3610  TGLDUP LDA P9
3620           EOR #$0C
3630           STA P9
3640 ;
3650 ; MAIN LOOP ENTRY POINT
3660 ;
3670  P0       JSR XIN
3680           BCC P3
3690           AND #$7F
3700           CMP #SI
3710           BNE P2
3720           JMP PRTXX
3730  P2       JSR CNSLOU
3740  P3       JSR CNSLIN
3750           BEQ P0
3760           CMP #CTRLD
3770           BEQ TGLDUP
3780           CMP #CTRLB
3790           BEQ BACK
3800  P6       CMP #DEL
3810           BEQ P7
3820           CMP #DEL+$20
3830           BNE P8
3840  P7       JSR STROUT
3850           .BYTE BS,SP,$00
3860           LDA #BS
3870  P8       JSR XMIT
3880  P9       BIT CNSLOU
3890           JMP P0
3900 ;
3910 ; ROUTINE TO SEND CHARACTER OUT MODEM
PORT
3920 ;
3930  XMIT     PHA
3940  X1       LDA STATUS
3950           LSR A
3960           LSR A
3970           BCC X1
3980           PLA
3990  X2       STA MODEM
4000           RTS
4010 ;
4020 ; MAIN EXIT POINT
4030 ;
4040  BACK     LDA #$03
```

```
4050  B2      STA STATUS
4060          LDA #$11
4070  B3      STA STATUS
4080          LDA #60
4090          STA MDCTRL
4100          JSR SCRCLR
4110          LDX STKPTR
4120          TXS
4130          LDA #WARM
4140          LDY #WARM/256
4150          JMP ERRSU
4160 ;
4170  WRIT    LDA #$01
4180          STA SECT
4190          LDA SRCSIZ
4200          CMP #$08
4210          BEQ WRIT1
4220          LDA #$0C
4230  WRIT1   STA PAGES
4240          JMP WRITE
4250 ;
4260  REED    LDA #$01
4270          STA SECT
4280          JMP READ
4290 ;
4300  SETPTR JSR SETADR
4310          LDA ADRLX
4320          STA SAVADR
4330          LDA ADRHX
4340          STA SAVADR+1
4350          RTS
4360 ;
4370  FILSEL JSR DRSEL
4380          JSR FNDFIL
4390          BCS NOTF
4420          RTS
4430 ;
4440  NOTF    JSR STROUT
4450          .BYTE 'FILE NOT FOUND',CR,LF,LF
4460          .BYTE 'Did you want to try
again ? ',$00
4470  NOTF1   JSR GETANS
4480          BEQ FILSEL
4490          LDX STKPTR
4500          TXS
4510          JMP P0
4520 ;
4530  CNSLIN JMP $FFFF
4540 ;
4550  CNSLOU JMP $FFFF
4560 ;
4570 ; SERIAL CONSOLE INPUT ROUTINE
4580 ;
4590  TTYIN   LDA STATUS
4600          LSR A
4610          BCC TTYIN2
4620  TTYIN1 LDA MODEM
4630          AND #$7F
4640          RTS
4650  TTYIN2 LDA #$00
4660          RTS
4670 ;
4680 ; SERIAL COUNSOLE OUTPUT ROUTINE
4690 ;
4700  TTYOUT PHA
4710  TTYO1   LDA STATUS
```

```
4720          LSR A
4730          LSR A
4740          BCC TTYO1
4750          PLA
4760  TTYO2   STA MODEM
4770          RTS
4780 ;
4790 ; MODEM INPUT ROUTINE
4800 ;
4810  XIN     LDA STATUS
4820          LSR A
4830          BCC X4
4840  X3      LDA MODEM
4850  X4      RTS
4860 ;
4870 ; READ A SECTOR OF THE DIRECTORY
4880 ;      TRACK INTO "DIRBUF"
4890 ;
4900  DIRIN   LDA #DIRBUF
4910          STA ADRLX
4920          LDA #DIRBUF/256
4930          STA ADRHX
4940          LDA COUNT
4950          STA SECT
4960          LDA DIRTRK
4970          JSR BCDH
4980          STA TRAKX
4990          JSR SEEKX
5000          JMP READ+3
5010 ;
5020 ; BCD TO HEX CONVERSION ROUTINE
5030 ;
5040  BCDH    PHA
5050          AND #$F0
5060          LSR A
5070          LSR A
5080          LSR A
5090          LSR A
5100          TAX
5110          LDA #$00
5120  BCDH1   CLC
5130          ADC #$A
5140          DEX
5150          BNE BCDH1
5160          STA TMP
5170          PLA
5180          AND #$F
5190          CLC
5200          ADC TMP
5210          RTS
5220 ;
5230 ; COMPUTE AND SET DISK BUFFER ADDRESS
5240 ;
5250  SETADR LDA #$00
5260          STA ADRLX
5270          LDA MAXMEM
5280          SEC
5290          SBC SRCSIZ
5300          SBC #$02
5310          STA ADRHX
5320          RTS
5330 ;
5340 ; WRITE BUFFER TO DISK
5350 ;
5360  WRITE   JSR SETADR
5370          JSR LOAD
```

```
5380         JSR SAVEX
5390         JMP UNLOAD
5400 ;
5410 ; READ DISK TO BUFFER
5420 ;
5430  READ   JSR SETADR
5440         JSR LOAD
5450         JSR CALLX
5460         JMP UNLOAD
5470 ;
5480  IDSTR  .BYTE '#CPMPMMI,CC,,PA,',CR
5490  ;
5500  PRTCL  LDY #$00
5510  PRTC1  LDA IDSTR,Y
5520         JSR XMIT
5530         INY
5540         CMP #CR
5550         BNE PRTC1
5560  PRTC2  RTS
5570 ;
5580  PRTP0  LDX STKPTR
5590         TXS
5600         LDA #ERRTRM
5610         LDY #ERRTRM/256
5620         JSR ERRSU
5630         LDA #171
5640         STA CRSCHR
5650         JMP P0
5660 ;
5670  PRTXX  LDA #22
5680         STA CRSCHR
5690 ;
5700  PRTXX0 JSR XIN
5710         BCC PRTXX1
5720         AND #$7F
5730         CMP #ESC
5740         BEQ PRTXX2
5750         CMP #SO
5760         BEQ PRTP0
5770         JSR CNSLOU
5780  PRTXX1 JSR CNSLIN
5790         BEQ PRTXX0
5800         CMP #CTRLC
5810         BEQ PRTP0
5820         CMP #DEL
5830         BEQ PRTXBS
5840         CMP #DEL+$20
5850         BNE PRTXX5
5860  PRTXBS JSR STROUT
5870         .BYTE BS,SP,$00
5880         LDA #BS
5890  PRTXX5 JSR XMIT
5900         JMP PRTXX0
5910 ;
5920  PRTXX2 JSR PRTXIN
5930         AND #$7F
5940         CMP #'I
5950         BNE PRTXX3
5960         JSR PRTCL
5970         JMP PRTXX0
5980 ;
5990  PRTXX3 CMP #'A
6000         BEQ PRTXX6
6010         JMP PRTXX0
6020 ;
6030  PRTXX6 LDA #PRTERR
```

```
6040         LDY #PRTERR/256
6050         JSR ERRSU
6060  PRTC3  JSR PRTXIN
6070         CMP #SOH
6080         BNE PRTC3
6090         LDA #$00
6100         STA CHKS
6110  PRTC4  JSR PRTXIN
6120         STA ORN
6130         JSR PRTCHK
6140  PRTC5  JSR PRTXIN
6150         STA SLEN
6160         JSR PRTCHK
6170         LDY #$00
6180         JSR PRTXIN
6190         STA KEYNUM
6200         JSR PRTCHK
6210  PRTH1D JSR PRTXIN
6220         CMP #ETX
6230         BEQ PRTH1
6240         JSR PRTCHK
6250         AND #$7F
6260         STA INBUF,Y
6270         JSR CNSLOU
6280         INY
6290         BNE PRTH1D
6300  PRTH1  JSR PRTXIN
6310         CMP CHKS
6320         BEQ PRTH2
6330         LDA #NAK
6340         JSR XMIT
6350         JMP PRTC3
6360  PRTH2  JSR SETPTR
6370         LDY #$00
6380  PRTH3  LDA INBUF,Y
6390         CMP #'/
6400         BEQ PRTH4
6410         INY
6420         BNE PRTH3
6430         JMP PRTH59
6440  PRTH4  LDA #CR
6450         STA INBUF,Y
6460         INY
6470         LDA INBUF,Y
6480         AND #$0F
6490         JSR DRS1
6500         LDY #$00
6510         LDA #SP
6520  PRTH5  STA CURFIL,Y
6530         INY
6540         CPY #$06
6550         BNE PRTH5
6560         LDY #$00
6570         JSR FNDF1
6580         BCC PRTH6
6590         JSR PNAME
6600         JSR STROUT
6610         .BYTE ' NOT FOUND',CR,LF,0
6620  PRTH59 JSR FILSEL
6630 ;
6640  PRTH6  LDA SLEN
6650         CMP #'D
6660         BEQ PRTH61
6670         JMP PRTSEN
6680  PRTH61 LDA STTK
6690         STA TRAKX
```

```
6700         JSR SWAP
6710         JSR SEEKX
6720         JSR SWAP
6730  PRTH7  LDA #ACK
6740         JSR XMIT
6750 ;
6760  PRTCN  LDY #$00
6770         STY EFFLAG
6780         STY CHKS
6790         STY PRTSAV+1
6800  PRTCN1 JSR PRTXIN
6810         CMP #SOH
6820         BEQ PRTCN2
6830  LABRT  LDA #NAK
6840         JSR XMIT
6850         BNE PRTCN
6860  PRTCN2 JSR PRTXIN
6870         JSR PRTCHK
6880         STA NRN
6890         CMP #'0
6900         BNE PRTCN3
6910         LDA #'9+1
6920  PRTCN3 SEC
6930         SBC ORN
6940         CMP #$01
6950         BNE LABRT
6960  ;
6970         LDA #$3B
6980         STA PRTSAV+2
6990  PRTCN4 JSR PRTXIN
7000         CMP #ETX
7010         BEQ PRTPEN
7020         CMP #DLE
7030         BNE PRTCN6
7040  PRTCN5 JSR PRTXIN
7050         SBC #$40
7060         JMP PRTCN8
7070  PRTCN6 CMP #EOT
7080         BNE PRTCN8
7090         INC EFFLAG
7100         BNE PRTCN9
7110  PRTCN8 PHA
7120         JSR CNSLOU
7130         PLA
7140  PRTSAV STA $FFFF
7150         INC PRTSAV+1
7160         BNE PRTCN9
7170         INC PRTSAV+2
7180  PRTCN9 JSR PRTCHK
7190         JMP PRTCN4
7200  ;
7210  PRTPEN JSR PRTXIN
7220         CMP #DLE
7230         BNE PRTPE1
7240         JSR PRTXIN
7250         SBC #$40
7260  PRTPE1 CMP CHKS
7270         BNE LABRT
7280         LDA #$00
7290         STA PRTXF1+1
7300         LDA #$3B
7310         STA PRTXF1+2
7320  PRTXFR LDY INDEX
7330  PRTXF0 LDA PRTXF1+1
7340         CMP PRTSAV+1
7350         BNE PRTXF1
```

```
7360         LDA PRTXF1+2
7370         CMP PRTSAV+2
7380         BEQ PRTXF3
7390  PRTXF1 LDA $FFFF
7400         STA (SAVADR),Y
7410         INY
7420         BNE PRTXF2
7430         INC SAVADR+1
7440         LDA SAVADR+1
7450         CMP BFENPG
7460         BEQ PRTNXT
7470  PRTXF2 INC PRTXF1+1
7480         BNE PRTXF0
7490         INC PRTXF1+2
7500         BNE PRTXF0
7510  PRTXF3 LDA #$FF
7520         STA (SAVADR),Y
7530         STY INDEX
7540         LDA EFFLAG
7550         BNE PRTDUN
7560         LDA NRN
7570         STA ORN
7580         LDA #ACK
7590         JSR XMIT
7600         JMP PRTCN
7610 ;
7620  PRTDUN JSR SWAP
7630         JSR WRIT
7640         JSR SWAP
7650         LDA #ACK
7660         BNE PRTQT
7670  ;
7680  PRTNXT JSR SWAP
7690         JSR WRIT
7700         LDA TRAKX
7710         CMP ENDTK
7720         BEQ PRTERR
7730         INC TRAKX
7740         JSR SEEKX
7750         JSR SWAP
7760         JSR SETPTR
7770         LDY #$00
7780         BEQ PRTXF2
7790  ;
7800  PRTERR JSR SWAP
7810         JSR CRLF
7820  PRTABT LDA #CTRLU
7830  PRTQT  JSR XMIT
7840         JMP PRTXX0
7850  ;
7860  PRTCHK PHA
7870         ASL CHKS
7880         ADC CHKS
7890         ADC #$00
7900         STA CHKS
7910         PLA
7920         RTS
7930  ;
7940  PRTXIN JSR XIN
7950         BCC PRTX1
7960         RTS
7970  PRTX1  JSR CNSLIN
7980         BEQ PRTXIN
7990         CMP #CTRLC
8000         BNE PRTXIN
8010         PLA
```

```
8020        PLA
8030        JMP PRTABT
8040 ;
8050 BUMPRN LDA ORN
8060        CLC
8070        ADC #$01
8080        CMP #'9+1
8090        BNE BUMPR1
8100        LDA #'0
8110 BUMPR1 STA ORN
8120        RTS
8130 ;
8140 PRTMSK CMP #SP
8150        BCS PRTMS1
8160        PHA
8170        LDA #DLE
8180        JSR XMIT
8190        PLA
8200        CLC
8210        ADC #$40
8220 PRTMS1 JMP XMIT
8230 ;
8240 PRTSEN LDA STTK
8250        STA TRAKX
8260        JSR SETPTR
8270        JSR SWAP
8280        JSR SEEKX
8290        JSR REED
8300        JSR SWAP
8310        LDA #$00
8320        STA COUNT
8330        JSR PRTRPG
8340        JSR BUMPRN
8350        LDA #ACK
8360        JSR XMIT
8370 PRTS41 JSR PRTXIN
8380        CMP #ACK
8390        BEQ PRTS5
8400        JSR CNSLOU
8410        BNE PRTS41
8420 ;
8430 PRTS5  LDY #$00
8440        STY CHKS
8450        STY EFFLAG
8460        LDA #SOH
8470        JSR XMIT
8480        LDA ORN
8490        JSR XMIT
8500        JSR PRTCHK
8510 PRTS6  LDA DIRBUF;Y
8520        CMP #$FF
8530        BNE PRTS8
8540        LDX KEYNUM
8550        CPX #'A
8560        BEQ PRTS10
8570 PRTS8  JSR PRTCHK
8580        PHA
8590        JSR PRTMSK
8600        PLA
8610        JSR CNSLOU
8620        INY
8630        BNE PRTS6
8640        BEQ PRTS11
8650 PRTS10 INC EFFLAG
8660 PRTS11 LDA #ETX
8670        JSR XMIT
8680        LDA CHKS
8690        JSR PRTMSK
8700 PRTS12 JSR PRTXIN
8710        CMP #ACK
8720        BEQ PRTS14
8730        CMP #NAK
8740        BEQ PRTS15
8750        JSR CNSLOU
8760        JMP PRTS12
8770 PRTS14 JSR BUMPRN
8780        JSR PRTRPG
8790        LDA EFFLAG
8800        BNE PRTS16
8810 PRTS15 JMP PRTS5
8820 ;
8830 PRTS16 LDA #$00
8840        STA CHKS
8850        LDA #SOH
8860        JSR XMIT
8870        LDA ORN
8880        JSR PRTCHK
8890        JSR XMIT
8900        LDA #EOT
8910        JSR PRTCHK
8920        JSR XMIT
8930        LDA #ETX
8940        JSR XMIT
8950        LDA CHKS
8960        JSR PRTMSK
8970        JSR PRTXIN
8980        CMP #ACK
8990        BNE PRTS16
9000 PRTSQ  JMP PRTXX0
9010 ;
9020 PRTRPG LDA COUNT
9030        CMP PAGES
9040        BNE PRTRP2
9050        LDA TRAKX
9060        CMP ENDTK
9070        BEQ PRTRP3
9080        INC TRAKX
9090        JSR SETPTR
9100        JSR SWAP
9110        JSR SEEKX
9120        JSR REED
9130        JSR SWAP
9140        LDA #$00
9150        STA COUNT
9160 PRTRP2 LDY #$00
9170        LDA (SAVADR),Y
9180        STA DIRBUF,Y
9190        INY
9200        BNE PRTRP2+2
9210        INC COUNT
9220        INC SAVADR+1
9230        RTS
9240 PRTRP3 INC EFFLAG
9250        RTS
9260 ;
9270        .END TRM
```

Term-Plus

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

Term-32

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

Term-65U

Patterned after Term-Plus, Term-65U is a smart terminal program for OS-65U (all versions) running in the single user mode. Allows capturing text to disk files. Term-65U will transmit text files, or BASIC programs as text. The program will also send WP-3 files as formatted text and can 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" 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

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 or numeric data or a formula which computes its results based on the contents of the other cells. 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).

PEEK (65)
The Unofficial OSI Users Journal

P.O. Box 586
Pacifica, CA 94044
415-993-6029

May 29

DELIVER TO:

GOODIES for OSI Users!

PEEK (65)
The Unofficial OSI Users Journal

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

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

POSTAGE MAY VARY FOR OVERSEAS

Name ______________________________
Street ______________________________
City ______________ State __________ Zip __________