PEEK[65]
The Unofficial OSI Journal

December, 1986
Volume 7, No. 12

Inside This Month:

For the two or three of you out there who haven't figured it out yet, I produce PEEK[65] on an Apple Macintosh computer. The reasons are many. This issue brings with it the culmination of my first year as owner/editor/chief cook and bottle-washer. My plan all along has been to bring the best tools to bear on this job. About 90% of them have now come into fruition.

Every issue prior to this one required that I make hard-copies of every article on a dot-matrix printer, and armed with all of that paper, I would literally cut and paste the articles onto a huge guide sheet that held two pages of PEEK. But in order to do this, I first had to measure each article, diagram, and advertisement and draw mock-ups of each page before I went to the light table to make the real masters. Printing out miles of articles and pasting snippets of paper onto these big guide sheets was extremely time consuming. It was the best method for producing the magazine I had until now.

This month my printer got an Apple LaserWriter Plus. That accounts for the improvements in the lettering you see here. In addition, I got a copy of some page-composition software for the Mac. Combined, these two tools reduced the time it took to produce this issue by at least 40%. Next month, it will surely be even more helpful since I've now worked with these things long enough to know what works and what doesn't.

OK, so why then did it take 2 months to get this issue out again? Mostly it was my fault. I just got out and out swamped by end of the year details. But since it was my fault, I made this issue larger (and intend to continue to do so) and didn't make it another of my imfamous "double issues". I hope you'll accept my apologies. You will see another issue of PEEK within 30 days. That issue will be a double issue - January and February, but you should never see another now that the production headaches have been so largely reduced.

Of course, I still desparately need your help. PEEK[65] is a users' journal. I depend on you to continue to supply me with articles and programs to publish. If you check out the 1986 Index in this issue, you'll see that only a handful of people have really been consistent contributors. I have yet to see an OSI owner who hasn't written at least one program on his own. If every one of you would send in a program with just a short description of it, PEEK[65] would double in length overnight. It doesn't need to be a new program. It doesn't need to be a super-spectacular program. Chances are that you have solved a problem that has been stumping someone else for ages, even if the original intent of your program doesn't match that of the reader's, it can still be a Godsend. So send 'em in boys and girls. We all depend on each other here.

In other news, Paul Chidley and David Livesay are both **very** close to being able to announce their 65816 CPU boards. As soon as the details and prices are fixed, I'll be announcing them here. I am confident that will happen next month. DBI is producing their board now for the higher-end users and I hear that their doing well. DBI has been hard at work on the software side for their implimentation. Only time will tell if their efforts will benefit video system owners as well, but I am hopeful.

I had a nice conversation with the Canadian firm of Becterm. They have been associated with the OSI world for a very long time, although I suspect most U.S. users and dealers aren't aware of them. The article on page seven of this issue discusses some of their systems and what they've been doing with them. You dealers out there who need a multi-user system with more capacity should certainly check out Becterm.

On the home front, we have a lot of great stuff this month. Former editor Eddie Gieske shows some of the pitfalls of using serial devices to enter data into your OSI. George Jennings graciously donated a technique that solve a problem that I know a lot of dealers have been struggling with - how to get a client's system to reboot via modem. Hardware fanatics will enjoy Dave Livesay's article on adding high-density drives to your OSI system. Dale King discusses the nuts and bolts of mortgages and annuities, providing some very helpful software along the way. John Horemans of TOSIE demonstrates how he improved the BASIC additions that come with Generic Computer Products' Color+ board. Finally, I begin a two-part article on a full-screen disk file editor for OS-65U.

Finally, don't forget that we are still alive and well on CompuServe. CompuServe gives you instant access to the OSI community around the world. If you got lost in the shuffle, you'll find us by entering "GO CLUB" at any "!" prompt. We populate section 8 of the Computer Club Forum.

Input Control and the 6850

by Eddie Gieske

One upon a time, "need" became the mother of, well, maybe not "invention", but some thinking and help along the way made the darned thing work.

Not too long ago, an outfit called simply "MSI" from somewhere in Georgia came up with this neat little hand-held computer that looks more like a walkie-talkie. You may have seen one in the store in the hands of the clerk taking inventory. He takes the wand and rubs it over the bar code on the shelf label to pick up the item number and then keys in the number of items on the shelf of that type. If programmed well, as he wanders the aisles, he can punch in some commands to give him all kinds of totals, values, or just step through his entries.

When the job is completer, he goes to the back room where he "dumps" the MSI into the store's computer. After a good dump, wipe the memory clean and it is ready for another trip to the shelves.

In my situation, up to 15 or 20K of data would be dumped at a time and the MSI doesn't support XON/XOFF flow control. So much for storing a track at a time. SO here sat an OSI 230E ready and willing, but would it be able?

The say that an OSI can be made to work with almost anything. Now I believe, but at the time, I wasn't too sure at all. I yanked out the CA-10 to rig up a port for 4800 baud while contemplating how I would get all those 15K of data into memory, if not on disk. Past efforts at dumping WP print-out from one OSI into another via modem had all failed, apparently due to the length of time for the receiving machine to execute the carriage return - line feed (it always lost the first character or two from the next line).

To test the new port, I hooked up a printer to it, just to see if I could send to it. I POKEd 11686,129 to set the output to the CA-10 and the serial console simultaneously, and POKEd 19798 with 6 to get port 4 on the CA-10 board (0, 2, 4, 6). It worked as expected, but that didn't solve my problem.

If I can POKE the output where I want it, I wonder if the same thing can be done with input? PEEK[65] to the rescue! Back to Roger Clegg's OS-U PEEK and POKE list. Yup! There is a location 11668 that handles input much like 11686 does for output. So, after the 19798 port selection, I POKEd 11668 with 128 (Editor's Note: the INFLAG at 11668 is checked by the operating system for the lowest bit number set. Once a bit has been found for this function, higher bit numbers that may also be set are ignored. The upshot of this is that you can only choose one port for input at a time). Tickle the input leads and.... nothing.

What on earth could be wrong? It is just like the book said. A quick call to Mike Sokol. "No wonder. You have to initialize the ACIA and establish the protocol."

One of these days, someone will hopefully do the 6850 ACIA a good turn and let us all in on its bag of tricks, but for the moment, let's approach this from the lay-hardwareman's point of view.

If you don't have the data sheet on the 6850, you had better get a copy from Motorola. A documentation sheet came with every CA-10-X board I ever saw. After wading through all the technicalities, one comes to the last two pages that tells you how to set it up.

Before trying to tell it what you want it to do, you must first initialize the 6850. It talks binary and has an eight bit word, so get out your calculator and be prepared to convert base 2 into base 10 or 16 and back again, or dig back in PEEK[65] to find the program that will do it for you.

OK, initialize! First, where does it live? Well, that depends upon your machine and where it thinks the CA-10 is located in memory. In most cases it will be at $CF00 or $CE00. Since two locations are used for each port (the first being the Control Register where we tell it what to do, and the second is the Data Register that actually passes the data in and out) the first port might be at $CF00 and $CF01, the second at $CF02 and $CF03, etc. So, pick your port and then convert these hexadecimal values of the memory addresses to decimal so that we can POKE it. In my case, the Control Register for port 4 converts to 52998.

Initialize at last! The 6850 manual says to put ones in CR0 and CR1 (Editor's Note: that's bit zero and one to us mere mortals) of the Control Register to reset the chip. Some quick calculations will tell you that this value is 3, so I POKE 52998 with 3.

How to behave? In the manual, there are several tables listing various values for CR0 through CR7 (the eight bits of the byte you store in the Control Register to configure the 6850). After consulting the manual for the MSI, modem, or whatever you will hang on the port, compare it's requirements for baud rate, word length (or number of data bits), parity, and number of stop bits with those in the 6850 manual to determine the value to POKE to the Control Register after you have reset it. Remember that each of the eight bits has a meaning and must be set properly.

In addition to controlling the reset function, bits CR0 and CR1 also set what the 6850 calls the "divide rate". This divide rate determines the baud rate, or the speed at which you want to communicate. You see, the speed at which the 6850 operates is determined by two factors: (1) the speed of the clock signal coming into it from your computer, and (2) the divide rate. The clock signal is divided by the divide rate in actual operation. The 6850 has three possible settings for this divide rate - 1, 16, and 64. Let's assume the clock signal coming into the 6850 is 19,200 cycles per second. If the divide rate on the 6850 is set to 1, the effective baud rate you will communicate at will be 19,200 baud. If the divide rate is set for 16, you will be set for 1200 baud. And if the divide rate is 64, you'll get 300 baud. You see? 19,200/1=19,200. 19,200/16=1200. 19,200/64=300. All of the settings are shown in Table 1.

A setting of 7 data bits, even parity, and 1 stop bit is the most common setting and suited my needs. After selecting the rest of the settings I wanted, my configuration byte ended up as "10010001". In decimal, that's 145. Since the 6850 was already initialized, I then POKEd 52998 with 145. Shift the input from the keyboard to the CA-10 with POKE 11668,128 and select the port with POKE 19798,6. Hitch up the MSI and tell it to dump. Violla! the dump appears on the screen!

So you write a simple little program that then says to INPUT A, or INPUT A$, and go get the next one. The operating system takes care of putting it on the screen and into memory.

But let's get a little more practical. The shorter the program, the more room in memory will be left for variables. I just DIMed A$(500) and made a little loop.

```
10 INPUT A$(X): X=X+1: GOTO 10
```

That was just great, but the darned thing just died on me when it finished. I noticed that the MSI sent "/END" as the last characters, so I installed;

```
IF A$(X) = "/END" THEN POKE 11668,1
```

to restore input control to the console. Then I, or the program can save the stuff to a file.

That's it. Sweet and simple, and very useful. Now that you can handle the ACIA and control INPUT sources, just let your mind ramble.

How's this for starters? Dealers have gone to all sorts of expensive ends to have a modem on a customer's machine so they can access it from their office. Some have even installed extra memory partitions - just for the modem. Now and extra menu selection can do the POKEs to turn the console over to the modem and it can be POKEd back to the console when finished.

CR1	CR0	Effect
0	0	Divide Rate = 1
0	1	Divide Rate = 16
1	0	Divide Rate = 64
1	1	Master Reset

CR4	CR3	CR2	Effect
0	0	0	7 data + Even Parity + 2 Stop
0	0	1	7 data + Odd Parity + 2 Stop
0	1	0	7 data + Even Parity + 1 Stop
0	1	1	7 data + Odd Parity + 1 Stop
1	0	0	8 data + No Parity + 2 Stop
1	0	1	8 data + No Parity + 1 Stop
1	1	0	8 data + Even Parity + 1 Stop
1	1	1	8 data + Odd Parity + 1 Stop

CR6	CR5	Effect
0	0	RTS = low, Interrupt Disabled
0	1	RTS = low, Interrupt Enabled
1	0	RTS = hi, Interrupt Disabled
1	1	RTS = hi, Interrupt Enabled and Transmits a <BREAK> level on the Transmit Data Output

Table 1

ATTENTION: DEALERS!

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

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

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

Remote <BREAK> and Boot

by George Jennings
Capitol City Stationers
3649 Market Street
Camp Hill, PA 17011

The purpose of this article is to demonstrate a technique to allow ISOTRON dealers to provide remote programming support for clients who are located several miles from the dealership.

Essentially, it is a matter of going on-line via modem with the customer's system, saving travel time and expense for software fixes that would otherwise require a trip to the client site.

One of the first problems encountered is the fact that when a programmer is trying out a software fix and it doesn't work properly, the machine often hangs - going off into the woodwork, requiring a reboot of the system. It can be a nuisance having to place a second phone call to the customer (assuming he has a second phone line) to get him to push the reset button. Figure 1 is a simple little 4-component circuit which allows a remote programmer - working through the supervisory (console #0) port to reboot the system by remote control.

It is a simple comparator and timer which monitors the RS-232 voltage at the console input to the computer. This voltage (regardless of whether it originates at a local terminal or a modem) sits normally at somewhere between minus 3 and minus 12 volts. When a character is received, the voltage momentarily switches to plus 3 to plus 12 volts at the baud rate employed. The comparator looks for a plus voltage excursion lasting more than a few seconds and when it detects one, it pulls pin 7 on the NE-555 low. This pulls the main reset line on the processor low and provides the familiar "H/D/M?" boot message.

OK, how do we put a plus voltage on that input pin? Many CRT terminals have a <BREAK> key which does just this. The problem is that a lot of them only produce a short positive break pulse which isn't long enough to activate the timer. The timer has to have a fairly long time constant so as not to respond to the baud rate pulses normally used for modem work (300 to 1200 baud or so). Figure 2 is a simple button and battery circuit which can be built into a little box at the dealer end to provide a <BREAK> signal for those terminals that don't provide a sustained <BREAK> signal. The programmer pushes the little button, counts slowly to 10, and lets go. The terminal responds with "H/D/M?" from the remote computer, and he proceeds from there.

If only an occasional "on-line" session is needed, it's more convenient to plug and unplug modems, etc. at each end and get the operators to switch baud rates each time. If frequent on-line sessions are required, it's worth the trouble to gear up a little more conveniently.

Prior to delivery of the customer system, the break timer (Figure 1) and a small 3-pole double-throw toggle switch can be installed in the client's machine. The break circuit can be mounted directly on the 510 boards using the NE-555 foil pads provided for the optional 110 baud clock timer - see schematic. The 515 board has an uncommitted 16-pin pad layout which can be used on this processor board for the same modification. The toggle switch is installed on the back panel, near the console input DB-25s connector. In one position, normally down, the switch connects the console input cable to the CPU board directly to the console terminal jack. This requires 2 of the 3 switch contacts.

The third switch section is tied into the baud rate selector pads and sets the console baud rate to whatever is desired (usually 4800 or 9600). In the other position, the switch ties the CPU input and output to a long pig-tailed DB-25 connector and ties to the external modem. The third section selects the modem baud rate (usually 300 or 1200). This provides a no-hassle way for the client to switch over. He just flips the switch, drops the phone in the modem cuff, and is ready to go. At the end of the on-line session, he hangs up the phone, flips the toggle switch the other way, and is back in business normally (assuming the software "fix" worked). CAUTION!! The baud rate clock signals on both the 510 and 515 boards are at 16 times the actual baud rates listed! It is imperative that the baud rate clock circuits to and from the toggle switch be run with shielded leads with the shields grounded!! The rise times on the clocks are sufficiently fast as to interact if you don't shield them. This will cause the CPU boards to try to clock at some unpredictable baud rate - and it won't work!

Meanwhile, back at the dealer's site, Figures 3A and 3B (with a little help from Figure 2 as noted above) provide 4 convenient modes of operation.

(A) is the normal setup at the shop. Terminal talking to local computer.

(B) ties the local dealer terminal into the modem. You have to switch terminal baud rates on the terminal to agree with the what the client and modems are set up to handle. In this mode, the programmer can go on-line with the remote client machine.

(C) with some software diddling to make the character transmission rate compatible with BASIC's somewhat slow internal housekeeping, can be used to download files and programs directly from the dealer's computer to the remote client machine. The remote machine will echo what its getting back to the dealer terminal. This takes some attention to things like eliminating automatic linefeeds and other stuff which could be troublesome, and I won't get into all that. This, however, will supply a hardware means to do it, with the software details left up to programmers more clever than I am.

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

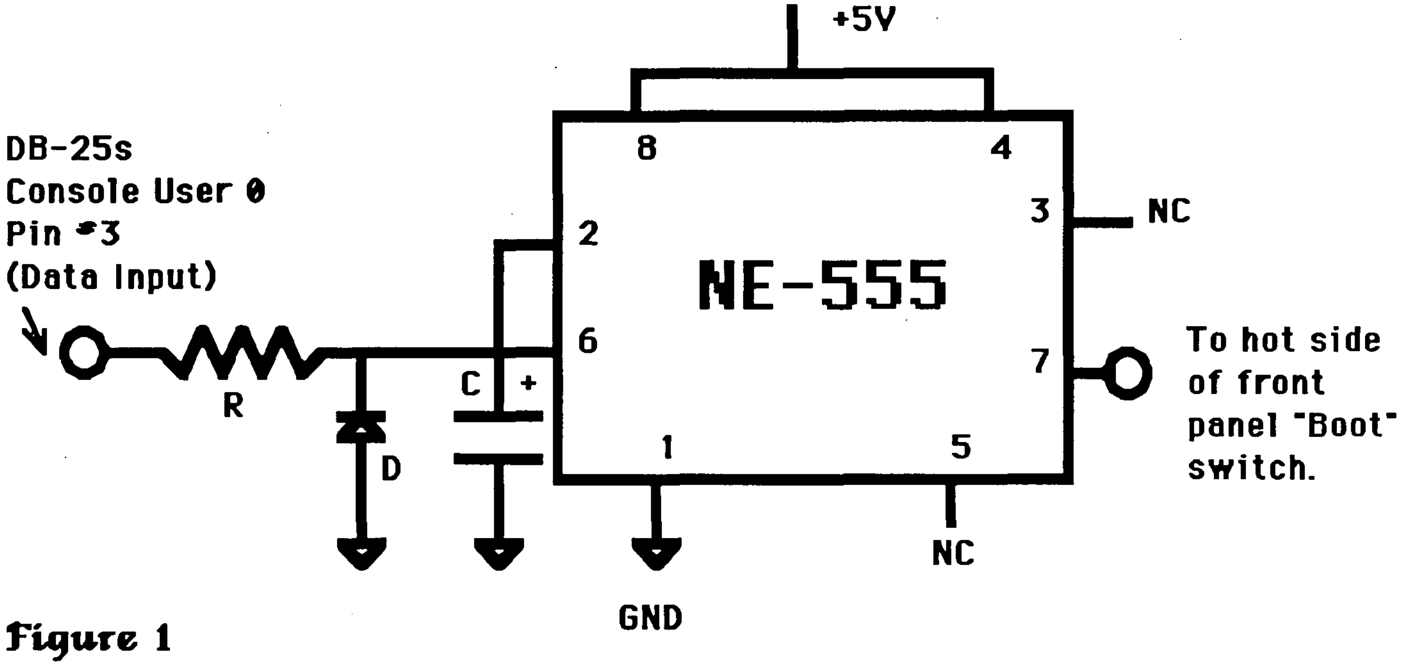


Figure 1

R = 100K 1/4W (sets time to reboot)

D = 1N914, etc.

C = 10 mfd, 15VDC, tantalum

(D) is useful in cases where the dealer or his programmer may want to access his shop computer from a client location (for look-up or demonstration purposes or whatever). The local dealer terminal monitors whatever is being sent out of the shop, so if somebody starts rooting around in proprietary files, the dealer operator can flip the switch and terminate the session. If's a security watchdog feature. Also handy to monitor the usage of dealer computer when leased or rented remotely to an outside user.

The stuff at the client end costs roughly $15.00 plus the cost of whatever modem is used. The dealer end stuff might run $50.00 or so, plus modem. This sort of setup can pay for itself in travel time, and extra "please reboot the system" phone calls in a fairly short time, depending on how busy the shop is and how far away the clients are located.

Another useful idea for multi-user systems at the client end; if ge is in time-share while the programmer is on-line through the console port, a little inter-terminal communications program can be used to POKE messages from the programmer to specific time-share user terminals for instructions, etc. In essence, the program inputs a message character string and then POKEs it one character at a time to the ACIA port for the particular user. Still another way to get the client's attention is to dump a message to his line printer: "Please insert the OS-65U System Disk. Thank-you." "Please pick up the telephone.", etc.

Add to Figure 3 if terminal used only provides momentary "break".

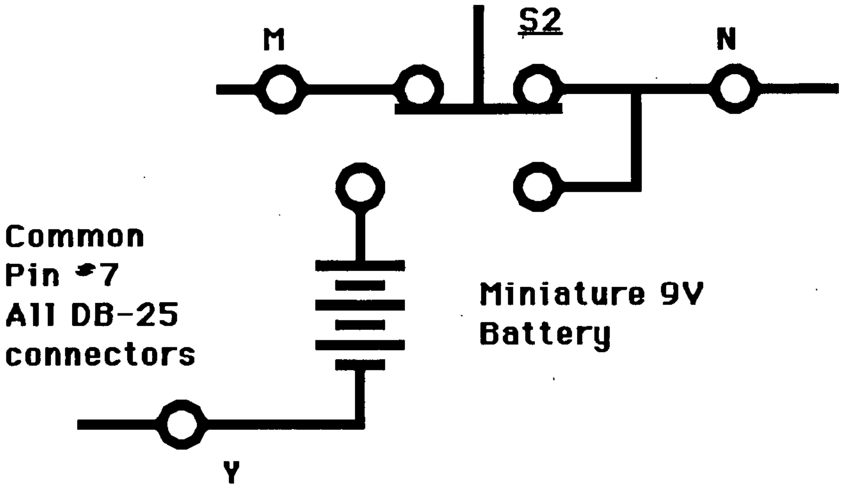


Figure 2

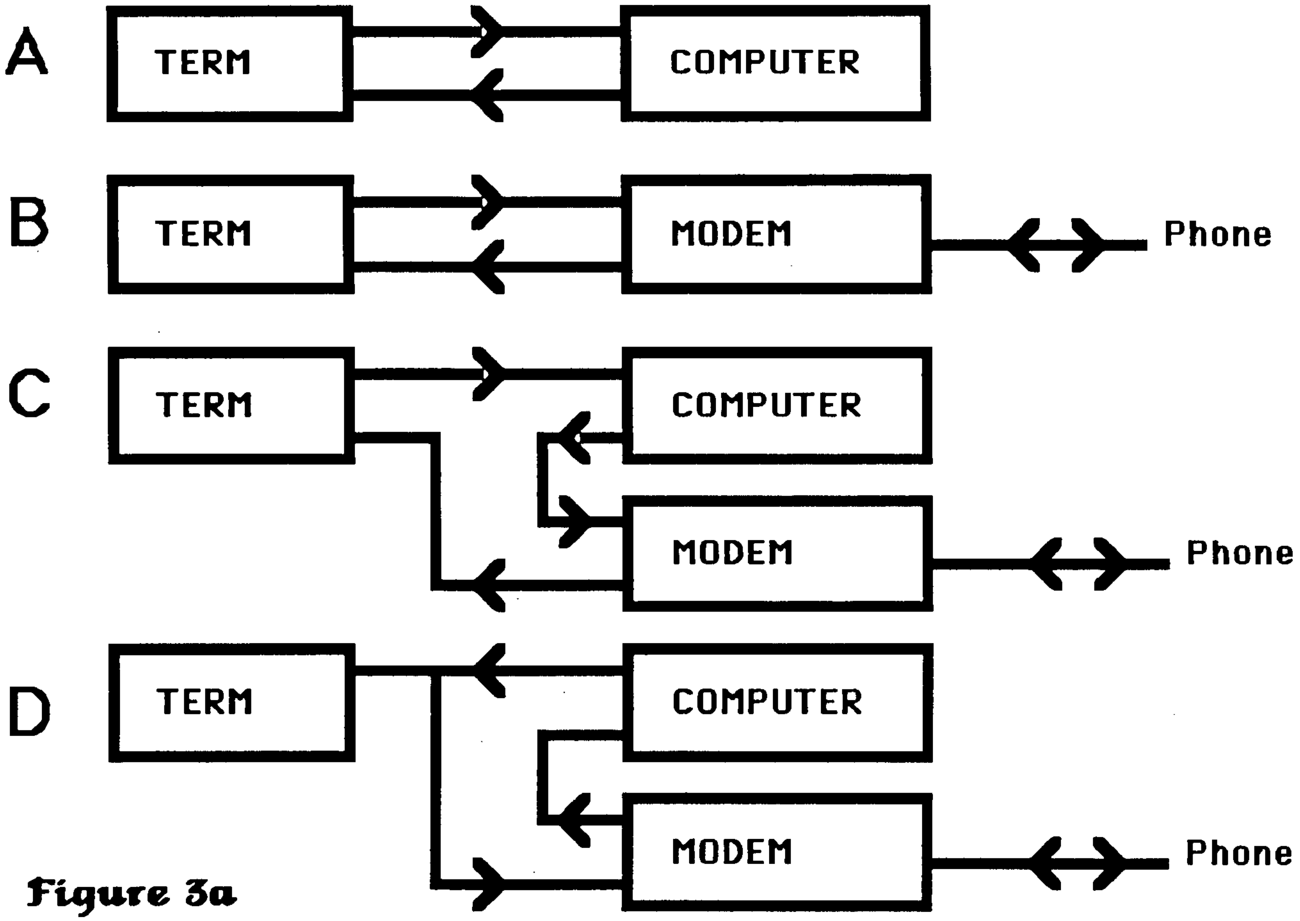


Figure 3a

Product Description: Becterm Multi-Micro(R) Systems

We all know that Denver Boards, Inc. makes OSI-compatible boards and systems, but did you know there was another company making boards that will run OS-65U? Neither did I until the company, Becterm of Quebec, Canada phoned to ask some questions. I recognized the name from the PEEK subscription list, but had always assumed they were a dealer or some other computer-related business.

I was astounded to hear of what they had been doing. They have a line of multi-user computers that use a proprietary operating system they call "BMOS". The BMOS environment allows several different operating systems to run on the system simultaneously. OK, I've heard of that sort of thing before, but this was the first time I'd heard of one that also supported differring microprocessors, attatched processors, and co-processors to run on a single system.

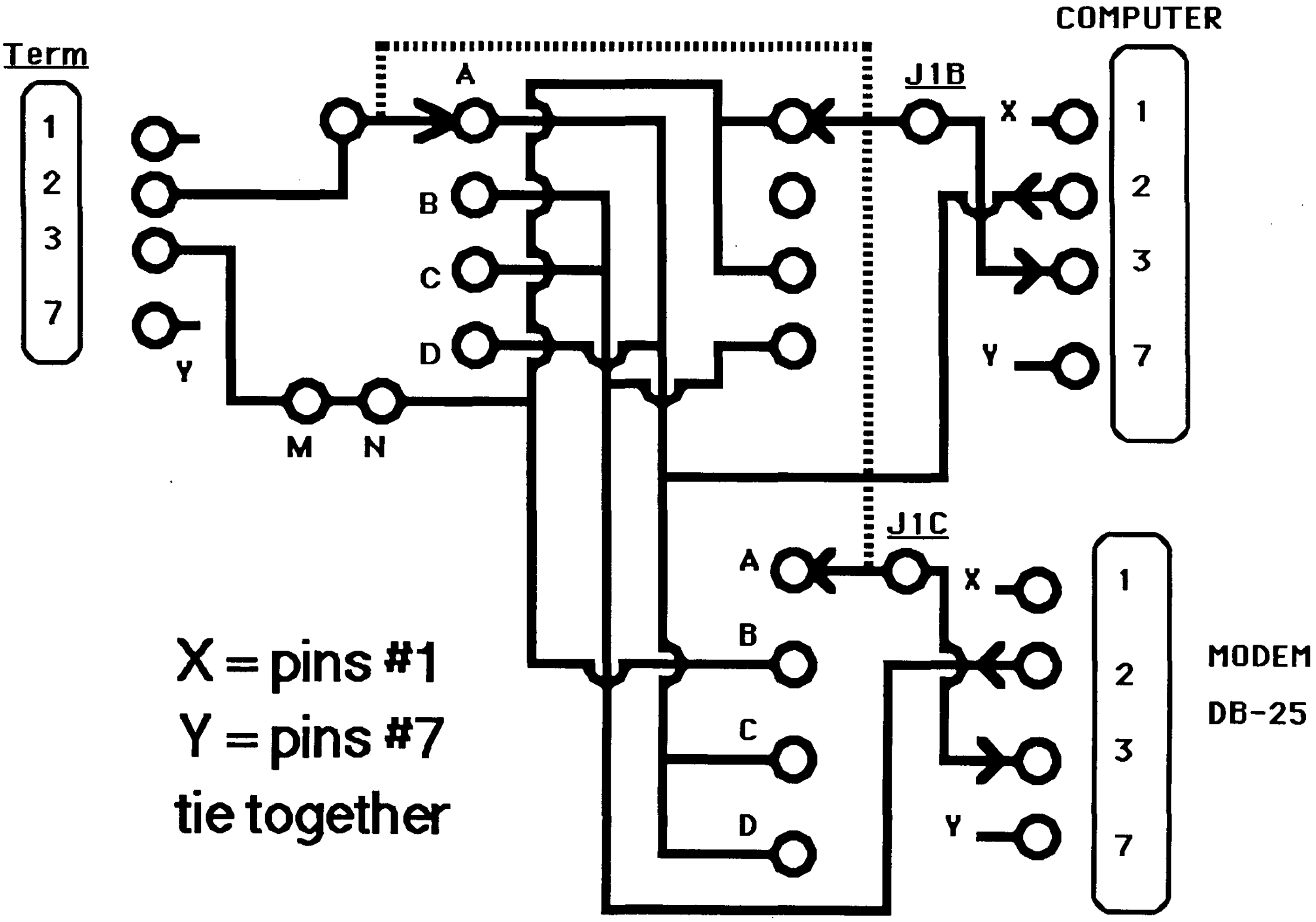
As you might expect from the above specifications, the Becterm systems give each user exclusive access to at least one processor. Their lowest entry level system, the model AZ-400, supports up to 20 users. At the high end, their model AZ-1400 supports up to a mix of 256 users, user processors, and peripheral processors.

Becterm supports a variety of operating systems, including OS-65U, UCSD, and IDRIS. On the hardware front, they support the 6502 and the 68000, and will apparently soon support the 8086 family.

In my conversation with Mr. Andre' Gareau, it was clear that Becterm had gone far beyond the traditional OSI multi-user and networked systems, with many features a lot of people have been begging for. How does 32 gigabytes of mass storage sound to you? Not impressed? How about 670 megabytes of RAM-disk storage?

For more information, contact:

Becterm
12, Trans-Canada Ouest
Levis, Quebec
Canada G6V 4Z2
(418)-835-1551



X = pins #1
Y = pins #7
tie together

Figure 3b

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

Mortgages, Discounted Mortgages, and the Annuity Equation

by Dale King
Box 419
Leonard, TX 75452

Did you ever wonder how the monthly payment is determined on a fixed-rate mortgage? Have you noticed the growing classified ads under "Mortgages For Sale"? How would we know how much to offer for these investment instruments in order to make them profitable to us? Can I use my OSI to analyze these instruments? Yes, of course.

Some terminology: A **note** is merely a **promise to pay** a sequence of payments in the future (possibly just one payment). In the case of real estate, a **mortgage** is a **legal agreement** that secures the note. In some states this is called a **deed of trust**. By an abuse of language we use these terms interchangably.

An **annuity** is merely a **sequence of equal payments**. They could be monthly, yearly, or daily, but they are equal. It is not hard to see that a payment to be received in the future is not worth as much in the presentt. For one could set aside the **present value** and let it earn interest until it has grown enough to be received at the future date.

Thus, every payment in an annuity has its own present value. Add all these present values up to get the present value, or **PV**, or the entire annuity. For example, the PV of a mortgage at day one is the original amount of the loan.

It takes a little algebra, but one can show that the equation in Figure 1 relates N, the number of payments; R, the amount of each payment; i, the interest rate per conversion period; an PV, the present value of the annuity.

The program ANUITP, shown in Listing 1, allows you to solve this equation given any three out of four of its variables. In the case of i, this is harder than you think. The equation is transcendental in i.

Your banker looks un the N, PV, R, and i in "Mortgage Interest Tables". These tables are widely available, but usually do not have the range that we seek. My father has an old CRC Math Tables. They handle i up to about 5%. Modern tables go up to roughly 20%. Why would anybody need tables higher than 20%? This question leads us to the subject of discounted Mortgages.

Suppose I am receiving monthly payments from a mortgage (I carried back a second when I sold my house). I might rather have the cash.

The value of my mortgage may be $10,000.00, but I might be willing to sell my mortgage to somebody else for say $5,000.00. You can be sure that such a buyer of my mortgage is going to earn a lot more interest than I am. How much more? 50% is not unheard of.

It happens every day. From the buyer's perspective, he is buying an annuity of N payments of R dollars (or pounds Sterling or francs) and he is paying a PV of $5,000.00. By selecting option 4 in ANUITP, he can determine his percentage of yield.

If you find any of this interesting, then let Rick Trethewey know (send a note to PEEK[65]) and we can continue this discussion and provide other programs and examples. I haven't mentioned the **AMOUNT** of an annuity, which is another powerful concept. Send me a SASE and I will send you a brief bibliography on the subject at no charge.

$$PV = R * \left[\frac{1-(1+i)^{-N}}{i} \right]$$

Figure 1

```
1 REM***************** ANNUITP ************************
2 REM
3 REM          SOLVES THE ANNUITY EQUATION
4 REM i.e. SOLVES FOR ANY ONE OF N, PV, R, AND I GIVEN ANY THREE
5 REM   WHERE N = NUMBER OF PERIODS
6 REM         R = PAYMENT PER PERIOD    ( e.  g. monthly payment)
7 REM        PV = PRESENT VALUE OF THIS INCOME STREAM
8 REM         I = INTEREST RATE PER CONVERSION PERIOD
9 REM            by Dale King, PO BOX 419, LEONARD TX, 75452
10 REM**************************************************
100 CL$ = CHR$(10): FOR I = 1 TO 5: CL$ = CL$+CL$: NEXT I
110 PRINT CL$
120 PRINT "**********************************"
130 PRINT "     SOLVE THE ANNUITY EQUATION"
140 PRINT
150 PRINT "(1) FOR PV GIVEN N,     R, AND i"
160 PRINT "(2) FOR N  GIVEN    PV, R, AND i"
170 PRINT "(3) FOR R  GIVEN N, PV,   AND i"
180 PRINT "(4) FOR i  GIVEN N, PV, R"
190 PRINT
200 INPUT "YOUR SELECTION"; Q$: Q = VAL(Q$)
205 IF Q$ = "X" THEN STOP
210 IF Q<1 OR Q>4 OR Q<>INT(Q) THEN 200
215 INPUT "ENTER CR FOR MONTHLY CONVERSION AN ANNUAL i"; DE$
216 DE = LEN(DE$)
220 ON Q GOTO 300, 400, 500, 600
297 REM
298 REM
299 REM*************************************
300 REM SOLVE FOR PV
301 REM
302 REM
310 GOSUB 700: GOSUB 745: GOSUB 760: REM GET N, R, i
320 PV = R * ((1 - (1+I) ^ (-N))) / I
330 GOSUB 785: REM PRINT SOLUTION
350 GOTO 120
397 REM
398 REM
399 REM************************************
400 REM SOLVE FOR N
401 REM
410 GOSUB 720: GOSUB 745: GOSUB 760: REM GET PV, R, i
420 N = -(LOG(1 - I * PV/R)) / LOG(1+I)
430 GOSUB 785
450 GOTO 120
497 REM
498 REM
499 REM************************************
500 REM SOLVE FOR R
501 REM
505 GOSUB 700: GOSUB 725: GOSUB 760: REM GET N, PV, i
510 R = PV / ((1 - (1+I) ^ -N) /I)
520 GOSUB 785
530 GOTO 120
597 REM
598 REM
```

```
599 REM************************************
600 REM SOLVE FOR i (note we must use a numerical method here)
601 REM
605 GOSUB 700: GOSUB 725: GOSUB 745: REM GET N, PV, R
610 DEF FNA(X) = PV - R * ((1 - (1+X) ^ -N) /X)
620 GOSUB 1000: REM FIND THE ZERO OF THE FUNCTION A(X)
625 PRINT
630 I=X: GOSUB 785
650 GOTO 120
697 REM
698 REM
699 REM********** SUBROUTINES *************
700 REM N
701 REM
705 INPUT "N - the number of periods"; N
710 RETURN
720 REM PV
725 INPUT "PV - the present value of the annuity"; PV
730 RETURN
740 REM R
745 INPUT "R - the amount of the periodic payment"; R
750 RETURN
760 REM i
765 IF DE THEN 775
770 INPUT "i - the annual %interest rate"; I: I = I/1200: RETURN
775 INPUT "i - the %interest per conversion period"; I: I=I/100:RETURN
785 PRINT CL$
786 PRINT " N", " PV", " R", " %IPP", " %ANNU"
790 PRINT N, PV, R, I*100, I*1200
792 PRINT: PRINT: PRINT
795 RETURN
997 REM
998 REM
999 REM*******************************************
1000 REM THIS SUBROUTINE SOLVES A(X)=0 FOR X
1005 REM      USING THE BISECTION METHOD
1006 REM
1007 REM
1010 A = .0001: B = 1: REM WE ASSUME THAT A < X < B IN THIS METHOD
1020 IF SGN(FNA(A) * FNA(B)) > 0 THEN PRINT "ERROR": RETURN
1022 IF ABS(A-B) < 10^-4 THEN X = (A+B)/2 : RETURN
1023 PRINT ". ";
1025 MIDPT = (A+B)/2
1030 CHECK = FNA(MIDPT) * FNA(A)
1040 IF SGN(CHECK) < 0 THEN B = MIDPT: GOTO 1020
1050 A = MIDPT: GOTO 1020
```

Listing 1

Using High Density 5.25" Disk Drives to Replace 8" Drives

by David Livesay
ave de la Resistance 6
B4920 Emourg, Belgium

How many of you have wished that you could silence your 8" drives? If not, how many of you have family that wish that you would turn off your computer to silence the 8" drives? Okay, so you like the noise. How would you like to increase your storage capacity?

If anything in the above paragraph strikes a chord, then read on. I will explain how you can replace your noisy 8" (most likely single-sided) drives with quiet high-density double-sided 5.25" drives. Today, two 5.25" double-sided drives with power supply and cabinet will cost less than one single-sided 8" drive with cabinet and power supply cost 6 years ago.

This article is a continuation of the article which appeared in the September issue of PEEK[65] and you will need to refer to that article for some of the information required to install the high-density drives.

For several years now, high-density 1.2 megabyte (when formatted in double-density format, but only about 500K in standard OSI format) 5.25" disk drives have been on the market which can be used to replace the 8" drives. Although the drive connector is different that the 8" drives, the signals are compatible. Table 1 shows the pin-out of the high-density drives. These drives spin at the same speed and have the same data transfer rate as the 8" drives. The only differences are that all of these drives are double-sided, have 80 tracks per side, and don't have built-in data separators. If you make an adaptor to connect these to your computer and provide a data separator, the computer won't know the difference between these drives and the 8" units.

Data Separator and Motor C.ntrol

You will need to either build or buy the data separator/motor control circuit described in the September issue of PEEK[65]. There are a couple of changes that you will need to make to the data separator described in that article. In place of the 470*pf* capacitor connected to U2, you should connect a 220*pf* capacitor. The 10K trim pot should be adjusted for a 2.75 to 3.0 microsecond positive pulse at pin 6 of U2. Most of the high-density drives also have a ready signal at pin 34, so this line can be connected to the OSI controller pins 20 and 24.

The high-density drives usually have provision for speed select. This allows you to use the high-density drive to read and write normal 80 track 5.25 formats. For high-density mode, the drive turns at 360 RPM (just like the 8" drives) and to read and write normal 5.25 disks, the drive turns at 300 RPM. If you have more than one OSI system and one is a mini-floopy, you could build two data separators, install a connector on the back of each computer for the disk drives, and then install a switch on the disk drive for selecting the speed. The switch should be connected between pin 2 and ground. Normally, grounding this pin will switch the drive to low speed mode. You should consult the disk drive manual. You can have automatic speed selection by connecting pin 2 of the drive cable connector to ground on the data separator used with the mini-floppy system. The data separator used with the 8" system should have pin 2 left open. Now when you plug the drives into the mini-floppy system, they will spin at 300 RPM and when plugged into the 8" system they will spin at 360 RPM.

Modification of OSI Controllers for use with Double-Sided Drives

You will need to modify your OSI drive controller for use with double-sided drives. The required modifications and described in the September PEEK[65] article. For those who have a different OSI controller than the one I described and can't figure out what to do to make the changes, write to me and I will give you instructions.

Replacing 8" Drives with High-Density 5.25 Drives

After you have the data separator built, you will need to make an adaptor to be able to connect one 8" and one 5.25" drive at the same time.

Figure 4 shows the connections required to connect an 8" drive to the high-density 5.25" drive cable while transferring data from the 8" disks to the 5.25" disks. You should set up the 8" drive as drive number one and the 5.25" drive as number two. The easiest way to connect the two drives is to connect a small prototype board with a 34 conductor edge connector into one of the drive connectors on the 5.25" drive cable. Find a prototype board with an edge connector with 2X17 connections on it spaced at .100" between conductors. Radio Shack sells some cards with 2X20 connections. You can modify this by cutting part of the edge with a hacksaw. Another choice would be a prototype board for an Apple. The Apple-compatible boards have 2X25 connections on it and will also have to be modified. To this board you will need to add a 50-pin female header for a ribbon cable to the 8" disk drive. You will then need to make the connections shown in Figure 4 between the 34-pin connector and the 50-pin connector. If there is enough interest, I will make a small adaptor board with the connector on it. You will then need to make up the cable for your 5.25" disk drives with connectors for the two drives, even if you only have one 5.25" drive, and a new cable to the 8" drives.

In order to control the head loading on the 8" drive, you will need to either run a wire from pin 1 on the OSI controller to pin 18 of the 50-pin cable going to the 8" drives, or you can use pin 4 of the 34-pin cable for controlling the head load. If you use pin 4 of the 34-pin cable, run a wire from pin 1 of the OSI controller connector on the data separator board to pin 4 of the connector for the disk drive cable. You will then need to make the connection from pin 4 of the 34-pin connector to pin 18 of the 50-pin connector on your adaptor for the 8" drives.

Now that you have all of the required hardware, install the data separator, connect the 5.25" disk drive cable to the disk drive - set up as drive number two - connect your adaptor board to the 5.25" drive cable, and connect your new 50 conductor cable to the 8" drive - set up as drive number one. Remember that some manufacturers of 5.25" drives number their drive select lines as 0-3 and others as 1-4. In either case, when I refer to drive number one, I mean set the drive so that it is selected by pin 10, and so that drive two is selected by pin 12.

Now with all of the connections made, you can boot your system with the 8" drive. Note that you are now using the data separator connected to the OSI controller and not the data separator built into your 8" drive.

At this point it should be mentioned that the disks that you use for the high-density 5.25" disk drives should be identified as being suitable for use with the IBM PC-AT. Don't try to use normal double-density disks. If you also use these drives as normal 80-track drives (in low speed mode) for a mini-floppy system, you should use normal double-density disks when using the drive with the mini-floppy system.

You should now POKE in the changes required for OS-65D to use 80-track drives (see section below), select drive "B" and try to initialize the disk by entering;

DISK!"INIT"

If all went well, the disk drive will initialize side one of the disk. You can now try reading and writing to the disk to make sure that everything is working properly. Don't forget that for the second drive, side one is device "B" and side two is device "D". You should now copy all of your 8" disks to the 5.25" disks. You can then disconnect the 8" drive and set up your 5.25" drive as drive number one. You should now be able to boot from the 5.25" drive.

Changing OS-65D for 80-Track Drives

In order to use 80-track drives in place of the 77-track 8" drives, you will need to make some changes to OS-65D and some of the utility programs.

There are three memory locations in OS-65D which need to be changed. There are two ways that we can do this. The first one is to POKE the correct values into memory by adding appropriate commands to your BEXEC* programs on all of your disks. The second way is to make permanent changes to the operating system on your disks.

The following memory locations are the ones to change. Values within parenthesis are the decimal equivilents of the hexadecimal values preceding them.

ADDRESS	OLD VALUE	NEW VALUE
$26CA	$77(119)	$80(128)
$2769	$76(118)	$79(121)
$2779	$76(118)	$79(121)

To make the changes to OS-65D, enter "EXIT" at the "OK" prompt in BASIC and load the Track Zero Read/Write Utility from track one, sector 2 into memory at $0200. You execute this program by entering "GO 0200". Follow the instructions to read track zero into memory at $6200. Load the Extended Monitor and change the three memory locations shown above. Remember to add $4000 to the memory addresses to reflect where we put the track zero contents (ie. instead of $26CA, you would enter $66CA). Run the Track Zero Read/Write Utility again and save the new version to disk. Remember that we will read and write 12 pages of data each time. At this point you will have a disk that will boot and be able to use all 80 tracks.

5.25" HIGH-DENSITY DISK DRIVE INTERFACE

PIN #	SIGNAL TYPE	FUNCTION
2	INPUT	SPEED SELECT
4	INPUT	IN USE or HEAD LOAD
6	INPUT	DRIVE 4 SELECT
8	OUTPUT	INDEX
10	INPUT	DRIVE SELECT 1
12	INPUT	DRIVE SELECT 2
14	INPUT	DRIVE SELECT 3
16	INPUT	MOTOR
18	INPUT	DIRECTION SELECT
20	INPUT	STEP
22	INPUT	WRITE DATA
24	INPUT	WRITE GATE
26	OUTPUT	TRACK 00
28	OUTPUT	READ DATA
30	INPUT	SIDE SELECT
32	OUTPUT	READY

- NOTE - ALL ODD PINS ARE GROUND

Table 1

You will also need to change the program CREATE to be able to use 80 tracks. The instructions for doing so are in the September article. In this case, changing line 20090 as stated will allow the use of tracks 0-7 as well as 8-80.

Using 8" and 5.25" Drives at the Same Time

For those who wish to have the possibilty of quickly using the 8" drives, you might wish to make up the 5.25" drive cables with three disk drive connectors on it. Two of these connectors would be used for your 5.25" drives and the third would be used to connect to your adaptor for the 8" drive. Although they may be a little bit difficult to find these days, you should be able to locate a cabinet suitable for a single 8" drive. You will then need to install a switch on the case which connects to the drive select jumper on the drive. You will also need to connect a switch to the drive select jumpers on one of the 5.25" drives. Both of the drives should be set up as drive number two. You should mark the position of each switch to indicate which is selected. If you accidentally leave both selected, they will not work.

Conclusion

At this time, you can purchase the high-density drives for about $150.00 each. These will most likely drop to about $125.00 each in the next 6 months. Remember these drives are usually identified as 1.2 megabyte drives for the IBM PC-AT. DO NOT get confused and purchase a 360K drive for the IBM PC-AT. Now you can enjoy the quietness and increased capacity of your new drives.

50 PIN 8" TO 34-PIN INTERFACE CONNECTIONS

8" DISK CONNECTOR	5.25" DISK CONNECTOR	
14	SIDE SELECT	32
18	HEAD LOAD	4*
20	INDEX	8
22	READY	34*
26	DS0	10
28	DS1	12
30	DS2	14
32	DS3	6
34	DIRECTION	18
36	STEP	20
38	WRITE DATA	22
40	WRITE GATE	35
42	TRACK 0	26
44	WRITE PROTECT	28
46	READ DATA	30

* Optional on some 5.25" drives

COLOR+ Additions

by John Horemans
TOSIE
Box 29
Streetsville, Ontario
Canada

Finally I have played with the Color+ board long enough that I feel confident enough to write a few words. Changes to the keywords, new words, and relocating the Color+ above BASIC are my main achievements. This has been aided greatly by Bob Ankeney of Generic Computer Products passing on to TOSIE the source code, and allowing distribution as long as we retained the header "Not to be used for profit".

My first efforts were to install words to operate the Commodore SID chip. This, as most will recognize, is the sound chip installed in the Commodore 64. Data is easy to get. For the circuit diagram, borrow a copy of the 64 programmer's guide. The circuit is at the back of the book. My 6581 SID is connected as shown in that diagram. I added a decoder for $C4xx and a DD line. Note that the original Commodore 64 location was $D400. This is a possibility for Supoerboard owners. I chose $C4xx as this area was free. When I started and was programming it with code copied from magazines, all I had to do was subtract 4096 from the decimal addresses. Now with the new code, I don't even need to remember where the chip is in memory. The commands added to the Color+ are as shown in Table 1 (PS - you could add them to BASIC+).

So far, that's it for new commands, but the syntax of several others was changed. I did not enjoy typing SPRITEPATTERN and other 12 character keywords. They were shortened to save typing and space. The parameters have not changed; use them as before.

OLD	NEW
SPRITEMOVE	- SMOV
HCOL	- HCOL
COLOR	- COL
SPRITESELECT	- SSEL
SPRITECOLOR	- SCOL
SPRITEPATTERN	- SPAT
SPRITEINIT	- SPINIT
SPRITESIZE	- SSIZ
TCOLOR	- TCOL
SCREEN	- SCR

All this saves typing, decoding at run time, memory space, space in the table, and has been easier to remember. Note for example that now none of the COLOR commands need an "=". I could never remember which ones needed it.

Of course, all this took space. The code has been expanded to just over 2 tracks on my 5-1/4" system. To leave room for more, I went to three tracks. The standard Color+ method of storing itself along with the program was never really satisfactory. It filled a disk with just 3 or 4 programs. My additions would only have made things worse.

The method used by BASIC+ was easy to impliment. Just search the directory for a file called "BASIC+" and load the tracks into the top of memory. Once again, the source code was needed. It was reassembled to $A800, and the hooks changed. See the partial listing of the BEXEC* for the details. If you use the original code, you could easily get away with only $B000 and up, and reserve only two tracks. Now a BASIC+ disk fills like any other. The trade-off is that a separate version is needed for systems with different amounts of memory. However, with the C4/C8, 48K seems standard. With the C1, there are many with 32K, and some with 40K, which would require two versions. This is probably why Bob Ankeney used the method of moving up BASIC to allow for Color+. He did not have to be concerned with memory sizes.

Sample programs called ART1 and HAPPY are included with this article. ART1 demonstrates some of the capabilities and the math routines. Note that ART1 is really a compilation of 15 separate routines. Each menu item is a separate program. Add GRINIT:HGR and each will run on its own. HAPPY Birthday demonstrates the ease of using the SID sound chip, and the use of sprites. Note too that the SID chip works will at 3 MHz (connected directly to the 48-pin bus), unlike the General Instruments AY3-8910/12 which is usually run through a 6821 PIA to allow for its slow access times.

Color+ (and in my installation TOSIE II) has opened another area of exploration. I don't own a color monitor and don't intend to get one. The high resolution graphics, sprites, and character sets have opened up another area of programming fun.

SCLR - SID clear, clears all the registers to zero.

VOLnn - Volume set (all 3 voices) with nn = 0 to 15.

WAVE r,n - Waveform where r is the register number with r = 1 to 3 and n is the type of wave. n=1 gives sine waves, n=2 gives triangle waves, n=3 gives pulsed waves, and n=4 gives the noise.

PLAY r,nnnn - Play the sound where r is the register, r = 1 to 3, and nnnn is the frequency value POKEd to the chip. This will be integerized and split into HI/LO values as needed.

OFFn - OFF0 starts the release cycle for all 3 voices. OFF1, OFF2, or OFF3 starts the release cycle for that voice.

ATK r,n - Sets the attack duration, where r is the register and n = 0 to 15.

Letters to the Editor

Dear Sir;

I have an offer which I believe would be of interest to your readers and expand the use of the OSI computers. As I wrote you previously, I have been working at converting an IBM BASIC program to OSI BASIC. I am pleased to report that the conversion is complete and the program runs on my C4P exactly like it runs on the IBM.

I would like to share what I have learned with any other PEEKers who are interested in conversions. I have access to an IBM PC-AT at work where I make my own hard copies and run the IBM programs and I would be happy to make copies for anyone who doesn't have access to an IBM. I also have access to an expert IBM programmer, and we have success making listings of programs protected while being saved. I'm sure that you will agree that the purchaser of a program is entitled to a listing of the program.

Anyway, I've always read PEEK[65] with great interest and learned a great deal from it, but I haven't been able to contribute much. Maybe in the area of conversions I can.

Sincerely,
(Name Withheld)

Dear Sir,

Thank-you for your kind offer to help folks move their IBM software to the OSI. In most cases, the OSI will run rings around IBM PC's BASIC. I've heard of people pitting the OSI against PC-AT's too, and the OSI held its own.

However, I am concerned about your proposal to share software. Commercial programs are copyrighted material. You cannot freely distribute copies of such programs. You may sell the original program as you please, although you would be obligated to destroy any additional copies of the software should you do so. I won't draw any crazy analogies between a book and a program. The bottom line is that its wrong to distribute copies of commercial software. The authors of such software deserve to be paid by everyone who benefits from it.

I also disagree that you have a right to the listing of any program that you purchase. I'm sure that you were never told the program came with a listing, or that it would be unprotected when you bought it. Chances are that the software came with a license agreement. Whether or not that license is totally binding is hard to say. Its a matter of some debate in the industry and in the courts. However, I'm sure that any license you did receive spelled out exactly what your rights were. If the program doesn't do what its supposed to do, then you have many routes of redress. Whether or not you have the right to de-crypt a program is even more nebulous. Under the concept of "fair use", you probably do have that right. However, I could envision arguments against this applying to licensed software.

I'm a software author, so naturally I'm sympathetic to the rights of other authors. However, I buy software too, so I'm not totally biased. Given the state of affairs we find ourselves in these days (both technologically and legally) restraints on copying software boil down to a matter of ethics. When you buy or use a commercial program, you know what the vendor considers his rights to be. If those conditions aren't satisfactory to you, find another vendor. The world is up to its armpits in 'em. But don't assume that any such unsatisfactory conditions confer upon you the right to violate that vendor's rights. If you don't like the deal, walk away from it.

Rick

Dear Sir;

I have owned an OSI C4P-MF with 48K and two disk drives since 1980. Right now, it is sitting, unused, near my Macintosh. In terms of operating speed, software availability, data storage, and general usability, there is really no comparison. The Macintosh is the winner. However, I stil: have a soft spot for the OSI and I am very interested in the project to upgrade to the 65C816 microprocessor. Please tell us more about the new CPU board from Paul Chidley at TOSIE. The closest OSI group is a two hour drive away from me, and my family and schedule makes it very difficult to attend.

You asked for responses to the programming project for a new OS-65D. Here are some of the items I think should be addressed:

Hardware

- the new CPU should be driven at the highest practical speed. The higher speed will allow better programs due to less restrictions in timing.
- A clock/calendar should be included.
- An ASCII-encoded port for keyboards as well as an unencoded input would allow those with video machines to get right of the old unencoded keyboards. An alternative to this is a small adapter board allowing the use of encoded keyboards.
- Serial ports for printer and modem, one of which should be RS-232/RS-422 compatible.
- a disk controller compatible with single and double-sided drives, including 8", 5-1/4", and 3-1/2".
- Hi-res graphics and color. At least 640x400 since that is becoming a standard.
- Obviously we will need new memory cards for all this. With the price of the 6264 static RAM chips down to about $3.00, a 256K card can be wired for less than $200.00

Software

- the memory map shoudl allow at least 4 MB of contiguous memory for future expansion, include a reserved area for disk buffers and RAM disks.
- a built-in directory function.
- Automatic file creation. This would remove the necessity of running a separate program.
- Dynamic file sizing. When developing a program, the file size changes constantly. Let the operating system figure out the size and save it. This could be implimented in one of two methods. UCSD Pascal requires contiguous space on the disk. MS-DOS allows fragmentation of the file and storage in any available sectors.
- a true random access file system built into the operating system with variable record size from 1 bytes to at least 1 K-byte.
- Automatic buffering when using files. Having to save the buffers with a program is a waste of precious disk space.
- Support for more than two disk buffers. In many cases, two buffers are enough, but more wuld make many operations easier and much faster.
- No built-in language. When a language is built in, the machine architecture sometimes is tailored for that language. That tailoring can make implimentation of other languages more difficult than it should be.

There are other things I would like to see, such as character generation in RAM, like the Macintosh, windowing, maybe even multi-tasking (how about a print spooler?). However, what I outlined here are things that should be addressed immediately if the OSI community is to survive and gain any support from the outside.

Sincerely,
Norman Thorsen
22225 Woodward Way NW
Poulsbo, WA 98370

Dear Norman,

Thanks for your comments. Many of the issues you raise are common complaints about OSI systems. Some of them have been addressed by my Hooks into BASIC. You might want to check out your back-issues of PEEK for the article I wrote that includes that software. You'll find a directory command and a file creation command that you can call from BASIC. This eliminates the need to keep a scratch file during program development.

Many of the other items you ask for are also on our list, but some may not be possible or practical. Just as you don't like a language built into the system, I am not in favor of building random file access into the operating system. That should be handled at the language or application program level. Whether or not we switch to non-contiguous files will probably depend on how well such a scheme works on our ancient 8" disk systems. I'm convinced this would be a better way to go, but I don't know how it would work in real life. In any event, keep thinking about it!

Rick

OS-65U Disk File Editor and Directory Utility
Part 1

by Richard L. Trethewey

Sooner or later it happens to everyone. There's a disk error, errant program, or pilot error waiting out there to mangle your precious data. When it happens, you face a difficult problem because there are few tools out there that will let you examine and repair disk files under OS-65U. Last year, while testing MC-DMS and some other software, I kept running into the problem of not being able to easily tell where my software was actually reading and writing to disk. After going through 4 or 5 little utility programs in BASIC, I sat down and wrote this editor in machine code to save time, memory, and hair. I call it DKEDIT.

As with any machine code routine for OS-65U, there are two components to DKEDIT - the assembly language source code and the support program in BASIC. The assembly language is broken into two separate files "DKED1" and "DKED2" and will require about 10 tracks each on your OS-65D diskette. They are written for my ASM-Plus assembler, so if you're using another assembler, you'll have to copy the starting equates into DKED2. Both files will also have to have all cross-references added if you're using a different assembler like the ones from OSI.

The idea behind this editor is very simple. You select a file to edit, and the program displays the contents of that file one page (256 bytes) at a time on the screen. You can page through the file to examine the contents or you can edit it. There are two modes of editing supported. You can enter the hexadecimal value to insert at the cursor position, or you can type in replacement ASCII characters for editing text.

The main reason the program is so large is because it contains all of the support for examining the directory track to locate files. This code came from the directory printing program I showed you last month. Since being able to examine and/or search disk directories is always a handy feature, I left that part of the code intact when I moved it.

The editor depends on a Hazeltine compatible console terminal. Serial systems using other terminals will have to alter the code to reflect any differences between their terminal and a Hazeltine. Video system owners are rather stuck unless they port the video driver from OS-65D V3.3 into OS-65U. I've done it, but I don't think there's any good way for me to write up the technique. However, if there is interest in the video community, I'll try to come up with a legal way of passing on the information without

```
10 REM- Disk File Screen Editor
20
K0=0:K1=1:K2=2:K3=3:K4=4:K5=5:K6=6:K7=7:K8=8:K9
=9:KT=10
30 CLS$=CHR$(27)+CHR$(28)
40 U1SER=PEEK(8778):U2SER=PEEK(8779)
50 POKE 8778,K0: POKE 8779,96
60 T=PEEK(9832): IF T>127 THEN T=T-124: IF T>63
THEN T=T-58
70 DD$=CHR$(T+65)
100 PRINT CLS$;"Disk File Screen Editor": PRINT
110 PRINT "(1) Directory"
120 PRINT "(2) Edit a File"
125 PRINT "(3) Quit"
130 PRINT: INPUT "    Your Choice ";Y$:
CMD=VAL(Y$)
140 PRINT: IF Y$="" THEN 500
150 IF CMD=K1 THEN 200
160 IF CMD=K2 THEN 300
165 IF CMD=K3 THEN 500
170 GOTO 100
200 REM- Vanilla Directory
210 PRINT CLS$
220 INPUT "DEVice ";DR$: L=LEN(DR$)
230 PRINT: IF L<>K1 THEN 220
240 C=ASC(DR$): IF C>ASC("Z") THEN C=C-32
250 DEV CHR$(C)
260 X=USR(K0),NP,NM,NK,NS: PRINT X;" Bytes in
Use"
270 PRINT: GOSUB 63000: GOTO 100
280 :
300 PRINT"Edit File on ";: X=USR(K6): GOTO 100
310 :
500 GOSUB 63000: IF DD$<>"" THEN DEV DD$
510 POKE 8778,U1SER: POKE 8779,U2SER
520 IF RP$="" THEN END
530 RUN RP$,PW$
540 :
60000 RP$="BEXEC*": PW$="PASS": GOTO 20
60010 :
63000 INPUT "Press <RETURN> to continue ";Y$
63010 IF Y$<>"STOP" AND Y$<>"stop" THEN RETURN
63020 GOTO 510
```

violating any copyrights. In any event, the directory program and the techniques involved will be of benefit to any 65U user regardless of their set-up.

As I said, the editor is very simple and I hope, simple for others to use. Whatever you're doing, there's always a prompt line along the bottom of the display which reflects the level you're at within the program. The top level is where you start when you've just opened the file. The contents of the first page of the file are displayed in hexadecimal, and the cursor is positioned at the upper left-hand corner of the contents display. Pressing the ">" key brings up the next page of data from the file and the "<" key brings back the previous page. The page number is also displayed so you know where you are. Pressing "Q" stops the program and returns you to the main menu.

Two other commands are available at this top level. These commands send you to the next level of the program and determine the editing mode. Entering "N" selects numeric editing in which you enter hexadecimal values to insert in the file. "A" selects ASCII editing in which your keystrokes are directly entered in the file.

Once you have selected the editing mode, the prompt line changes. Entering "M" sends you to a level where you can move the cursor with the "U", "D", "L", and "R" keys (for Up, Down, Left, and Right respectively).

Entering "E" enables editing and your changes are made effective at the current cursor position. After each byte is changed, the cursor is automatically moved to the next byte in the page. When the end of the page is reached, the cursor is returned to the top of the same page. Yes, I probably should have written it to advance to the next page, but I didn't so that you could abort without making any changes you weren't sure of. In the editing mode, you must press the <ESC> key to stop editing and return to the next higher level.

That's all there is to it. As I said, its a simple program. However, there are a number of things within the program worth examining closer.

First of all, there is the interface to BASIC where the machine code calculates what command you've issued from the main menu. Naturally, the USR(X) vectors pointing to the machine code (ie. locations 8778 and 8779) have been set up. I have mentioned this before, but it bears repeating. Whenever you alter the USR(X) vector to your own code, you should always retain a copy of the initial contents of these locations and restore the vector when your program is finished

```
10 .PAGE 'OS-65U DISK FILE EDITOR'
20; WRITTEN BY RICHARD L. TRETHEWEY
30; COPYRIGHT 9/7/85   ALL RIGHTS RESERVED
40;
50; BASIC EXTERNALS
60;
70 STRFLG =$000E  STRING FLAG
80 INTFLG =$000F  INTEGER FLAG
90 POSCNT =$0016  CURSOR POSITION
100 POKER  =$0019  UTILITY POINTER
110 BUF    =$001B  BASIC Z-PAGE BUFFER (71 CHARS.)
120 INDEX  =$006F  UTLITY POINTER
130 MEMSIZ =$0084  END OF BASIC MEMORY
140 VARNAM =$0092  VARIABLE NAME STORAGE
150 VARPNT =$0094  POINTER TO VARIABLE STORAGE
160 FORPNT =$0096  PTR. TO VAR. FOR STORING
170 VARPTR =$00AC  VARIABLE POINTER
180 FACEXP =$00AE  F.P. ACC. EXPONENT
190 FACHI  =$00AF  F.P. ACC. MSB
200 FACMHI =$00B0  F.P. ACC. NMSB
210 FACMLO =$00B1  F.P. ACC. NLSB
220 FACLO  =$00B2  F.P. ACC. LSB
230 FACSGN =$00B3  F.P. ACC. SIGN (+/-)
240 FACGRD =$00BD  F.P. ACC. EXPONENT GUARD BYTE
250 CHRGET =$00C0  FETCH NEXT CHARACTER
260 CHRGOT =$00C6  RETRIEVE LAST CHAR. SEEN
270 TXTPTR =$00C7  PTR. TO PROGRAM FOR CHRGET/GOT
280 CRDO   =$0A73  OUTPUT CR/LF PAIR
290 OUTSTR =$0ACC  OUTPUT STRING POINTED TO BY A/Y
300 OUTDO  =$0AEE  OUTPUT CHARACTER IN ACC.
310 CHKTYP =$0CBC  MAKE SURE NUMERIC TYPE EXPRESSION
320 CHKSTR =$0CBE  MAKE SURE STRING EXPRESSION
330 FRMEVL =$0CCD  FORMULA EVALUATOR
340 CHKCLS =$0E0D  INSURE ")", EXIT THROUGH CHRGET
350 CHKOPN =$0E10  INSURE "(", EXIT THROUGH CHRGET
360 CHKCOM =$0E13  INSURE ",", EXIT THROUGH CHRGET
370 SNERR  =$0E1E  SYNTAX ERROR
380 PTRGET =$0F2E  FIND VARIABLE IN STORAGE TABLE
390 FCERR  =$10D0  FUNCTION CALL ERROR
400 GIVAYF =$1218  GIVE A/Y PAIR TO F.P. ACC.
410 FREFAC =$1520  FIND STRING LOCATION & LENGTH
420 GETBYT =$1618  EVALUATE EXPRESSION<256 --> X REG.
430 GETVAR =$1A9D  PUT VARIABLE IN F.P. ACC.
440 FLOAT  =$1B44  CONVERT INTEGER TO F.P. TYPE
450 QUINT  =$1B96  CONVERT F.P. TO INTEGER
460 ASCFP  =$1BEE  CONVERT ASCII AT 'TXTPTR' TO FP
470 ASCII  =$1CEC  CONVERT F.P. ACC. TO ASCII STRING
480;
490; OS-65U EXTERNALS
500;
510 DISCN  =$2668  CURRENT DRIVE
520 DUN    =$26A1  DISK UNIT CONTROL BLOCK
530 DIRADR =$26AB  DIRECTORY DISK ADDR. STORAGE
540 DIRSIZ =$26AE  DIRECTORY SIZE STORAGE
550 DIRBUF =$26F2  DIRECTORY BUFFER
560 OUTCH  =$2808  OUTPUT CHARACTER IN ACC.
570 GET    =$28E8  READ DISK
580 PUT    =$28F3  WRITE TO DISK
590 FLUSH  =$2C23  FLUSH SYSTEM DISK BUFFER/CLOSE
600 OUFLAG =$2DA6  CURRENT OUTPUT DEVICE #
610 SWBUFF =$4700  PAGE 0/1 SWAP BUFFER
620 SWAP   =$4907  SWAP 0/1 WITH SWAP BUFFER
630;
640; OS-65U DISK CONTROL BLOCK DEFINITION
650;
660; DUN     = DISC UNIT NUMBER TO READ/WRITE
670; DUN+1   = DISK ADDRESS LSB
```

```
680; DUN+2    = DISK ADDRESS NLSB
690; DUN+3    = DISK ADDRESS NMSB
700; DUN+4    = DISK ADDRESS MSB
710; DUN+5    = NUMBER OF BYTES LSB
720; DUN+6    = NUMBER OF BYTES MSB
730; DUN+7    = MEMORY ADDRESS LSB
740; DUN+8    = MEMORY ADDRESS MSB
750;
760; ASSEMBLY CONSTANTS
770;
780 BS      =$08
790 LF      =$0A
800 CR      =$0D
810 ESC     =$1B
820 SP      =$20
830 SKIP2   =$2C
840 DEL     =$5F
850 STACK   =$100
860;
870; EDITOR EXTERNALS
880;
890 PTR     =$50
900 STRPTR  =$52
910 TMP     =$54
920 TMP1    =$55
930;
940         *=$6000
950;
960         LDA FORPNT        FETCH ENTRY FORPNT
970         STA OLDFOR        SAVE FOR RESTORE ON EXIT
980         LDA FORPNT+1
990         STA OLDFOR+1
1000         JSR $1047        MAKE CMD# AN INTEGER
1010         LDA FACLO        PICK UP CMD#
1020         CMP #TYPE-CMDTBL/2
1030         BCS BADCMD
1040         STA CMD          SAVE COMMAND #
1050         ASL A            *2!
1060         TAX
1070         LDA CMDTBL,X
1080         STA DOCMD+1
1090         LDA CMDTBL+1,X
1100         STA DOCMD+2
1110 DOCMD   JMP $FFFF        MODIFIED CODE!!!!
1120 BADCMD  JMP FCERR
1130;
1140 USRDIR  JSR CRDO
1150         JSR DIRSU
1160         JSR HEADER
1170         JMP D2           GO TO DISPLAY
1180;
1190 DIRSU   LDA DISCN        GET DEVICE NUMBER
1200         STA DUN          GIVE TO 65U CONTROL BLOCK
1210         LDA #$00         INIZ
1220         STA DUN+1        CLEAR DISK ADDR. LSB
1230         STA DUN+3
1240         STA DUN+4
1250         STA DUN+5        CLEAR # BYTES LSB
1260         LDA #$01
1270         STA DUN+6        SET R/W FOR 1 PAGE
1280         LDA #DIRBUF
1290         STA DUN+7        SET RAM ADDRESS LSB
1300         LDA #DIRBUF/256
1310         STA DUN+8        SET RAM ADDRESS MSB
1320         LDA #25088/256
1330         STA DUN+2        POINT TO DIREC*
1340         JSR GETDSK       READ IT
```

because it is a common practice for software packages to install machine code routines in the BEXEC* program and to install a USR(X) vector at boot up. Sometimes, other programs within such packages will assume that the vectors are untouched since they were installed. Thus, if you alter these locations without restoring them afterward, you can get hit with some mysterious crashes.

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

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

The directory printer will probably interest a lot of people for a couple of reasons. First of all, it's fast. I mean **REALLY FAST!** Have your fingers ready on <CTRL>'S' when you use this baby. Second, the program prints out a valid password for each file. Note that due to the encoding method used by OSI, the password displayed may not be identical to the one you selected when the file was created, but it will work nonetheless. Third, the code used does several interesting things. First, it expands the normal format of the USR(??) function. Second, it demonstrates how to access the disk drives and the directory under OS-65U. Third, it demonstrates several useful techniques for calling routines in BASIC from your own machine code.

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

```
1350        LDY #$00         INIZ
1360 D1     LDA DIRBUF+$C,Y  READ DIREC* SIZE
1370        STA SIZE,Y       SAVE IT LOCALLY
1380        INY
1390        CPY #$03
1400        BNE D1           LOOP 'TIL DONE
1410        LDY #$00
1420        TYA
1430 D6     STA INUSE,Y
1440        INY
1450        CPY #TABTO-INUSE
1460        BNE D6
1470        LDA #98
1480        STA INUSE+1         SHOW DIR OFFSET
1490        RTS
1500;
1510; MAIN LOOP
1520;
1530 D2     JSR GETDSK           READ IN DIR PAGE
1540        LDA #DIRBUF          LOAD DIRBUF LSB
1550        STA POKER            GIVE TO POKER
1560        LDA #DIRBUF/256      LOAD MSB
1570        STA POKER+1          SET IT UP TOO
1580        LDA #$00
1590        STA EC
1600        LDA COUNT            BUMP COUNTER LSB
1610        BNE D3               WATCH FOR PAGING
1620        LDA COUNT+1          BUMP NMSB ON PAGING
1630        BNE D3               AND WATCH AGAIN
1640        LDA COUNT+2          BUMP MSB ON PAGING
1650        BNE D3
1660        INC EC
1670        LDA #$10
1680        CLC
1690        ADC POKER
1700        STA POKER
1710        BCC D3
1720        INC POKER+1
1730 D3     JSR DIROUT           DISPLAY CONTENTS
1740        INC COUNT            BUMP COUNTER LSB
1750        BNE D4               WATCH FOR PAGING
1760        INC COUNT+1          BUMP NMSB ON PAGING
1770        BNE D4               AND WATCH AGAIN
1780        INC COUNT+2          BUMP MSB ON PAGING
1790 D4     LDA COUNT            FETHC LSB
1800        CMP SIZE             READ ENTIRE DIR?
1810        BNE D5               NO! ==> D5
1820        LDA COUNT+1          MAYBE, CHECK NMSB
1830        CMP SIZE+1           SAME?
1840        BNE D5               NO! ==> D5
1850        LDA COUNT+2          FETCH MSB
1860        CMP SIZE+2           SAME?
1870        BEQ DIRQT            YES! END!
1880 D5     JSR DBUMP            BUMP DIRECTORY PTRS
1890        JMP D2               AND LOOP!
1900 DIRQT  JSR SAVVAL           SAVE FILE COUNTS
1910        LDA INUSE
1920        STA FACLO
1930        LDA INUSE+1
1940        STA FACMLO
1950        LDA INUSE+2
1960        STA FACMHI
1970        LDA INUSE+3
1980        STA FACHI
1990        LDA OLDFOR           GET X= FORPNT
2000        STA FORPNT           RESTORE IT FOR BASIC
2010        LDA OLDFOR+1         GET MSB TOO
```

```
2020        STA FORPNT+1
2030        JMP NORMAL    EXIT VIA NORMAL
2040;
2050 SAVVAL LDA NUMPRG+1  GET #P FILES MSB
2060        LDY NUMPRG    GET #P FILES LSB
2070        JSR GIVAYF    GIVE TO FPACC.
2080        JSR SAVNUM    GIVE TO "NP" VAR
2090        LDA NUMMF+1   GET # OF MF MSB
2100        LDY NUMMF     AND LSB
2110        JSR GIVAYF    GIVE TO FPACC.
2120        JSR SAVNUM    GIVE FP TO "NM"
2130        LDA NUMKF+1   GET #KEY MSB
2140        LDY NUMKF     GET #KEY LSB
2150        JSR GIVAYF    GIVE TO FPACC.
2160        JSR SAVNUM    GIVE TO "NK"
2170        LDA NUMSCR+1  GET #SCR FILES
2180        LDY NUMSCR
2190        JSR GIVAYF    GIVE TO FPACC.
2200;
2210 SAVNUM JSR CHKCOM    FIND OUR FRIEND
2220        JSR PTRGET    FIND THE VAR.
2230        STA FORPNT    SAVE PTR TO VAR
2240        STY FORPNT+1
2250        LDA STRFLG
2260        BNE SAVNU2
2270        LDA INTFLG
2280        BPL SAVNU1
2290        JMP $09C5   GIVE F.P. TO % VAR
2300 SAVNU1 JMP $1ACB   FACC. TO F.P. VAR
2310 SAVNU2 JMP FCERR   CAN'T USE STRINGS!
2320;
2330 DBUMP  INC DUN+2
2340        BNE DBUM1
2350        INC DUN+3
2360        BNE DBUM1
2370        INC DUN+4
2380 DBUM1  RTS
2390;
2400 CMDTBL .WORD USRDIR   DISPLAY ALL
2410        .WORD USRDIR   DATA FILES ONLY
2420        .WORD USRDIR   PROGRAMS ONLY
2430        .WORD USRFIL   FIND DISK ADDR.
2440        .WORD WILD     WILD CARD DIR
2450        .WORD KEYGET   GET KEYPRESS
2460        .WORD EDIT     FILE EDITOR
2470;
2480 TYPE   .BYTE 'DATA '
2490        .BYTE 'BASIC'
2500        .BYTE 'OTHER'
2510 AR     .BYTE 'NONE '
2520        .BYTE 'READ '
2530        .BYTE 'WRITE'
2540        .BYTE 'R/W  '
2550 DELTYP .BYTE '[----]  Deleted File',
$00
2560 MFTYP  .BYTE 'Master',$00
2570 KFTYP  .BYTE 'Key',$00
2580 SCRTYP .BYTE 'Scratch',$00
2590 HEAD   .BYTE 'Name     Password  '
2600        .BYTE 'Type   Access  '
2610        .BYTE 'Address     '
2620        .BYTE 'Size       Special'
2630        .BYTE CR,LF,$00
2640 CURFIL .BYTE 'XXXXXX',CR
2650 TMPTYP .BYTE $00   TEMP. TYPE STORAGE
2660 CMD    .BYTE $00        COMMAND
```

```
2670 OLDFOR .WORD $FFFF
2680 DRIVE  .BYTE $00
2690 SIZE   .BYTE $00,$00,$00
2700 STADDR .BYTE $00,$00,$00,$00
2710 ENADDR .BYTE $00,$00,$00,$00
2720 CADDR  .BYTE $00,$00,$00,$00
2730 FSIZE  .BYTE $00,$00,$00
2740 BFENPG .BYTE $00
2750 BSIZE  .BYTE $00
2760 INUSE  .BYTE $00,$00,$00,$00
2770 RECOV  .BYTE $00,$00,$00,$00
2780 COUNT  .BYTE $00,$00,$00
2790 NUMMF  .WORD $0000
2800 NUMKF  .WORD $0000
2810 NUMSCR .WORD $0000
2820 NUMPRG .WORD $0000
2830 TEMP   .BYTE $00
2840 EC     .WORD $0000
2850 PW     .BYTE $00,$00,$00,$00
2860 TABTO  .BYTE $00
2870;
2880 TABER  LDA POSCNT
2890        CMP TABTO
2900        BCS TABER1
2910        LDA #SP
2920        JSR OUTDO
2930        JMP TABER
2940 TABER1 RTS
2950;
2960 HEADER LDA #HEAD
2970        LDY #HEAD/256
2980        JSR OUTSTR
2990        LDY #$00
3000        LDA #'-
3010 HEADE1 JSR OUTDO
3020        INY
3030        CPY #62
3040        BNE HEADE1
3050        JMP CRDO
3060;
3070 TYPCHK LDY #$08
3080        LDA (POKER),Y
3090        AND #%11100
3100        LSR A
3110        LSR A
3120        PHA
3130        STA TYPCH1+1
3140        ASL A
3150        ASL A
3160 TYPCH1 ADC #$FF          *5!
3170        STA TMPTYP   SAVE FOR LATER
3180        PLA
3190        TAX
3200        INX          +1!
3210        LDY CMD      CHECK COMMAND #
3220        BEQ TYPCH2   CMD 0? --> PASS
3230        CPX CMD      CMD = TYPE?
3240        BNE TYPCH3   NO! ==>
3250 TYPCH2 SEC
3260        RTS
3270 TYPCH3 LDY #$0C
3280        LDA (POKER),Y
3290        CLC
3300        ADC INUSE+1
3310        STA INUSE+1
3320        INY
3330        LDA (POKER),Y
```

```
3340        ADC INUSE+2
3350        STA INUSE+2
3360        INY
3370        LDA (POKER),Y
3380        ADC INUSE+3
3390        STA INUSE+3
3400        CLC
3410        RTS
3420;
3430 DIRDUN JSR CRDO
3440        PLA
3450        PLA
3460        JMP DIRQT    RETURN TO CALLER
3470;
3480 DIRNX0 LDA #DELTYP
3490        LDY #DELTYP
3500        JSR OUTSTR
3510        JSR TYPE4
3520        JSR CRDO
3530        JMP DIRNXT
3540;
3550 DIROUT JSR TYPCHK    CHECK ENTRY TYPE
3560        BCS DIRO1
3570        JMP DIRNXT    NOT WANTED!
SKIP!
3580 DIRO1  LDY #$00       INIZ
3590        LDA (POKER),Y  FETCH CHAR.
3600        BEQ DIRDUN    0? YES! END DIR!
3610        CMP #$01      DELETED ENTRY?
3620        BEQ DIRNX0   YES! SKIP TO NEXT
3630        JSR PNAME    PRINT NAME/PW
3640        JSR TYPER  PRINT TYPE & RIGHTS
3650        JSR FTYPE
3660 DIRNXT LDA POKER
3670        CLC
3680        ADC #$10
3690        STA POKER
3700        BCC DIRNX1
3710        INC POKER+1
3720 DIRNX1 INC EC
3730        LDA EC
3740        CMP #256/16
3750        BNE DIROUT
3760        RTS
3770;
3780 GETDSK JSR SWAP
3790        LDA #GETD1-1/256
3800        PHA
3810        LDA #GETD1-1
3820        PHA
3830        JMP GET
3840 GETD1  .WORD DUN
3850        JSR SWAP
3860        TAY
3870        BNE GETD2
3880        RTS
3890;
3900 GETD2  JMP FCERR   ABORT ON DISK ERR.
3910;
3920 PUTDSK JSR SWAP
3930        LDA #PUTD1-1/256
3940        PHA
3950        LDA #PUTD1-1
3960        PHA
3970        JMP PUT
3980 PUTD1  .WORD DUN
3990        JSR SWAP
```

```
4000        TAY
4010        BNE GETD2
4020        RTS
4030;
4040 TYPER  LDA #18
4050        STA TABTO
4060        JSR TABER
4070        LDX #$00
4080        LDY TMPTYP
4090 TYPE1  STX TEMP
4100        LDA TYPE,Y
4110        JSR OUTDO
4120        LDX TEMP
4130        INX
4140        INY
4150        CPX #$05
4160        BNE TYPE1
4170        LDA #25
4180        STA TABTO
4190        JSR TABER
4200        LDY #$08
4210        LDA (POKER),Y
4220        AND #$03
4230        STA TYPE2+1
4240        ASL A
4250        ASL A
4260 TYPE2  ADC #$FF
4270        TAY
4280        LDX #$00
4290 TYPE3  STX TEMP
4300        LDA AR,Y
4310        JSR OUTDO
4320        LDX TEMP
4330        INY
4340        INX
4350        CPX #$05
4360        BNE TYPE3
4370 TYPE4  LDA #32
4380        STA TABTO
4390        JSR TABER
4400        LDA #$00
4410        STA FACLO
4420        LDY #$09
4430        LDA (POKER),Y
4440        STA FACMLO
4450        INY
4460        LDA (POKER),Y
4470        STA FACMHI
4480        INY
4490        LDA (POKER),Y
4500        STA FACHI
4510        JSR NORMAL
4520        JSR ASCII
4530        LDA #STACK
4540        LDY #STACK/256
4550        JSR OUTSTR
4560        LDA #43
4570        STA TABTO
4580        JSR TABER
4590        LDA #$00
4600        STA FACLO
4610        LDY #$0C
4620        LDA (POKER),Y
4630        STA FACMLO
4640        CLC
4650        ADC INUSE+1
```

```
4660        STA INUSE+1
4670        INY
4680        LDA (POKER),Y
4690        STA FACMHI
4700        ADC INUSE+2
4710        STA INUSE+2
4720        INY
4730        LDA (POKER),Y
4740        STA FACHI
4750        ADC INUSE+3
4760        STA INUSE+3
4770        JSR NORMAL
4780        JSR ASCII
4790        LDA #STACK
4800        LDY #STACK/256
4810        JSR OUTSTR
4820        RTS
4830;
4840 FTYPE  LDY #$08
4850        LDA (POKER),Y
4860        AND #%11100
4870        BNE FTYPE6  NOT DATA! PRG?
4880        LDA #55
4890        STA TABTO
4900        JSR TABER
4910        LDY #$05
4920        LDA (POKER),Y
4930        CMP #'0
4940        BEQ FTYPE4     MASTER
4950        CMP #'1
4960        BCC FTYPE2     SCRATCH
4970        CMP #'8
4980        BCS FTYPE2     SCRATCH
4990        LDA #KFTYP
5000        LDY #KFTYP/256
5010        JSR OUTSTR
5020        INC NUMKF
5030        BNE FTYPE1
5040        INC NUMKF+1
5050 FTYPE1 JMP CRDO
5060 FTYPE2 LDA #SCRTYP
5070        LDY #SCRTYP/256
5080        JSR OUTSTR
5090        INC NUMSCR
5100        BNE FTYPE3
5110        INC NUMSCR+1
5120 FTYPE3 JMP CRDO
5130 FTYPE4 LDA #MFTYP
5140        LDY #MFTYP/256
5150        JSR OUTSTR
5160        INC NUMMF
5170        BNE FTYPE5
5180        INC NUMMF+1
5190 FTYPE5 JMP CRDO
5200 FTYPE6 CMP #%100
5210        BNE FTYPE7
5220        INC NUMPRG
5230        BNE FTYPE7
5240        INC NUMPRG+1
5250 FTYPE7 JMP CRDO
5260;
5270 USRFIL JSR GTFNAM       GET FILE NAME
5280        JSR FNDREM    REMOTE FILE FIND
5290        BCC USRF1
5300        LDA #$FF
5310        TAY
5320        LDX OLDFOR
5330        STX FORPNT
5340        LDX OLDFOR+1
5350        STX FORPNT+1
5360        JMP GIVAYF     SHOW NO MATCH!
5370;
5380 USRF1  LDY #$09
5390        LDA (POKER),Y
5400        STA FACMLO
5410        INY
5420        LDA (POKER),Y
5430        STA FACMHI
5440        INY
5450        LDA (POKER),Y
5460        STA FACHI
5470        LDA #$00
5480        STA FACLO
5490        JSR NORMAL
5500        LDX OLDFOR
5510        STX FORPNT
5520        LDX OLDFOR+1
5530        STX FORPNT+1
5540        RTS        AND RETURN TO CALLER
5550;
5560; NORMALIZE FLOATING POINT ACCUMULATOR
5570;
5580 NORMAL LDA #32+$80
5590        STA FACEXP
5600        LDA FACHI
5610        BMI NORMA2
5620        BNE NORMA1
5630        LDA FACMHI
5640        BNE NORMA1
5650        LDA FACMLO
5660        BNE NORMA1
5670        LDA FACLO
5680        BEQ NORMA3        0! ==>
5690 NORMA1 DEC FACEXP
5700        ASL FACLO
5710        ROL FACMLO
5720        ROL FACMHI
5730        ROL FACHI
5740        BPL NORMA1
5750 NORMA2 RTS
5760 NORMA3 STA FACEXP
5770        RTS
5780;
5790 KEYGET JSR $0587
5800        TAY
5810        LDA #$00
5820        JMP GIVAYF
5830;
5840 PNAME  LDY #$00         INIZ
5850 PNAME1 LDA (POKER),Y    FETCH CHAR.
5860        JSR OUTDO        PRINT IT
5870        INY              BUMP IT
5880        CPY #$06   PRINTED WHOLE NAME?
5890        BNE PNAME1        NO! LOOP!
5900        LDA #8
5910        STA TABTO
5920        JSR TABER
5930        LDY #$06     GET INDEX TO PW
5940        LDX #$00
5950 PNAME2 LDA (POKER),Y GET 1ST PW CHAR.
5960        PHA               SAVE IT
5970        AND #$0F   MASK TO LOW NIBBLE
5980        CMP #$0F
```

```
5990        BNE PNAME3    NOT DEFAULT ==>
6000        LDA #'-
6010        BNE PNAME4
6020 PNAME3 CLC
6030        ADC #78
6040 PNAME4 STA PW+1,X
6050        PLA
6060        LSR A
6070        LSR A
6080        LSR A
6090        LSR A
6100        CMP #$0F
6110        BNE PNAME5
6120        LDA #'-
6130        BNE PNAME6
6140 PNAME5 CLC
6150        ADC #65
6160 PNAME6 STA PW,X
6170        CPX #$02
6180        BEQ PNAME7
6190        INY
6200        LDX #$02
6210        BNE PNAME2
6220 PNAME7 LDY #$00
6230 PNAME8 LDA PW,Y
6240        JSR OUTDO
6250        INY
6260        CPY #$04
6270        BNE PNAME8
6280        RTS
6290;
6300 GTFNAM JSR CHKCOM      FIND THE COMMA
6310        JSR FRMEVL    EVALUATE EXPRES.
6320        JSR FREFAC-3    CHKSTR & FIND
6330        CMP #$07        CHECK LENGTH
6340        BCC GTFN1     O.K. ==> CONT.
6350        JMP SNERR       BAD! ERROR!
6360 GTFN1  STA GTFN3+1     SAVE LENGTH
6370        LDY #$00        INIZ
6380        STY STRFLG  CLEAR STRFLG EARLY
6390 GTFN2  LDA (INDEX),Y   FETCH A CHAR.
6400        JSR CASECK    MAKE IT ALL CAPS
6410        STA BUF,Y       SAVE IT
6420        INY
6430 GTFN3  CPY #$FF
6440        BNE GTFN2
6450        LDA #SP
6460 GTFN4  CPY #$06
6470        BEQ GTFN5
6480        STA BUF,Y
6490        INY
6500        BNE GTFN4
6510 GTFN5  LDA #CR
6520        STA BUF,Y
6530        RTS
6540;
6550 WILD   JSR GTFNAM        GET FILE NAME
6560        JSR DIRSU  SET UP FOR DIR READ
6570        JSR HEADER
6580        LDY #$00
6590 WILD0  LDA BUF,Y
6600        CMP #SP
6610        BNE WILD1
6620        LDA #'?
6630        STA BUF,Y
6640 WILD1  INY
6650        CPY #$06
6660        BNE WILD0
6670 WILD2  JSR GETDSK
6680        LDA #DIRBUF
6690        STA POKER
6700        LDA #DIRBUF/256
6710        STA POKER+1
6720        LDA #$00
6730        STA EC
6740 WILD3  LDY #$00
6750 WILD4  LDA (POKER),Y
6760        BEQ WILDC      END OF DIR! ==>
6770        CMP #$01
6780        BEQ WILD9    SKIP DELETED'S
6790        LDA BUF,Y
6800        CMP #'?
6810        BEQ WILD8
6820        CMP #'#         LOOK FOR #?
6830        BNE WILD6
6840        LDA (POKER),Y
6850        CMP #'0
6860        BCC WILD9
6870        CMP #'9+1
6880        BCS WILD9
6890        BCC WILD8
6900 WILD6  LDA BUF,Y
6910        CMP (POKER),Y
6920        BNE WILD9
6930 WILD8  INY
6940        CPY #$06
6950        BNE WILD4
6960        JSR TYPCHK
6970        JSR PNAME
6980        JSR TYPER
6990        JSR FTYPE
7000 WILD9  LDA POKER
7010        CLC
7020        ADC #$10
7030        STA POKER
7040        LDA POKER+1
7050        ADC #$00
7060        STA POKER+1
7070        INC EC
7080        LDA EC
7090        CMP #256/16
7100        BNE WILD3
7110        INC COUNT
7120        BNE WILDA
7130        INC COUNT+1
7140        BNE WILDA
7150        INC COUNT+2
7160 WILDA  LDA COUNT+2
7170        CMP SIZE+2
7180        BNE WILDB
7190        LDA COUNT+1
7200        CMP SIZE+1
7210        BNE WILDB
7220        LDA COUNT
7230        CMP SIZE
7240        BEQ WILDC
7250 WILDB  JSR DBUMP
7260        JMP WILD2
7270 WILDC  LDA #$00
7280        TAY
7290        LDX OLDFOR
7300        STX FORPNT
7310        LDX OLDFOR+1
7320        STX FORPNT+1
```

```
7330        JMP GIVAYF
7340;
7350        .FILE DKED2
```

Continued Next Month

Book Bargains!

Sam's Service Manuals

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

C1P Sam's Regular Price: $7.95 **Sale Price:** $4.00
C4P Sam's Regular Price: $15.95 **Sale Price:** $10.00
C2/C3 Regular Price: $39.95 **Sale Price:** $25.00

65V Primer

This is an introductory guide to machine code that shows you how to program your video system u Monitor ROM. An excellent tutorial on the fundamentals of machine code.

Regular Price: $5.95 **Sale Price:** $3.00

Assembler/Editor - Extended Monitor Manual

Until recently, OSI included the Assembler/Editor and Extended Monitor software with all copies of However, even when it was free, there was little documentation accompanying the disks. If you've beer for instructions on these two programs, this is the book for you!

Regular Price: $6.95 **Sale Price:** $4.00

How To Program Microcomputers

By William Barden, this book explains the instruction set of the 8000, 6500, and 6800 s microprocessors. While not OSI-specific, this book contains many valuable algorithms for solving prot machine code using the microprocessors available in OSI computers.

Regular Price: $8.95 **Sale Price:** $4.00

Professional Computers Set Up and Operations Manual

A valuable guide for installing and using OSI serial systems. Includes an overview of classic OSI soft these systems. The book also provides information on how to program the C3 series using the Z-80 a microprocessors.

Regular Price: $9.95 **Sale Price:** $6.00

Introductory Manuals

These books don't contain a lot of information that isn't duplicated in many other places. Still, for the user, they can be a valuable reference to keep by your system while you're learning. Specify C1P/C1P-I cassette, C4P-MF, or C8P-DF.

Regular Price: $6.95 **Sale Price:** $2.00

User Guides

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

C1P-MF Regular Price: $8.95 **Sale Price:** $4.00
C4P-MF Regular Price: $8.95 **Sale Price:** $5.00
C8P-DF Regular Price: $8.95 **Sale Price:** $5.00

1986 Index to PEEK[65]

Public Digital Radio Service	D. Stoner	Oct/Nov, p7
Remote Break and Boot	G. Jennings	Dec, p4
SAM, the Self-Aware Microcomputer	R. Reed	Sept, p13
Speed Hints for 65U BASIC	R. Clegg	Oct/Nov, p13
String Swap for 65D BASIC	E. Morris	Apr, p7
A Tale of Two Ads	E. Morris	Feb, p3
User Survey: Final Results	R. Trethewey	Sum, p2
User Survey: Preliminary Results	R. Trethewey	Apr, p11
Using High Density 5-1/4" Drives	D. Livesay	Dec, p12
Using OSI SIG on CompuServe	R. Trethewey	Mar, p7
USR(X) (Y) (Z) (1) (2) (3)	E. Morris	Jan, p10
U-Word: 65U Word Processor	R. Trethewey	Sept, p9
Wazzat Corner: Printer control	L. E. Jankowski	Feb, p2

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!

Classy-AD$

C8P-DF New, never used. 12-inch BMC monitor. $495. Al Stark 703-524-5455 DWH

C3B with 80MB hard disk, dual floppies, Hazeltine 1420 terminal with Polaroid screen, OSI rack cabinet, D&N CP/M card, D&N serial card, OSI serial card. New disks, manuals, software. Al Stark 703-524-5455

OSI C1P 32K, $100; C2P, 8K, $75; unused ISID 5-1/4" drive, $75; 5" monitor, $50; parts for RS-232 and disc cable. All for $250. Bob Oldham, 344 White Oak Creek Road, Burnsville, NC 28714

Isotron OSI Work System 200, forty meg hard disk, Terminal and Printer, like new. $3400.00. Cleveland, 216-524-8686 or 216-467-2652

C3 system with four user memory paritions, two SS floppy drives, CD-74 hard disk, serial and parallel printer ports, two Hazeltine 1500 terminals, lots of software. Best offer. Call Ron in Denver at 303-771-6990.

OSI boards, cases, power supplies, disc drives, and software for sale. Send SASE for list and prices. R. Groome, 824 West Main Street, Richmond, IN 47374

For Sale: C3-S1, Hazeltine 1420, Remex 8" double-sided floppies, serial board, 56K RAM, extras. Asking $600.00. Three 1 MHz 520 boards, asking $50. 470 parallel output board, asking $35. Contact William Haas, Box 361, Lowell, OH 45744, 614-896-2344

Solve Your Disk Drive Problems. Increase storage capacity. Introducing the DL Systems E-15 data separator/motor control board. With this board you can use any standard 5.25" drive to replace your mini-floppy drives or use high-density 5.25" drives to replace 8" drives. The board comes completely assembled and tested. A 34-pin connector with ejector clips for the disk drive cable is installed on the board. All IC's are on machined gold-plated socket. The board comes with an instruction manual describing how to install it and instructions for converting to 80-track usage. For mini-floppy systems, order P/N E-15. For 8" systems, order P/N E-15HD. Price: $49.00 including shipping and handling. For information, contact David Livesay. Order from PEEK[65].

NOTICE!

Your mailing label has the day and month transposed.

PEEK (65)
The Unofficial OSI Users Journal

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

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 0S-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 ____________