PEEK (65)

The Unofficial OSI Users Journal

$1.75

MAY 1984
VOL.5,NO.5

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

INSIDE

Column One

Last month's column has aroused a good deal of interest. Readers want to know "Where can we get this laser-beam printer?"

For now, you can't. But you should be able to by the end of the year, but it won't say Canon on it...

Let me explain. The Canon Laser Beam Printer, inexpensive version, was shown at the Comdex conference last fall, by Canon. However, they will only sell them in lots of many hundreds or thousands, and only with a "video" interface. In other words, you must send it a signal not unlike the signal sent to a TV set to make it show a picture.

But who can send a video interface signal? Anybody who makes a board to generate the signal. Probably not a project for the home hacker (though I bet some PEEK(65) readers could do it). More likely companies who now manufacture letter-quality printers will get smart and design a board which will receive RS-232 signals and generate video signals to drive the printer.

The beauty of the whole thing, in addition to its quiet operation and lovely print quality, is that Canon is offering the machinery for about $1,000 in large quantities! This means that even after adding a board and the cost of selling the thing and their profit, manufacturers will be able to sell the whole kaboodle at maybe $3,500!

So, how do you get one? Just wait. You will see lots of ads by Diablo or NEC or whoever buys the mechanism in quantity. Then just go to your dealer and buy one!

Our faces are red. In the December issue, on page 10, the following words appeared: "..enter and save the BEXEC* program listed here..." You guessed it -- there was a long listing with the article, but no BEXEC*. Discovered too late to insert this month, the missing BEXEC* will be printed next issue. If you are too anxious to wait another month, send us a self addressed stamped envelope and we will mail the listing to you right away.

The Canon Printer story brings up an interesting point. What to do if you need a letter-quality printer NOW. It looks like whatever one you buy may be rendered thoroughly obsolete within the year. Or maybe not. Other seemingly brilliant products have hit the market with fatal flaws, or never hit it at all, after very promising announcements. So do you spend your money for something you are pretty sure will be outdated before the new wears off? Or struggle along with something which doesn't really do the job while you wait for a new product which may never show at all, or may be late, or may not be as good as it sounds?

Actually, everyone in the computer game is in that situation all the time. Buy an IBM PC, then out comes Macintosh. Buy a Mac, and IBM announces the expected super-PC multi-user system. Buy that, and the Japanese will jump in with a 5th generation "knowledge-based" system.

What we all have to do, every time, is buy something which will do the job we need done, at a price which is fair, and then when the inevitable better system comes out next month, console ourselves with the thought that when it comes time to replace our system, we will get the one which will render THEIRS obsolete.

al

DEPRECIATION ANALYSIS

By: Robert S. Baldassano
4045 Ashbrook Circle
San Jose, CA 95124

In 1982 I was looking for a good depreciation program for my OSI 8PDF, to help me with my income tax. I came upon a program written for a TRS-80 Model I in the December 1982 issue of Popular Computing that looked like it would do the trick if I converted it.

I should know better, because every time I go about converting a program written for another computer, I usually wind up re-writing a major portion of it, as I either want to add features that the original author did not include, or I find that his coding is not efficient or worse yet I find errors in concept or coding.

In this case it was all of the above, admittedly in stages, with a healthy review of IRS rules and regulations to boot.

To be fair, Mr. Gitt's program was basically a good one giving a broad range of options and swiftly doing a task that could be tedious at best, thus my first attempt at re-coding was a straight forward conversion from TRS-80 Basic to OSI Basic. This was a three track program that ran OK, but the program had a lot of unnecessary comparisons of the type IF C > zzz THEN etc. So I worked on condensing those to the minimum, also eliminating unnecessary PRINT routines along the way.

Now things looked better and it took up fewer tracks, but I was still not satisfied. Depreciation is primarily allowed for equipment used in business, even if it is part time like writing this article. It is also allowed for other income generating uses such as

published monthly

Editor - Al Peabody
Technical Editor - Brian Hartson
Circulation & Advertising Mgr. - Karin Q. Gieske
Production Dept. - A. Fusselbaugh, Ginny Mays

Subscription Rates

US (surface)	$15
Canada & Mexico (1st class)	$23
So. & Cen. America (Air)	$35
Europe (Air)	$35
Other Foreign (Air)	$40

All subscriptions are for 1 year and are payable in advance in US Dollars.

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

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


```
10 REM DEPRECIATION PROGRAM REV2, COPYRIGHT (C) 1983 BY BOB BALDASSANO
12 REM ALL RIGHTS RESERVED.
15 REM THIS VERSION IS WRITTEN FOR OSI SYSTEMS USING OS65D AND IS
20 REM BASED ON PROGRAM BY a.P.(TONY GITT),DEC'82 POP COMPUTING
22 REM ALL ERRORS OF THAT VERSION CORRECTED,PROGRAM EXPANDED, AND
24 REM UPDATED FOR 1983 TAX RETURN RULES 11/83. DISK FILE NAME DEPANL
40 CLEAR:DIMD(30):PRINT!(20):PRINT&(8,15)"TAX DEPRECIATION"
50 FOR I=1TO2500:NEXT:PRINT!(21):U$="######.##"
60 DEF FNA(X)=INT(100*X+.5)/100
100 REM INPUT DATA
110 PRINT:INPUT"NAME OF ITEM";I$
120 A=0:B=0:F=0:F1=0:D(1)=0:A9$=""
130 INPUT"WHAT IS TODAYS DATE (MM/DD/YY)";DP$
140 INPUT"PURCHASE DATE (MM/DD/YY)";D$
142 INPUT"COST OF ITEM";OC
144 INPUT"IS IT USED FOR BUSINESS";AP$
146 IF ASC(AP$)<>89 THEN BA=0:GOTO152
150 AP$="BUSINESS":BA=1:GOTO154
152 INPUT"APPLICATION";AP$
154 PRINT"PERCENT USED IN THIS "AP$;:INPUT" APPLICATION";PU
155 PU$=STR$(PU)+"%"
156 C=FNA(OC*PU/100)
160 PRINT"DO YOU WISH TO DISREGARD THE ACCELERATED COST"
170 INPUT"RECOVERY SYSTEM (ACRS)";A9$
180 GOTO340
185 INPUT"WAS THE ITEM ORIGINALLY PURCHASED FOR BUSINESS";Z$
187 BU=0:IF LEFT$(Z$,1)="Y"THEN BU=1
190 A=1:INPUT"USEFUL LIFE IN YEARS";L
192 INPUT"STRAIGHT LINE OR  DECLINING BALANCE METHOD";Z$
194 IF LEFT$(Z$,1)<>"S"THEN 200
196 R=100:IFL=80RL=10ORL=12THEN SV=0:GOTO210
198 INPUT"WHAT IS THE SALVAGE VALUE";SV:GOTO210
200 INPUT"DEPRECIATION RATE IN % (NEW =200: USED = 150)";R
210 R$=STR$(R)+"%"
220 REM CALCULATE YEARLY RATE
230 YR=R/(100*L)
240 REM CALCULATE EXTRA FIRST YEAR DEPRECIATION
250 ED=0:IF BU THENIFL>6THEN ED=.2*C
260 CV=C-ED-SV
270 REM CALCULATE INVESTMENT CREDIT
280 IFL<3THENIC=0
290 IFL>=3THENIC=C/30
300 IFL>=5THENIC=C/15
310 IFL>=7THENIC=C/10
320 GOTO 410
330 REM DETERMINE YEAR/DAY/MONTH
340 MM=VAL(LEFT$(D$,2)):DD=VAL(MID$(D$,4,2)):YY=VAL(RIGHT$(D$,2))
370 FY=12-MM:IFDD=<15THEN FY=13-MM
380 LY=12-FY
390 IFYY<810RA9$="Y"THEN185
395 GOTO1370
400 REM FIRST YEAR DEPRECIATION
410 CY=1:D(CY)=CV*YR*(FY/12):IFR<>100THENCV=CV-D(CY)
440 REM MIDDLE YEAR DEPRECIATION
450 FORCY=2TOL:D(CY)=YR*CV:IFR<>100THENCV=CV-D(CY)
460 NEXT
490 REM LAST YEAR DEPRECIATION
500 IFCY=0THEN 540
510 CY=L+1:D(CY)=YR*CV*(LY/12):IFR<>100THENCV=CV-D(CY)
520 IFR=100THEN IFL<>80RL<>10ORL<>12THENCV=SV
530 REM OUTPUT DATA TO VIDEO SCREEN OR PRINTER
540 INPUT"OUTPUT TO PRINTER INSTEAD OF VIDEO";Z$
542 DV=2:DW=64:LF=0:IF LEFT$(Z$,1)="Y"THEN DV=1:DW=80:LF=8
543 T=INT((DW-(LEN(I$)+26))/2)
544 IFDV=2THEN PRINT!(21):PRINT:PRINT
545 IFDV=1THENPRINT#DV:PRINT#DV
550 PRINT#DV,TAB(T);"DEPRECIATION ANALYSIS FOR ";I$:PRINT#DV
560 PRINT#DV,TAB(40+LF);"DATE PREPARED: ";DP$:PRINT#DV
570 PRINT#DV,TAB(LF);"INPUT DATA:":PRINT#DV
580 PRINT#DV,TAB(LF);"ITEM NAME: ";TAB(30);I$
582 PRINT#DV,TAB(LF);"DATE OF PURCHASE: ";TAB(30);D$
584 PRINT#DV,TAB(LF);"COST OF ITEM: ";TAB(30);"$";OC
585 PRINT#DV,TAB(LF);"APPLICATION: ";TAB(30);AP$
587 PRINT#DV,TAB(LF);"APPLICATION RATE:";TAB(30);PU$
605 PRINT#DV,TAB(LF);"APPLICATION BASIS:";TAB(30);"$";C
606 IF(BA=0)OR(YD<3)THEN610
607 PRINT#DV,TAB(LF);"BASIS AFTER INV. CREDIT : "TAB(30);"$";RB
610 PRINT#DV,TAB(LF);"USEFUL LIFE: "L" YEARS"
620 PRINT#DV,TAB(LF);"DEPRECIATION RATE:";TAB(30);R$:PRINT#DV
640 CY=1:Y=1899+YY:IF DV=2THENINPUT"CONTINUE";Z$:PRINT!(21)
670 PRINT#DV:IFB=1THEN740
680 PRINT#DV,TAB(LF);"DEPRECIATION CALCULATIONS:":PRINT#DV
690 PRINT#DV:IFBATHENPRINT#DV,TAB(LF);" YEAR";TAB(10+LF);"INVESTMENT";
695 IFBATHEN PRINT#DV,"CREDIT";
700 IF A=1THENPRINT#DV,TAB(35+LF)"EXTRA FIRST YEAR DEPRECIATION"
710 IF A=0THENPRINT#DV
720 IFBATHEN PRINT#DV,TAB(LF);Y+CY;
725 IFBATHEN PRINTUSINGU$$#DV,TAB(13+LF);"$"FNA(IC);
730 IF A=1THENPRINTUSINGU$$#DV,TAB(44+LF)"$"FNA(ED):GOTO740
735 PRINT#DV,TAB(35+LF)CHR$(10)
740 PRINT#DV:PRINT#DV,TAB(LF)" YEAR";TAB(10+LF);"DEPRECIATION":PRINT#DV
750 IFR1=1THEN860
760 IFF1=1THEN800
770 IFF=1THEN790
780 IFA<>1THEN800
790 IFLY<>0THEN810
800 FORCY=1TOL:GOTO820
810 FORCY=1TOL+1
820 PRINT#DV,TAB(LF);Y+CY;:PRINTUSINGU$$#DV,TAB(11+LF);"$"FNA(D(CY))
830 IFCY<>10ORCY<>20ORCY<>30THEN870
840 IFDV=2THENINPUT"CONTINUE";Z$:PRINT!(21)
850 R1=1:PRINT#DV:GOTO740
860 R1=0
870 NEXT:IFCV=<0THEN925
880 PRINT#DV:IFA<>1THEN925
```

Continued

investing. But if you also use the computer to play games or do other non-income generating chores (with the possible exception of educational expenses in some cases), you have to calculate the business use as a percentage of the total use and deduct depreciation using what the IRS calls an adjusted basis.

Mr. Gitt mentioned this in his article, but did not allow for it in his program, assuming 100% business use. Well when you have a computer, why do these things by hand? So I added an automatic basis calculation. That was OK, but now I compounded my problem as his program also computed such things as investment credit and first year expensing; calculations only allowed for business use, not investment or educational use. So now I had to trap out these calculations.

As I was doing this -(remember the third bugaboo, error in concept or coding?) - I found that Mr. Gitt had not followed IRS convention for use of straight line depreciation when using the Alternate Accelerated Cost Recovery System (ACRS). He used the old method where depreciation is allocated by months of service in the first year, and not the required half-year convention that says you are only allowed one-half of the years depreciation in the year you place it in service, and you get the other half in the year following the recovery period. If anyone used his program for his 1982 tax and used the alternate ACRS, they would have had an error in their return, and a nasty note from the IRS.

After all this I had a nice program to do my 1982 depreciation calculations with, but if I had done it by hand, I would have had it done sooner! Ah, but I had a jump on 1983 taxes and beyond I said.

That was almost correct. When I got my J. K. Lasser's tax guide for my 1983 return this year, the IRS did it to me again -(Sob)- they changed the rules on investment credit for equipment placed in service in 1983 and beyond! This amounted to a reduction of the properties depreciable basis by 50% of the credit claimed.

Not to worry, I added this to my program too, and cleaned it up some more, so now YOU can do all this neat depreciation without having to do more than punch a few buttons after you type in the enclosed program. For those of you who find that too taxing (HeHe), I will provide an 8 inch disk with the program on it suitable for video systems for $10.

For those who want to know more on the subject, the IRS publishes a number of good pamphlets on depreciation and other business deductions.

```
910 SV$="SALVAGE VALUE AT END OF"
920 PRINT#DV,TAB(LF);SV$;CY+Y-1;"IS $";:PRINTUSINGU$#DV,FNA(CV)
925 PRINT:PRINT:IFDV=1THENPRINT#DV,CHR$(12)
930 DV=2:INPUT"ANOTHER ANALYSIS";Z$:IFLEFT$(Z$,1)<>"Y"THENRUN"BEXEC*
945 GOTO40
1360 REM DETERMINE YEAR/TYPE DEPRECIATION
1370 PRINT!(21):PRINT"PLEASE SELECT DESIRED DEDUCTION/DEPRECIATION":PRINT
1380 PRINTTAB(10)"<1> SEC. 179, MAX. SHORT TERM DEDUCTIONS/DEPRECIATION
1385 PRINTTAB(10)"<2> STANDARD ACRS ACCELERATED DEPRECIATION/DEDUCTION
1390 PRINTTAB(10)"<3> ALT. ACRS METHOD LONG TERM DEPRECIATION"
1400 PRINT:INPUT"YOUR SELECTION NUMBER";Z7
1405 IFZ7<1ORZ7>3THEN1370
1406 IFZ7=1AND BA<>1THENPRINT"ONLY BUSINESS PROPERTY ELIGIBLE":GOTO1380
1410 YD=YY-80:ON YD GOTO1470,1500,1500,1610,1610
1450 IFYY>85THENGOTO1720
1460 REM 1981 DEPRECIATION
1470 ED=0:C1=C:IFZ7<>3THEN1960
1480 GOTO2260
1490 REM 1982/83 DEPRECIATION
1500 ED=0:C1=C:ONZ7GOTO1520,1960,2260
1520 IFC>5000THEND(1)=5000:C1=C:C=C-5000:GOTO1960
1560 R$="100% WRITE OFF":IFC<5001THEND(1)=C:B=1:GOTO540
1600 REM 1984/85 DEPRECIATION
1610 ED=0:C1=C:ONZ7GOTO1630,1960,2260
1630 IFC>7500THEND(1)=7500:C1=C:C=C-7500:B=1:GOTO1960
1670 IFC<7501THEND(1)=C:R$="100% WRITE OFF":B=1:GOTO540
1710 REM 1986 AND BEYOND DEPRECIATION
1720 ED=0:C1=C:ONZ7GOTO1740,1960,2260
1740 IFC>10000THEND(1)=10000:C1=C:C=C-10000:B=1:GOTO1960
1780 IFC<10001THEN D(1)=C:R$="100% WRITE OFF":B=1:GOTO540
1820 REM 5 YEAR ASSET COST RECOVERY STSYEM
1830 IFBATHEN IC=C*.1:IFYD>2THENC=C-.5*IC:RB=C
1835 CY=1:F1=1:D(CY)=D(1)+C*.15:CY=2:D(CY)=C*0.22
1870 FORCY=3TO5:D(CY)=C*.21:NEXT:CV=0:F=1:C=C1
1930 R$="ASSET COST RECOVERY SYSTEM":GOTO540
1950 REM ACCELERATED USEFUL LIFE SELECTION
1960 PRINT!(21):PRINT"SELECT TYPE OF LIFE USE FROM THE FOLLOWING LIST:
1970 PRINT:PRINTTAB(10)"<1> RESEARCH AND DEVELOPMENT - 3 YEAR LIFE"
1980 PRINTTAB(10)"<2> NORMAL BUSINESS USE - 5 YEARS"
1990 PRINT:INPUT"WHAT IS THE NUMBER OF YOUR SELECTION";Z
2010 L=INT(Z*2+1):IFZ>2ORZ<1THENGOTO1960
2020 ON Z GOTO 2170,1830
2040 REM ALT ACRS STRAIGHT LINE DEPRECIATION WITH HALF-YEAR CONVENTION
2050 C1=C:IFBATHEN IC=C*.1:IFYD>2THENC=C-.5*IC:RB=C
2055 ED=0:F=1:CY=1:HY=.5:DP=C/L:D(CY)=DP*HY:FORCY=2TOL:D(CY)=DP:NEXT
2060 CY=L+1:D(CY)=DP*HY:C=C1:R$="ALT ACRS STRAIGHT LINE":GOTO540
2160 REM 3 YEAR ACRS
2170 IFBATHEN IC=C*.06:IFYD>2THENC=C-.5*IC:RB=C
2175 F1=1:D(1)=D(1)+C*.25:D(2)=C*.38:D(3)=C*.37:CV=0:F=1
2180 R$="ASSET COST RECOVERY SYSTEM":C=C1:GOTO540
2240 END
2250 REM ALT ACRS LONG TERM USEFUL LIFE SELECTION
2260 PRINT!(21):PRINT"SELECT TYPE OF USEFUL LIFE FROM THE FOLLOWING :"
2270 PRINT:PRINTTAB(10)"<1> ALT ACRS TERM - 5 YEARS"
2280 PRINTTAB(10)"<2> ALT ACRS LONG TERM - 12 YEARS"
2285 PRINTTAB(10)"<3> ALT ACRS LONG TERM - 25 YEARS"
2290 PRINT:INPUT"WHAT IS THE NUMBER OF YOUR SELECTION";Z
2295 IFZ>3ORZ<1THENGOTO2260
2300 L=INT((Z-1)*12.5):IFZ=1THENL=5
2310 GOTO2050
```

★

TANDON 848-1 TO C4P

By: L. Z. Jankowski
Otaio Rd 1
Timaru, New Zealand

Ye Ed asked out-of-the box experiences (see PEEK 12/83). Well here's one which began as a horror story but has a happy ending.

I bought a TANDON 848-1 8" drive to go with my converted Superboard. A year later, it was running! I bought the drive on the strength that it possessed the Standard Industry Interface (the big discount helped too), and the assurance that the 'Tasker Bus Motherboard System' disk controller would drive it. From the beginning, the problem was that the radial track alignment on the drive was out (clunk, clunk, clunk on track zero). But I did not know that at the time. Was the data-separator faulty? No, it ran fine in another system. So did the disk-controller. The power supplies were investigated for ripple and checked out OK. Everything that could be taken apart and checked was checked. The drive was tested with a TRS-80 III, no problems! Well, I gave up and put the drive back on the shelf. Months went by. The service manual arrived! The first copy was lost in the post. The service manual revealed vital information, like the misleading strapping information that comes with the drive (see below). Once the strapping was right. the data separator pots were set midway and the drive booted - every so often. It was

noticed that the drive could not find a track when it went back to the directory for a track number. Aha! Seventy dollars and a week later, it was at long last, fully operational!

Settings the pots on the data separator was a breeze, virtually any setting worked. But more precisely, find the two extreme settings of the pot at which the drive will boot, then set the pot midway between these two points. The settings are important (2.6 and 2.9 usec) but more important is that one timing should be slower than the other (.3 of a usec).

The TANDON 848-1 is a belt-drive and is not microprocessor controlled. This is in contrast to the 848E direct-drive which is microprocessor controlled, and therefore probably difficult to interface to OSI hardware.

This is the strapping I did to make my drive work, (see diagram):

1. Strap DS1 - drive select 1.

2. Programmable shunt U3, open 1-16 and 2-15. Leave all others closed.

3. Close DS.

4. Close Y.

5. Cut M1 to remove 20 second delay.

6. Remove 2-pin micro-switch socket, situated immediately behind diskette lever. Again, this is to remove the irritating 20 second delay before the drive stops.

7. The drive can be stopped in software by -

(I) POKE 49154,0
(II) Selecting another drive

Throwing a switch is quicker.

Run two wires from 3-14 on U3 to a small switch. I placed the switch on the front of the drive, in the top right-hand corner. (Drill a hole to take the switch).

The switch selects or de-selects the drive in drive hardware. No ill effects have been noticed.

A possible alternative is to use the Motor Control lines.

The 848-2 is a two sided drive. OSI software can see it as drive A and drive B. A little more work is involved here - see the service manual. You must remove the resistor pack (the blue IC by the connector) in drive 1, if using 2 physically separate drives.

In spite of my experiences, if things don't work, it is still a good idea to first check the power supply:

(I) Good grounding is essential.

(II) The drive surge-current requirement is 2 1/2 amps at 24v.

(III) A voltage regulator must have at least 3v difference between input and output voltages. Check the level of the AC ripple.

To convert from 1MHz to 2MHz operation is straightforward. Even though the computer/disk controller/drive will read at 2MHz (separate Rx clock), it will not write. One strap change is required on the Tasker disk-controller board: the disk-controller Tx clock line must be restrapped to give 250KHz. (The 02 clock line is changed to 2MHz on the computer). This could be checked with a frequency meter.

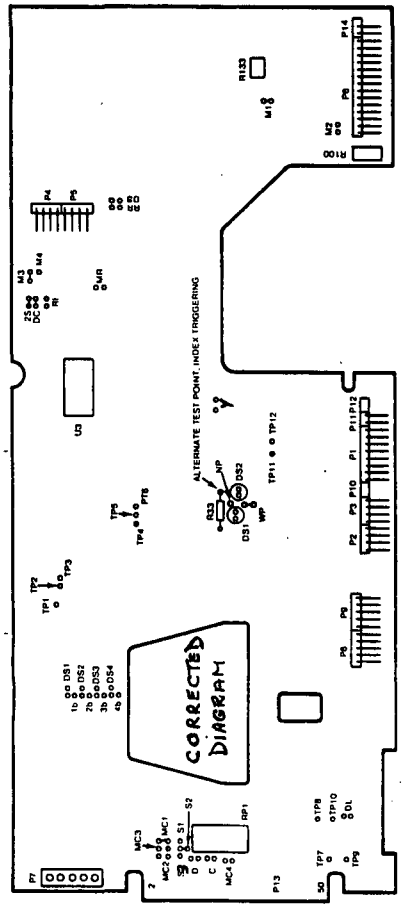


Figure 1-6
Circuit Board Assembly (C)

Finally, the shipping disk and Manual circuit board assembly diagram, contain omissions and errors with regard to 'Y', 'S2', 'MC1'. The shipping disk has DS marked incorrectly as (U3,2-15): (U3,2-15) is HL.

★

A REAL TIME CLOCK

Courtesy of TOSIE
Toronto Ohio Scientific Idea Exchange
P. O. Box 29
Streetsville, Ont.
Canada L5M 2B7

Here is a no hardware (almost) real time clock for Superboard. As described here it will work with the OSI monitor only, but by using different memory locations to hold the hours, minutes, and seconds you can make it compatible with any system.

As a bonus, there is a demonstration of a method of tacking a machine language load right onto a BASIC load. Even more, there is a demo of an auto-run routine, and a way to have the computer go back to the program after doing a LIST. Usually it gives you the O.K. message after doing a LISTing.

HOW IT WORKS:

A wire is run from the TV sync signal to the NMI (Non Maskable Interrupt - see a 6502 manual for more information) pin of the 6502. Every sixtieth of a second this pulse interrupts the program and increments the clock as needed. You will not notice that this is being done as the machine code does this very quickly. As a matter of fact, I did not even have to turn it off during cassette saves or loads, however, this might be a wise precaution.

WHAT TO DO - HARDWARE:

Run a wire from U65 pin 4 through an SPST switch to pin 2 of the expansion connector. Leave the switch OFF until the software is entered (otherwise the interrupt tries to execute a program that isn't there yet). That must be the easiest modification to your machine you have ever done! If you can't find U65, a 74LS123 chip, get help with this work. It will only take a minute.

SOFTWARE:

Use the OSI or other monitor

to enter the code as in the block dump, or as you have modified it. Type in the BASIC, both the clock demonstration, which will prove it all works, and the portion at line 50000 on, that makes an auto machine language loader.

You can test the clock (after all the code is in) by flipping the switch to on, and typing RUN. Answer the questions to reset the time, and watch the time displayed on the screen. Of course, you will want to develop your own software to use this new feature on your machine. Maybe a timed skill test, checking the speed of a routine or other uses such as controlling the lights via a BSR or other interface are possible.

To make a self loading copy of this program AND the machine code from $0130 to $0173, all you have to do is type: RUN 50000, press record, and then press RETURN. An autoload version of the program will be made. Don't turn off the recorder until the OK message. You will not see anything happening on the screen while the ML is being written to tape. The Basic routine used would be too slow.

SOFTWARE EXPLANATION:

Memory locations $E5 to E0 are reserved for the real time clock. Put these elsewhere for ROMTERM or other monitors that use this area of memory. $E4 and $E5 are used to temporarily save the X and A register during the interrupt. $E3 stores the number of sixtieths of a second. Retrieve it by a PEEK to location 227 (decimal).

$E2 stores the number of seconds. PEEK location 226. $E1 stores the number of minutes. PEEK location 225. $E0 stores the number of hours. PEEK 224.

The machine code at $0130 up, basically stores the A and X registers, increments $E3, the sixtieths counter. If this is 60, then it resets this to zero, but adds one to the seconds counter,...and so on for the minutes and hours.

I have added two places to adjust the clock. I found that my clock was running quite slow. This is due to the fact that the video sync is only approximately a sixtieth of a second, as derived from the computer's crystal. The 01 at $0148 adds an extra sixtieth of a second every

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0130  85 E4 86 E5 A9 3C A2 00 E6 E3 C5 E3 D0 26 86 E3
0140  E6 E2 C5 E2 D0 1E 86 E2 A2 01 86 E3 E6 E1 C5 E1
0150  D0 12 A2 01 86 E3 A2 00 86 E1 A9 18 E6 E0 C5 E0
0160  D0 02 86 E0 A5 E4 A6 E5 40 A9 00 8D 03 02 20 77
0170  A4 4C C2 A5
:Q0130
0130 85E4    STA $E4
0132 86E5    STX $E5
0134 A93C    LDA #$3C
0136 A200    LDX #$00
0138 E6E3    INC $E3
013A C5E3    CMP $E3
013C D026    BNE $0164
013E 86E3    STX $E3
0140 E6E2    INC $E2
0142 C5E2    CMP $E2
0144 D01E    BNE $0164
0146 86E2    STX $E2
0148 A201    LDX #$01
014A 86E3    STX $E3
014C E6E1    INC $E1
014E C5E1    CMP $E1
0150 D012    BNE $0164
0152 A201    LDX #$01
0154 86E3    STX $E3
0156 A200    LDX #$00
0158 86E1    STX $E1
015A A918    LDA #$18
015C E6E0    INC $E0
015E C5E0    CMP $E0
0160 D002    BNE $0164

0162 86E0    STX $E0
0164 A5E4    LDA $E4
0166 A6E5    LDX $E5
0168 40      RTI
0169 A900    LDA #$00
016B 8D0302  STA $0203
016E 2077A4  JSR $A477
0171 4CC2A5  JMP $A5C2
0174 60      RTS
```

```
0 POKE515,0
5 FORI=1TO30:PRINT:NEXT
10 REM CLOCK DRIVER
20 A=224:B=A+1:C=A+2:D=A+3:SC=53419
22 INPUT"RESET TIME Y/N";N$
23 IFN$="N"THEN 40
25 PRINT:PRINT
31 PRINT:INPUT"TIME IN H,M,S";H,M,S
35 POKEA,H:POKEB,M:POKEC,S:POKED,0
40 FORI=1TO30:PRINT:NEXT
42 H$="TIME:":FORI=1TOLEN(H$)
43 POKE53414+I,ASC(MID$(H$,I,1)):NEXT
50 H=PEEK(A):H$=STR$(H):IFH<10THENH$=" 0"+RIGHT$(H$,1)
60 M=PEEK(B):M$=STR$(M):IFM<10THENM$=" 0"+RIGHT$(M$,1)
70 S=PEEK(C):S$=STR$(S):IFS<10THENS$=" 0"+RIGHT$(S$,1)
80 J=PEEK(D):J$=STR$(J):IFJ<10THENJ$=" 0"+RIGHT$(J$,1)
90 T$=H$+M$+S$+J$:FORI=1TO12:POKESC+I,ASC(MID$(T$,I,1)):NEXT:
50000 REM ROUTINE TO RECOVER AFTER LIST                    GOTO50
50010 SAVE:POKE4,194:POKE5,165
50015 LIST:POKE4,195:POKE5,168
50020 PRINT:PRINT"0POKE515,0
50030 REM TURN ON THE MONITOR
50040 PRINT"POKE251,1:POKE11,67:POKE12,254:X=USR(X)
50050 PRINT".0130/";
50060 A1=304:A2=372:GOSUB50100
50090 PRINT".0169G":END
50095 REM WRITES M/L TO TAPE
50100 FORA=A1TOA2
50110 OP=PEEK(A):H=INT(OP/16):L=OP-16*H
50120 H=H+48-7*(H>9):L=L+48-7*(L>9)
50130 WAIT61440,2:POKE61441,H
50140 WAIT61440,2:POKE61441,L
50150 WAIT61440,2:POKE61441,13
50160 NEXTA:RETURN:
```

D&N MICRO PRODUCTS, INC.

3702 N. Wells St.
Fort Wayne, Ind. 46808
(219) 484-6414

TERMS $3.00 shipping. Foreign orders add 15%, Indiana residents add 5% sales tax.

COMPUTER

MICRO-80 COMPUTER

Z-80A CPU with 4Mhz clock and CP/M 2.2 operating system. 64K low power static memory. Centronics parallel printer port. 3 serial ports. 4" cooling fan. Two 8" single or double sided floppy disk drives. IBM single density 3740 format for 243K or storage, double density format for 604K of storage. Double sided drives allow 1.2 meg on each drive. Satin finish extruded aluminum with vinyl woodgrain decorative finish. 8 slot backplane, 48 pin buss compatible with OSI boards.

MODEL 80-1200 **$2995**
2 8" Single sided drives

MODEL 80-2400 **$3495**
2 8" Double sided drives

MICRO-65 COMPUTER

6502 CPU with 2Mhz clock and DOS-65 operating system. 48K of low power static memory. 2 serial ports and 1 Centronics parallel port. 2 8" single or double sided drives. Satin finish extruded aluminum with vinyl woodgrain finish. 8 slot backplane, 48 pin buss compatible with OSI. Will run OSI 65D and 65U software.

MODEL 65-1 **$2995**
2 8" Single sided drives

MODEL 65-2 **$3495**
2 8" Double sided drives

BP-580 8 Slot Backplane ..... **$ 47**
OSI 48 pin Buss compatible

MEM-CM9 MEMORY/ FLOPPY CONTROLLER

24K memory/floppy controller card uses 2114 memory chips, 1 8K and 1 16K partition. Supports OSI type disk interface

24MEM-CM9 ..... **$325**
16MEM-CM9 ..... **$260**
8MEM-CM9 ..... **$180**
BARE MEM-CM9 ..... **$ 50**
Controller on assembled unit add ..... **$ 90**

BIO-1600 Bare IO card ..... **$ 50**
Supports 8K of memory, 2 16 bit parallel ports, 5 serial ports, with manual and Molex connectors.

PRINTERS

Okidata

ML82A, 120 cps, 10" **.$409**
ML83A, 120 cps, 15" **.$895**
ML84 Parallel, 200 caps, 15" **.$1150**

C. Ioth

8510AP Prowriter, parallel ... **$419**
120 cps, correspondence quality
8510APD Prowriter, serial **$585**
F10-40PU Starwriter, parallel **$1319**
Letter quality daisy wheel
F10-40RU Starwriter, serial .. **$1319**
F10-55PU Printmaster **$1610**
parallel, Letter quality daisy wheel
F10-55RU Printmaster, serial **$1610**

DISK DRIVES AND CABLES

8" Shugart SA801 ..... **$385**
single sided
8" Shugart SA851 **$585**
double sided
FLC-6 6 ft cable from D&N **$69**
or OSI disk controller to 8" drive
5¼" MPI B51 disk drive with .. **$450**
cable, power supply and cabinet. Specify computer type.
FLC-5¼ cable for connection . **$75**
to 5¼ drive and D&N or OSI controller, with data separator and disk switch. Specify computer type

HARDWARE
OSI COMPATIBLE

IO-CA10X Serial Printer Port .. **$125**
Specify Device #3 or #8
IO-CA9 Parallel Printer Port .. **$150**

CMOS-MEM

64K CMOS static memory board, uses 6116 chips, 3 16K, 1 8K and 2 4K blocks, Partitionable for multi-user, OSI type disk controller, 2 IO mapped serial ports for use with D&N-80 CPU. Ideal way to upgrade from cassette to disk.

64K CMOS-MEM **$490**
48K CMOS-MEM **$390**
24K CMOS-MEM **$250**
16K CMOS-MEM **$200**

Controller add . **$ 90**
2 IO mapped serial ports add. **$125**
on assembled memory board
Z80-IO 2 IO mapped serial **$160**
ports for use with D&N-80 CPU card
FL470 Disk Controller **$155**
Specify 5¼ or 8" drive

STANDARD CP/M FOR OSI

D&N-80 CPU CARD

The D&N-80 CPU allows the owner of an OSI static memory computer to convert to Industrial Standard IBM 3740 single density disk format and CP/M operating system. Double density disk operation is also supported for 608K of storage on an 8" diskette. When used with a 5¼" disk system 200K of storage is provided. Includes parallel printer and real time clock. Also available for polled keyboard and video systems. Compatible with C2, C3, C4 and 200 series OSI computers.

D&N-80-P **$349**

CP/M 2.2 ..... **$150**

64K CMOS-MEM with D&N-80 CPU card **$450**

HARD DISK DRIVER **$140**
Allows D&N-80 CPU board to control OSI 40 or 80 meg hard disk unit. Will not destroy OSI files. Will also allow for a true 56K CP/M system. Specify 40 or 80 meg drive.

BUSS TRANSFER **$135**
Allows for D&N-80 and OSI CPU to be in the computer at the same time. Toggle switch provides for alternate CPU operation.

DISK TRANSFER **$100**
Utility program to transfer OSI CP/M format disk to IBM 3740 single density format. Will also transfer IBM to OSI format.

SYSTEM HARDWARE REQUIREMENTS

D&N-80 CPU, D&N FL470 or OSI 470 controller, 48K memory at 0000-BFFF, 4K memory at D000--DFFF, two disk drive cables.

FORMAT TRANSFER **$15**
You supply software on 8" diskette D&N will transfer OSI CP/M format to IBM 3740 CP/M format. Can also transfer IBM 3740 CP/M format to OSI CP/M format. Original diskette returned.

minute. You can increase this if your clock is too slow. There is another Ø1 at $Ø153. This adds a sixtieth of a second every hour, and can be changed for really fine adjustments. With these two values at Ø1 my clock ran as accurately as my LCD wristwatch. Your crystal frequency may be a little different, so adjustments may be needed.

Exit from the routine is at $Ø164, by reloading the X and A registers, and doing an RTI, return from interrupt, instruction.

The Basic program from Ø to 9Ø is quite straightforward. It is a demonstration to put the clock on the screen. Line Ø makes sure the computer is no longer in SAVE. A, B, C, and D are memory locations 224, 225, 226, and 227 that store hours, minutes, seconds, and sixtieths. These are PEEKed, converted to strings, and then POKEd onto the screen in line 9Ø. A simple routine.

The autoloader routine is a little trickier. As you know, the normal response after LIST is for the computer to say OK. If we want another statement to be executed, we will have to point the vector at location 4 and 5 to a CONT command instead. This is what happens in line 5ØØ1Ø. Let's start again.

5ØØ1Ø The SAVE sets the cassette output pointer, but before LIST is executed in the next line, the program is pointed at the CONT code by POKEing the appropriate address into locations 4 and 5. The computer will now CONTinue instead of saying OK.

5ØØ15 LIST is executed, but the program continues. First we will reset the pointer back to its original setting. (Note: - change this for ROMTERM or you will crash).

5ØØ2Ø puts line Ø in. POKE 515,Ø turns off the LOAD.

5ØØ4Ø This puts the following commands onto the tape. They will be executed as if you had typed them from the keyboard. If the LOAD flag is set, the cassette input can operate the computer as if it was the keyboard. The machine doesn't know the difference. This gimmick saved OSI a lot of monitor space. Immediate commands executed from tape: POKE 251,1 same as L, to set monitor load. The POKES to 11 and 12 set up the user vector to the monitor, which can be entered at $FE43. The X=USR(X) does that.

The rest of the program writes the ML to the tape. If you look you will see that it is identical to you entering the code manually, except that the entering device is the tape - via the above mentioned gimmick.

5ØØ5Ø set the memory to $Ø13Ø, the / starts data entry.

5ØØ6Ø Data from 3Ø4 decimal ($Ø13Ø hex) to 372 ($Ø173) is to be read, converted to hex High and Low bytes by the calculations in 59119 and 5Ø12Ø, then written to the cassette port H byte first, Low next, then a carriage return (13). These things are done in 5Ø13Ø to 5Ø15Ø.

The last thing executed is a .Ø169G (see line 5ØØ9Ø). Yes, exactly as if it came from the keyboard, it starts executing at $Ø169. Here is a RUN command executed from a ML program. The code at $Ø169 and $Ø16B, turn off the Load that the monitor was in, and the next two lines of code, at $Ø16E and $Ø171 set up and execute the RUN.

I know that this article is quite a lot to chew if you are a beginner. You may be able to work it through with the help of the various other resources that you have. If not, why not write me a note asking for clarification of this or that point. I'd be glad to explain the details of the above if someone asks. On the other hand, if you all have it figured out there is no sense my wasting valuable space.

Lastly, if you get this running properly, you can, and this would be the ideal solution, put the real time clock into EPROM, and thus it would always be available. It will also avoid the shock you get if you turn on the interrupts (remember the switch?) before putting the software in. Your machine will behave rather strangely! You are forewarned! The cure, of course, is to flip the switch only after entering the software.

★

EXTENDED MONITOR IN EPROM

By: Rolf B. Johannesen
13917 Congress Drive
Rockville, MD 2Ø853

It is very convenient to have an extended monitor (EXMON) program always available. In case of a crash of any kind, it is always possible to go to the extended monitor on rebooting with a minimum of disturbance to the contents of memory. This is true for both cassette systems and disk systems. Rebooting a disk system always rewrites not only the disk operating system (DOS), but also about 24 pages of memory, with the standard BEXEC* program. The error responsible for the crash may well be overwritten in this process. Additionally, if the memory contains a program that had not yet been saved, it will be irretrievably lost upon rebooting. An EXMON is also useful for installing short patches or pieces of hand-assembled code, for examination of memory in various formats, and for setting of breakpoints while debugging a program.

An extended monitor program in EPROM can never be wiped out by any software flaw. By going first to the extended monitor after a crash, diagnostics can be run, including any kind of core dump that appears to be indicated. Programs still in memory can be moved to high core where they will not be disturbed, and the system then booted in the normal way. After booting, the program can be moved back to normal program space and the work in hand before the crash can be continued.

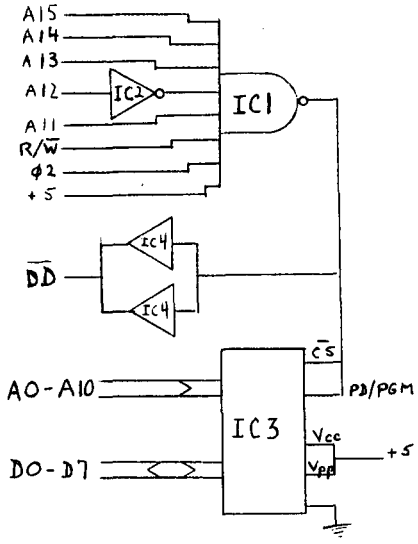
The OSI cassette based extended monitor program is not self-modifying, so it can be successfully transferred to EPROM. In the OSI C1P, memory space from $E000 to $F000 is free, and is not used by most add-on peripherals. I have relocated the OSI EXMON to $E800-$EFFF and burned it into an EPROM. A simple relocation of the code will not work: locations $66 to $15F from the start of code comprise a table which must not be changed. Also, locations $64,65 and $160 to $199 contain pointers to subroutines which must be relocated by the magnitude of the offset. This is not done automatically by the "R" command, since these are pointers and not 3-byte instructions.

The figure shows the wiring diagram that I used. IC1 is a 74LS30 and is used as an address decoder. In addition to address lines A15-A13, A11, and inverted A12; lines R/W and Ø2 of the clock are connected to the inputs of IC1 to select addresses in the range $E800-$EFFF; but only when a

read is wanted and only when the address is valid (Ø2 high). The eighth input to IC1 is tied high (+5v). Obviously, another address could be chosen if desired by the correct choice of lines to be inverted. IC2 inverts A12 to provide the inverted A12 needed. IC3 is the EPROM, a 2516 single supply 2Kx8 chip. My system runs at 1 MHz; in adapting this circuit to a 2 MHz system, it will be necessary to insure that the EPROM can function at the shorter read time. Finally, IC4 is a pair of 7417 open collector drivers to switch the bus buffers from sending (from the CPU) to receiving (by the CPU), by driving line DD low. Two chips in parallel are used to drive DD since the line is quite heavily loaded. For this reason, too, a pull-up resistor is not required at the open collector. This circuit can be connected to either the 6ØØ board or the 61Ø board. The pinouts for the expansion connectors are not the same on these two boards, so I have not given either of them. The circuit is a good candidate for wire-wrap construction. I have rebuilt this circuit 3 times as I have expanded and altered the extras connected to the expansion connector, and it has worked each time. Therefore, I suppose that the wiring arrangement is not especially critical. A source of +5V power is required. Both expansion sockets have one pin unused. It is possible to connect the +5V bus on the board to this unused pin if desired, or a separate +5V and ground supply can be run to the added circuit from another power supply. If you use this with the 6ØØ board, remember that it is necessary to plug 8T28 bus buffers into sockets U6 and U7.

For those interested in the construction of this circuit, I am willing to provide a wire-wrap list if you include a SASE with your request. Be sure to indicate whether you plan to use this with the 6ØØ board or 61Ø board. I hope to be able to offer the EPROMS for sale, if I can get OSI's permission to distribute their copyrighted software in this way. My own EPROM has two changes in the EXMON program which I will mention only briefly here. First, the "J" command restores the locations on page zero that are used by both EXMON and BASIC and returns to the warm start location for BASIC. Otherwise, it is impossible to return to BASIC-In-ROM. The "Z" command now dumps 24 characters in ASCII format. To add these commands, it was necessary to recover a few bytes of space, and the hexadecimal add was taken out. The other hexadecimal arithmetic commands are still present, however.

This article has presented a useful addition to the OSI C1P computer in the form of an EPROM containing the extended monitor program.



★

24K RAM/EPROM CARD FOR THE TASKER BUS

By: David Tasker
111 Bass Highway
Tasmania, Australia 7303

INTRODUCTION

The 24K RAM/EPROM card is a double sided printed circuit board that has been designed as if it were to be manufactured as a plated-through-hole printed circuit board. No thru hole plating is used at this time and results in some small scale integrated circuits being directly soldered into the board with solder points at the top and bottom of the board.

The memory integrated circuits are all socketed. Some 24 pin compatible types are CMOS RAM type 6116, STATIC RAM type 2128, and single supply 2716 EPROMS. For RAM the CMOS 6116 types are recommended as they use the least power. The Data buffers also are socketed to allow for inverting or non-inverting data buffer types to be used. If you are using the board for RAM ONLY, then either 8T28's or 8T26's may be used. If you plan to use this board for EPROMs, then you must check to see what type of data buffers you are currently using within your system. Check U5 and U6 on the C1P or Superboard. As supplied, the board may have either type of buffer as the board is intended primarily as a RAM card.

RAM CARD

The card is pre strapped for RAM either 6116 or 2128. The strap to allow for EPROMs is just to the right of pins 21 on the 24 pin sockets marked R4 and R8.

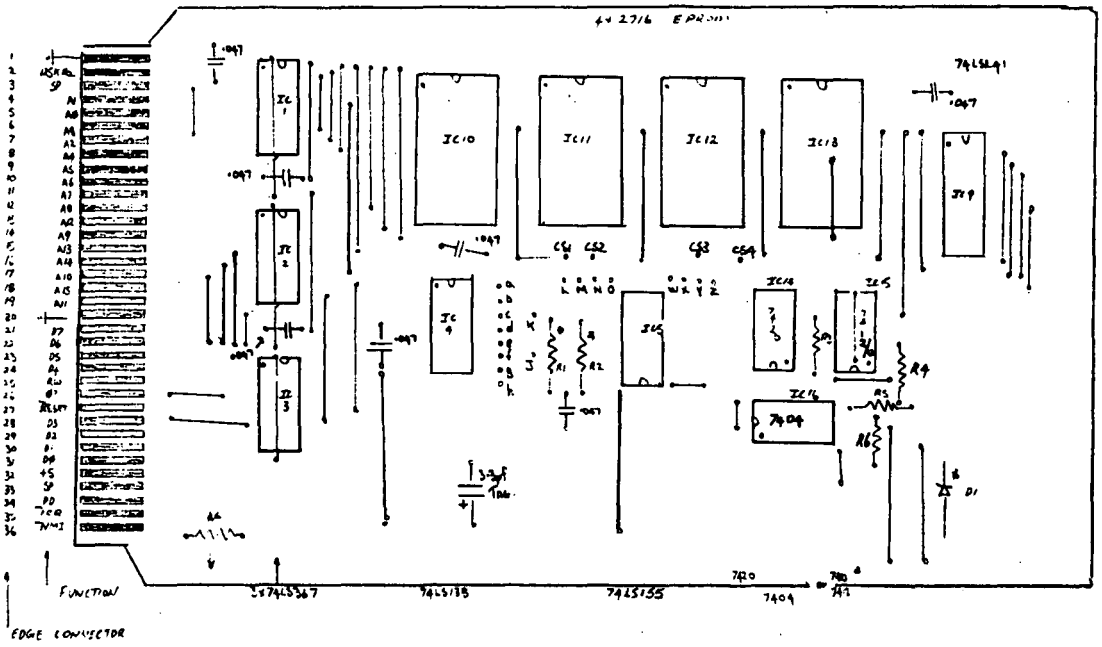
EPROM STRAP

For the EPROM to function correctly, pin 21 of a 2716 should be logic 1 (+5volts). The RAM devices have the R/W (Read/Write) line connected to pin 21. It is possible that a 2716 device will function correctly with R/W connected to this pin as we only read EPROMs and the R/W line is logic 1 during a read.

If you find you get read errors with the EPROM, then you will need to cut the copper link on the underside of the printed circuit board and connect pin 21 to the +5 volt rail. Provision has been made to do this on only 2 rows of 24 pin sockets, i.e. sockets R1 to R4 and R5 to R8. Sockets R9 to R12 should be used only for RAM. However, technically there is no reason that these could not support EPROMs, other than it is a little more difficult to organize the straps.

continued

8K 2716 EPROM BOARD LAYOUT



** Not required ClP * RESISTORS only required if jack not strapped

Note: Even though IC10-13 may not be used, points CS1-CS4 must be tied temporarily to one of the points L thru Z or to +5 via a 1K resistor is better.

* D1 If you wish you may use a 7410 in place of 7412 if 7410 used, then insert Diode D1 in lieu of strap and leave out resistors R4, R6.

PARTS LIST:

2 x 74LS241 U1 U2.
2 x 8T26/28 U3 U4.
1 x 74LS138 U5.
2 x 74LS155 U6 U7.
1 x 74LS133 U8.
1 x 7438 U9.

3 x 1K ohm resistors R1 R2 R3.
15 x 0.1 or 0.047 uf caps' C1 to C15.

2 x 16 pin sockets. U3 U4.
12 x 24 pin sockets U10 to U21 (R1 to R12).

1 x PRINTED CIRCUIT BOARD.

STRAPPING

STANDARD STRAPS

The board is strapped and tested as a RAM only card. It has 11 of the 12 2K blocks enabled. The reason for this is that the equipment used for testing has the BASIC 4 chip still in for the use of the Video Driver in this chip. The BASIC 4 chip occupies memory at B800-BFFF (all addresses used here are Hexadecimal). B800 - BFFF is the top of 48K and this board as provided is for the second 24K of memory from the 25K to 48K, however, as mentioned socket R12 is not strapped in.

This card is provided as shown.

Socket number.

R1	6000 - 67FF
R2	6800 - 6FFF
R3	7000 - 77FF
R4	7800 - 7FFF
R5	8000 - 87FF
R6	8800 - 8FFF
R7	9000 - 97FF
R8	9800 - 9FFF
R9	A000 - A7FF
R10	A800 - AFFF
R11	B000 - B7FF
R12	not strapped.

You will note that the strap point marked R12 is not connected and that point Y4 is also not strapped. This could easily be added for the memory at B800 - BFFF.

IMPORTANT NOTE! - If you have BASIC-IN-ROM fitted to your computer then cut the link M6 to Sy. otherwise this board will clash with the BASIC.

SELECTING YOUR MEMORY

The total addressable memory on the 6502 computer is 64K (actually 65000+) this is divided into 8 x 8K blocks by Integrated circuit U5.

U5 provides 8 outputs each representing an 8K block of memory. These outputs are brought out to the point marked M1 to M8. Shown in detail here.

U5 Pin.

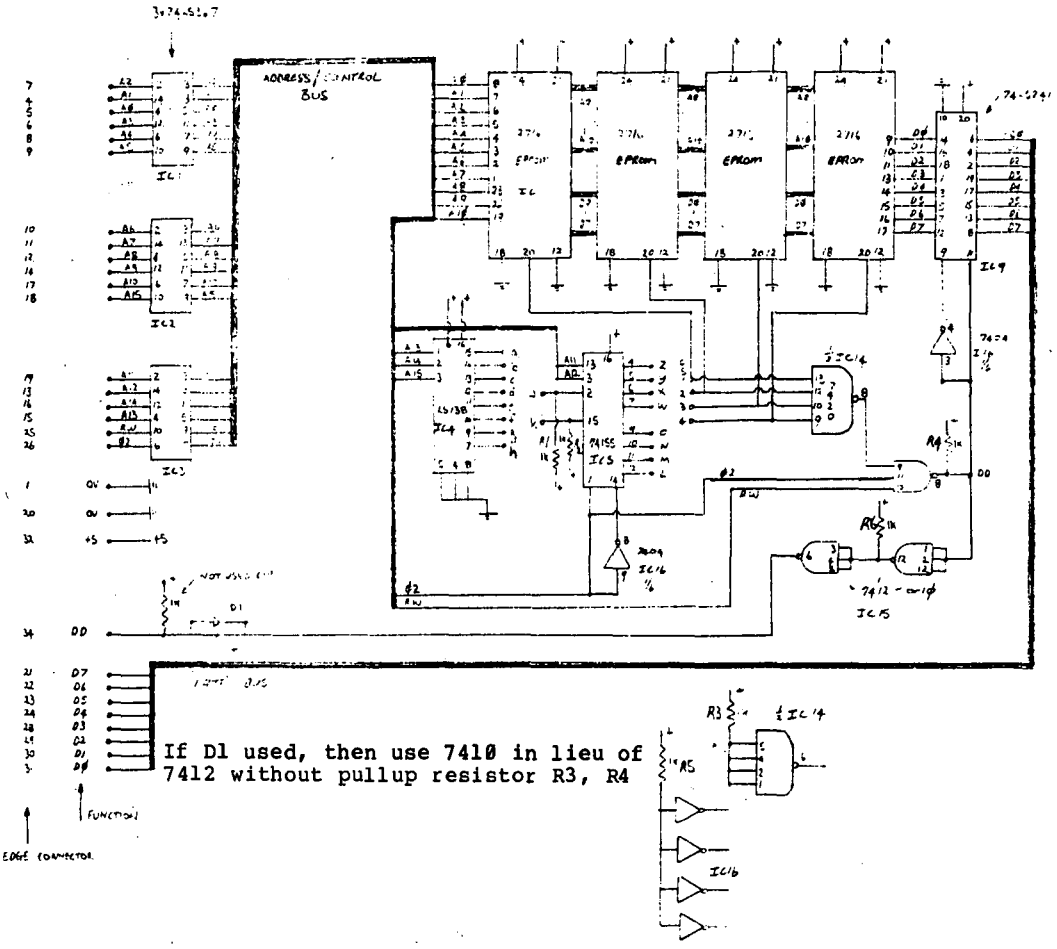
15 M1 8k of memory from 0000 to 1FFF
14 M2.....2000 -- 3FFF
13 M3.....4000 -- 5FFF
12 M4.....6000 -- 7FFF
11 M5.....8000 -- 9FFF
10 M6.....A000 -- BFFF
9 M7.....C000 -- DFFF
7 M8.....E000 -- FFFF

M1 represents the 8K of RAM normally found on the C1 or superboard.

M6 represents the 8K BASIC-IN-ROM on the C1, Superboard and C4.

Continued

8K EPROM BOARD 4 x 2K 2716 EPROMS CIRCUIT DIAGRAM



If D1 used, then use 7410 in lieu of 7412 without pullup resistor R3, R4

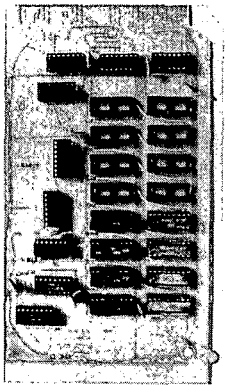
2716 EPROM BOARD

IC4. Defines the 8K Block. Strap points a-h over to either J or K of IC5. The following table is for each 2K Block of memory in the top 32K of memory.

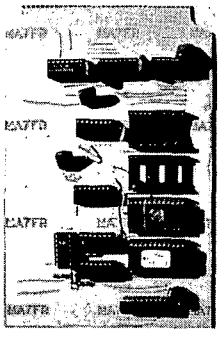
IC5. This is in two identical sections. The strap information is applicable to either.

IC4	IC5 2or 15	7	9	W	Q	= 8000--87FF
11.		6	10	X	N	8800--8FFF 33K - 40K
	jor	5	11	Y	M	9000--97FF
e.	k.	4	12	Z	L	9800--9FFF
10.	"	"				A000--A7FF
						A800--AFFF 41K - 48K
f.						B000--B7FF
	"	"				B800--BFFF
9.	"	"				C000--C7FF
						C800--CFFF
g.						D000--D7FF 49K - 56K
	"	"				D800--DFFF
7.	"	"				E000--E7FF
						E800--EFFF 57K - 64K
						F000--F7FF
h.						F800--FFFF

RAM 2 x 4K Blocks



EPROM 4 x 2K Blocks



M7 represents input / output, disk, video and keyboard.

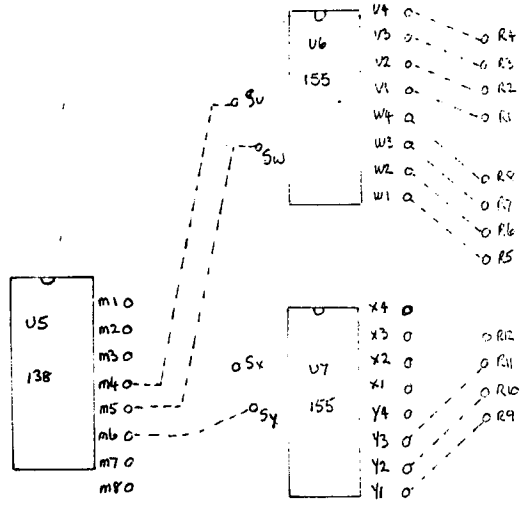
M8 represents Color video on a TASAN video (C4 video). ACIA cassette port at F000 on a C1 and the Monitor ROM at F800 (DABUG or CEGMON) or FD00 for the original ROM.

There is an unused 2K block at E800 to EFFF. I have EXTENDED MONITOR in EPROM at this address. EXMON uses routines in the monitor ROM which call the video driver in BAS 4.

SELECTING A 2K BLOCK

U6 AND U7 are packages, each with two decoders. Each decoder section handles an 8K block and breaks it down into 4 x 2K sections. By connecting an 8K block (M1 to M8) into one of the inputs of U6 (Sv or Sw) or U7 (Sx or Sy) the corresponding outputs will be activated. These outputs are then in turn connected to ANY one of the 24 pin sockets via points marked R1 to R12. You can connect any 2K block to any R point (socket), there is no need other than neatness to keep the straps in any order.

U6 input Sv connects to outputs V1 V2 V3 V4



Dotted lines show straps as provided.

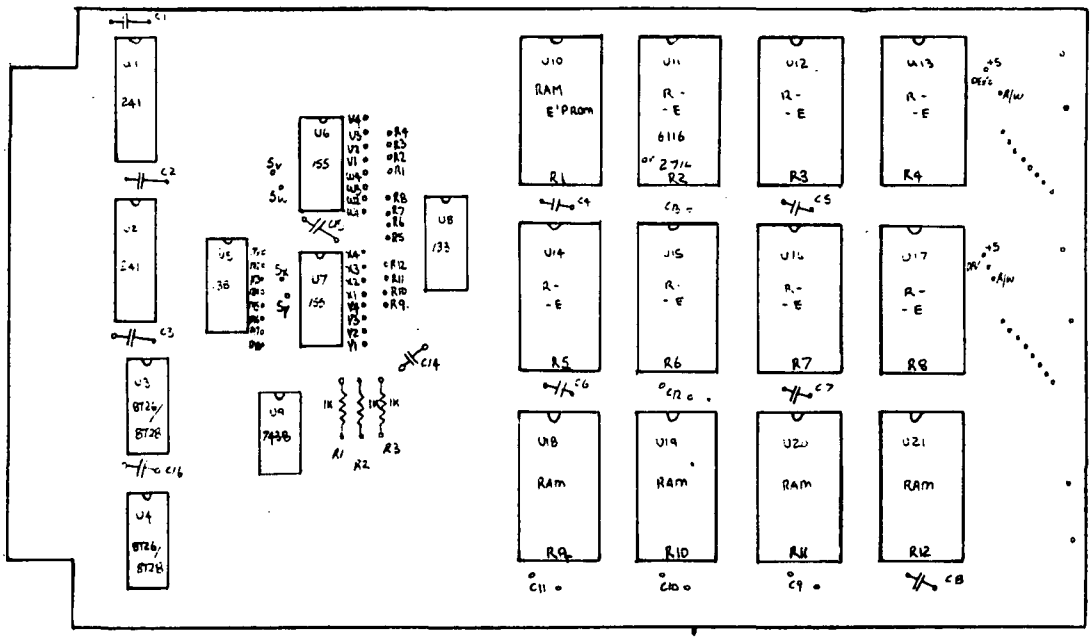
U6 input Sw connects to outputs W1 W2 W3 W4

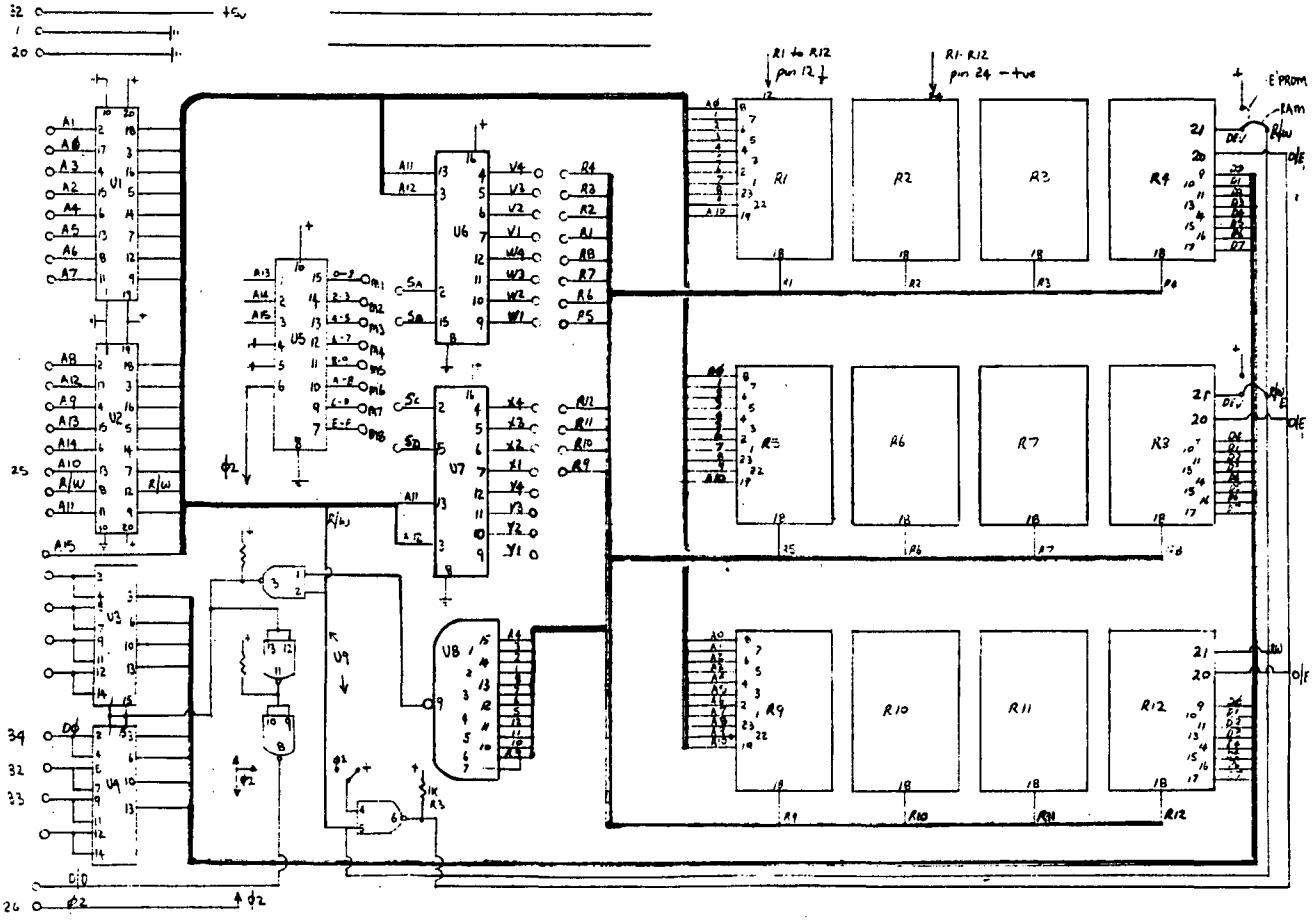
U7 input Sx , connects to outputs x1 x2 x3 x4
U7 input Sy connects to output Y1 Y2 Y3 Y4

At any one time 4 x 8K blocks may be active on the board. Any 2K device can be selected into any of the 4 blocks. If an 8K block of memory should happen to contain some other device that is not on this board, then that is ok providing that the relevant 2K sections do not get connected across to one of the R1 to R12 points. It is only when one of the R strapping points become active that there is a DD (data direction) signal sent to the host computer.

More Schematics on page 14.

24K STATIC RAM CARD - C'MOS WITH PROVISION 16K 2716 EPROM





From Gander Software *The Ultimate Data Base Manager*

THE DATA SYSTEM

AFTER 2½ YEARS OF DEVELOPMENT, THE MUCH RUMORED & SOUGHT DBM IS HERE!
WITH IMPROVEMENTS OVER THE OTHER SIMILAR ITEM

- Stored Report Formats
- Stored Jobs, Formats, Calcs.
- Multiple Condition Reports
- Multiple File Reports
- Calc. Rules Massage Data
- Up to 100 Fields Per Record
- User Designed Entry/Edit Screens
- Powerful Editor
- Merges - Append, Overlay, Match
- Posting - Batch Input
- Nested Sorts - 6 Deep
- Abundant Utilities

HARDWARE REQUIREMENTS: 48K OSI, Hard Disk, serial system, OS-65U 1.42 or Later; Space required: 1.3 megabytes for programs and data.

TECHNICALITIES: User configurable to either 1.42 or 1.43 and Later; 9 job files, each saving up to 40 report formats, 40 calculations, 40 merge routines, or 40 posting routines, 40 file editors, and all those on up to 9 conditions each! Will convert older Type 10 and Type 20 file headers to the new Type 30; Etc., etc.

PRICE: $650.00 (User Manual $35.00, credited towards TDS purchase). Michigan residents add 4% sales tax. 30 day free trial, if not satisfied, full refund upon return.

DEALERS: This package is the new standard in DBM's. We are out to make it your standard too! Give us a call to find out how we can make TDS very worth your while.

SIMPLY POWERFUL AND COMPLETE: This DBM has external simplicity, and a manual that is written in English; complete from Tutorial to Definition of Terms. Although Gander provides support by the author, it shouldn't be necessary.

TDS's power is derived from all the things you would expect of a DBM, plus a list of capabilities (menu selectable and self-instructing) which include: Reports in any format, constructed on screen, giving only the desired data from up to three files with key file access; Move fields from one record to another; Sub Totals and Totals where wanted, etc.; Calc. Rules are similar in syntax to OSI's Planner Plus; PRTMAP modified to work with all system printers, with paging and still user independent; Quick File's QF Sort speeds sorting with no record length limits; Many machine language Utilities expedite the system; for instance Pack File is 2850% faster.

GANDER SOFTWARE

3223 Bross Road
"The Ponds"
Hastings, MI 49058
(616) 945-2821

"It Flies"

FROM THE FOLKS WHO BROUGHT YOU:
Financial Planner
Time & Task Planner

AND THERE IS MORE COMING SOON:
Program Generator for TDS
Proposal Planner
Time and Billing A/R

ZAP, GAME WITH SOUND FOR C1P

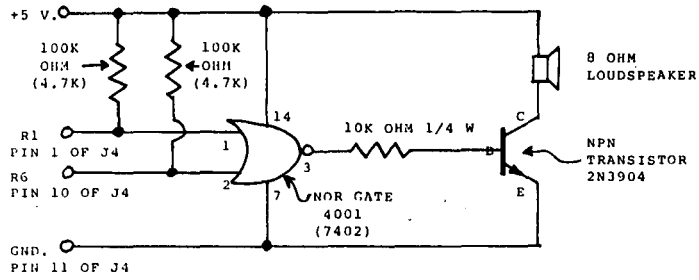
By: Michael A. Bryson
203 Meadow View Dr.
Buchanan, MI 49107

A couple of years ago, I modified a game program written for a VIC 20 computer for use on the OSI C1P, I'd like to share it with you here. This program uses a lot of little features I gathered from many sources. I put all the repeatedly called subroutines and the guts of the program with low line numbers for speed of operation. A machine code screen clear taken from the BASIC ROM is used to improve the action. My contribution was to use the sound generation capabilities of the computer that OSI considered when they designed the C1P but left out when it came to getting the price within the selling range OSI wanted. A small amount of hardware is required, but you don't have to make any changes to the computer itself, like many hardware modifications require. All the circuitry necessary connects to the C1P through jack J4 on the 600 board. If you buy all the parts new it will cost about $5.00. All the parts shown are available at Radio Shack stores if you are not a collector of electronic parts like I am. None of the component values are critical, for example any NPN transistor with a gain of at least 50 could be used. Enough said, I will explain the circuitry and some of the unusual program operation.

THE CIRCUIT

Normally the keyboard of a C1P is scanned one row at a time to check for key closure. By designing a circuit to continuously monitor two or more rows of the keyboard simultaneously, it is possible to use the keyboard as an output port. The normal operation of the keyboard is not affected. All you need to monitor the keyboard is a dual input NOR gate such as a CMOS type 4001 or a TTL type 7402 integrated circuit. The output of the integrated circuit used to monitor the keyboard is not powerful enough to operate a small loudspeaker. A transistor is added to the circuit to provide the additional power to produce sound. Only three more components are necessary to complete the circuit as shown in Figure 1. Two pull up resistors are used to keep the output of the NOR gate in the off state until the two rows being monitored are both

KEYBOARD MONITOR CIRCUIT



NOTE: Resistor values for 7402 IC are shown in parenthesis.

FIGURE 1

```
 10 1E00                *=$1E00       ;NOISE
 20 000A=               ADDR=$A       ;GENERATOR PROGRAM
 30 1E00 A50A           LDA ADDR      ;SAVE 0 PAGE LOCATIONS
 40 1E02 48             PHA
 50 1E03 A50B           LDA ADDR+1
 60 1E05 48             PHA
 70 1E06 A900           LDA #0        ;SET POINTER TO TABLE
 80 1E08 8D9A1E         STA $1E9A     ;TABLE AT $1F00
 90 1E0B A91F           LDA #$1F
100 1E0D 8D9B1E         STA $1E9B
110 1E10 18             CLC
120 1E11 AD9A1E NEXT    LDA $1E9A
130 1E14 850A           STA ADDR      ;TABLE VALUES
140 1E16 AD9B1E         LDA $1E9B
150 1E19 850B           STA ADDR+1
160 1E1B A003           LDY #3        ;STORE LOOP QUANTITY
170 1E1D B10A           LDA (ADDR),Y
180 1E1F F07F           BEQ NOISE     ;IF=0 MAKE EXPLOSION SOUND
190 1E21 8D9F1E         STA $1E9F
200 1E24 A000   WARB    LDY #0
210 1E26 B10A           LDA (ADDR),Y  ;READ FIRST TABLE VALUE
220 1E28 F039           BEQ RSTR      ;IF=0 THEN RETURN TO BASIC
230 1E2A 8D9C1E         STA $1E9C     ;WARBLE IF GREATER THAN 1
240 1E2D A001           LDY #1
250 1E2F B10A           LDA (ADDR),Y  ;TONE DELAY VALUES
260 1E31 8D9D1E         STA $1E9D
270 1E34 A002           LDY #2
280 1E36 B10A           LDA (ADDR),Y
290 1E38 8D9E1E         STA $1E9E
300 1E3B A900   TONE    LDA #0        ;OUT LOW VALUE TO PORT
310 1E3D 8D00DF         STA $DF00
320 1E40 20801E         JSR $1E80     ;JUMP TO DELAY SUBROUTINE
330 1E43 A9FF           LDA #$FF      ;OUT HIGH VALUE TO PORT
340 1E45 8D00DF         STA $DF00
350 1E48 20801E         JSR $1E80
360 1E4B CE9D1E         DEC $1E9D
370 1E4E CE9C1E         DEC $1E9C
380 1E51 D0E8           BNE TONE
390 1E53 CE9F1E         DEC $1E9F
400 1E56 D0CC           BNE WARB
410 1E58 A904           LDA #4        ;SET POINTER TO NEXT
420 1E5A 6D9A1E         ADC $1E9A     ;GROUP IN TABLE
430 1E5D 8D9A1E         STA $1E9A
440 1E60 4C111E         JMP NEXT
450 1E63 68     RSTR    PLA
460 1E64 850B           STA ADDR+1    ;RECOVER PAGE 0 INFO
470 1E66 68             PLA
480 1E67 850A           STA ADDR
490 1E69 60             RTS
500 1E80                *=$1E80       ;DELAY SUBROUTINE
510 1E80 AE9E1E         LDX $1E9E
520 1E83 AC9D1E LOOP    LDY $1E9D
530 1E86 88     DELAY   DEY
540 1E87 D0FD           BNE DELAY
550 1E89 CA             DEX
560 1E8A D0F7           BNE LOOP
570 1E8C 60             RTS
580 1EA0                *=$1EA0
590 1EA0 A9A0   NOISE   LDA #$A0      ;NOISE SUBROUTINE
600 1EA2 850B           STA ADDR+1    ;DUMP PORTION OF ROM
610 1EA4 A900           LDA #0        ;BASIC TO SOUND PORT
620 1EA6 850A           STA ADDR
630 1EA8 A20A           LDX #10
640 1EAA A000           LDY #0
650 1EAC B10A   LOOP1   LDA (ADDR),Y
660 1EAE 8D00DF         STA $DF00
670 1EB1 A915           LDA #$15
680 1EB3 8D9F1E         STA $1E9F
690 1EB6 CE9F1E DEL1    DEC $1E9F
700 1EB9 D0FB           BNE DEL1
710 1EBB C8             INY
720 1EBC D0EE           BNE LOOP1
730 1EBE E60B           INC ADDR+1
740 1EC0 CA             DEX
750 1EC1 10E9           BPL LOOP1
760 1EC3 4C631E         JMP RSTR
770                     .END
```

TRY US!

WE MEAN BUSINESS
SC5/80-1 Computer System

With 1 Denver Board Multi Processor
Wired for 6 Users
Expandable to 14 Users

$6990.00

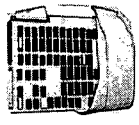
For Each Add'l Multi Processor User Add $1,000.00

- **Includes Super Utility Package**

Plus our incredible new super system data base manager. With brand new Disk Tech One, 2 year warranty on internal hard disk components. Beautiful hand finished oak cabinet over steel frame. Instant access to all components. Available with casters or plastic feet

BEAUTIFUL AS WELL AS FUNCTIONAL

NEW CONTROLLER/INTERFACE!

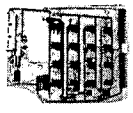


$599.00

The new 9590 Controller replaces the OSI 590/525 set with 1 single board. Unlike others you've seen, it works perfectly with OSI or Denver boards, and all versions of 65U

It is compatible with old Style 592 Interface with 3 cable adapter, or attaches with single ribbon cable to our new 9592 H.D. Interface Board. Unlike others, our interface completely supports the ready/fault indicator on the 80 meg. hard disk.

$269.00



12 SLOT BUS

Fits standard OSI mount holes can be connected in "T" or daisy chained by cable for expansion.

$79.00

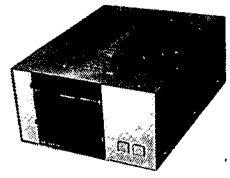
CLOSE OUT!
C3-OEM

2 MHz/56K*
With 6502 and Z80 Processors

With Centronics Parallel Printer Interface

While they Last!

$1499.00



*Includes 8K Exec Memory for CP/M or Multi-User

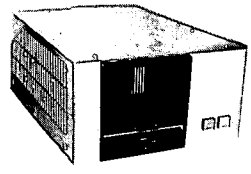
- Buy for a Spare
- Add to a Spare H.D.
- Develop Programs

Spare 510CPU Board with Z80+6502 $169.00

MULTI PROCESSOR!

10 COMPLETE MEGABYTE SYSTEM

$3999.00



Boot From Hard or Floppy!

- 8" Hard Disk
- 8" Floppy Disk

Single user with Centronics parallel printer interface expandable to eight users.

Enclosed in table top cabinet as shown or mounted in deluxe floor cabinet, as above left. Add $400.00

10 meg. Subsystem WOW!

$1999.00

Add to any existing OSI floppy based computer.

Just plug in one board and set this little gem on top or next to your computer and voila! Welcome to speed and convenience. Completely self contained with power supply. 9598 Hard Disk Controller also available separately at $699.00

DEALERS! **If you're still alive, call us and ask about our aggressive new price structure. We'll send you an unbelievable spares price list.**

SPACE-COM International
22991 La Cadena Drive, Laguna Hills, CA 92653 (714) 951-4648

in the low state. The third resistor limits the current output of the NOR gate to prevent damage to the integrated circuit when the transistor goes into saturation. That is all the hardware you need besides a small loudspeaker (I used a 2 inch transistor radio type) to make sound with your ClP. I could have complicated the circuit by adding a volume control but I found this works quite well and wanted to make this easy enough that many people will decide to add sound to their ClP.

The output of the circuit will be a train of digital pulses generated by a machine language routine. BASIC is too slow to produce a wide range of tones in the audio range. The wiring is not critical, as a good practice all unused inputs of the integrated circuit should be connected to the supply ground, the unused outputs should have no connections to them. Jack J4 of the 600 board has connections for rows 1, 6, and 7 of the keyboard. Any two of the rows on J4 can be used for the monitoring circuit. The ground connection is made at pin 11 of J4. None of the jacks on the ClP provide a +5 Volt supply. The +5 volts can be taken directly from the power supply unit if you do not wish to attach any wires directly to the 600 board, just get the +5 volts from the terminal with the red wire going from the power supply to the computer.

MAKING SOUND

Owners of Series 2 ClP computers do not need to add the monitor circuit since the DAC converter already does this for you. Since I do not have a Series 2, I cannot verify the operation of this game but it appears all you need do is turn on the DAC with a POKE 55296,16 at about line 430 in the program. The noise subroutine in this game will allow you to be able to generate various sounds such as phasors, crashes (explosions) and tones (constant or warbling). To generate these sounds required my first attempt at machine language programming, this may not be the most efficient routine and it's not immediately relocatable but it does work. Maybe this article will stimulate some interest in sound for the ClP and a better sound routine will be printed in a future issue. Just experiment with values in the sound table to make your own special tones as

SOUND TABLE

BYTE	LOCATION DECIMAL	HEX	DESCRIPTION
B1	7936	6100	WARBLE(2-255), TONE(1), EXIT(0)
B2	7937	6101	FREQUENCY LOW BYTE(0-255)
B3	7938	6102	FREQUENCY HIGH BYTE(0-255)
B4	7939	6103	LOOP(1-255), CRASH(0)

NEXT SET OF DATA

FIGURE 3

```
1 REM**********************
2 REM* OSI ZAP WITH SOUND *
3 REM*    by Mike Bryson  *
4 REM*    January, 1982   *
5 REM**********************
10 GOTO 980:REM INITIALIZE
20 FORV=1TOLEN(A$):POKEA+V,ASC(MID$(A$,V,1)):NEXT:RETURN
30 POKE530,1:POKEKE,253:K1=PEEK(KE)
40 IFK1=223THENDR=-32:H=1:GOTO90
50 IFK1=247THENDR=32:H=2:GOTO90
60 POKEKE,254:K1=PEEK(KE)
70 IFK1=252THENDR=1:H=3:GOTO90
80 IFK1=250THENDR=-1:H=4
90 POKE7938,H:X=USR(X)
100 FORT=1TO30-5*S:NEXTT
110 POKEB,32:B=B+DR:POKE530,0
120 RETURN
130 SC=0
140 K1=PEEK(B)
150 IFK1=42THENRETURN
160 IFK1=1THENSC=1:GOTO220
170 IFK1=226THENSC=2:GOTO220
180 IFK1=30THENSC=3:GOTO220
190 IFK1=232THENSC=5:GOTO220
200 IFK1=15THENSC=10:GOTO220
210 GOTO230
220 TS=TS+SC
230 FORT=1TO30-5*S:NEXTT
240 RETURN
250 REM GAME STARTS HERE
260 FORR=1TORN
270 FORP=1TOPN:PRINT"PLAYER #";P
280 PRINT:PRINT"What skill level"
290 INPUT"Enter 0,1,2,3 or 4";S
300 IFS>4ORS<0THENPRINT"HUH?":GOTO290
310 X=USR(X):A$="SCORE TO BEAT:":A=53642:GOSUB20
320 A$=STR$(TB):A=53710:GOSUB20
330 A$="SKILL LEVEL:"+STR$(SL):A=53802:GOSUB20
340 A$="PLAYER #"+STR$(PB):A=53900:GOSUB20
350 FORT=1TO2000:NEXTT:X=USR(X)
360 POKE11,0:POKE12,30
370 DEFFNA(L)=INT(RND(1)*L)+LS
380 FORF=1TO50-2*S:D=FNA(643):POKED,1:NEXTF
390 FORF=1TO30:D=FNA(673):POKED,226:NEXTF
400 FORF=1TO13+4*S:D=FNA(673):POKED,42:NEXTF
410 FORF=1TO24:D=FNA(643):POKED,30:NEXTF
420 FORF=1TO18:D=FNA(643):POKED,232:NEXTF
430 FORF=1TO12+S:D=FNA(673):POKED,15:NEXTF
440 LC=0:B=53777
450 IFLC>1000ORRND(1)<.001THEN680
460 IFB<LSTHENB=B+672
470 IFB>HSTHENB=B-672
480 POKEB,240
490 GOSUB30
500 GOSUB130:IFK1=42THEN530
510 LC=LC+1
520 GOTO450
530 POKE7939,0:X=USR(X):POKE7939,8
540 POKE11,34:POKE12,2:X=USR(X)
550 FORT=1TO6:PRINT:NEXTT
560 PRINT"#### #### #### ## ##"
570 PRINT"#### #### #### ## ##"
580 PRINT"   # #  # #  # ## ##"
590 PRINT"  #  #### #### ## ##"
600 PRINT" #   #### #### ## ##"
610 PRINT"#### #  # #"
620 PRINT"#### #  # #    ## ##"
630 FORT=1TO2000:NEXTT
640 X=USR(X)
650 A$="YOU LASTED:":A=53642:GOSUB20
660 A$=STR$(LC)+" MOVES":A=53706:GOSUB20
670 GOTO710
680 POKE7936,85:X=USR(X):POKE7936,1
690 POKE11,34:POKE12,2:X=USR(X)
700 A$="TIME IS UP....":A=53642:GOSUB20
710 A$="SCORE="+STR$(TS):A=53802:GOSUB20
720 Z(P,R)=Z(P,R-1)+TS
730 IFTS>TBTHENTB=TS:SL=S:PB=P
740 A$="ROUND #:"+STR$(R):A=53866:GOSUB20
750 FORX=1TOPN
```

Continued

described next and add to your game programs.

Figure 2 shows the assembly code listing of the sound generator. In order to make different types of tones I decided to have a table of values which contain the instructions to a sound generator routine. Four values are stored in the table for each tone sequence as shown in Figure 3. Byte B1 serves three functions, if it equals zero and the value in B4 is non-zero then the routine returns to BASIC. I added the special crash noise routine after writing the tone generator so the routine has to check B4 before checking B1. If B1 has a value between 2 and 255 then the tone repeats B1 times with B2 decreased by one each time the tone is repeated, thus changing the frequency. When B1 equals one then only a single tone is produced. Byte B4 determines how long the tone stays on by repeating the whole pulse sequence B4 times. The values which are stored in B2 and B3 determine the delay between pulses, in other words the frequency of the tone to be generated. The routine used decrements the values in the table before testing for a zero value. If a zero is stored in B2, the computer will loop through 256 values before

```
760 PRINT"PLAYER #";X;":";Z(X,R)
770 NEXTX
780 TS=0:DR=0:H=0:PRINT
790 NEXTP,R
800 POKE133,LB:POKE134,HB
810 END
820 REM DATA FOR NOISE GENERATOR
830 DATA165,10,72,165,11,72,169,0,141,154,30,169,31,141,155,30,24
840 DATA173,154,30,133,10,173,155,30,133,11,160,3,177,10,240,127
850 DATA141,159,30,160,0,177,10,240,57,141,156,30,160,1,177,10,141
860 DATA157,30,160,2,177,10,141,158,30,169,0,141,0,223,32,128,30,169
870 DATA255,141,0,223,32,128,30,206,157,30,206,156,30,208,232,206
880 DATA159,30,208,204,169,4,109,154,30,141,154,30,76,17,30,104,133
890 DATA11,104,133,10,96
900 REM DATA FOR DELAY SUBROUTINE
910 DATA174,158,30,172,157,30,136,208,253,202,208,247,96
920 REM DATA FOR CRASH ROUTINE
930 DATA169,160,133,11,169,0,133,10,162,10,160,0,177,10,141,0,223
940 DATA169,21,141,159,30,206,159,30,208,251,200,208,238,230,11
950 DATA202,16,233,76,99,30
960 REM DATA FOR SOUND TABLE
970 DATA1,90,1,8,0
980 LB=PEEK(133):HB=PEEK(134)
990 POKE133,255:POKE134,29
1000 REM SCREEN CLEAR
1010 POKE11,34:POKE12,2:POKE574,96
1020 FORX=0TO27:V=PEEK(65036+X):POKE546+X,V:NEXT
1030 X=USR(X)
1040 A$="LOADING":A=53645:GOSUB20
1050 A$="DATA":A=53710:GOSUB20
1060 B=53777:D=0:DR=0:F=0:H=0:HS=54140:K1=0:KE=57088
1070 L=0:LC=0:LS=53445:P=0:PB=0
1080 PN=0:R=0:S=0:SC=0:SL=0:T=0:TS=0
1090 DIM PL(6),R(5),Z(6,6)
1100 FORX=768TO785:READV:POKEX,V:NEXT
1110 FORX=7808TO7820:READV:POKEX,V:NEXT
1120 FORX=7840TO7877:READV:POKEX,V:NEXT
1130 FORX=7936TO7940:READV:POKEX,V:NEXT
1140 FORV=1TO5:FORX=1TO6:Z(X,V)=0:NEXTX,V
1150 X=USR(X)
1160 PRINT"WELCOME TO OSI ZAP"
1170 PRINT:PRINT"by Mike Bryson"
1180 PRINT"January, 1982"
1190 PRINT"Game uses these keys"
1200 PRINT"for movement:"
1210 PRINT"'Z' UP":PRINT"'/' DOWN"
1220 PRINT"'LEFT SHIFT' LEFT":PRINT"'RIGHT SHIFT' RIGHT"
1230 PRINT:PRINT"SCORING:"
1240 PRINTCHR$(1);"=1":PRINTCHR$(226);"=2":PRINTCHR$(30);"=3"
1250 PRINTCHR$(232);"=5":PRINTCHR$(15);"=10"
1260 PRINT"You are:";CHR$(240)
1270 PRINT"Don't hit a * or"
1280 PRINT"you will get ZAPPED."
1290 INPUT"How many rounds (1-5)";RN
1300 IFRN<1ORRN>5THENPRINT"HUH?":GOTO1290
1310 PRINT:INPUT"How many players (1-6)";PN
1320 IFPN<1ORPN>6THENPRINT"HUH?":GOTO1310
1330 GOTO260
```

saved scratch Dsk #25

continuing the next step. You should be careful about putting a zero in these bytes because you may think the computer is stuck in an endless loop. All this information is provided for those who may want to add sound to their own game programs. I wrote this game to work on any C1P computer with at least 8K of RAM memory.

After experimenting with the tone generator I found I could not produce a sound suitable to simulate a crash or explosion needed for many games. A very complex string of data is needed to produce such a sound. Just for fun I decided to output the contents of the BASIC ROM to the sound port. This is a readily available string of random bytes. This works quite well as an explosion. I merely added the new routine and provided a check in byte B4 of the sound table.

PROGRAM DESCRIPTION

Figure 4 contains a listing of the ZAP game which demonstrates three of the possible types of sound; tones, warble and crash, which the machine language program can produce. To use the machine language routine you must first reserve space in the upper part of memory as in lines 980-990. Lines 830-970 and lines 1100-1140 place the sound routine in memory. Sound is produced by poking in values to the sound table and calling the USR function. The pointer to the USR function must be set as shown in line 360. This game program also uses a machine code screen clear so you will notice the pointers in location 11 and 12 to the USR function are changed throughout the game.

Because of the fast action of the higher skill levels, you may wish to change the keys which control the direction of the player on the screen to suit your own preferences. Lines 30-80 contain the keyboard scanning routine so you need only make changes there using data from your user's manual. The game starts when you press any direction key. You can't stop moving once the game starts, you can only change direction. It is not necessary to hold down the direction keys as the player will continue in the same direction until a new direction is entered.

The game ends when you hit a '*' or a random number cuts out your playing time (a real time clock is needed, but I haven't worked that out). Now OSI C1P's will not have to be silent any longer.

LETTERS

ED:

I have had several complimentary communications on my Word Processor article in PEEK(65) (Dec. 1983, page 2). All, however, refer to typos, such as the missing comma in line 460, and several places where "L" and "1" are interchanged. A more persistent and serious problem is differences in OSI hardware & DOS memory locations between different models. Apparently, memory locations 9504, 9815, and 9834 are not the only ones that OSI uses for their GET routine. You will have to experiment if your model uses another.

Assuming that PEEK (57088) senses the right & left shift in your OSI, use the following test to see if your model will return the values given below:

```
10 Q = PEEK (57088)
20 PRINT Q
30 GOTO 10
```

```
Note: right shift =   252
      left shift  =   250
      left shift/L =  255
      left shift/O  = 218
      right shift/O = 220

      line 11 checks for
       "return"
      line 12 checks for "L"
      line 13 checks for " "
        (right shift/N)
      line 15 provides a de-
        structive backspace
        (right shift/O)
      line 16 converts caps to
        lowercase
```

Perhaps these clues will allow you to experiment with the peek values to find the right combination for your OSI. After all, the name for this publication is PEEK(65), isn't it?

Stanley Harshfield
Memphis, TN 38115

* * * * *

ED:

I finally got tired of only being able to get 48 characters on a line. I've got a C1P-MF W/OS65D 3.3 and a monitor. The computer can easily support more characters, but the operating system stops conservatively short. Thanks to an article in PEEK(65) by

David Larson and Charles Stewart, I found the following to provide a 55 character line with no other modification.

1 POKE 13034,54:POKE 13035,136 :POKE 13049,54:POKE 13050,136

Just add this to BEXEC* and resave, or in DOS "SAVE 13,1= 3274/8.

Brick Rule
Sarasota, FL 33582

* * * * *

ED:

I would like to get information / answers regarding the following items:

1. How do you convert a video based C8P-DF to a serial based machine?

2. Are spare parts available from the new owners of OSI?

3. Who are the new owners of OSI, and how can they be contacted?

4. Can CP/M be added to a C8P-DF and if so how? Who sells boards and software?

Thank you very much for any information that you may be able to give me.

Archie P. Berry
Akron, OH 44313

Archie:

1. Too much to handle here, but it's not too hard. We'll try to have some details next month.

2. Yes. Call Tech Support - see ISOTRON Feb ad.

3. You guessed it. ISOTRON, Inc., 140 Sherman St., Fairfield, CT 06430.

4. Depending which CPU Board you have, you may already have the Z80, if you will really use CP/M, see ads by D&N for their Proxy board.

Peek Staff

* * * * *

ED:

Having recently purchased an OSI computer, I've found that there are many questions I need answered. So far my search has proved useless. After reading a couple issues of Peek, I am going to try them on you. Currently, I have a C2-OEM, Double sided Double density disks, Intertec Terminal, and an Epson printer, all controlled under 65U.

1. Where can I get info on this unit? The only thing I have is a second rate users manual that tells me very little about what I can and cannot do.

2. What types of software will run on it, and where can the software be purchased?

3. Who still supports it?

4. What hardware is out there to work with this system?

5. I would like to change the operating system (what is being loaded on boot) but have no idea how to get into it.

What it all comes down to is that I have a machine that I really like (but who writes an operating system in BASIC) and would like to keep but I really need some help. I need information on the system.

I don't know what your policy is as far as answering this type of question (throw it out to readers) but I would like an answer if possible. I'll pay postage (and a modest fee) if it will help. I need answers. Help!!!

Eugene S. Mellon
Pittsburgh, PA 15205

Eugene:

Welcome aboard! It sounds like you have bought a used machine with only partial documentation. As more time goes by, more used machines are being bought, many with less than complete documentations. So you are not alone. Hopefully, the following will help you and others in the same boat.

1. To start with, early OEM documentations left something to be desired. Hence the pages of back issues of Peek are full of useful information. As for your missing documentions, these should still be available from your dealer or ISOTRON (OSI). Call one of the new Tech Support numbers (ISOTRON ad March) or the Aurora, OH plant (216) 562-2020. Additionally, Peek's "Goodies" list on the back cover has other supportive manuals including the Howard Sams schematic manuals.

2. What will run on the OEM? Programs for OS65-D, OS65-U, HEXDOS, to name a few, and if your CPU board happens to also have a Z80 CPU and a couple of modifications are made. CP/M,

but it's slow! If you order CP/M from OSI, specify the CPU board number and Rev. number. PASCAL and FORTH may also be available, with some restriction and/or limitations. Lastly, many programs written in MICROSOFT BASIC can be converted to OSI's disk operating system. Where do you get the above? Some from OSI. The others have all been advertised or mentioned in Peek at one time or another. Start with the annual "index".

3. The software is supported by the respective vendors and/or authors.

4. By hardware we presume you mean terminals, modems, printers, and the like. The answer is - just about anything you can buy. One of the beauties of OSI is that the innards are accessible and modifiable to meet most any condition.

5. Modifying the operating system is not a good idea unless you have sufficient back-ups of the original, have a good general knowledge of operating systems, and have read up on OSI's in particular. Again, see old Peeks. One non-destructive way to get at it is to POKE the changes in at the beginning of your program. That way, the next time you boot, everything is back to normal. As for operating systems in BASIC, - forget it! Slow! Slow!

Now, the "modest fee," you are charged with telling friends about Peek and writing to us about the things you have done and problems you have solved.

Peek Staff

* * * * *

ED:

I'm currently using a C1P series II upgraded to 32k, with the D&N MPI B51 drive and operating under OS65DV3.3 (a whole new world compared to BASIC-in-ROM for me!), and I would like to add a second mini-floppy drive, being the Shugart 400. I understand there is a system patch allowing the stepping rate to be changed to the 20msec rate rather than the 5msec rate required by the MPI. What is the nature and location of such a patch?

Any information you might provide would be greatly appreciated.

Scottie Cantrell
Tucson, AZ 85749

* * * * *

ED:

I have an OSI C1P with the OS65D3 V3.3 disk operating system. Recently, I ran into a deal too good to pass up: two Teac FD50 disk drives in a nice small enclosure and power supply for under two hundred dollars, brand new! But now I'm faced with the problem of interfacing these drives with the C1P. I say problem because these drives have a track to track access time of 25u seconds as opposed to MPI's 83. If anyone knows of how I could step up the software "step rate" I would be very thankful if they would share it with me!!

Also, if anyone knows of any good modem drivers, dumb and/or intelligent, which will operate at 1200 baud, could you please let me know?

Steve Meisner
Arlington, MA 02174

Steve:

Regarding your question about OS-65D's step rate location, it is stored at $EF. I'm not going to be of much help beyond that since I'm a real hardware rookie. The problem is that 3.3 installs the value at this location on cold boot in the routine at $2E79. As you may know, OSI is famous for playing with this routine in attempting to compute the system's clock speed. I advise you to call Bill Thompson at ISOTRON Tech. Support 800-321-5805. Give Bill the above information so he doesn't have to look it up, and cry for help... a small crack in your voice helps a lot. Okay, that's my "official" response. If you are ambitious and have another version of 65D running, you may be able to patch your 3.3 diskette with something along the following lines:

(1) Boot the working copy of OS-65D and make a back-up copy of the 3.3 diskette.

(2) Invoke the Extended Monitor and insert the back-up copy of the 3.3 diskette in the drive.

(3) At the EM's ":" prompt, enter "!CA 4A00=01,1".

(4) If a printer is available, enter "!IO ,03" if it's a serial printer, or "!IO ,06" if it's a parallel printer (assuming this is a video system).

(5) Again at the ":" prompt, enter "Q4E79".

(6) Somewhere between $4E79 and $4F78, there is a table of values that run something like:

```
.BYTE $33,$33,$33,$33
.BYTE $66,$66,$66,$66
.BYTE $xx,$xx,$xx,$xx
```

The above is from memory and is likely in error as to the specific values, but the pattern of 3 sets of repeating values should appear in some fashion. I believe that by adjusting the row that corresponds to your system's clock speed (i.e. 1st row = 1MHz, 2nd row = 2MHz, 3rd row = GT), you can make the change you desire. The mini-floppy and 8" version of this routine

differ significantly, and have been altered in subsequent releases of 3.3 I believe. Hopefully, you will be able to disassemble the routine and get a better handle on the effects of the routine.

(7) Assuming the change has been made with the "@" command in the Extended Monitor, this last command will make the change permanent on the 3.3 diskette. At the ":" prompt, enter "!SA 01,1=4E79/8" for mini-floppies or enter "!SA 01,1=4E79/5" for 8" disks.

The last thing that may help is that if you are able to boot 3.3 and the BEXEC* runs, but have problems with subsequent disk accesses, POKEing ($EF+$2F79) will allow you to make adjustments under BASIC. I trust you can handle the HEX conversion.

I hope this is of some help.

Richard L. Trethewey

* Re your questions - Modem Drivers - See PEEK back issues Sept '83 page 11 and Oct '83 page 5.

Peek Staff

* * * * *

ED:

Here is a goodie for assuring that a Key Search always FINDs what you want.

Line 760 of EDKFMF pads the Key File with spaces (SP$) when it is loaded to the length of the search field.

Line 1200, the Key File Search Routine, pads field X$ with spaces to the right - so as to match the key.

This mod cuts out any checking except to see if the field found matches. If not, Line 1290 is executed - "no match for X$".

K. E. Orgel
Memphis, TN 38118

* * * * *

ED:

Re Al Adams's letter, April '84, page 17. Al suggested to POKE 207,36 (I am using OS65D v3.2). Well, I did that just for fun to see what he was talking about, and it seems to me to be a completely worthless and unproductive thing to do. It causes BASIC programs to bomb out. It gives SN ERROR messages.

It does allow leading spaces to be typed on a BASIC line, but that would only use up memory space for nothing, so who would want to do that?

Carl M. King
Sarasota, FL 33579

ERRATA

April '84 issue, page 16, paragraph in Bruce Showalter's letter should read:

If speed is important to you, consider that the prices I quoted are for 200n.s. 2016s and 450n.s. 2114s. The 2016's also require less power.

```
699 REM       LOAD KEY FILE FROM MASTER
700 RPTR=1:ML=0:FL=FL(EF)-2:S$="
705 IF EF=N THEN ML=0:GOTO710
706 FOR I=(EF+1) TO N: ML=ML+FL(I):NEXT I
710 BDF=43+LEN(KN$):INDEX<1>=BDF
720 O$="^":DEV DV$(3)
730 IF EF=1 THEN FOS=0: GOTO 750
739 FOS=0
740 FOR K=1 TO (EF-1):FOS=FOS+(FL(K)):NEXT K
750 INDEX<2>=((RPTR-1)*RL)+FOS+BODF
751 IPTR=BODF+((RPTR-1)*RL)
760 INPUT%2,T$:U$=LEFT$(S$,(FL-LEN(T$))):PRINT%1,O$+T$+U$:PRINT%1,IPTR
765 IF INDEX(2)>=(E-ML) GOTO 800
766 PRINT"RECORD NUMBER "RPTR
770 RPTR=RPTR+1
780 IFRPTR<=RNGOTO750
800 EODF=INDEX(1):INDEX<1>=9:PRINT%1,EODF
900 REM
950 PRINT:PRINT"KEYFILE: ";F$(1)+RIGHT$(STR$(Y),1)
960 PRINT"LABEL ";FDLB$(EF)
965 PRINT
970 PRINT"              KEY FILE LOADED "

1180 OPENF$(3),PW$,3
1190 INDEX<3>=42:INPUT%3,T$
1200 Y$="^"+X$+LEFT$(SS$,(19-LEN(X$)))
1210 FINDY$,3
1220 IFINDEX(3)>1E8THENCLOSE3:MS=K1:GOTO2750
1230 INPUT%3,T$:INPUT%3,PA:PRINT
1250 CLOSE3:RPTR(A)=K1+((PA-BODF(A))/RL(A))
1260 X=((RPTR(A)-1)*RL(A))+BODF(A)
1270 FORI=1TON(A):FC$(A,I)="":INDEX<A>=X+FP(A,I):INPUT%A,FC$(A,I):NEXTI
1280 IFFC$(A,1)=X$THEN:PRINTCS$:GOTO1310
1290 PRINT:PRINTFC$(A,1);"NO MATCH FOR "X$;" KEYFILE ERROR"
1300 INPUT"ENTER A  C  TO CONTINUE  (MUST FIX FILE)";A$:GOTO2750
1310 REM
```

PEEK (65)
The Unofficial OSI Users Journal

P.O. Box 347
Owings Mills, Md. 21117

BULK RATE
U.S. POSTAGE
PAID
Owings Mills, MD
PERMIT NO. 18

DELIVER TO:

GOODIES for OSI Users!

PEEK (65)
The Unofficial OSI Users Journal

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

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

( ) Cash enclosed ( ) Master Charge ( ) VISA

Account No. _____ Expiration Date _____

Signature _____

Name _____

Street _____

City _____ State _____ Zip _____

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

POSTAGE MAY VARY FOR OVERSEAS