January/February 1987
Volume 8 Nos. 1&2

PEEK[65]

The Unofficial OSI Journal

Hello there... remember me? Sorry to take so long to get this to you, but there have been some late breaking developments that I wanted to let you in on as soon as possible.

The great news is that Isotron has new owners - again. I just received the following letter:

Dear Rick,

Isotron stock has been purchased by The Phoenix Group. Our objective is to return the Ohio Scientific product line to the pre-eminence it once held, through continued support of the 200 series and new product developments in the 700 series. Stronger customer product support is a key element in our plans. To help sales, we are adding an in-house leasing arm so that dealers can have a convenient means of ditributing product and customers can more economically deal with acquiring our products.

Isotron has been a company that passed its product line from stepfather to stepfather with little concern for the user "orphans".

The prior owners were probably the best of the managers going all the way back to the founders of the company. Even so, they sustained significant losses which, in my opinion, were not due to the products making up the Ohio Scientific family.

Isotron has never signed an agreement with DBI, whether for distributing OS-65U or any other product. DBI is a welcome competitor in the marketplace and we look forward to agressive competition with them in the future.

Inside This Month:

Our approach will be to make OSI a dynamic company that will fully support its dealers and hardware users.

Sincerely,
Franc R. Richardson
President

Mr. Richardson, let me first thank you for keeping PEEK[65] and its readers informed of developments. I look forward to working with you and your staff. Thanks as well for correcting my remarks about DBI.

PEEK[65] was created out of users' frustration with past OSI owners. We have usually been left out in the cold. Letters and phone calls were seldom answered and *never* followed up. Even when I worked for the company under MA/COM's stewardship, information was always "just around the corner". If I could sum up the needs of the user in one word, it would be just that, information.

I'm sure that our future discussions will be benficial to all of us. PEEK[65] is waiting to help.

Now then back to the home front, Paul Chidley of TOSIE describes his CxP, the 65816-based system as well as providing information on RLE graphics. I've included part 2 of my 65U disk editor and part 2 of John Horeman's article on extending the commands in the Color+ software. Robert S. Runyon gives us a cute utility for listing BASIC programs to make them easier to follow. And finally, Gerald Van Horn shares his program for working on Cryptograms.

Let me sign off here with two points. First, I'm very excited about the new 65816 systems. We'll be following their progress very closely and will let you know when and how to get your hands on one. Second, PEEK[65] is in desparate need of articles. So warm up your word processor and have at it! Share that little utility you wrote. Tell us how you use your system. What problems did you find and how did you overcome them? Thanks folks... it's going to be a great year!

Rick

The CxP, A New Start?

by Paul Chidley
TOSIE

Over the past few years there have been lots of letters and articles about the "new" OSI Challenger. Well, OSI has not and may never come out with anything so after two years of work, I did it myself. The system I am writing the article on is my idea of the next Challenger. It contains a 65816, 128K of RAM (expandable to 16MB) and is running at 4MHz. It is hard to sum up two years' work in one short article, but I will try to give as accurate a picture as possible.

The CxP is a single OSI 48-pin board with CPU, RAM, ROM, 2 serial ports, paralell port, OSI disk controller, and a 2793 double-density disk controller. Each section of the hardware will be covered later. First, an understanding of the board's layout is required. A handfull of OSIers, myself included, are not using OSI hardware. The system is based on Eurocards from England, originally published in Elektor, a Brittish electronics magazine. Elektor wanted a disk interface for their Junior computer. They chose the OSI design and 65D and came out with their own version on a single Eurocard for the Junior. They then continued with a CPU, video, RAM, etc. Before long we were able to build a whole OSI system from Eurocards (Eurocards are 100x160 mm). I have not had any OSI hardware for several years and so my designs for a new disk and new CPU are based on Eurocards. Making these available to the rest of the OSI community then presented a problem since redesigning the artwork just to put the same hardware on a different size board was too big a task. Instead, I chose to use the Eurocard format AND the OSI 48-pin format (see figure 1). The CxP consists of three major sections: the CPU "board"; the disk "board"; and the interface section that allows these two Eurocards to plug directly into the 48-pin bus. People like myself literally cut the board in two to use it in our Eurocard systems. To avoid confusion, I will refer to the CPU and disk as "sections", rather than "boards". There is also a hidden bonus, the Elektor Eurocard backplane can plug directly into an OSI backplane. In other words, you can run both OSI and Eurocard boards at the same time in the same system (see figure 2). I alread have a 64K RAM Eurocard built, as well as a version of the Color+ in Eurocard. Converting the 64K RAM card to a 256K RAM card is simple as soon as the price of the chips drop a little more. Now that you have an idea of the board layout, let's talk about the individual sections and the hardware on each.

published monthly
Editor: Richard L. Trethewey

Subscription Rates

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

The CPU Section

The microprocessor is the 65816. The 65816 is a huge topic all by itself and I won't even start to describe it here. If you don't know how fantastic this chip is, I suggest you read up on it ASAP. The CxP does have jumpers on it so a 65802 or 65C02 can be run in its place. The monitor ROM is a 2764, although there is only 4K available in the memory map. Address line 12 is brought out to a jumper so that it can be tied high or low. The idea here is that diagnostic routines may be put in the unused half of the ROM. If your machine develops problems, just pull out the jumper and see if the diagnostics can help. This can be helpful in trouble-shooting disk problems since most of us have our disk test programs on floppy where they don't do any good if your drives crash. At the moment, I haven't written any diagnostics for the ROM except for a memory test, but the capability is there. The RAM is made up of two 43256 (32Kx8) static RAM chips. A header of jumpers is provided to "mask out" the portions of the 64K memory map that are required for things other than RAM. The jumper is organized in 2K blocks from $00:C000 to $00:E800. The space from $00:F000 through 00FFFF is always masked out for the monitor ROM.

The system clock is a dual speed circuit running at 4MHz and shifting to 2 MHz. The shift to 2 MHz is automatic for any address between $00:C000 and $00:FFFF or when the "WAIT" line is pulled low on the backplane. A jumper is also provided on the CPU section to force the clock to stay at 2 MHz if necessary.

Status LEDs are provided to show the state of the CPU section visually. One indicates the speed of the system clock. It is off for 4 MHz and comes on when 2 MHz is selected. One indicates that the RESET line is active. When you press the reset button, the LED comes on. The other three LEDs indicate the state of three of the 65816 status bits. They are the E, X, and M bits. For more information, consult the 65816 data sheet. Circuits on the board are provided to latch the additional address lines provided by the 65816, A16-A23. All eight are available to the Eurocard bus, but only A16 -A19 are passed to the OSI bus. This shouldn't cause a problem for anyone. There aren't many OSI machines around with over 1 MB of memory.

And last, but not least, there are two serial ports on the CPU section, driven by 6551 chips. While not software compatible with the older 6850, the 6551 has its own internal baud rate generator. This improvement warrants the needed software changes to the serial drivers. The changes are simple. I have been using a 6551 as a serial port for several years. The signals from both 6551s are taken to a header. To use the ports, a driver board is needed to plug into this header. At first this may seem like a disadvantage, but I like to think of it as a feature. The connectors used for the serial ports can be anything from a DB-25 to a DB-9, to RJ-11, to ??. Likewise, the driver chips can be 1488/89, max232, rs422, etc. As someone's need change, only the little driver board needs to be redesigned. At this time I have only done up one using the old 1488/89 with DB-9 connectors, but the combinations possible are endless. Using a ribbon cable to connect this board to the CPU section also makes it possible to put the connectors on the case for access to the outside world. The status LEDs may also be mounted on the case and connected by ribbon cable.

The Disk Section

The disk section contains the old familiar OSI style disk interface. The only exception is the data separator, which is based on the Elektor design rather than the OSI. It is a simple circuit, but very reliable. The adjustment for the data separator doesn't even require an oscilloscope, you just center the pots. It is so reliable that I designed most of the board to use it for 8" drives as well as 5". Most 8" drives can separate the data themselves, but only for single-density, so I let the interface do it. Motor control for 5" drives is handled by the READY line. If you have older drives with no READY lines there are a couple of work-arounds. 8" don't usually have any motor control anyway so READY line or no READY line doesn't matter.

The disk section also contains a 2793 disk controller. This chip is capable of all the standard FM and MFM formats. This makes it possibleto use double-density as well as read or write disks in formats used by other computers. Although we have not yet written much software to use this controller, we did write enough to verify that the hardware does work. We have read and written IBM PC disks. The design of the 2793 section is the work of several people in and around the TOSIE user group. Thanks to all of them. The disk section is capable of controlling 5" or 8" drives, but not both at the same time. However, it is possible to build two disk boards and run both in the same system as I do.

The disk section also contains the Centronics parallel port. And last, both disk controllers use the same ribbon cable to talk to the same drives. This makes it possible to have an OSI formatted disk in one drive and an MFM format disk in the other, or you could read an OSI disk into memory and write it back out to the same diskette in some other format. The switching is controlled by software.

The CxP Software

The CxP will run any software you currently have with few exceptions. In most cases, few if any changes are needed. As I said, I am currently running this hardware, so I know what it can or can't do. Here, as an example, is 65D. The operating system 65D will run as is except for the default CPU speed on track 0 (see V3.3 bug in the October '85 issue of PEEK[65]). The changes to use the new serial and parallel ports fit over the old drivers with room to spare. The only programs that cause trouble are those that address old hardware directly. Of all the software tested, only on program does not run - the OSI Assembler/Editor. If you have a 65816 and bank addressing hardware, then the OSI Assembler gets confused. You see, the 65816 is 100% downward compatible with the 6502 when in the emulation mode. However, opcodes that on the old 6502 were illegal or ignored are now legal and perfectly valid on the 65816. It seems that the OSI Assembler does something wrong that on the 6502 and 65C02 had no effect. However, on the 65816 it messes up the line numbers and only if the bank addressing circuits are installed. (**Editor's Note:** Illegal or unrecognized opcodes are supposed to cause the microprocessor to halt or produce some other specific response. In the early days of the 6502, it was noted by astute hackers that some of these illegal opcodes in fact had real function and would execute. By deduction and trial and error, inquisitive programmers noted those opcodes and what they did. However, the designer of the 6502 never supported these opcodes - presumably for reasons of reliability. I have heard in the past that the OSI Assembler in fact used such illegal opcodes. As one of the primary design goals of the 65C02 was to duplicate the 6502, including the errors and foibles, it is not altogether surprising that the OSI Assembler works on that chip. Neither is it surprising that the 65816 accepts only documented commands in the emulation mode. "Do as I say, not as I do", eh?) It is a simple bug and by the time you read this I will probably have found it. Any other assembler that I have tried works fine. In summary, this project is a starting point. It is up to us to write the software. No one will do it for us. If you are worried that this project might die because of lack of software, then buy a PC clone. I have invested two years and $500 into this project. I am not about to stop. If I have to write every scrap myself for my one-of-a-kind system, I will. I have already started the outline of a new operating system. It is very modular, so updating the code to 65816 can be done in sections with no effect to the user. The I/O is based on a personal BIOS much like CP/M, but even more so. It is another huge project that is likely to take several months. When it is completed, you can be sure that I will make it available as public or at least shareware.

Summary

The CxP is the ultimate in basement hardware. It is not, however, developed by a team of engineers with an R&D budget. It is just me and my basement 6502. I have tested the design as much as my budget can stand. I now want to make the board available to other OSIers. This will encourage more software development than I can do myself and it will help offset the money I put into this one board. The cost of making the boards in such small quantities is high. Most of the board manufacturers I talked to are not even interested. The ones that are want about $60 for a bare board of less than average quality. At this time, I cannot give a firm price. If you wish to write or phone, I'm sure I will know more by the time this is printed. The detail of the traces on this board is very fine. Only the above average solderer could handle putting this board together. I am willing to sell bare boards, but only to people who assure me that they are capable and that they realize I can guarantee nothing. What I am working on is producing a "lazy board". All parts that need to be soldered are included - you only have to buy the chips and plug them into the sockets. The boards will be tested and adjusted. At this time, the cost of a "lazy board" appears to be about $150.00 Canadian ($180-$200 U.S.). The cost of the components is currently around another $150.00, the most expensive being the 65816 at $30 and the two RAM chips at $25 each. The rest is relatively cheap.

One last note, it may sound like a lot of money to upgrade the OLD OSI, but look at it this way. The new IIgs from the company with a name like a fruit has the 65816, but running at only 2.8 MHz and it costs about $2000.00. They say 2.8, but it waits for RAM, so it is really around 2.5, and only for some hardware. For a lot of stuff it shifts down to 1 MHz. The CxP runs the 65816 at 4 MHz for all of the RAM and only shifts down to 2 MHz for the old OSI disk drives, video/keyboards, and the monitor ROM. It may not have lots of color graphics or super sound, but for raw speed, the CxP is in a class of its own.

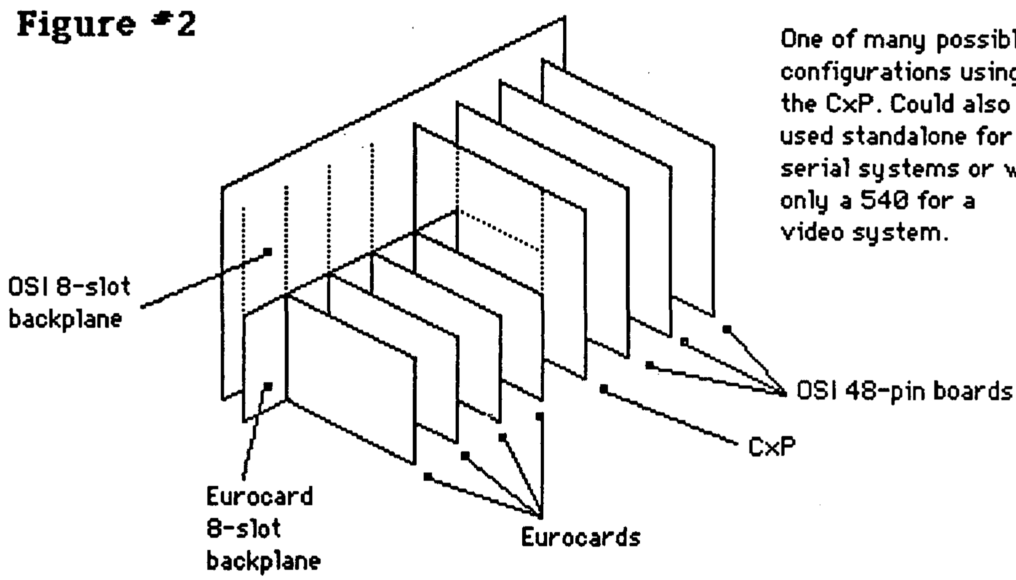
For your information, here is the complete list of books available at this time about programming the 65816. I cannot offer a fair review of the books since I only have two of them. However, the review I did see liked the two that I have the best. The M. Fischer book is very technical. It makes an excellent reference-style manual. The Lichty and Eyes book is, on the other hand, more of a tutorial. I have found that I tend to look things up in the Fischer book, and if I don't understand it, I go read about it in the other. The Lichty and Eyes book also contains a complete listing for DEBUG16, a 65816 machine code debugger and disassembler program. If is, of course, for the Apple, but the machine-dependent sections are clearly marked and documented. I will be very easy to convert it for OS-65D or whatever. Each time I pick up these books I find things the 65816 can do that I wasn't aware of. After all, isn't that why we play with these things? The continued discovery of new knowledge. It sounds good to me. Have fun reading!

65816/65802 Assembly Language Programming
by Michael Fischer
Osborne/McGraw-Hill
2600 Tenth Street
Berkeley, CA 94710
(415)-548-2805
$19.95

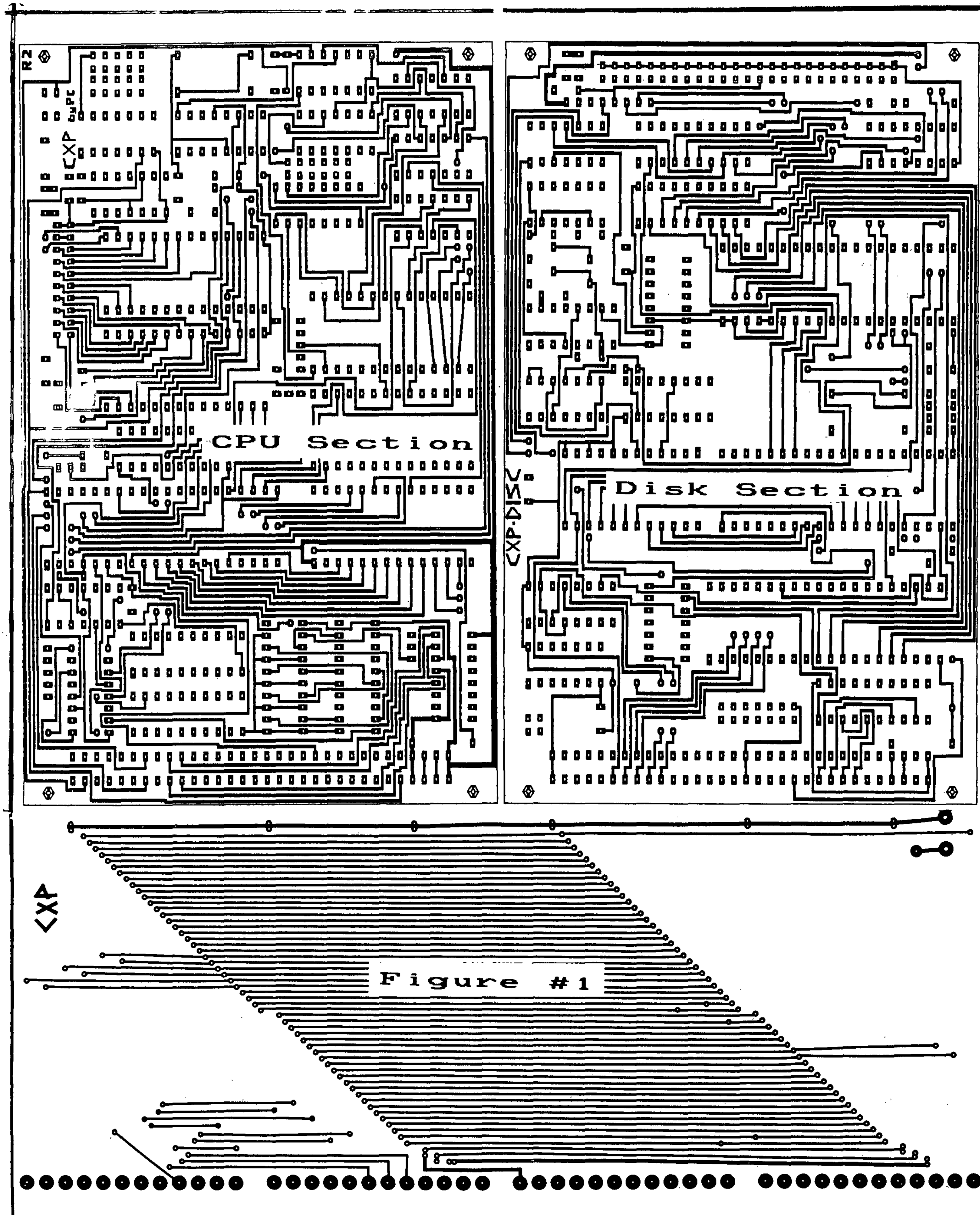
Programming the 65816 Microprocessor
Including the 6502, 65C02, and 65802
by Ron Lichty and David Eyes
Prentice-Hall
Englewood Cliffs, NJ 07632
(201)-592-2240

Programming the 65816
by William Labiak
Sybex, Inc.
2344 Sixth Street
Berkeley, CA 94710
(800)-227-2346

Figure #2



One of many possible configurations using the CxP. Could also be used standalone for serial systems or with only a 540 for a video system.


R2
<XP
CPU Section
Disk Section
<XP
Figure #1

Color+ Additions Part II

by John Horemans
TOSIE

(Editor's Note: These listings were inadvertantly omitted in the original article in the December issue. The sample programs John mentioned will be printed in a future issue. My apologies to you readers and to John.)

```
12130 CMDLST    .BYTE 5,'HPLOT',4,'SMOV'
12140           .BYTE 4,'PLOT',4,HLIN',4,'VLIN'
12150           .BYTE 4,'HCOL',3,'COL'
12160           .BYTE 4,'SSEL',4,'SCOL'
12170           .BYTE 4,'SPAT',6,'SPINIT'
12180           .BYTE 4,'SSIZ',6,'GRINIT'
12190           .BYTE 3,'HGR',2,'GR',5,'HBACK'
12200           .BYTE 4,'H',$A9,'IG',4,'L',$A9,'IG'
12210           .BYTE 5,'HMODE'
12220           .BYTE 4,'TCOL',6,'WINDOW'
12230           .BYTE 4,'HTAB',4,'VTAB',4,'HOME'
12240           .BYTE 5,'BLANK',3,'SCR'
12250           .BYTE 4,'CSET',4,'SCLR',3,'VOL'
12255           .BYTE 4,'WAVE',4,'PLAY',3,'OFF'
12256           .BYTE 3,'ATK',3,'DEC',3,'SUS',3,'REL'
12257           .BYTE 5,'PULSE',4,'SYNC',4,'RING',4,'DUMP'
12260           .BYTE 5,'MOVIE',3,'REC',3,'BOX',4,'SQRT',0    ; END
```

LISTING 1

```
1 POKE 14172,7: POKE 14170,16
2 MAX = PEEK(8960): DEST = MAX-24: POKE 133,DE: CLEAR: DE=PEEK(133)
3 POKE 2888,0: POKE 8722,0: POKE 50950,0
4 X=PEEK(10950): POKE 8993,X: POKE 8994,X: DIM AL%(76)
5 IF PEEK(57088)=224 THEN POKE 9794,37
6 DEF FNA(X)= 10*INT(X/16)+X-16*INT(X/16)
7 DEF FNB(X)= 16*INT(X/10)+X-10*INT(X/10)
8 CP=168*256:D=11897:F$="COLOR+":S=1: IF PEEK(CP+5)=165 THEN 16
9 DISK!"CA 2E79=12,"+RIGHT$(STR$(S),1)
10 FORI=DTOD+255STEP8:F1$="":FORJ=0TO5:F1$=F1$+CHR$(PEEK(I+J)):NEXTJ
11 IF F1$ = F$ THEN TT=FNA(PEEK(I+J)): I = 99999
12 NEXT I: IF S<2 THEN S=S+1: GOTO 9
13 TT$ = RIGHT$(STR$(TT+100),2): DISK!"CA A800="+TT$+",1"
14 TT=TT+1: TT$=RIGHT$(STR$(TT+100),2): DISK!"CA B000="+TT$+",1"
15 TT=TT+1: TT$=RIGHT$(STR$(TT+100),2): DISK!"CA B800="+TT$+",1"
16 DV=2: PRINT!(28)&(20,2)"OS-65D V3.3 & Color+ V2.0": PRINT
17 PRINT " 1-Dir 2-Create 3-Change 4-Delete 9-Color+       ";
18 INPUT "Which";S$: IF S$="9" OR S$="PASS" THEN 93
19 IF LEN(S$)>1 THEN RUN
```

```
20 S=INT(VAL(S$)): IF S<1 OR S>8 THEN 16
21 GOSUB 92: ON S GOSUB 24, 26, 38, 47
22 IF P$="PASS" OR P$="9" THEN 93
23 GOTO 16
24 PRINT "DIRECTORY:  ";: GOSUB 54: GOSUB 56
25 GOSUB 60: PRINT#DV: GOTO 59
26 PRINT "Create Utility - ";: GOSUB 54
27 GOSUB 92: INPUT "Filename";A$: IF A$="" THEN RUN
28 IF LEN(A$)<6 THEN A$=A$+" ": GOTO 28
29 S=5: GOSUB 60: IF E$="Y" THEN 89
30 IF NF=0 THEN 91
31 INPUT "# tks";N
32 FOR T1 = 0 TO 39-N+1: FOR TS = 0 TO N-1: IF AL%(T1+TS) THEN 35
33 NEXT TS
34 PRINT "** OK **": T2=T1+N-1: N=0: PRINT: GOTO 37
35 NEXT T1
36 PRINT "** NO ROOM FOR **": PRINT: GOTO 59
37 S=2: GOTO 60
38 PRINT "Rename";: GOSUB 54
39 INPUT "File";A$: IF A$="" THEN RUN
40 IF LEN(A$)<6 THEN A$=A$+" ": GOTO 40
41 S=5: GOSUB 60: IF E$="N" THEN 88
42 O$ = A$: PRINT
43 INPUT "New";A$
44 IF LEN(A$)<6 THEN A$=A$+" ": GOTO 44
45 GOSUB 60: IF E$="Y" THEN 89
46 S=3: GOTO 60
47 PRINT "Delete  on";: GOSUB 54
48 INPUT "File ";A$: IF A$="" THEN RUN
49 IF LEN(A$)<6 THEN A$=A$+" ": GOTO 49
50 S=5: GOSUB 60: IF E$="N" THEN 88
51 S=4: GOTO 60
52 P$="8": T1=1: T2=10: GOSUB 2152
53 T1=13: T2=27: GOTO 2153
54 INPUT " A,B,C, or D ";D$: IF D$="" THEN D$="A"
55 DISK!"SE "+D$: RETURN
56 INPUT "Printer";P$: IF LEFT$(P$+" ",1)="Y" THEN DV=4
58 RETURN
59 INPUT "Continue";P$: RETURN
60 NF=0: E$="N": FOR K = 0 TO 39: AL%(K)=0: NEXT: M=0
61 IF S=5 THEN PRINT: PRINT" wait.";
62 DISK!"CA 2E79=12,1": GOSUB 68
63 IF S>1 AND S<5 THEN DISK!"SA 12,1=2E79/1"
64 IF S>1 AND DE$="Y" THEN RETURN
65 DISK!"CA 2E79=12,2": GOSUB 68
66 IF S>1 AND S<5 THEN DISK!"SA 12,1=2E79/1"
67 RETURN
68 FOR I=11897 RO 12145 STEP 8
69 ON S GOSUB 72,80,76,78,80
70 IF NF=1 AND S=2 THEN GOSUB 87: E$="Y": RETURN
71 NEXT: RETURN
72 IF PEEK(I)=35 THEN NF=NF+1: RETURN
73 GOSUB 84: T1=FNA(PEEK(I+6)): T2=FNA(PEEK(I+7))
74 PRINT#DV, TAB(M);N$;" ";T1;" -";T2;
75 M=M+21: IF M>50 THEN M=0: PRINT#DV: RETURN
76 GOSUB 84: IF O$<>N$ THEN RETURN
77 E$="Y": GOTO 86
78 GOSUB 84: OF A$<>N$ THEN RETURN
79 E$="Y": A$="######": GOSUB 86: POKE I+6,0: POKEI+7,0: RETURN
80 IF PEEK(I)=35 THEN NF=NF+1: RETURN
```

```
81 GOSUB 84: IF N$=A$ THEN E$="Y": RETURN
82 T0 = FNA(PEEK(I+6)): T9 = FNA(PEEK(I+7))
83 FOR K = T0 TO T9: AL%(K)=-1: NEXT: RETURN
84 N$="": FOR J=I TO I+5: N$=N$+PEEK(J)): NEXT
85 PRINT " ";: RETURN
87 GOSUB 86: POKE I+6,FNB(T1): POKE I+7,FNB(T2): NF=255: RETURN
88 PRINT: PRINT"** ";CHR$(34);A$;CHR$(34);" not found **": GOTO 59
89 PRINT: PRINT"** ";CHR$(34);A$;CHR$(34);" exists ";: GOTO 59
90 PRINT "in the directory. **": PRINT: GOTO 59
91 PRINT: PRINT "** Directory Full **": PRINT: GOTO 59
92 PRINT !(28): RETURN
93 POKE 578,10: POKE 579,168: POKE 576,13: POKE 577,168
94 POKE 2470,32: POKE 2471,7: POKE 2472,168: POKE 2888,27
95 POKE 741,76: POKE 750,78: POKE 2073,173: POKE 2893,55: POKE 2894,8
96 POKE 13*4096+14*256,1: PRINT!(12)CHR$(13)!(15);: END
98 PRINT "No Color+ on this disk *** ERROR ***": END
```

Listing 2

```
10;*****************************************************
20;***  USED WITH BASIC+ OR OTHER HOOKS INTO BASIC TO  ***
30;***   DRIVE COMMODORE SID (6581) CHIP FROM BASIC    ***
40;***            BY JOHN HOREMANS TOSIE               ***
50;*****************************************************
60;***            PRINTER DUMP FOR COLOR+              ***
70;***            BY PAUL CHIDLEY TOSIE                ***
80;*****************************************************
90;** PUBLIC DOMAIN SOFTWARE NOT TO BE USED FOR PROFIT ***
100;****************************************************
110;* SID CHIP IS LOCATED AT $C4XX ON MY SYSTEM        ***
120;* ALSO USES SUBROUTINES IN COLOR+                  ***
130;****************************************************
140;
150; THS SID CHIP IS A READ ONLY CHIP (mostly)
160; so we need a copy of the registers in RAM
170 SIDTBL   .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0
180          .BYTE 0,0,0,0,0,0,0,0,0,0,0,0,0,0
190;
200 SCLR     LDX #27           ; SID CLEAR COMMAND
210          LDA #$2A
220 LCLR     STA SID,X
230          STA SIDTBL,X
240          DEX
250          BNE LCLR
260          JMP HRETRN
265;
270 VOL      LDA SIDTBL+24     ; VOLnn  VOLUME COMMAND
280          AND #$F0          ; GET TOP NYBBLE
290          STA SIDTBL+24
300          JSR GETNIB
310          ORA SIDTBL+24     ; PUT THEM TOGETHER
320          STA SIDTBL+24
330          STA SID+24
340          JMP HRETRN
350;
360 PLTABL   .BYTE 0,7,14
370 PLAY     JSR GETREG        ; PLAY r,nnnn
```

```
380            STX TEMP          ; SOUND #
390            JSR OFFDO
400            LDX TEMP
410            LDA PLTABL,X
420            STA TEMPX
430            JSR GETNUM
440            LDX TEMPX
450            STA SID,X
460            INX
470            LDA HIVAL
480            STA SID,X
490            INX
500            INX
510            INX
520            LDA SIDTBL,X
530            ORA #%00000001    ; BE SURE ON (BIT 0)
540            STA SID,X         ; TURN ON SOUND
550            JMP HRETRN
555;
560 WAVTBL     .BYTE 4,11,18
570 WVAL       .BYTE 16,32,64,128
580 WAVE       JSR GETREG
590            LDA WAVTBL,X
600            STA TEMPX         ; REG # FOUND
610            LDA #4            ; 4 POSSIBLE WAVE TYPES 1-SINE
620            JSR GETBYT        ; 2-TRIANGLE 3-PULSE 4-NOISE
630            TAX
640            DEX
650            BMI ZERROR
660            LDA WVAL,X
670            STA TEMP
680            LDX TEMPX         ; IS REG #
690            LDA SIDTBL,X
700            AND #$0F          ; CLEAR UPPER 4 BITS
710            ORA TEMP
720            STA SIDTBL,X
730            JMP HRETRN
740 ZERROR     JMP FCERR
745;
750 OFF        LDA #3
760            JSR GETBYT
770            TAX
780            BEQ ALLOFF
790            DEX
800            JSR OFFDO
810            JMP HRETRN
820 OFFDO      LDA WAVTBL,X      ; OFF SOUND IN X REG
830            TAX
840            LDA SIDTBL,X
850            STA SID,X
860            RTS
870 ALLOFF     LDX #0
880 OFFLP      STX TEMP
890            JSR OFFDO
900            LDX TEMP
910            INX
920            CPX #4
930            BNE OFFLP
940            JMP HRETRN
```

```
950 GETREG   LDA #3             ; GET THE REG#, CHECK COMMA
960          JSR BYTGET         ; VOICE# - 1
970          TAX                ; RETURNS IN X
980          DEX
990          BMI ZERROR         ; ERROR IF SOUND REG 0
1000         RTS
1001;
1010 SUSTBL  .BYTE 6,13,20
1020 SUS     JSR GETREG
1030         LDA SUSTBL,X
1040 HIHALF  TAX                ; REGISTER #
1050         STA TEMPX
1060         LDA SIDTBL,X
1070         AND #$0F
1080         STA SIDTBL,X
1090         JSR GETNIB
1100         ASL A
1110         ASL A
1120         ASL A
1130         ASL A              ; HIGH NYBBLE
1140         LDX TEMPX
1150         ORA SIDTBL,X
1160         STA SID,X
1170         STA SIDTBL,X
1180         JMP HRETRN
1185;
1190 ATKTBL  .BYTE 5,12,19
1200 ATK     JSR GETREG
1210         LDA ATKTBL,X
1220         BNE HIHALF
1230 DECAY   JSR GETREG
1240         LDA ATKTBL,X
1250 LOHALF  TAX
1260         STA TEMPX
1270         LDA SIDTBL,X
1280         AND #$F0           ; ERASE LOW 4 BITS
1290         STA SIDTBL,X
1300         JSR GETNIB
1310         LDX TEMPX
1320         ORA SIDTBL,X
1330         STA SID,X
1340         STA SIDTBL,X
1350         JMP HRETRN
1355;
1360 REL     JSR GETREG
1370         LDA SUSTBL,X
1380         BNE LOHALF
1385;
1390 PULTBL  .BYTE 2,9,16
1400 PERR    JMP FCERR
1410 PULSE   JSR GETREG
1420         LDA PULTBL,X       ; GET REG#
1430         STA TEMPX
1440         JSR GETNUM
1450         LDX TEMPX
1460         STA SID,X
1470         INX
1480         LDA HIVAL
1481;
```

```
1490 SYNC    JSR GETREG
1500         LDA WAVTBL,X
1510         STA TEMP
1520         LDA #1
1530         JSR GETBYT
1540         TAX
1550         DEX
1560         BMI ZSYNC
1570         LDX TEMP
1580         LDA SIDTBL,X
1590         EOR #%00000010
1600         STA SIDTBL,X
1610         JMP HRETRN
1620 ZSYNC   LDX TEMP
1630         LDA SIDTBL,X
1640         AND #%11111101
1650         STA SIDTBL,X
1660         JMP HRETRN
1661;
1670 RING    JSR GETREG
1680         LDA WAVTBL,X
1690         STA TEMP
1700         LDA #1
1710         JSR GETBYT
1720         TAX
1730         DEX
1740         BMI ZRING
1750         LDX TEMP
1760         LDA SIDTBL,X
1770         ORA #%00000100
1780         STA SIDTBL,X
1790         JMP HRETRN
1800 ZRING   LDX TEMP
1810         LDA SIDTBL,X
1820         AND #%11111011
1830         STA SIDTBL,X
1840         JMP HRETRN
1850;
1860         PRTOUIT = $249F    ; DEVICE #4 PARALLEL
1865; **** FIX FOR DEVICE # USED FOR PRINTER ****
1870         PRINT  = $2343
1880         PRT2HX = $2D92
1890         CRLF   = $2D6A
1900         STROUT = $2D73
1910;
1920;******* PRINTER DUMP **** PR1011/PX1080,1091
1921;******* THESE ARE EPSON COMPATIBLE PRINTERS (they say)
1930         TEMPR  = $20       ; TEMPORARY STORAGE
1940         RCOUNT = TEMPR+1   ; ROW COUNT
1950         CCOUNT = RCOUNT+1  ; COLUMN COUNT
1960         TABLE2 = CCOUNT+1  ; 8 BYTES OF STORAGE
1970         TABLE3 = TABLE2+8  ; 8 BYTES OF STORAGE
1980;
1990;
2000 DUMPST  JSR INTPRT         ; INITIALIZE PRINTER
2010         JSR INTVDP         ; INITIALIZE VDP TO 1ST
2020         LDA #24            ; SET # ROWS
2030         STA RCOUNT         ; STORE IN ROW COUNT
2040 LOOP1   JSR PLHDR          ; PRINT START OF LINE H
```

```
2050         JSR PLINE        ; JSR PRINT A LINE OF H
2060         JSR PCRLF        ; END THE LINE WITH A CR
2070         DEC RCOUNT       ; DECREMENT ROW COUNT
2080         LDA RCOUNT
2090         BNE LOOP1        ; BRANCH IF NOT DONE
2100         LDY #0
2110 ENDLP   LDA TABLE4,Y
2120         CMP #$FF
2130         BEQ END
2140         INY
2150         JSR PRTOUT
2160         JMP ENDLP
2170 END     JMP HRETRN
2180 TABLE4  .BYTE 10,10,10,10,10,27,50,27,108,0,$FF ; 4 CR, RESET
2200;
2210 PLINE   LDA #32          ; SET COLS
2220         STA CCOUNT       ; STORE IN COLUMN COUNT
2230 LOOP2   JSR PBLOCK       ; PRINT A BLOCK OF HI-RES
2240         DEC CCOUNT       ; DECREMENT COLUMN COUNT
2250         LDA CCOUNT
2260         BNE LOOP2        ; BRANCH IF NOT DONE
2270         RTS
2280;
2290 PBLOCK  LDY #8           ; SET Y TO # BYTES
2300 LOOP3   LDA VDP          ; GET DATA
2310         STA TABLE2,Y     ; SAVE IT
2320         JSR DELAY
2330         DEY              ; DECREMENT COUNT
2340         BNE LOOP3        ; BRANCH IF NOT DONE
2350         JSR CONVRT       ; JSR CONVERT DATA FORM
2360         LDY #8           ; SET COUNT .
2370 LOOP4   LDA TABLE3,Y     ; CONVERTED DATA
2380         JSR PRTOUT       ; SEND IT TO PRINTER
2390         DEY              ; DECREMENT COUNT
2400         BNE LOOP4        ; BRANCH IF NOT DONE
2410         RTS
2420;
2430 CONVRT  LDX #8           ; INVERT AND FLIPOVER DATA
2440 CON1    LDY #8           ; PUTS CONVERTED DATA IN
2450 CON2    LDA TABLE2,Y     ; TABLE3 FROM TABLE2
2460         ROL A
2470         STA TABLE2,Y
2480         ROL TEMPR
2490         DEY
2500         BNE CON2
2510         LDA TEMPR
2520         STA TABLE3,X
2530         DEX
2540         BNE CON1
2550         RTS
2560;
2570 INTPTR  LDY #$00         ; INITIALIZE FROM TABLE
2580         BNE LOOP4        ; BRANCH IF NOT DONE
2590 INT1    LDA TABLE1,Y
2600         CMP #$FF
2610         BEQ INTEND
2620         JSR PRTOUT
2630         INY
2640         JMP INT1
```

```
2650 INTEND  RTS
2660 TABLE1  .BYTE 27,65,8,27,108,19,10,10,10,10,$FF
2670;
2680 INTVDP  LDA #0               ; FIRST ADDRESS IN VIDEO
2690         STA VDP+1
2700         STA VDP+1
2710         JSR DELAY            ; MIN 2 uSEC. REQUIRED
2720         RTS
2730;
2750;
2760 PLHDR   LDA #27              ; START OF A LINE
2770         JSR PRTOUT
2780         LDA #42
2790         JSR PRTOUT
2800         LDA #$5
2810         JSR PRTOUT
2820         LDA #0
2830         JSR PRTOUT
2840         LDA #1
2850         JSR PRTOUT
2860         RTS
2870;
2880 PCRLF   LDA #$0D
2890         JSR PRTOUT
2900         LDA #$0A
2910         JSR PRTOUT
2920         RTS
2930         .FILE CP65D5
```

```
30000 ;*************** MC DRAW ROUTINES ****************
30010 ; APRIL 5, 1986   J. HOREMANS
30020 ;************************************************
30030 ; CP65D6
30040 ;
30060           *=*+2
30070 ;************************************************
30080 VAL4     JSR GETPAR          ; GET X,Y, ADD XORIG,YORIG
30090          LDA XVAL
30100          STA XSTART
30110          STA TEMPR
30120          LDA YVAL
30130          STA YSTART
30140          STA YEND
30150          JSR CHKCOM
30160          JSR GETPAR
30170          LDA XVAL
30180          STA XEND
30190          LDA YVAL
30200          CMP YSTART
30205          BEQ DONREQ
30210          BCS CONT4           ; SWAP Y VALUES IF NEEDED
30220          TAX
30230          LDA YSTART
30240          STA YVAL
30250          STX YSTART
30260          STX YEND
30270 CONT4    JSR LINE
```

```
30280         RTS
30290 REC     JSR VAL4
30300 CONREC  JSR BUMP
30310         JSR LINE
30320         JMP CNREC
30330 BUMP    LDA TEMPR
30340         STA XSTART
30350         INC YEND
30360         LDA YEND
30370         STA YSTART
30380         CMP YVAL
30390         BEQ DONREC
30400         RTS
30410 DONREC  JSR LINE
30420         PLA
30430         PLA
30440         JMP HRETRN
30450 BOX     JSR VAL4          ; FIND 4 PARAMETERS X1,Y1,X2,Y2
30460 CNBOX   JSR BUMP
30470         JSR SETBIT
30480         LDA XEND
30490         STA XSTART
30500         LDA YEND
30510         STA YSTART
30520         JSR SETBIT
30530         JMP CNBOX
30540         CPEND = *
30550         .FILE CP65DS
```

```
20010 ; MOVIES assembly language driver
20020 ; syntax:  MOVIE expr (give start address)
20030 ; Data is read from memory starting at location NNNN
20040 ; A byte >= $C0 specifies a special command:
20050 ;    $FF - signals end of picture
20060 ;    $E0-$EF - set HCOLOR to $0 - $F (# AND $0F)
20070 ;    $C0 - next 2 bytes are coordinate to move to
20080 ; Coordinates are specified as 8-bit Y then X.
20090 ; Y is inverted (subtracted from $C0).
20100 ;
20110 ;
20130 MOVIE   JSR GETNUM
20140         STA PTCHAR+1        ; LOW VALUE OF ADDR
20150         LDA HIVAL
20160         STA PTCHAR+2
20170;
20180 NXTPT   JSR PTCHAR
20190         CMP #$FF            ; DONE?
20200         BEQ DNMOVI
20210         CMP #$E0
20220         BCC NOCOLR
20230         AND #$0F
20240         STA COLOR
20250         JMP NXTPT
20260;
20270 NOCOLR  CMP #$C0            ; MOVE?
20280         BNE DRAW
20290         JSR PTCHAR          ; YES!
20300         EOR #$FF
20310         CLC
20320         ADC #$C0
20330         STA YSTART
20340         JSR PTCHAR
20350         STA XSTART
20360         JMP NXTPT
20370;
20380 DRAW    EOR #$FF
20390         CLC                 ; DRAW
20400         ADC #$C0
20410         STA YEND
20420         JSR PTCHAR
20430         STA XEND
20440         JSR LINE
20450         JMP NXTPT
20460;
20470 PTCHAR  LDA $1234           ; GET X OR Y COORDINATE
20480         INC PTCHAR+1
20490         BNE RETURN
20500         INC PTCHAR+2
20510 RETURN  RTS
20520 DNMOVI  JMP HRETRN
20530         .FILE CP65D6
```

Sluething BASIC

by Robert S. Runyon
7015 Brookview Road
Hollins, VA 24019

For your readers that use OS-65D V3.2, here is a short BASIC program that is best described by direct observation. Enter the BASIC source code as listed and run the program. Then load your favorite hard-to-crack BASIC program and LIST it. You will find the listing much easier to follow than before.

The BASIC loader routine installs a patch in an unused corner of the page zero swap buffer, and connects it to intervene during execution of the LIST command. Once installed, it will remain so unless you cold-start BASIC or re-boot.

The patch works with v3.3 also, but OSI has already used the swap buffer for some of their own code. This means you will have to find another place to hide the patch, which will run as written anywhere in available memory. Just change the base location in line 140 and modify the POKEd values in lines 170 and 180 to correspond.

(Editor's Note: To use this program with v3.3, it is probably best to reserve a section of high memory to hold the patch. PEEK(133) holds the current maximum RAM address MSB. Fetch that value and POKE 133 with one less. Then multiply the original value by 4096 to obtain the base address to be installed in line 140. Use the original value again in line 180, and 0 in line 170.)

inserts blank line
after GOTO, Return, END

```
100 REM - MODIFIED LIST ROUTINE
110 POKE 2976,44
120 FOR N = 0 TO 44
130 READ X
140 POKE 12154+N, X
150 NEXT N
160 POKE 1816, 32
170 POKE 1817, 122
180 POKE 1818, 47
190 END
200 DATA 160,4,177,172,240,34,201,128,240,27,201,136,240,23
210 DATA 201,137,240,19,201,141,240,15,201,138,240,14,200
220 DATA 201,58,240,227,177,172,208,243,240,3,32,115,10,160,0
230 DATA 177,172,96
```

```
2F7A  A004    LDY #$04
2F7C  B1AC    LDA ($AC),Y
2F7E  F022    BEQ $2FA2
2F80  C980    CMP #$80
2F82  F01B    BEQ $2F9F
2F84  C988    CMP #$88
2F86  F017    BEQ $2F9F
2F88  C989    CMP #$89
2F8A  F013    BEQ $2F9F
2F8C  C98D    CMP #$8D
2F8E  F00F    BEQ $2F9F
2F90  C98A    CMP #$8A
2F92  F00E    BEQ $2FA2
2F94  C8      INY
2F95  C93A    CMP #$3A
2F97  F0E3    BEQ $2F7C
2F99  B1AC    LDA ($AC),Y
2F9B  D0F3    BNE $2F90
2F9D  F003    BEQ $2FA2
2F9F  20730A  JSR $0A73
2FA2  A000    LDY #$00
2FA4  B1AC    LDA ($AC),Y
2FA6  60      RTS
```

END
GOTO
Run
Return

Write for PEEK!

Software Spectacular!

C1P/Superboard Cassettes

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

Assortment of 10 for just $20.00!

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

C4P/C8P Cassettes

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

Sargon II Chess Software

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

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

Extended Monitor

Cassette version for all systems
Regular $50.00

Sale Price $15.00

```
10; DISK FILE EDITOR
20; PART 2
30;
40 ASC     CLC
50         ADC #'0
60         CMP #'9+1
70         BCC ASCII1
80         ADC #06
90 ASCII1 RTS
100;
110; STRING INPUT ROUTINE
120;
130 GETSTR LDY #$00
140 GETS1  JSR $0587        BASIC
150        BEQ GETS1
160        CMP #CR
170        BEQ GETS2
180        CMP #SP
190        BCC GETS1        SUPRESS <CTRL> CHARS
200        CMP #DEL
210        BEQ BKSPC
220        CMP #DEL+$20
230        BEQ BKSPC
240        STA BUF,Y
250        JSR OUTDO
260        INY
270        BNE GETS1
280 GETS2  STY STRLEN      SAVE LENGTH
290        STA BUF,Y
300        JMP CRDO         CLEAN UP AND QUIT
310;
320 STRLEN .BYTE $00
330 DIRTY  .BYTE $00
340 PAGNUM .BYTE $00,$00,$00,$00
350 XCOORD .BYTE $00
360 YCOORD .BYTE $00
370 MASTER .BYTE $00
380 OF$    .BYTE ' of','xxxxxxxxxx',$00
390;
400 BKSPC  TYA
410        BEQ GETS1
420        PHA
430        JSR STROUT
440        .BYTE BS,SP,BS,0
450        PLA
460        TAY
470        DEY
480        JMP GETS1
490;
500 GETANS JSR GETSTR     GET "YES" OR "NO" FROM
USER
510        LDA BUF
520        JSR CASECK
530        CMP #'Y
540        RTS
550;
560 DRSEL  JSR STROUT
570        .BYTE 'DEVice ? ',$00
580        JSR GETSTR
590        LDA STRLEN
600        CMP #$01
610        BNE DRSEL
620        LDA BUF
630 DRS1   JSR CASECK
640        CMP #'A
650        BCC DRSEL
660        SBC #$41
670        CMP #$04
680        BCC DRSEL2
690        SBC #$04
700        CMP #$04
710        BCC DRSEL1
720        BCS DRSEL
730 DRSEL1 ORA #$80
740 DRSEL2 STA DRIVE
750        STA DISCN
760        RTS
770;
780 STROUT PLA
790        STA STROU1+1
800        PLA
810        STA STROU1+2
820        LDY #$01
830 STROU1 LDA $FFFF,Y
840        BEQ STROU2
850        JSR OUTDO
860        INY
870        BNE STROU1
880 STROU2 TYA
890        SEC
900        ADC STROU1+1
910        STA STROU3+1
920        LDA STROU1+2
930        ADC #$00
940        STA STROU3+2
950 STROU3 JMP $FFFF
960;
970 PRBYTE PHA
980        LSR A
990        LSR A
1000       LSR A
1010       LSR A
1020       JSR ASC
1030       JSR OUTDO
1040       PLA
1050       AND #$0F
1060       JSR ASC
1070       JMP OUTDO
1080;
1090 CASECK CMP #'a
1100       BCC CASE1
1110       CMP #'z+1
1120       BCS CASE1
1130       EOR #$20
1140 CASE1 RTS
1150;
1160 FNDFIL JSR DRSEL       SELECT A DRIVE
1170 FNDF1 JSR STROUT       ASK FOR NAME
1180       .BYTE CR,LF
1190       .BYTE 'File Name ? ',$00
1200       JSR GETSTR
1210       LDA STRLEN       FETCH RESPONSE LENGTH
1220       CMP #$07         CHECK IT
1230       BCS FNDF1        BAD! ==> TRY AGAIN!
1240       LDA BUF          FETCH 1ST CHARACTER
1250       JSR CASECK       MAKE IT ALL CAPS
1260       CMP #'A          CHECK FOR LEGAL
1270       BCC FNDF1
1280       CMP #'Z+1
1290       BCS FNDF1
1300 FNDREM JSR DIRSU       SET UP DIR READ
1310       LDY #$00         INIZ
```

```
1320 FNDF2  LDA BUF,Y        FETCH A CHARACTER
1330        JSR CASECK       MAKE IT ALL CAPS
1340        CMP #CR         END OF STRING?
1350        BEQ FNDF3       YES! ==>
1360        STA CURFIL,Y     SAVE IT
1370        INY              BUMP INDEX TO CURFIL
1380        BNE FNDF2        NOT YET, LOOP!
1390 FNDF3  LDA #SP          YES! LOAD A <SP>
1400 FNDF4  CPY #$06         AT LENGTH MAX?
1410        BEQ FNDF5        YES! ==> FNDF5
1420        STA CURFIL,Y     NO, CLEAR THIS SPOT
1430        INY              BUMP INDEX
1440        BNE FNDF4        AND LOOP
1450;
1460 FNDF5  JSR GETDSK       READ DIRECTORY
1470        BEQ FNDF10
1480        JMP FNDF11
1490 FNDF10 LDA #DIRBUF      INIZ
1500        STA POKER          SET POKER TO DIRBUF
1510        LDA #DIRBUF/256  HANDLE MSB AS WELL
1520        STA POKER+1
1530 FNDF6  LDY #$00         INIZ
1540 FNDF7  LDX #$00         INIZ
1550 FNDF8  LDA (POKER),Y      FETCH ENTRY
CHARACTER
1560        BEQ FNDFC        END OF DIR! ==>
1570        CMP CURFIL,X     COMPARE TO CURFIL
1580        BNE FNDF9        NO MATCH! ==> FNDF9
1590        INY              BUMP FETCH POINTER
1600        INX              BUMP MATCH COUNTER
1610        CPX #$06         MATCHED ALL 6 ?
1620        BNE FNDF8        NOT YET, LOOP! ==>
FNDF8
1630        BEQ FNDFD        FOUND IT!
1640 FNDF9  TYA              PUT INDEX IN ACC.
1650        AND #$F0         MASK TO ENTRY
LENGTH(S)
1660        CLC
1670        ADC #$10         ADD ONE ENTRY LENGTH
1680        TAY              PUT IT BACK IN Y
1690        BNE FNDF7        LOOP ON NO PAGING ==>
FNDF7
1700        LDA COUNT        FETCH PAGE COUNTER
1710        CLC
1720        ADC #$01         ADD ONE
1730        STA COUNT        SAVE IT
1740        BCC FNDFA        WATCH PAGING
1750        INC COUNT+1      BUMP NLSB ON PAGING
1760        BNE FNDFA
1770        INC COUNT+2      BUMP MSB ON PAGING
1780 FNDFA  LDA COUNT+2      FETCH MSB
1790        CMP SIZE+2       CMP TO DIRECTORY SIZE
1800        BNE FNDFB        NOT THERE YET ==>
FNDFB
1810        LDA COUNT+1      FETCH NLSB
1820        CMP SIZE+1       CHECK IT
1830        BNE FNDFB        NO ==> FNDFB
1840        LDA COUNT        FETCH LSB
1850        CMP SIZE         CHECK
1860        BNE FNDFB        NO ==> FNDFB
1870 FNDFC  SEC              YES! SHOW NO MATCH IN
DIR!
1880        RTS              AND QUIT!
1890 FNDFB  JSR DBUMP        BUMP DIR POINTERS
1900        JMP FNDF5        AND LOOP!
1910;
1920 FNDFD  TYA
1930        AND #$F0
1940        CLC
1950        ADC POKER
1960        STA POKER
1970        BCC FNDFE
1980        INC POKER+1
1990 FNDFE  LDY #$09
2000        LDX #$00
2010        STX CADDR
2020        STX STADDR
2030        STX ENADDR
2040        STX FACHI          SAVE FOR "OF"
2050 FNDFF  LDA (POKER),Y
2060        STA STADDR+1,X
2070        STA CADDR+1,X
2080        INY
2090        INX
2100        CPY #$0C
2110        BNE FNDFF
2120        CLC
2130        LDA (POKER),Y
2140        STA FSIZE
2150        STA FACLO          SAVE FOR "OF"
2160        ADC STADDR+1
2170        STA ENADDR+1
2180        INY
2190        LDA (POKER),Y
2200        STA FSIZE+1
2210        STA FACMLO         SAVE FOR "OF"
2220        ADC STADDR+2
2230        STA ENADDR+2
2240        INY
2250        LDA (POKER),Y
2260        STA FSIZE+2
2270        STA FACMHI         SAVE FOR "OF"
2280        ADC STADDR+3
2290        STA ENADDR+3
2300        CLC
2310        RTS
2320;
2330 FNDF11 JSR STROUT
2340        .BYTE CR,LF
2350        .BYTE 'DISK ERROR',CR,LF,$00
2360        SEC
2370        RTS
2380;
2390 WRITE  LDA #$00
2400        STA DIRTY          CLEAR BUFFER DIRTY
FLAG!
2410        STA DUN+5          # BYTES LSB
2420        STA DUN+7          MEMORY ADDR. LSB
2430        LDA #$01           ONE PAGE BUFFER!
2440        STA DUN+6          SET # BYTES MSB
2450        LDA DRIVE          GET SAVE FILE DRIVE
2460        STA DUN            SET IT
2470        LDA #BUFFER/256    LOAD BUFFER ADDR MSB
2480        STA DUN+8          SAVE AS MEMORY MSB
2490        LDY #$00           INIZ
2500 WRITE1 LDA CADDR,Y        FETCH FILE START ADDI
2510        STA DUN+1,Y        GIVE TO 65U
2520        INY                BUMP INDEX
2530        CPY #$04           SENT ALL 4?
2540        BNE WRITE1         NOT YET, LOOP!
2550        JSR PUTDSK         AND WRITE BUFFER TO
DISK
```

```
2560        BNE WRITE2       ERROR? ==> WRITE2
2570        RTS              ALL O.K., QUIT
2580 WRITE2 JMP FCERR
2590;
2600 READ   LDA #$00
2610        STA DIRTY        CLEAR BUFFER DIRTY
FLAG!
2620        STA DUN+5        # BYTES LSB
2630        STA DUN+7        MEMORY ADDR. LSB
2640        LDA #$01         GET BUFFER SIZE
2650        STA DUN+6        SET # BYTES MSB
2660        LDA DRIVE        GET SAVE FILE DRIVE
2670        STA DUN          SET IT
2680        LDA #BUFFER/256  LOAD BUFFER ADDR. MSB
2690        STA DUN+8        SAVE AS MEMORY MSB
2700        LDY #$00         INIZ
2710 READ1  LDA CADDR,Y      FETCH CURRENT FILE
ADDR
2720        STA DUN+1,Y      GIVE TO 65U
2730        INY              BUMP INDEX
2740        CPY #$04         SENT ALL 4?
2750        BNE READ1        NOT YET, LOOP!
2760        JSR GETDSK       AND READ BUFFER TO
DISK
2770        BNE READ2        ERROR? ==> READ2
2780        RTS              ALL O.K., QUIT
2790 READ2  JMP FCERR
2800;
2810 NOTF   JSR STROUT
2820        .BYTE 'FILE NOT FOUND',CR,LF,$00
2830        JMP FCERR
2840;
2850 MAKEOF JSR NORMAL
2860        JSR ASCII
2870        LDY #$00
2880 MAKEO1 LDA STACK,Y
2890        STA OF$+3,Y
2900        BEQ MAKEO2
2910        INY
2920        BNE MAKEO1
2930 MAKEO2 RTS
2940;
2950 EDIT   JSR FNDFIL        FIND THE FILE
2960        BCS NOTF          NO LUCK? --> NOTF
2970        JSR MAKEOF        MAKE "OF" STRING
2980        LDA #$00
2990        STA MASTER        CLEAR MASTER DIRTY
FLAG
3000        TAY
3010 EDIT1  STA PAGNUM,Y
3020        INY
3030        CPY #$03
3040        BNE EDIT1
3050        INC PAGNUM        START AT PAGE "1"
3060 EDIT2  JSR READ         READ IN 1ST PAGE
3070 EDIT3  JSR PGDSP
3080 EDIT4  JSR CLRCMD
3090        LDA #CHOICE
3100        LDY #CHOICE/256
3110        JSR OUTSTR
3120 EDIT41 JSR $0587
3130        LDY #$00
3140        STY POSCNT
3150        JSR CASECK
3160 EDIT5  CMP EDCMD,Y
3170        BEQ EDIT6
3180        INY
3190        CPY #EDVEC-EDCMD
3200        BNE EDIT5
3210 EDIT51 LDA #7
3220        JSR OUTDO
3230        JMP EDIT41
3240 EDIT6  TYA
3250        ASL A
3260        TAY
3270        LDA EDVEC,Y
3280        STA EDIT7+1
3290        LDA EDVEC+1,Y
3300        STA EDIT7+2
3310 EDIT7  JMP $FFFF
3320;
3330 EDCMD  .BYTE '><NAQ'
3340;
3350 EDVEC  .WORD NEXT,PREV,NEDIT,AEDIT,QUIT
3360;
3370 CHOICE .BYTE '> = NEXT, < = PREVIOUS, '
3380        .BYTE 'N = NUMERIC EDIT, '
3390        .BYTE 'A = ASCII EDIT, '
3400        .BYTE 'Q = QUIT',$00
3410;
3420 CHO1$  .BYTE 'Q = QUIT, M = MOVE CURSOR, '
3430        .BYTE 'E = EDIT',$00
3440;
3450 MOVE$  .BYTE 'Q = QUIT, U = UP, '
3460        .BYTE 'D = DOWN, L = LEFT, '
3470        .BYTE 'R = RIGHT',$00
3480;
3490 EDIT$  .BYTE '<ESC> = STOP',$00
3500;
3510 QUIT   LDA DIRTY
3520        BEQ QUIT1
3530        JSR WRITE
3540 QUIT1  LDA MASTER
3550        BEQ QUIT2
3560        JSR FLUSH
3570 QUIT2  LDA #$00
3580        TAY
3590        JMP GIVAYF
3600;
3610 NEXT   LDA CADDR+1
3620        CLC
3630        ADC #$01
3640        STA FACLO
3650        LDA CADDR+2
3660        ADC #$00
3670        STA FACMLO
3680        LDA CADDR+3
3690        ADC #$00
3700        CMP ENADDR+3
3710        BNE NEXT2
3720        LDA FACMLO
3730        CMP ENADDR+2
3740        BNE NEXT2
3750        LDA FACLO
3760        CMP ENADDR+1
3770        BNE NEXT2
3780        JMP EDIT51
3790 NEXT1  LDA DIRTY
3800        BEQ NEXT2
3810        JSR WRITE
3820 NEXT2  INC PAGNUM
3830        BNE NEXT3
```

```
3840         INC PAGNUM+1
3850         BNE NEXT3
3860         INC PAGNUM+2
3870 NEXT3   INC CADDR+1
3880         BNE NEXT4
3890         INC CADDR+2
3900         BNE NEXT4
3910         INC CADDR+3
3920 NEXT4   JMP EDIT2
3930;
3940 PREV    LDA CADDR+3
3950         CMP STADDR+3
3960         BNE PREV1
3970         LDA CADDR+2
3980         CMP STADDR+2
3990         BNE PREV1
4000         LDA CADDR+1
4010         CMP STADDR+1
4020         BNE PREV1
4030         LDA CADDR
4040         CMP STADDR
4050         BNE PREV1
4060         JMP EDIT51        YELL AT USER!
4070 PREV1   LDA DIRTY
4080         BEQ PREV2
4090         JSR WRITE
4100 PREV2   LDA CADDR+1
4110         SEC
4120         SBC #$01
4130         STA CADDR+1
4140         LDA CADDR+2
4150         SBC #$00
4160         STA CADDR+2
4170         LDA CADDR+3
4180         SBC #$00
4190         STA CADDR+3
4200         LDA PAGNUM
4210         SEC
4220         SBC #$01
4230         STA PAGNUM
4240         LDA PAGNUM+1
4250         SBC #$00
4260         STA PAGNUM+1
4270         LDA PAGNUM+2
4280         SBC #$00
4290         STA PAGNUM+2
4300         JMP EDIT2        JUMP TO MAIN LOC. !
4310;
4320 PGDSP   JSR STROUT
4330         .BYTE 27,28,'Page #',$00
4340         LDA PAGNUM
4350         STA FACLO
4360         LDA PAGNUM+1
4370         STA FACMLO
4380         LDA PAGNUM+2
4390         STA FACMHI
4400         LDA PAGNUM+3
4410         STA FACHI
4420         JSR NORMAL
4430         JSR ASCII
4440         LDA #STACK
4450         LDY #STACK/256
4460         JSR OUTSTR
4470         LDA #OF$
4480         LDY #OF$/256
4490         JSR OUTSTR        SHOW SIZE
4500         JSR CRDO
4510         JSR CRDO
4520;
4530         JSR STROUT
4540         .BYTE '    00 01 02 03 04 05 06 07 '
4550         .BYTE '08 09 0A 0B 0C 0D 0E 0F',$00
4560         JSR CRDO
4570         JSR CRDO
4580         LDA #BUFFER
4590         STA PTR
4600         LDA #BUFFER/256
4610         STA PTR+1
4620         LDA #$00
4630         STA COUNT
4640 PGDSP1  LDA COUNT
4650         JSR PRBYTE
4660         LDA #SP
4670         JSR OUTDO
4680         JSR OUTDO
4690         LDY #$00
4700 PGDSP2  LDA (PTR),Y
4710         JSR PRBYTE
4720         LDA #SP
4730         JSR OUTDO
4740         INY
4750         CPY #$10
4760         BNE PGDSP2
4770         LDA #SP
4780         JSR OUTDO
4790         JSR OUTDO
4800         LDY #$00
4810 PGDSP3  LDA (PTR),Y
4820         BMI PGDSP4-2
4830         CMP #SP
4840         BCS PGDSP4
4850         LDA #SP
4860 PGDSP4  JSR OUTDO
4870         INY
4880         CPY #$10
4890         BNE PGDSP3
4900         JSR CRDO
4910         LDA #$10
4920         CLC
4930         ADC PTR
4940         STA PTR
4950         INC COUNT
4960         BCC PGDSP1
4970         JSR CRDO
4980         JMP CRDO
4990;
5000 AEDIT   LDA #BUFFER
5010         STA PTR
5020         LDA #BUFFER/256
5030         STA PTR+1
5040         JSR HOME
5050 AEDIT1  JSR CLRCMD         CLEAR COMMAND LINE
5060         LDA #CHO1$
5070         LDY #CHO1$/256
5080         JSR OUTSTR         DISPLAY CHOICES
5090         JSR CPUT7
5100 AEDIT2  JSR $0587
5110         JSR CASECK
5120         CMP #'Q       QUIT?
5130         BNE AEDIT3
5140         JMP EDIT4       YES! DONE!
5150 AEDIT3  CMP #'M       MOVE CURSOR?
```

```
5160        BNE AEDIT4
5170        JSR CMOVE
5180        JMP AEDIT1
5190 AEDIT4 CMP #'E       EDIT?
5200        BEQ AEDIT6       YES! HERE!
5210 AEDIT5 LDA #7
5220        STA POSCNT
5230        JSR OUTDO
5240        JMP AEDIT2
5250 AEDIT6 JSR CLRCMD       CLEAR CMD LINE
5260        LDA #EDIT$
5270        LDY #EDIT$/256
5280        JSR OUTSTR
5290        JSR CPUT7
5300 AEDIT7 JSR $0587
5310        CMP #ESC
5320        BEQ AEDIT8
5330        JSR CPUT          PUT CHAR. AT (X,Y)
5340        JMP AEDIT7        AND LOOP!
5350 AEDIT8 JMP AEDIT1       RESET/HOME!
5360;
5370 NEDIT  LDA #BUFFER
5380        STA PTR
5390        LDA #BUFFER/256
5400        STA PTR+1
5410        JSR HOME
5420 NEDIT1 JSR CLRCMD       CLEAR COMMAND LINE
5430        LDA #CHO1$
5440        LDY #CHO1$/256
5450        JSR OUTSTR       DISPLAY CHOICES
5460        JSR CPUT7
5470 NEDIT2 JSR $0587
5480        JSR CASECK
5490        CMP #'Q       QUIT?
5500        BNE NEDIT3
5510        JMP EDIT4        YES! DONE!
5520 NEDIT3 CMP #'M       MOVE CURSOR?
5530        BNE NEDIT4
5540        JSR CMOVE
5550        JMP NEDIT1
5560 NEDIT4 CMP #'E       EDIT?
5570        BEQ NEDIT6       YES! HERE!
5580 NEDIT5 LDA #7
5590        STA POSCNT
5600        JSR OUTDO
5610        JMP NEDIT2
5620 NEDIT6 JSR CLRCMD       CLEAR CMD LINE
5630        LDA #EDIT$
5640        LDY #EDIT$/256
5650        JSR OUTSTR
5660        JSR CPUT7
5670 NEDIT7 LDA #$00
5680        STA TMP          CLEAR NUMBER
5690        STA TMP1        CLEAR NYBBLE COUNT
5700 NEDIT8 JSR $0587        GET KEYPRESS
5710        CMP #ESC         DONE?
5720        BEQ NEDIT1      YES! ==>
5730        JSR CASECK        MAKE IT CAPS
5740        CMP #'0           LEGAL?
5750        BCC NEDITC
5760        CMP #'9+1
5770        BCC NEDIT9
5780        CMP #'A
5790        BCC NEDITC
5800        CMP #'G
5810        BCS NEDITC
5820 NEDIT9 SEC
5830        SBC #'0
5840        CMP #$A
5850        BCC NEDITA
5860        SBC #$07
5870 NEDITA LDX TMP1
5880        BEQ NEDITB
5890        ASL TMP
5900        ASL TMP
5910        ASL TMP
5920        ASL TMP
5930        ORA TMP
5940        JSR CPUT
5950        JMP NEDIT7          AND LOOP!
5960 NEDITB STA TMP
5970        INC TMP1
5980        JMP NEDIT8
5990 NEDITC LDA #7
6000        STA POSCNT
6010        JSR OUTDO
6020        JMP NEDIT8
6030;
6040 CPUT   PHA                 SAVE CHARACTER ON
STACK
6050        LDA YCOORD          GET Y COORDINATE
6060        ASL A
6070        ASL A
6080        ASL A               *16!
6090        ASL A
6100        CLC
6110        ADC XCOORD          ADD X COORDINATE
6120        TAY                PUT IN Y REGISTER
6130        PLA
6140        STY TMP            SAVE FOR LATER
6150        STA (PTR),Y        SAVE IN BUFFER
6160        LDA #$01
6170        STA DIRTY          SHOW BUFFER DIRTY
6180        STA MASTER          SHOW MASTER WRITE
6190        STA POSCNT          FOOL BASIC HERE!
6200        LDA XCOORD
6210        ASL A               *2
6220        ADC XCOORD          +1  = *3
6230        ADC #$04            (XCOORD*3) + 4
6240        STA PRAT1+1         GIVE TO PRINT AT
6250        LDA YCOORD
6260        CLC
6270        ADC #$04            YCOORD+4
6280        STA PRAT2+1         GIVE TO PRINT AT
6290        JSR PRAT
6300        LDY TMP
6310        LDA (PTR),Y        RETRIEVE BYTE
6320        JSR PRBYTE
6330        LDA XCOORD
6340        CLC
6350        ADC #54
6360        STA PRAT1+1
6370        JSR PRAT
6380        LDY TMP
6390        LDA (PTR),Y
6400        BMI CPUT5
6410        CMP #SP
6420        BCS CPUT6
6430 CPUT5  LDA #SP
6440 CPUT6  JSR OUTDO
6450        INC XCOORD
6460        LDA XCOORD
```

```
6470        CMP #$10
6480        BCC CPUT7
6490        LDA #$00
6500        STA XCOORD
6510        INC YCOORD
6520        LDA YCOORD
6530        CMP #$10
6540        BCC CPUT7
6550        LDA #$00
6560        STA YCOORD
6570 CPUT7  LDA XCOORD
6580        ASL A
6590        ADC XCOORD
6600        ADC #$03
6610        STA PRAT1+1
6620        LDA YCOORD
6630        CLC
6640        ADC #$04
6650        STA PRAT2+1
6660        JMP PRAT          AND EXIT THRU PRINT AT
6670;
6680 HOME   LDA #$00
6690        STA XCOORD
6700        STA YCOORD
6710        JMP CPUT7
6720;
6730 PRAT   LDA #27
6740        JSR OUTDO
6750        LDA #17
6760        JSR OUTDO
6770 PRAT1  LDA #$FF
6780        STA POSCNT
6790        JSR OUTDO
6800 PRAT2  LDA #$FF
6810        JMP OUTDO
6820;
6830 CLRCMD LDA #0
6840        STA PRAT1+1
6850        LDA #22
6860        STA PRAT2+1
6870        JSR PRAT
6880        LDA #27
6890        JSR OUTDO
6900        LDA #15
6910        JMP OUTDO
6920;
6930 CMOVE  JSR CLRCMD
6940        LDA #MOVE$
6950        LDY #MOVE$/256
6960        JSR OUTSTR
6970        JSR CPUT7
6980 CMOVE1 JSR $0587
6990        JSR CASECK
7000        CMP #'Q
7010        BNE CMOVE3
7020 CMOVE2 RTS
7030 CMOVE3 CMP #'U
7040        BNE CMOVE4
7050        LDA YCOORD
7060        BEQ CMOVE7
7070        DEC YCOORD
7080        JSR CPUT7
7090        JMP CMOVE1
7100 CMOVE4 CMP #'D
7110        BNE CMOVE5
7120        LDA YCOORD
7130        CMP #$F
7140        BEQ CMOVE7
7150        INC YCOORD
7160        JSR CPUT7
7170        JMP CMOVE1
7180 CMOVE5 CMP #'L
7190        BNE CMOVE6
7200        LDA XCOORD
7210        BEQ CMOVE7
7220        DEC XCOORD
7230        JSR CPUT7
7240        JMP CMOVE1
7250 CMOVE6 CMP #'R
7260        BNE CMOVE7
7270        LDA XCOORD
7280        CMP #$F
7290        BEQ CMOVE7
7300        INC XCOORD
7310        JSR CPUT7
7320        JMP CMOVE1
7330 CMOVE7 LDA #7
7340        STA POSCNT
7350        JSR OUTDO
7360        JMP CMOVE1
7370;
7380        *=*/256+1*256
7390 BUFFER=*
7400        *=*+$100
7410;
7420        .END DKED1
```

Sam's Service Manuals

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

C1P Sam's Regular 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

Don't forget to add postage!

65V Primer

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

Regular 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 OS-65D. However, even when it was free, there was little documentation accompanying the disks. If you've been looking for instructions on these two programs, this is the book for you!

Regular 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 series of microprocessors. While not OSI-specific, this book contains many valuable algorithms for solving problems in machine code using the microprocessors available in OSI computers.

Regular 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 software for these systems. The book also provides information on how to program the C3 series using the Z-80 and 6800 microprocessors. Regular Price: $9.95 Sale Price: $6.00

User Guides

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

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

Cryptograms

by Gerald M Van Horn
640 SW Addison Avenue
Junction City, OR 97448

After reading the latest PEEK[65]s, it seems that most readers are not interested in the frivolous type of programs I have here, but here goes anyway.

I used to work on the Cryptograms found in most newspapers, but gave it up because the time consuming necessity of marking each occurance of a letter with what I thought the letter should be, then finding out I was wrong and having to go back and erase all my marks and put in the new one, which also might be wrong. Now I have taken up working these puzzles again because of the included program. It makes it easy to use the trial and error method to a solution.

A few notes on the program: First, it is simple enough to be adaptable to any of the OSI machines. I adapted it from a program listing in BYTE magazine (Feb. 1984, p383). However, I added the SAVE facility because sometimes I have interruptions and it is handy to be able to save my work on disk until I can return. I started entering the program with a buffer for this reason. Two tracks are all that is necessary on an 8" disk. (**Editor's Note:** When you begin to install this program, don't forget to run the program "CHANGE" after you run "CREATE", so that one disk buffer will be installed at the start of BASIC's workspace.)

The program is straightforward. I think there are plenty of REMarks to explain the operation. The asterisk is used with GET because otherwise the program would write it as a word of the cryptogram. Also SAVE and QUIT must be typed out, or else the program recognizes "S" or "Q" as just another letter to be changed. "C*" is clear screen and "Q*0" sets my C4P with Rick's Hooks into BASIC, in the 32x32 screen presentation. Have fun!

```
10 REM   CRYPTO ASSISTANT     BYTE FEB '84 P384
20 C* : REM- CLEAR SCREEN
22 Q*0 : REM- SELECT 32 X 32 SCREEN
30 DIM GN(95), AM(95), Z$(20), LL(20)
40 PRINT "TYPE THE CRYPTOGRAM (OR GET*)"
41 PRINT "END WITH SPACE<CR>"
42 PRINT: POKE 2972, 13: POKE 2976,13: REM- DISABLE COMMA & COLON
50 INPUT Z$(A)
60 IF Z$(A) = "GET*" THEN PF=2: GOTO 120
70 LL(A) = LEN(Z$(A)): REM- LL(A) IS LINE LENGTH
80 IF LL(A)<>0 THEN A = A+1: GOTO 50
90 A = A-1
100 GOSUB 390
110 FOR X = 0 TO A: X$(X)=Z$(X):NEXT: REM- MAKE CRYPTO & WKSPACE =
120 C*
122 REM- PRINT CRYPTO AND WORKSPACE
130 FOR Y = 0 TO A: PRINT Z$(Y): PRINT X$(Y): PRINT: NEXT
140 IF PF<>1 THEN 150
141 REM- SAVES CRYPTO AND WORKSPACE TO DISK
142 DISK OPEN, 6, "CRFILE"
143 PRINT #6, A
144 FOR Y= 0TOA: PRINT#6,LL(Y): PRINT #6,Z$(Y): PRINT#6,X$(Y):NEXT
146 DISK CLOSE, 6: GOTO 160
150 IF PF<>2 THEN 160
151 REM- RECOVER CRYPTO & WORKSPACE FROM DISK
152 DISK OPEN,6,"CRFILE": INPUT #6, A
153 FOR Y= 0TOA: INPUT#6,LL(Y): INPUT#6, Z$(Y):INPUT#6,X$(Y): NEXT
154 DISK CLOSE, 6
156 PF=0: GOSUB 390: GOTO 120
160 PRINT
180 PF=0: REM- SAVE OR GET FLAG
200 PRINT "CRYPTOGRAM: ";
202 REM- PRINT THE MOST USED CHARACTERS
210 FOR Y = 1 TO 5
220 IF G(Y) <> 0 THEN PRINT CHR$(G(Y));" ";
230 NEXT
232 PRINT
```

```
250 PRINT "PLAIN TEXT: E T L A N"
252 PRINT "TEST LENGTH:"; TEXT
260 PRINT: PRINT "ENTER THE LETTER TO BE"
262 PRINT "CHANGED (OR QUIT OR SAVE)"
264 INPUT A$
266 REM- FINDS ALL A$'S AND CHANGE WKSPACE A$ TO B$
270 IF A$="SAVE" THEN PF=1: GOTO 120
275 IF A$="QUIT" THEN 999
290 PRINT "ENTER THE LETTER IT IS TO BE"
292 INPUT "CHANGED TO"; B$
320 FOR Y = 0 TO A
322 X$ = X$(Y)
330 FOR I = 1 TO LL(Y)
332 C$ = MID$(Z$(Y),I,1)
334 IF C$=A$ AND I=1 THEN X$= B$+MID$(X$(Y),I+1,LL(Y)-1): GOTO 348
340 IF C$=A$ THEN X$= LEFT$(X$(Y),I-1)+B$+MID$(X$(Y),I+1,LL(Y)-I)
348 X$(Y) = X$
350 NEXT I
360 NEXT Y
370 GOTO 120
390 C*: PRINT "COUNTING LETTERS"
400 FOR Y = 0 TO A
410 FOR X = 1 TO LL(Y)
420 Q$ = MID$(Z$(Y),X,1): Q = ASC(Q$): AM(Q) = AM(Q) +1
430 NEXT X
440 NEXT Y
450 FOR X = 0 TO A: TEXT = TEXT + LEN(Z$(X)): NEXT X
460 TEXT = TEXT - AM(32)
480 FOR Y = 1 TO 5
490 FOR X = 65 TO 90
500 IF AM(X) => G(Y) THEN G(Y) = AM(X): GN(Y)=X
510 NEXT X
520 AM(GN(Y)) = 0
530 NEXT Y
540 RETURN
999 POKE 2972, 58: POKE 2976,44: Q*1: END
```

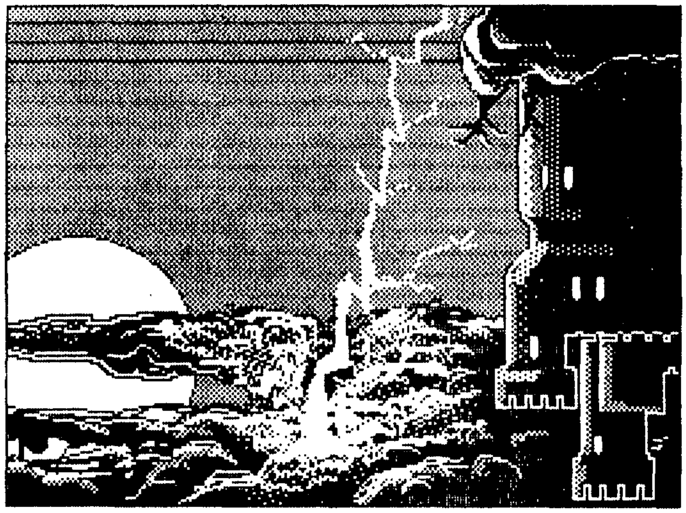
RLE Graphics

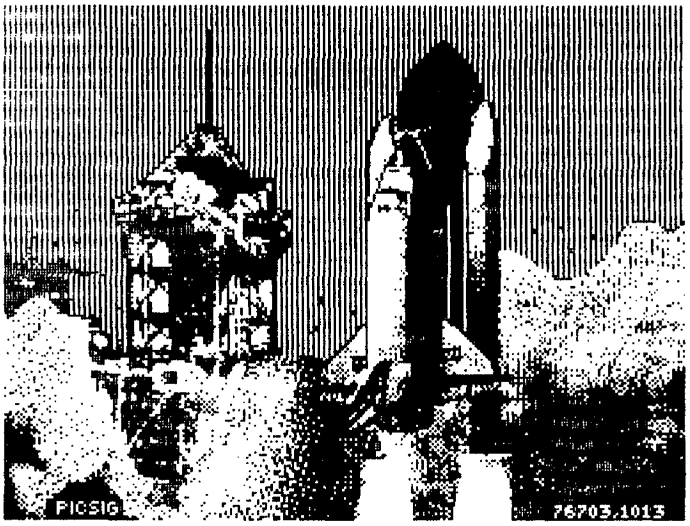
by Paul Chidley
TOSIE

Run Length Encoded (RLE) graphics files are "graphic monochrome pictures" encoded using only ASCII characters. The encoding scheme is simple, in most cases efficient and best of all, standard between all machines. Over the past few months, I have seen several articles in various magazines about RLE graphics. All of them showed these great picturers, but none of them showed what a programming would need to know to display them. So with the thought that "a picture is worth a thousand words", I have included several pictures derived from RLE files and printed with my OSI system. Now that you see what your trusty OSI can do, here is the information I found that is needed to write programs to work with RLE files.

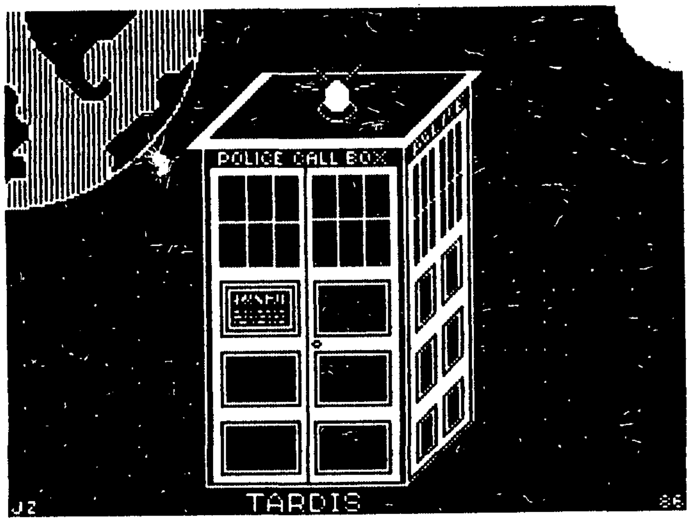
An RLE file starts with an opening sequence of three characters, an <ESC>ape, a "G" (for "Graphics" mode), and a third character - either "H" for high resolution, or "M" for medium resolution. High resolution RLE produces a bit map that is 256x192. Medium resolution produces a bit map of 128x96.

The opening sequence is then followed by a data sequence. The data sequence is best thought of as a set of ASCII character pairs. The first character represents the number of OFF (or background) pixels, and the second character represents the number of ON (or foreground) pixels. Each character is equal to the number of pixels plus 32 decimal. In other words, the smallest ASCII value used is 32 decimal (the <SPACE> character) and this represents 32-32=0 pixels. Since the parity bit is ignored, the largest character value is 127 representing 127-32=95 pixels. However, 127 decimal is the DEL (destructive backspace) character which is a non-printable character usually given special treatment by terminals and/or drivers. The highest character used should therefore be restricted to 126





decimal. This sequence of data pairs continues until the total number of pixels needed (to completely fill the bit map) have been accounted for. The end of the file is marked by a 7 decimal, <BELL> character, followed by a closing sequence of <ESC>, a "G", and an "N" (for "normal"). The pixels are then assumed to move left to right and top to bottom. When plotting the pixels, the end of one row wraps to the start of the next row.



Standard RLE File Format

$1B	<ESC>
$47	"G"
$48	"H" for "High" or "M" for "Medium" res.
$nn,$nn	ASCII pair where "$nn" is equal to or
$nn,$nn	greater than $20 and less than or
$nn,$nn	equal to $7E.
...,...	
...,...	
...,...	
$07	<BELL> (optional)
$1B	<ESC>
$47	"G"
$4E	"N"

Now that you know the format, you would probably like to display some of these pictures on your OSI. Well, this will require some kind of graphics medium. Most dot-matrix printers are well-suited to displaying these pictures. The ones included here were printed on my Panasonic KX-P1091. For displaying on a monitor, I use my Color+ board. Its resolution just happens to be 256x192. The first night I got my hands on an RLE file, I wrote a quick little program in BASIC that could display the picture in as little as 2 to 5 minutes. Not bad by most home computer standards, but not good enough for OSIers. So the next night I wrote a machine code version that takes only 2 to 5 seconds. My original version could display a picture (normal or inverse) and print it. Since then, John Horemans (also of TOSIE) has added the ability to take what is on the Color+ pattern screen and convert it to an RLE file on disk.I included the program so Rick may upload it to the OSI area on CompuServe. The program is public domain (for non-profit use), so enjoy. If you cannot download it, send me a disk and postal remuneration. And don't forget that US stamps don't work up here in Canada.

Transferring files between machines is a snap, so get your Apple and C64 friends to download the public software for their machines and you'll be trading pictures in no time. Yes, even an IBM or Mac can do this. When downloading actual RLE files, I just do an ASCII read and save the info in a RAM buffer. The CompuServe software, however, stops after the <BELL> to give the user time to view the picture. You must hit <RETURN> when the file has been completely transmitted this way. Since my software is, however, counting pixels, I don't even bother looking for the closing sequence anyway. This isn't really important, but I thought I'd mention it.

A last few points. The current version of the program "RLEGM" (RLE Graphics Manager) requires the Color+. It cannot, at this time, be used to print files unless the Color+ is also present. There is no real reason it can't. We just never wrote it to do that. If you don't already have a Color+, you may be out of luck. TOSIE once offered an OSI 16-pin I/O version, but we sold all 20 of them. Bob Ankeney of Generic Computer Products in Portland, OR is the creator of the board and may be able to help. The third choice would be to wait. For the past two years I have been desparately trying to get a source for the V-9938 chip from Yamaha. This is the video controller used in the MSX-II machines common in Japan. The chip is fully software compatible with the older TI-9918 series as used on the TI-99, Color+, etc. It adds 80 character text display, double the resolution, more color, more sprites, etc. I already have the design for a new board that includes this chip with an IBM PC keyboard interface. If I can ever find a source for the chip, we'll be set. Until then, enjoy RLE.

For more information, check the Picture Support Forum on CompuServe ("GO PICS"). Section 0 of their Data Library holds several files that describe the RLE standard, as well as several utility programs that could be converted for OSI. Besides PICS, you can find RLE graphics in several other areas of CompuServe, including the CompuServe CB Interest Forum (GO CBIG), the FBI's Ten Most Wanted list (GO FBI), weather maps (GO AWX-4), and others.

(**Editor's Notes:** RLE evolved with CompuServe's graphics protocols for their VIDTEX terminal programs. The <ESC>"G" sequences are defined in that standard to allow services like the FBI 10 Most Wanted List, interactive games, weather maps, and others to display information graphically, rather than just as simple text in real time. Extending that standard to a file format only expands the possible uses, however there are some subtle differences when the user is working from a file in a CompuServe Data Library.

A VIDTEX-compatible terminal program automatically converts the incoming characters into a graphics display as soon as the proper <ESC> sequence is received. Such programs would not be bothered by characters that others might treat as a backspace or whatever, as they would be in the "graphics mode", not the "text mode". But non-VIDTEX users *might* have trouble. In any event, this is only crucial when the terminal software being used thinks it must try to display every character received. When you use the (R)ead command in a CompuServe Data Library, the contents of that file are simply spewed out. If you want to save or manipulate that information, it is up to you and your terminal program to capture it in some way.

Of course, telecommunication involves phone lines and a whole host of other variables which can induce errors in transmission. To overcome this, special communications methods known as error-checking protocols have been developed. CompuServe supports three error-checking protocols for transferring files to and from their network. When using one of these file transfer protocols, the receiving terminal program may have trouble printing a character it receives, but it will still accurately record the character. And since many files will be composed of 8-bit data, these protocols could eliminate all limits on what characters can be in a RLE file for all practical purposes. However, since there is a standard you will probably have to follow it in order to be sure that others will be able to use files that you create.

The 256x192 image did not spring into use by random chance. It is the highest resolution of the older Apple II, which was the lowest common denominator among popular micros when VIDTEX was designed. Of course, since that time much higher resolutions have become commonplace. Several extensions to the RLE standard have been discussed, but they will probably never become widespread. CompuServe is in the process of developing and introducing a new graphics protocol which will be announced later this year. But even after that announcement, I'm sure that the current RLE format will be widely used for a long time.)

AD$

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

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

DELIVER TO: april 87

GOODIES for OSI Users!

PEEK (65)
The Unofficial OSI Users Journal

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

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

POSTAGE MAY VARY FOR OVERSEAS

Name ______________________
Street ______________________
City ______________ State ________ Zip ________