PEEK (65)

The Unofficial OSI Users Journal

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

$1.75

JULY 1984
VOL.5, NO.7

INSIDE

Column One

It's hard to know where to begin with this issue. If I had to pick just three items they would have to be:

1. Ken Schacter's review of DOS/65. Now there is a review you can really get your teeth into (and you only get the first half this month).

2. We're very excited about Rick Trethewey's Assembler Language course. Not only does Rick (Systems Operator for the OSI Special Interest Group on CompuServe) know his stuff, but he has written it in such a way that there is no longer an excuse for even fumble-foot beginners not to become functionally familiar with assembly code. We urge you to stick with it in the months to come.

3. The announcement of PEEK's second attack on free software listings. Last year, large portions of two issues were devoted to software. With your help, we hope to do the same thing again. After all, it's software that makes the machine run. As we have said before, don't be bashful about your programms - even if a similar one was listed last year. Who knows, yours might be tenfold better!

Things seem to be looking up at ISOTRON too. Following up on the COMDEX show: twenty some current dealers made the long trip and the show generated well over 100 serious dealer inquiries, of which about a quarter have already submitted papers. This swelling of the dealer ranks can only bode good for you and me. The list of dealers is in preparation now and will appear here as soon as it is available - so you will know where to get help and service.

Vertical market packages seem to be the thing at OSI now. A medical/dental package by Medical Business Systems is currently being shown at a show in San Francisco and OSI is in negotiations with four other well known OSI dealers for additional packages.

There's a new Official OSI word-processor for OS-U called Text Processor 2 (TP-2). After studying nearly a dozen, they picked Tom Jablonski's, much overhauled package. Look for a review in PEEK in the coming months.

Have you had trouble trying to figure out Planner Plus? The new manual is out under the name of KeyCalc. As soon as we see it, we will let you know if it really does the job and is worth the $50 price tag.

Speaking of prices, the Johny-come-latelies are going to pay through the nose for OS-U upgrades. What used to cost about $150 has gone up to $700. We hope that they are not serious!

At long last, PEEK has joined the stalwarts on CompuServe's OSI SIG. There is an awful lot out there and we are just learning our way around, but what we see, we like. What does it all mean to you? First of all, you are invited to join the hundred or so regulars and who knows how many others. Second, here is a new avenue for you to communicate with us. Messages and letters will be received and answered there (and appear in PEEK as well). And you can address articles and programs to us too.

Articles have appeared elsewhere on The Source and CompuServe and thus their general services should be well known, but little has been said about the OSI SIG. We'll try to have some of the details for you next month.

As if all that wasn't enough, just breeze through the ads for new data bases, second source boards and things plus new machines. Somebody must be doing something right. Hang in there. We will be at your side.

Eddie

For the vacationing Al.

****SPECIAL****
PEEK SOFTWARE ISSUE

One of the most often heard complaints about OSI is the lack of software. We don't believe it! We have talked to and received letters from too many of you who are proud as punch about your little "ditty". But, you don't think it's good enough or professional enough to put your name on it. All the same, it might just be the thing that someone else has been crying for and doesn't know where to find it or even if it exists at all.

If you are in the business arena, you probably have even less exposure than the "hackers" to what's available - besides you haven't had the time to go digging beyond your local dealer.

So PEEK(65) has decided to give every program author (hacker and pro alike) another opportunity to say, "here's what I have". That's right, PEEK is planning again to **devote a large segment of the October issue to software.** That sounds a long way off, but remember that the deadline for the October issue is **September 1st** - and that's not far away!

Business, games, utilities, neat little tricks and anything else that can be called a program. And the listing will be entirely FREE.

Listings will be sorted by Basic, Type and Machine to make it easy for you to zero in on the programs that may be of interest to you.

Each listing will have an encoded "head line" that should tell you everything you need to determine if it is interesting and will run on your machine. Next will be the program name and the author's name and address. Lastly, will be the author's description and any special comments. So what will the listing actually look like? Try this little gem!

Copyright ©1984 by **PEEK (65)** Inc. All Rights Reserved.
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

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

....

B/1.43/2/82/MR/M/D/12/1ØØØ
WONDER ACCOUNTING SYSTEM
John Q. Smith
123 Flatire St.
Cupcake, NZ 12345
123-456-7890

This system will handle up to three A/R and four R/P accounts at one time. Complete record locking, provided that no more than one user at a time is active. Average storage space required is 8MB. Mammoth overhaul required to run on SSII cassette system. And that makes 10 lines.

....

What's all the "gooble-de-gook" on the first line? Well, (B) it's a business package, under BASIC vers. No. (1.43) on a (2) C3A/B or 2ØØ series with a minimum of (82) two 8" FD's. It's (MR) Multi-user with Record locking, supported by (M) modem and sold by (D) dealers. There are (12) 12 x 1Ø = at least 12Ø copies in use at (1ØØØ) bucks retail. If that doesn't give you enough to tell you if you're interested, we give up!

Got you interested? Just "X" the appropriate boxes and fill in the remaining blanks on the form on the back page of this issue. If you have more than one program to submit (we certainly hope that you do), please feel free to make photo-copies of the form - one for each program!

The hard part will be writing a description that will not exceed PEEK's physical limits (10 lines, each not to exceed 30 characters). We would like to give you more room, but 10 lines of carefully chosen words should be adequate to whet the appetite.

If your software is not directly supported by you, the author, please fill in the DEALER ADDRESS as well as your own address block. If both blocks are filled in, only the dealer address will appear in PEEK. The dealer address may be either the selling dealer or an address where those inquiring may get a list of vendors.

Now it is up to you. Our guess is that our mail box will be full, but don't let us down and make liars out of us.

★

ADDING A PRINTER DRIVER TO THE UCSD SYSTEMS

By: William Beshures
294 Milford St., Apt. 32
Rochester, NY 14615

The UCSD System was originally developed at the University of California, San Diego. SofTech Microsystems markets an upgraded system based on the initial UCSD operating system. Currently SofTech is marketing version IV of the operating system. The version that was (is?) available for OSI machines is version IIs (version II, swapable). This version of the operating system supports computers with less than 64K; however, it requires 48K and two disk drives. Due to limited memory, some of the operating system routines may have to be made disk resident. When they are disk resident, they are only read into memory when they are required. The system has an excellent user interface and is a very good environment for learning PASCAL. Note that it is very similar to Apple PASCAL. The operating system itself is written in PASCAL with the exception of the interpreter and the BIOS; the interpreter and the BIOS are written in assembler. The interpreter does many things, but its primary job is to execute the psuedo-machine language that the PASCAL compiler generates. The BIOS is a group of routines to handle input and output to the hardware devices peculiar to a particular computer; it is the Basic Input/Output System (basic meaning 'simple', not the language).

The UCSD System that I obtained from SofTech Microsystems contained a device driver for a printer that was a 'do nothing routine'; it indicates to the operating system that the printer is off-line. When I purchased a printer for my C4P, I thought it might be nice to be able to use it with the PASCAL operating system. 'The UCSD System Supplemental Users Document for Use with the Ohio Scientific C3, C4 and C8', provided with the UCSD System, presents an example 'empty' printer driver. This example is useful in illustrating how to interface a device driver with the operating system. The supplement also briefly describes the process of linking new device drivers to the BIOS (Basic Input/ Output System). In the text that follows, I will describe this process using the 6850 ACIA (Asychronous Communications Interface Adapter) located at hexadecimal addresses FC00 and FC01.

First, locate the diskette labeled UTIL2: that was provided by SofTech Microsystems with the UCSD System. On this diskette are the files that are needed to create a new SYSTEM.BIOS. These files include:

SQUISH.CODE - a utility to compress the new BIOS routines

BIOS.CODE - an incomplete SYSTEM.BIOS that contains the code to initialize and halt the system, and to handle the disk drives

INTCON. CODE - code necessary to handle console functions using the internal video and keyboard of the C4 and C8

EXTCON.CODE - code necessary to handle console functions using an external terminal

EMPTYPRN.CODE - the code that we will replace

EMPTYREM.CODE - an empty REMOTE: device driver that indicates to the UCSD System that the REMOTE: device is off-line

EMPTYUSR.CODE - a similar routine that indicates that the user-defined device is offline

Second, enter the source contained in Listing #1. This routine will use the serial port controlled by the 6850 ACIA. When the code assembles without error, invoke F(iler and save it as UTIL2:PRINTER; F(iler will save SYSTEM.WRK.-TEXT and SYSTEM.WRK.CODE as PRINTER.TEXT and PRINTER.CODE respectively on the disk volume UTIL2:.

The third step is to link the BIOS routines together to form the new SYSTEM.BIOS. Invoke L(inker and respond to the 'Host file?' prompt with UTIL2:BIOS.CODE. Next enter the name of the assembled driver code files to the 'Lib file?' prompt one at a time. After all device driver code files have been identified, press <RETURN> at the 'Lib File?' prompt. When L(inker prompts for the name of the output file enter NEWSYS.BIOS (see Listing #2). L(inker will create a new code file that is destined to become SYSTEM.BIOS.

LISTING #1.

```
       .PROC PRINTER ; Bill's printer
                     ; Improved 01-May-83
                     ; 8 data bits
                     ; no parity
                     ; 2 stop bits
                     ; 1200 BAUD
;
; Define entry points that the system may reference
;
       .DEF  PRNINIT ; Printer Initialization
       .DEF  PRNSTAT ; Printer Status
       .DEF  PRNREAD ; Printer Read
       .DEF  PRNWRIT ; Printer Write
;
; Equate ACIA Status and Data register symbols to
;     their physical memory locations in the C4P
;     (note that leading zero is required)
;
ACIAST .EQU  0FC00H  ; ACIA Status Register
ACIADT .EQU  0FC01H  ; ACIA Data Register
;
; Initaialize the ACIA that controls the serial port
;     (note that default radix is hexadecimal)
;
PRNINIT LDA  #3      ; Master Reset
       STA   ACIAST  ; Reset ACIA
       LDA   #11     ; 8 data, no parity,
                     ; 2 stop, 1200 BAUD (hex.)
       STA   ACIAST  ; Initialize Status Register
       LDX   #0      ; Tell UCSD it's on-line
       RTS
;
; Always tell UCSD that printer status is good
;
PRNSTAT LDA  #0      ; Tell UCSD everything is O.K.
       RTS
;
; Don't bother reading from the printer
;
PRNREAD LDX  #9      ; Not wired for reads
       RTS
;
; Write a character to the printer
;
PRNWRIT PHA          ; Save character on stack
FOREVER LDA  ACIAST  ; Get ACIA status
       LSR   A       ; Logical Shift Right
       LSR   A       ; Again (to check status)
       BCC   FOREVER ; I hope it's on-line
       PLA           ; Get character from stack
       STA   ACIADT  ; Sent it to the printer
       LDX   #0      ; Return successful write
       RTS

       .END
```

The fourth step is to compress NEWSYS.BIOS to get rid of some extra stuff in the code file. The program to do this is SQUISH. However, SQUISH likes to consume large amounts of preciously spare memory and is likely to cause a fatal '*STK OFLOW*' error. '*STK OFLOW*' is the operating system's way of telling you that it ran out of memory; when it runs out of memory it re-initializes itself and you have to start over. Before we X(ecute SQUISH we must set part of the operating system to be disk resident. To do this enter 'S' for S(ystem State at the system command prompt:

Command: E(dit, R(un, F(ile, C(omp, L(ink, X(ecute,? [IIsl]

The following will appear:

Segment: SEGMENT#, Q(UIT

A) File handlers
   Disk 2 bytes

B) Soft buffer handlers
   Memory 910 bytes

C) Screen handlers
   Memory 1536 bytes

This indicates that of the segments that can be disk resident, one is disk resident and the other two are memory resident. We must set all three segments to be disk resident. Enter 'B' for the 'Soft buffer handlers' and the system will respond with the prompt 'New Status: M(emory, D(isk'. Enter 'D' to make that segment disk resident. Do the same for segment 'C'. Now X(ecute SQUISH. When SQUISH asks for an input file, respond with NEWSYS.BIOS. When it asks for the output file, respond with SQUISHED.-BIOS. After SQUISH has done its thing, restore the S(ystem State by setting segments 'B' and 'C' memory resident again.

Fifth, create a copy of your BOOT: disk and delete the SYSTEM.BIOS file on the new disk. Put the old disk in a safe place. Copy SQUISHED.-BIOS to the new BOOT: disk giving it the name SYSTEM.-BIOS. When the computer is booted using this disk the operating system will support a serial printer using the 6850 ACIA at FC00 and FC01 at 1200 bits per second. Boot the computer with this new disk and invoke F(iler. Select the V(olume option. F(iler will list the volumes on-line; this list should now include PRINTER:. Now you have a BIOS that supports a serial printer at 1200 bits per second. Your system has become 'more useful'. Produce listings 'til your heart's content.

LISTING #2.

Example L(inker execution

```
Linker [II.0 b1]

Host file? UTIL2:BIOS.CODE

Opening UTIL2:BIOS.CODE

Lib file? UTIL2:INTCON.CODE

Opening UTIL2:INTCON.CODE

Lib file? UTIL2:PRINTER.CODE

Opening UTIL2:PRINTER.CODE

Lib file? UTIL2:EMPTYREM.CODE

Opening UTIL2:EMPTYREM.CODE

Lib file? UTIL2:EMPTYUSR.CODE

Opening UTIL2:EMPTYUSR.CODE

Lib file? <RETURN>

Map name? <RETURN>

Reading  OSIBIOS

Reading  INTERNAL

Reading  PRINTER

Reading  NOREM

Reading  USRCLK

Output file? NEWSYS.BIOS
```

SOFTWARE REVIEW

AURORA SOFTWARE'S INTELLIGENT TERMINAL

By: Norman Thorsen
22225 Woodward Way NW
Poulsbo, WA 98370

The program is written in OSI BASIC with OS65D version 3.2 and is supplied with the systems disk, catalog of other software available, and six pages of documentation at a cost of $24.95. Two pages, plus part of a third, are command explanations and format specifications. The format is used to install your own codes for two-key transmission of entry codes such as Compuserve or Source requires. The other three pages are program listings, two pages for the terminal program and one for TMLIST.

Within the documentation is a memory map showing addresses for the program, the machine language I/O routine, disk buffers and indirect files. There is also a list of locations to POKE; top of memory (default is 24K), start of disk buffer, end of disk buffer, and indirect file start.

The program operates in full or half duplex by toggling ctrl-D. Exit to BASIC is accomplished by ctrl-E. In addition, there are nine more commands available:

ctrl-1 & 2 are command streams to be set up by the user with a choice of two formats. Separate bytes or text can be sent and may be up to 256 bytes long. Log on/off, set terminal characteristics, etc.

ctrl-3 clears the screen.

ctrl-4 will save incoming data in the indirect file.

ctrl-5 incoming data is sent to the disk buffer and to the disk.

ctrl-6 sends the indirect file.

ctrl-7 sends a disk file named by the operator.

ctrl-8 is used to download OSI BASIC programs.

ctrl-9 clears the screen and prints the menu of commands.

The information is concise and includes some examples. In particular, ctrl-1 and ctrl-2 are explained in great detail, as is ctrl-5. The inclusion of the byte stream and character stream will be appreciated by anyone who regularly has to log on to a system with passwords.

There is also a note to all users about pin 5 of the modem port. Later model C4P's and other models may not have pin 5 connectd to pin 7 (ground). Connect pin 5 to pin 7 and it should remove most problems in getting the modem to work.

TMLIST, mentioned above, is a program to list downloaded files to the screen, modem or printer ports. It is short, concise and does the job very well.

In all, this is a good program. It does everything claimed for it and hasn't glitched once in the time I have used it. It does not support OS65D version 3.3, but you will probably not notice except during screen clear. The documentation is not meant for beginners, but there aren't very many of them in OSI ranks. Aurora Software has put out a very good product at a fair price.

My system consists of a C4P MF, 48K, two disk drives, Amdek color monitor, and Epson printer. I bought the computer in July 1981 to help me on my way to a BS in computer science. Except for the lack of support software, I have no complaints and I don't regret the purchase at all. In the three years I have owned it, there has not been a day of trouble with any of the hardware and the glitches in software were programming problems, easily solved.

24 HOUR VIA CLOCK

By: L. Z. Jankowski
Otaio Rdl
Timaru
New Zealand

I noticed in the May issue of PEEK(65) there is a clock program. However, I thought this article '24 Hour VIA Clock' would be of some interest. If

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 aoo 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.

the article is a little terse, my excuse is that I have tried to keep it as short as possible. Combating the 6522 VIA is definitely for hardware/math freaks! Any kind of text on the VIA becomes technical in a very short time.

The listing presents a self-contained '24 Hour Clock' program. No BASIC driver program is required. The program has been tested successfully both in ROM and DISK BASIC.

The VIA 6522 has two 16 bit timers, Tl and T2. Tl can be used as a source of Interrupts, ideal for an Interrupt driven clock. Connect VIA pin 21 to the IRQ line. The IRQ vector sits in the middle of the stack and is best moved elsewhere. To do this, change the two bytes at $FFFE & $FFFF.

PREPARING THE VIA

Set bit 7 of register 14 (Interrupt Enable Register) to enable Interrupt Control. Set bit 6 to enable Tl as a source of Interrupts. Write 192 to IER.

Continuous Interrupts are required - set bit 6 in register 11 (Auxilliary Control Register). Write 64 to ACR.

Tl counts down at the 02 clock rate. When it hits zero an Interrupt is generated. The countdown value for Tl goes into registers 4 & 5. Register 5 counts in 256s.

CALCULATING THE COUNTDOWN VALUE

It is possible to have a clock accuracy of about 1 second error every 16 hours. To achieve this the value to be counted down by Tl must be calculated accurately.

The equation for this value is: V = 256 * Hi-byte + Lo-byte +2.

Hi is in Register 5 and Lo is in Register 4. The 2 is the delay inside the counter. To calculate Hi-byte and Lo-byte it is necessary to know the 02 clock rate - about 1 000 000. A more precise answer could be found by measuring it with a frequency meter. Failing that, it can be calculated. A ClP may have a 3.93216 MHz crystal. This would give an 02 clock speed of 983.040. Timer Tl being a mere 16 bit counter is not capable of counting such a large number, but it could count 1/16 of 983,040, which is 61,440 ex-

```
  10 0000                ; 24 HOUR CLOCK by LZ JANKOWSKI
  20 0000                ; Lines 50-110 are for OSI BASIC.
  30 0000                ;
  40 1F00                     * = $1F00     start of assembly
  50 1F00                JUMP   = $0000     jump to C1P/C4P warm start.
  60 1F00                OUTCHR = $FFEE     output to screen etc.
  70 1F00                KBINP  = $FFEB     keyboard input.
  80 1F00                IRQ    = $01C0     interrupt request vector.
  90 1F00                SCREEN = $D073     first of 8 addr used on screen.
 100 1F00                FRCADR = $D8       first of 4 clock addresses.
 110 1F00                VIA    = $C020     first VIA address.
 120 1F00                LOTIME = 254       lo byte for VIA timer T1.
 130 1F00                HITIME = 239       hi byte for VIA timer T1.
 140 1F00                FRACTN = 16        set for 1/16 of a second.
 150 1F00                ACR    = VIA+11    auxilliary control register.
 160 1F00                IER    = VIA+14    interrupt enable register.
 170 1F00                TL     = VIA+4     timer T1, lo byte.
 180 1F00                TH     = VIA+5     timer T1, hi byte.
 190 1F00                ;
 200 1F00 A0FF    START  LDY #$FF      print startup message.
 210 1F02 C8      MESGS  INY
 220 1F03 B9CF1F         LDA MESG1,Y
 230 1F06 F005           BEQ GET
 240 1F08 20EEFF         JSR OUTCHR
 250 1F0B D0F5           BNE MESGS
 260 1F0D A203    GET    LDX #3        input 3 pairs of numbers
 270 1F0F D005           BNE NEXT
 280 1F11 A93A    COLON  LDA #$3A      insert colons after pairs.
 290 1F13 20EEFF         JSR OUTCHR
 300 1F16 20EDFF  NEXT   JSR KBINP     get TENS digit
 310 1F19 20EEFF         JSR OUTCHR    and print it.
 320 1F1C 0A             ASL A         place TENS digit in
 330 1F1D 0A             ASL A         the tens column.
 340 1F1E 0A             ASL A
 350 1F1F 0A             ASL A
 360 1F20 95D8           STA FRCADR,X and store it.
 370 1F22 20EBFF         JSR KBINP     get UNITS digit &
 380 1F25 20EEFF         JSR OUTCHR    print it.
 390 1F28 290F           AND #$0F      get rid of high 4 bits.
 400 1F2A F8             SED
 410 1F2B 18             CLC
 420 1F2C 75D8           ADC FRCADR,X
 430 1F2E 95D8           STA FRCADR,X
 440 1F30 D8             CLD           2 digits now stored in 1 byte
 450 1F31 CA             DEX           next pair of numbers
 460 1F32 D0DD           BNE COLON
 470 1F34                ;
 480 1F34 A94C    INIT   LDA #$4C      initialise IRQ.
 490 1F36 8DC001         STA IRQ
 500 1F39 ADCD1F         LDA LOHI
 510 1F3C 8DC101         STA IRQ+1
 520 1F3F ADCE1F         LDA LOHI+1
 530 1F42 8DC201         STA IRQ+2
 540 1F45.A510           LDA FRACTN    initialise FRACTN.
 550 1F47 85D8           STA FRCADR
 560 1F49 A940           LDA #$40
 570 1F4B 8D2BC0         STA ACR
 580 1F4E A9C0           LDA #$C0
 590 1F50 8D2EC0         STA IER
 600 1F53 58             CLI
 610 1F54 A9FE           LDA #LOTIME
 620 1F56 8D24C0         STA TL
 630 1F59 A9EF           LDA #HITIME
 640 1F5B 8D25C0         STA TH
 650 1F5E 4C0000         JMP JUMP      jump to C1P/C4P warm start.
 660 1F61                ;
 670 1F61 48      CLOCK  PHA
 680 1F62 AD24C0         LDA TL
 690 1F65 C6D8           DEC FRCADR    countdown in fractions of sec
 700 1F67 D046           BNE OUT       1 sec not counted yet.
 710 1F69 A910           LDA #FRACTN   reset for next 1 sec countdow
 720 1F6B 85D8           STA FRCADR
 730 1F6D 8A             TXA
 740 1F6E 48             PHA
 750 1F6F A201           LDX #1        sec, min and hrs to update.
 760 1F71 20C31F  SECMIN JSR BUMP      increments secs or mins.
 770 1F74 B5D8           LDA FRCADR,X
 780 1F76 C960           CMP #$60      one min or one hr counted?
 790 1F78 D016           BNE EXIT      no.
 800 1F7A A900           LDA #0        yes.
 810 1F7C 95D8           STA FRCADR,X
 820 1F7E E8             INX
 830 1F7F E003           CPX #3        if X=2 then
 840 1F81 D0EE           BNE SECMIN    add 1 min & check for 60 mins
 850 1F83 20C31F  ONEHR  JSR BUMP      60 min counted. Add 1 to hour
 860 1F86 B5D8           LDA FRCADR,X
 870 1F88 C924           CMP #$24      24 hours up?
 880 1F8A D004           BNE EXIT      no
 890 1F8C A900           LDA #0        yes.
 900 1F8E 95D8           STA FRCADR,X
 910 1F90 98      EXIT   TYA           Put ':' and numerals on scree
 920 1F91 48             PHA
 930 1F92 A000           LDY #0
 940 1F94 A203           LDX #3
 950 1F96 B5D8    OUTPUT LDA FRCADR,X get TENS numeral.
 960 1F98 4A             LSR A         shift to low 4 bits.
 970 1F99 4A             LSR A
 980 1F9A 4A             LSR A
 990 1F9B 4A             LSR A
1000 1F9C 20B11F         JSR VIDEO     put to screen.
1010 1F9F B5D8           LDA FRCADR,X get UNITS numeral.
1020 1FA1 290F           AND #$0F      get rid of high 4 bits.
1030 1FA3 20B11F         JSR VIDEO
1040 1FA6 CA             DEX
1050 1FA7 E000           CPX #0
```

Continued on page 8

!!!!! MORE GOOD NEWS !!!!!
From
D.B.I., INC.

The reliable Denver Board has found a new home.
It has become the heart of a NEW,

Multi-User/Multi-Processing

SUPER-MICRO COMPUTER SYSTEM

Built exclusively by D.B.I., INC.

- * Up to 16 Users
- * SCSI interface (connect to most SCSI devices on market)
- * ERROR CORRECTING controller (no more intermittent read errors)
- * LOW POWER consumption
- * RUNS existing OS-65U OPERATING SYSTEMS and SOFTWARE
- * NEW DB-DOS runs existing OS-65U SOFTWARE, but has changes and extensions, to take advantage of the MICRO's features (faster)
- * SUPER QUIET
- * PRINTER INTERFACES (both serial and parallel)
- * REAL TIME CLOCK (day, date, 24 hour, day of the week)
- * NEW SIZE (approximately the same size as an average terminal)
- * TRANSPORTABLE, about 40 LBS. (single user 5 megabyte)
- * SHIPPABLE (can be sent by U.P.S.)
- * DISK SIZE available NOW! includes 5, 10, 20, 40 & 80 megabytes.
- * Taking orders for 160 megabyte systems (35 mil/sec access)
- * LOW COST PER BYTE BREAK-THROUGH, single user 5 meg-sug-$4995.

CALL YOUR DBI DEALER TODAY FOR FURTHER DETAILS!

DBi, inc.
p.o. box 7276
denver, colorado 80207
phone: (303) 428-0222

Continued from page 6

```
1060 1FA9 D0EB            BNE OUTPUT   print next pair of numerals.
1070 1FAB 68              PLA
1080 1FAC A8              TAY
1090 1FAD 68              PLA
1100 1FAE AA              TAX
1110 1FAF 68      OUT     PLA
1120 1FB0 40              RTI          return from Interrupt.
1130 1FB1         ;
1140 1FB1 18      VIDEO   CLC
1150 1FB2 6930            ADC #$30     convert to ASCII.
1160 1FB4 9973D0  SCR     STA SCREEN,Y
1170 1FB7 A93A            LDA #$3A     the colon.
1180 1FB9 C8              INY
1190 1FBA C002            CPY #2       print a colon?
1200 1FBC F0F6            BEQ SCR
1210 1FBE C005            CPY #5       print a colon?
1220 1FC0 F0F2            BEQ SCR
1230 1FC2 60              RTS
1240 1FC3         ;
1250 1FC3 F8      BUMP    SED          add 1 in decimal to S,M or H.
1260 1FC4 B5D8            LDA FRCADR,X
1270 1FC6 18              CLC
1280 1FC7 6901            ADC #1
1290 1FC9 95D8            STA FRCADR,X
1300 1FCB D8              CLD
1310 1FCC 60              RTS
1320 1FCD         ;
1330 1FCD 611F    LOHI    .WORD CLOCK
1340 1FCF 0D      MESG1   .BYTE $0D,$0A,'Type 6 digits',$0D,$0A,$0A
1340 1FD0 0A
1340 1FD1 54
1340 1FD2 79
1340 1FD3 70
1340 1FD4 65
1340 1FD5 20
1340 1FD6 36
1340 1FD7 20
1340 1FD8 64
1340 1FD9 69
1340 1FDA 67
1340 1FDB 69
1340 1FDC 74
1340 1FDD 73
1340 1FDE 0D
1340 1FDF 0A
1340 1FE0 0A
1350 1FE1 20              .BYTE ' HhMmSs = # ',$0
1350 1FE2 48
1350 1FE3 68
1350 1FE4 4D
1350 1FE5 6D
1350 1FE6 53
1350 1FE7 73
1350 1FE8 20
1350 1FE9 3D
1350 1FEA 20
1350 1FEB 23
1350 1FEC 20
1350 1FED 00
```

actly. Now, 61,440 is exactly divisible by 256, giving 240 with remainder zero. Therefore, Hi=240 and Lo=0. Remember that Hi decrements by one every time Lo decrements by 256. Hi will then decrement from 240 to 0 and then the VIA will generate an Interrupt. This is done 16 times a second! - (240 * 256) * 16 = 983,040. Finally, subtract 2 to allow for counter delay. Write 239 to R5 and 254 to R4.

If the clock runs slow reduce the value LOTIME by 17 (=1/3600 of 61,440) for each second lost. If the clock is running fast then let HITIME= 240 and increase the value in LOTIME. For a 2MHz CPU clock, double the value in FRACTN, from 16 to 32.

To run the program, enter it at START! Eg. POKE 11,#: POKE 12,#: X=USR(X), or use the Monitor - .####G.

THE SIMPLIFIED DUAL DRIVE SYSTEM

By: Michael E. Sherman
12951 Westchester Trail
Chesterland, OH 44026

This article describes a method for saving valuable disk space and simplifying the use of dual disk drive OSI computers. A great deal of disk space can be saved with one very simple program. The program is an executive program for disks which do not contain the operating system. In a dual drive disk system, there is no need to have the operating system or any of its utilities on all disks. Doing so wastes more than twenty five percent of your available disk space. A few simple modifications to the BEXEC* in conjunction with this can enhance the usability of the entire system.

SYSTEM DISKS vs. NON-SYSTEM DISKS

In my system, I have system disks and non-system disks. A system disk is a bootable disk with OS65D, BEXEC* and the most often used utilities. This disk is used only in drive A. It is used to boot the system and kept in drive A for quick access to any utility programs. A non-system disk is one used for program development and storage only. It is not bootable. When a non-system disk is created it has only the directory, a dummy file to cover track zero, and the non-system disk executive program I call "E".

THE BEXEC*

Modifications to, and complete rewrites to OSI's BEXEC* programs, are published frequently. A common theme seems to be to shorten it. I do not see as great a need for compression as there is for functional streamlining to make the system more easily accessable. User friendliness is of prime importance for fast efficient use of any computer system. I have built upon OSI's exec for their 3.3 operating system. I have increased its length by one track, and added several functions. All of these are aimed at user convenience.

To make up for the extra disk space required for my lengthened BEXEC*, I incorporated BASIC's trace function into it. The track was used, or some might say wasted, by the TRACE program which really only needs two lines. These lines were simply added to the exec with menu functions to enable and disable trace. I added functions to select either drive. These can be a great convenience. Another function added was a direct exit to OS65D. Another feature added to BEXEC*'s initialization routine was the pokes to set the indirect file boundaries to the appropriate values for my system memory. Now indirect files can be used with no calculations or pokes whenever needed. I also added direct calls for my modem software (TERM) and my text editor (TECO). Lines 920 and 950 are a form of error correction.

There are many other possibilities. Almost any function which you find that you use frequently, or program that you run often can either be incorporated directly into BEXEC* or called from it. The

LISTINGS:

```
10 REM          2/4/84
20 PRINT!(28):POKE2888,0:POKE8722,0
30 PRINT"       :::::::::::::::::::::::::
40 PRINT"       :     NON-SYSTEM DISK     :
50 PRINT"       :        EXECUTIVE        :
60 PRINT"       :::::::::::::::::::::::::
70 REM
80 REM
100 PRINT:PRINT"     Menu:":PRINT"    1> Run BEXEC*   A> Select A"
110 PRINT"      2> Run NUMBER       B> Select B
115 PRINT"      3> EXIT             E> Enable TRACE
120 PRINT"                          D> Disable TRACE
150 INPUT"         Select one";A$
152 IFA$="A"THENDISK!"SE A":GOTO1000
154 IFA$="B"THENDISK!"SE B":GOTO1000
160 IFA$="D"THEN450
165 IFA$="E"THEN400
170 IFA$=""THENA$="3"
180 A=VAL(A$)
190 ONAGOTO200,300,1000
200 DISK!"SE A"
210 RUN"BEXEC*"
300 DISK!"SE A"
310 RUN"NUM"
400 L=2011:POKEL,32:POKEL+1,216:POKEL+2,28:POKEL+3,234:POKEL+4,234
410 GOTO1000
450 L=2011:POKEL,24:POKEL+1,144:POKEL+2,2:POKEL+3,230:POKEL+4,200
460 GOTO1000
1000 POKE2888,27:POKE8722,27:PRINT!(28):END
```

Modifications to BEXEC*

```
110 PRINT"OS-65D v.3.3  system disk   - April 6, 1984"
120 PRINT" 1 > Directory                  A > Select A"
130 PRINT" 2 > Create a new file          B > Select B"
140 PRINT" 3 > Change a file name         X > Exit to OS"
150 PRINT" 4 > Delete file from disk      E > Trace on"
160 PRINT" 5 > Create blank data disk     D > Trace off"
170 PRINT" 6 > Create data disk + files"
180 PRINT" 7 > Create buffer space            TECO"
190 PRINT" 8 > Disk copier                    TERM"
200 PRINT" 9 > Enter OS-65D system":PRINT
900 PRINT&(0,10)!(24);"Type your selection and depress RETURN ";
910 INPUTS$
911 IFS$="PASS"ORS$="9"ORS$=""THEN60000
912 IFS$="A"THENDISK!"SE A":GOTO900
913 IFS$="B"THENDISK!"SE B":GOTO900
914 IFS$="X"THENPRINT!(28):EXIT
915 IFLEFT$(S$,3)="TEC"THENDISK!"SE A":RUN"TECO"
916 IFLEFT$(S$,3)="TER"THENDISK!"SE A":RUN"TERM"
917 IFS$="D"THENGOTO60110
918 IFS$="E"THENGOTO60100
920 IFLEFT$(S$,5)="DISK!"THENS$=RIGHT$(S$,LEN(S$)-6):DISK!+S$
950 IFLEFT$(S$,3)="RUN"THENS$=RIGHT$(S$,LEN(S$)-4):RUNS$
960 IFLEN(S$)>1THEN900
970 S=INT(VAL(S$)):IFS<1ORS>8THEN900
980 GOSUB50000
989 PRINT"        ";
990 ONSGOSUB1000,2000,3000,4000,5000,5000,7000,8000
```

design of our disk storage format favors this approach. Programs are stored in two killobyte blocks. This obviously favors long programs, and makes the storage of short programs such as trace, wasteful. When adding program calls to the BEXEC*, all possible calls need not be displayed on the menu. Large computer systems do not display all options.

THE EXECUTIVE

The executive program performs several useful functions. The first function on my menu is the systems BASIC executive program, BEXEC*. This function selects drive A and runs BEXEC*. Also included are single key functions to select either disk drive, to enable or disable BASIC's trace function, and to run another useful program called "number" that I have on my system disk. That happens to be a number base conversion program, but any utilities you use regularly could be called from E.

This program has several other advantages. The most important to me is the way that use of the system is simplified. I think that we are all a bit lazy, especially when it comes to pounding the keyboard. Typing 'RUN"E' is very fast and easy. Even though I type fairly quickly, that's still much faster than typing 'DISK!"SE A' and then 'RUN"BEXEC*'. This saves a great deal of time that could be spent thinking about more important things, like why the program that you are working on crashes!

★

A COMPUTERIZED INVENTORY SERVICE

By: Willis H. Cook
1298 Renee Drive
Lilburn, GA 3Ø247

This is the story of a small business venture undertaken with an OSI C2-4P computer. It is a very small business and is really just a hobbyist sort of effort except that it is really a money-making endeavor. How it came about, what it involves and how much money it generates might be of some interest, if not for inspiration, at least for entertainment.

PROLOGUE

I am a typical hobbyist and have owned a C2-4P since Feb 1979. The machine has grown since its 8k, cassette recorder days and now has an 8" floppy drive, 32k RAM and an Epson MX-8Ø printer. That seemed like a lot of computer when the disk was added and the memory expanded in 198Ø.

My employer, a consulting engineering firm, has used a mini-computer for years to run engineering studies for its clients, so everybody in the firm is quite familiar with computer operations and is used to working from computer printouts. This familiarity has no doubt helped me to get started, since my client, a fellow-engineer, already knew what computers could do for him.

ACT ONE

About a year ago, my co-worker decided to start his own business on the side by opening a liquor store. He still works for the consulting firm; he has a clerk to run the store so its operation takes up a minimum of his time during normal working hours.

One of his first needs was to establish the selling prices for his products. He asked me if I could make up a list giving the retail selling price for each bottle of liquor for a range of mark-up values. That was no problem and I gave him the listings arranged by bottle size. Standard liquor containers are now metric: .2 liter, .375, .5, .75, 1 and 1.75 liters. At that time, about 200 individual items were involved and a trivial little BASIC program did the calculations and made the printouts. He was very pleased and I charged him $35.

ACT TWO

My new customer was so impressed with the results of his first request, he was encouraged to ask if it might be possible to develop some sort of sales report on the computer. He was calculating his monthly sales by hand, so, although I knew nothing about his business or what would be required in such a report, I assured him that I could duplicate his hand calculations.

This effort lead to our first program called PROFIT. (I love these enigmatic computer names.) The program was designed to maintain a data base of all the liquor items supplied by eight distributors. Each distributor's products were kept in a separate file since he wanted totals by distributor and it kept each file small; a disaster with one wouldn't jeopardize the data in the others. Data kept on each item included product description, bottle size, quantity sold during the month, wholesale cost, suggested retail price, and actual selling price.

While the format of the output report was designed to satisfy the client, the operation of the program was designed for me since I was the one running it. Niceties included an editor function based on the keyboard scan routine to make entries and changes as easy as possible. The files are random-access and in addition to finding any record by number, a scrolling routine was incorporated. Hold down the right SHIFT key to scroll forward through the file, and the left SHIFT key to scroll backward. After the initial data entry, wholesale and retail prices change for various items every month and the scrolling feature made it easy to go through each file and make them.

A total of 11 values for each sales item was included on the printout and the width of the printout was 125 columns. I use 9 1/2 x 11 inch paper and used the condensed (17 columns /inch) print mode of the Epson printer. The print was small but legible.

ACT 3

The information tabulated by PROFIT was satisfactory, but we hadn't as yet included any information on inventory, just monthly sales. It was also tedious to enter the monthly sales of each item, even with my elaborate editor. The next version of the program was known as INVENY, which is how you spell inventory in six letters. As is probably very common when computerizing a new business, the owner requested frequent changes in what data should be kept in the data file and what quantities were to be calculated and included on the printout. For example, we used to calculate the mark-up for each item and also list a maximum mark-up allowed by competitors' prices. Profits were calculated from both mark-ups to see how close the actual profit came to the maximum. After a few months of operation, he developed his feel for allowable prices, so the maximum mark-ups were dropped from the printout.

Somewhere along the way, I decided that each program run would be worth $10.00 and any programming time I spent on the project would be at $5.00 an hour. For the basic ten dollars, my client got a program run and also the data input necessary to make the run; purchases of each item and the quantity of each item on hand taken from a physical inventory he makes every month. There are also typically a few price changes every month, and these are included too. Anything else, such as extensive new product listings, since they take a good bit of key punching to get them into the data base, and of course, any program changes, are charged at the $5.00/hr rate.

ACT FOUR

At this point we had two programs, PROFIT and INVENY, that each calculated information based on the same data files and that used similar output formats. It was logical to combine them into one program that handled both monthly sales and inventory. Some more items were dropped from the printouts to make room for the combined information and the next version of the program, called SALES, was born. See the Figure for a sample printout. SALES has actually become a series of four programs as the product categories have been expanded. SALES itself still handles the liquor items. The other programs are WINE, BEER and MIXES, each one handling its respective type of product. I don't advocate making different programs to do approximately the same thing, but in this case, the data kept in the data files was slightly different for each category and the printouts were different. Generally, I found out about wanted changes when I got the data for the next month's run, so there wasn't a lot of time available for leisurely program design. Expediency suggested making a copy of an existing program, making the changes and running each program with its own data sets. Another unforeseen benefit to keeping the different programs separate is that when I am occasionally asked to change one or another of them, it is easy to make changes without having to worry about effecting the other routines.

Several file-maintenance programs were also written to make things easier for both of us. REVISE replaces last month's bottles-on-hand figures with the ending inventory amounts. In other words, the end of the period quantities are re-written in the data file as beginning period quantities, to be ready for the next month's run. Also, the quantities purchased are zeroed out since they change every month. Another utility is SPREAD which spreads out data in each data file to allow insertions of new products in their respective locations. (Products are grouped by size in each file.) A final utility is called LLIST. It creates a data entry form for each item with blanks for purchases and for the ending inventory which the store clerk fills in. It also lists the wholesale cost and retail price for each item and they are changed as necessary.

EPILOGUE

My entrepeneur client and I have been computerizing his business now for about seven months. He is pleased because he gets a neat, accurate report of his business activity prepared each month for $50: ten dollars each for SALES, WINE, BEER, MIXES and LLIST. He has a completely customized set of programs that include exactly the information he needs and in just the format he wants. I am happy to have a modest little business and have enjoyed (usually) building up the programs that were needed. It takes me about five hours to input the monthly inventory and sales figures and run the sales reports and data input sheets for the next month's figures. Of course, I am not sitting at the keyboard all that time: the printing of about 45 pages takes over an hour -- 132-column lines hold a lot of characters.

From time to time the temptation arises to go out and solicit business from other liquor stores in the area. They might be interested in a computer service specially tailored to their business. But I only have the one machine and no dealer service is available for it. So far, I have been able to resurrect

Sample of the
INVENTORY AND SALES REPORT
Products from Acme Distributors.
April 10 - May 12 -- 1984

```
---------------------------------------------------------------------------------------------------------------
                                        Purchases                                                        $ Sales
                              Bottles   04/10/84             Ending               Avg           Retail Inventory 04/10/84
                      Size    On hand   through   On hand   Inventory   Unit     Cost/   Mark   Price/   Value   through
      Brand          (Ltrs)   04/10/84  05/12/84  + purch   05/12/84    sales    bottle   up    bottle  05/12/84 05/12/84
---------------------------------------------------------------------------------------------------------------

Bacardi Silver Rum    .20        6         6        12         4          8       1.98    1.26    2.49     7.92    19.92
Bacardi 151           .20        4        12        16         8          8       2.90    1.27    3.69    23.20    29.52
Paul Masson           .20        6         6        12         1         11       2.29    1.22    2.79     2.29    30.69
Bacardi Amber Rum     .20        3        12        15         8          7       2.12    1.17    2.49    16.96    17.43
Lord Calvert Can.     .20       11         0        11         0         11       1.79    1.28    2.29     0.00    25.19
Calvert Extra         .20       18         0        18         6         12       1.79    1.34    2.39    10.74    28.68
Karlof 90 Pf.         .20        3         3         6         2          4       1.34    1.19    1.59     2.68     6.36

Bacardi Silver Rum    .37        5        10        15        11          4       3.29    1.27    4.19    36.19    16.76
Lord Calvert          .37        3         4         7         5          2       3.39    1.27    4.29    16.95     8.58

Bacardi Silver Rum    .50        0        12        12         7          5       4.05    1.23    4.99    28.35    24.95
Bacardi 151           .50        2         4         6         4          2       5.59    1.23    6.89    22.36    13.78
Paul Masson           .50        4        24        28        18         10       4.69    1.23    5.79    84.42    57.90


                              TOTALS                                                                 252.06   279.76

---------------------------------------------------------------------------------------------------------------
```

The only data required from the store clerk each month are the quantities of each item purchased and the end-of-the-period inventory. The product descriptions, sizes, beginning and ending amounts and wholesale and retail prices are stored in a data file. The other quantities are calculated by the program. The total number of items tabulated is now over 600.

it whenever it has died, but it is not a pleasant experience to start on a long evening's work and have your computer suddenly tell you it has only 8k of memory free or that it can't boot up from the disk. Both these things have happened and more. In fact, it is sort of a grim joke around the house that whenever I bring home the liquor store data, the computer breaks. Right now, I don't know whether to get another machine for back up (anybody like to sell one?) or just to keep the effort at its current hobbyist level. Overall, I rate it as successfully miniscule, or minisculely successful. It's amazing that people are willing to pay for the output of my computer, but it is frightening to make commitments and then hold my breath when I hit BREAK-D to see what will happen.

★

DOS/65 REVIEW

By: K. B. Shacter
P.O. Box 61ØØØ
New Orleans, LA 7Ø161-1ØØØ

How often have you, an owner of a 65Ø2-based computer, wished you had an operating system that emulated the much touted Z8Ø/8Ø8Ø DOS called CP/M? Well, want no more, for MicroSystems Technology (45Ø Forrest Avenue, Apt. D312, Norristown, PA 194Ø1) has introduced a CP/M clone for 65Ø2-based machines. It is interesting to note that I did not just limit the DOS to just OSI machines. More on this later.

When I received my copy of DOS/65 for review, I was a little taken aback at the number of disks and the amount of documentation. Listed below are the manuals and other documents that came with the DOS. I read them all, but found some a bit beyond my ken, mostly written for the hard-core hardware or software junkie. The documentation provided is as follows:

Product Information
Comments on Copying
Diskette Allocation
DOS/65 OSI Notes
System Documentation
consisting of:
  Introduction and Index
  System Description
  System Interface Guide
  Bringing the System Up
  Non-Standard Controller
  SYSGEN Manual
  ASM Manual
  DEBUG Manual
  EDIT Manual
  BASIC-E/65 Manual
  S-1ØØ Interface Guide
  SIM Listing
  BOOT Listing
  LOADER Listing
  Tarbell 8Ø8Ø BIOS Listing
  UPGRADE Manual

PRODUCT INFORMATION

A brief advertising flyer is included, describing the system, support capabilities (in terms of hardware needed/accepted), why you would want to use DOS/65 instead of OS65D or HEXDOS, major software packages included, and a rundown on the 'compiled' BASIC-E/65. This little package also includes ordering instructions (a bit late if you already have the system). You may request this flyer prior to purchasing the DOS.

The cost of DOS/65 is $135 for 5-1/4 inch OSI disk systems, and $13Ø for the 8 inch disk OSI system. Customization (non-standard systems) costs extra. The documentation by itself is $3Ø, credited to the purchase of the system. The advertising really doesn't do justice compared to what you get for your money. Read on!

COMMENTS ON COPYING

How many software products do you know of that encourage you to make back-ups of their diskettes? DOS/65 does, but with some warnings as to the legality (or lack thereof) of you making copies to give (or sell) to friends. In addition, all of the software manuals contain copyright notices, disclaimers of warranty, and limitations of liability. The system is warranted for one year, and any bug fixes and documentaion fixes for the system will be provided (on paper or via an update disk) for the year warranty period.

DISKETTE ALLOCATION

What is a diskette allocation, you may ask. Well, the DOS comes on ten disks. Right there, that is about $35 from your local computer shop. And they aren't cheap disks either (no insinuation to the OSI diskettes!) They are Verbatim 5-1/4 inch floppies, with the re-enforced center hub. The diskette allocation is a listing of what is provided on what disk, including DOS utility programs (BASIC, EDITOR, ASM, etc), user programs (data base, address book, games, etc), and information about the user programs.

DOS/65D OSI NOTES

Information in this flyer tells you about how peculiar our 65Ø2-based computers really are, compared to the rest of the world. The biggest problem is OSI's disk controller and the format of track Ø. Additional aspects of disk format, disk I/O, keyboard I/O, the BOOT procedure, miscellaneous program insight, and possible hardware enhancements. OSI-specific programs are included with the DOS for extracting OS65D BASIC, ASM and data disk files to DOS/65 disks. These copy routines also do a lot of leg work in converting the source files to DOS/65 syntax conventions. More on this later.

The OSI notes also provide information on how to load the DOS. DOS/65 is booted in an unusual manner. First, you must boot OS65D. You then exit BASIC to the kernel, load in the DOS/65 LOADER disk, and move a machine language program into memory with the CALL statement. Next, you execute the program with the GO command. You place another DOS/65 diskette in the drive (the one containing your DOS) and depress the space bar. Next, you are prompted to state how many physical disk drives you have (1 or 2, although the DOS will handle up to 8, including hard disks). Finally, you are presented with the DOS/65 prompt: A>. You are now in DOS/65.

THE 'MANUAL'

The manual is provided on good quality paper in exceptionally legible word processed type, bound with a removable metal strap. The manual consists of the 16 individually noted sections above under system documentation. These little manuals are separated into sections within the bound document by tab dividers. I recommend the document be placed in a 3-ring binder for easiest use.

A brief INDEX is included up front, providing the major section headings (as listed above). A useful addition would be a compilation of all commands and error messages for the programs described in the various sections of the manual, so once you become familiar with the document, you won't have to go seaching through the individual sections to get the bit of information you need. However, with a little effort and access to your local copying machine, you can accomplish this task on your own.

As mentioned above, a warranty is provided for the documentation, ensuring updating/correcting the manual for one year from date of purchase. Each section has a thorough Table of Contents, listing the sub-sections of each section, including appendices, and lists of tables and figures (if applicable).

SYSTEM DESCRIPTION

The 'SYSTEM DESCRIPTION' portion of the manual explains DOS/65 basics. DOS/65 was written to provide the 65Ø2 user with a standardized, low cost operating system that will run on a variety of 65Ø2-based computers. Among those already having DOS/65 available are: OSI, KIM, SYM, and AIM. Plans for APPLE, COMMODORE and ATARI are in the works.

DOS/65 follows the same basic structure as CP/M, and is supposed to be file compatible with CP/M 1.4 (meaning CP/M files, whether source or data, should be able to be read by the DOS, converted to the syntax of DOS/65 BASIC/ASM, and finally executed). Unfortunately, it does not mean that the OSI controller format provides this degree of freedom, whether or not the DOS can provide for it. I couldn't get an APPLE (with a CP/M card) or a North Star to read the disks. I kept getting 'CAN'T FIND DIRECTORY' errors. However, it is supposed to support (on OSI machines) the OSI CP/M disk format.

The CP/M model for a DOS was stated to be chosen for the following reasons:

1. Nearly total hardware independence.

2. Easy access to operating system primitives.

3. Easy alteration to accommodate system unique characteristics.

4. A large library of compatible software exists (CPMUG).

5. Simplicity (at least, once you get the hang of the system!).

The DOS consists of three layers, named:

CCM:Console Command Module
PEM:Primitive Execution Module
SIM:Systems Interface Module

CCM is the primary interface between the DOS and the user. PEM is the core of the DOS, which contains the directives for console (terminal), peripheral, and disk control. SIM contains the user peculiar interface between PEM and the user's system. CCM contains five high level commands (DIR, SAVE, ERA, TYPE, and REN), drive selection (up to eight logical drives are supported, even if you only have one physical device), program execution (a .COM file), and five control-key editing functions (cancel current line, tab, list current buffer contents, warm boot, and list the screen to hard copy device). I never could get the 'list current buffer contents' command to work, perhaps because it's primary function would be for serial machines, and I have a 1-disk C1P (video based). The rubout/delete is a destructive back space, akin to the OS65D shift-O.

PEM handles disk files more like 'grown-up' (or main frame) operating systems. By this I mean you don't have to pre-allocate disk space before you start developing a program or writing a file (as with OS65D). The DOS writes information to the disk as blocks made up of 128-byte sectors. Blocks vary in length from 1K to 16K bytes, in multiples of 2 (information here is confusing). I did not gleem from the documentation whether the block length depended on your disk size, controller, disk format (floppy or hard, yes I said HARD), or if it is set in the software somehow. These blocks are assigned to a directory that contains pointers to either 8K or 16K of disk storage. Within each of these directory spaces the disk storage is ordered logically, but is not necessarily continuous. This means your program or data file may be physically scattered all over the disk (heavens!), being extended to include more room as you need it.

This is all transparent to the user, however. If you delete a file, it frees up all the space (wherever it was physically located) it used on the disk, and as new files are created, this space fills back up. This may not be the most aesthetically pleasing notion, but it certainly is efficient! You cannot, however, list the directory as with OS65D or HEXDOS and see exactly which programs or files lie on which tracks. The 'DIR' command of CCM provides the names of the files on the disk, without giving a hint as to where they are located. Two programs provided with DOS/65 do allow the user to determine how much free space is on a disk, and where it is located (although this doesn't matter). The programs are named ALLOC and FILESTAT, which are covered later under the supplied software section.

There are four Appendices in the System Description section:

1. Supplied transients/operational precedures.

2. CCM/PEM error messages.

3. Command line parsing.

4. Programming notes on:
   a) software UARTS
   b) single drive systems
   c) automatic command execution (auto-run, akin to BEXEC* or the autostart feature in HEXDOS 4.Ø)

I can personally attest to the fact that DOS/65 was not designed to be used with the single-drive system. For those who have more than one physical drive, the system would be heaven; for those with one drive, you will have to get used to swapping disks in and out an awful lot!

The file structure is set up to handle file names that take the form OBJECT.TYPE, where OBJECT is 8 characters or less and describes the file contents. TYPE is three characters or less and defines the file type. Files, of course, are ASCII or BINARY, but the TYPE qualifier helps you segregate text (.TXT), BASIC source (.BAS), assembly source (.ASM), backup source (.BAK), compiled BASIC (.INT), assembled ASM file (.KIM), or loaded (executable) .KIM file (.COM), among others. By following the above conventions, you can make file management simpler. Also, many supplied programs expect certain TYPE qualifiers.

SYSTEM INTERFACE GUIDE

This section covers the details of PEM and SIM, explaining the general concept and commands/functions of each. To digress a moment, the various modules that comprise DOS/65 may be related to their counterparts in CP/M:

DOS/65

CCM:Console Command Module
PEM:Primitive Execution Module

SIM:Systems Interface Module
TEA:Transient Execution Area

CP/M

CCP: Console Command Processor
BDOS:Basic Disk Operating System
BIOS:Basic Input/Output System
TPA: Transient Program Area

PEM is accessed by storing a specific function value in the X-register and jumping to a fixed memory location (JMP PEM). PEM handles such tasks as character oriented I/O, system control, and disk I/O. SIM's functions cover executing the system boot, actual I/O to the system specific hardware, and governing the actions of the disk controller (i.e., head movement and disk I/O).

Each subsection in the S.I.G. (one each for PEM and SIM) is broken down into brief (and sometimes not so brief) descriptions of how each function/action for PEM and SIM are used and operate. Appendices cover disk content layout (with respect to BOOT, CCM, PEM, and SIM only, not transient or user programs), memory usage (including a page one memory map and a generic system memory map), flag and interrupt use, and Standard Interface Formats (SIF), which defines the standard DOS/65 disk utilization and sector translation between the logical sector and the physical sector on the disk.

The S.I.G. manual is an indispensable aid to the hard core hacker, or the student interested in how to interface and manipulate the power of the DOS from a user-written program (BASIC or ASM). Let the reader beware, it is not easy reading, and contains a great deal of information. The concepts presented may be unfamiliar to some.

BRINGING THE SYSTEM UP

Once again the DOS/65 documentation goes above and beyond the call of duty for the standard OSI user (at least in the C1, C2, and C4 ranks). OEM users may find this chapter of interest. This manual discusses how to invoke DOS/65 on a system that does not conform to a standard release (SIF A through SIF E). Those C8 users who have added non-standard equipment (drives, controllers and the like) will make use of the information provided to allow adaptation of the DOS to your system configuration.

NON-STANDARD CONTROLLER

This manual is used in conjunction with 'Bringing the System Up' and the 'System Interface Guide' to help the user customize DOS/65 to their system requirements. Again, it is a bit over most C1/2/4 user's needs.

SYSGEN: DOS/65 SYSTEM GENERATION

Suppose you purchased DOS/65 configured for a 24K C1P. Let us further suppose you added an additional 8K of RAM and modified the video display circuitry to provide a 64 character display. Now what? With SYSGEN, you can re-tailor DOS/65 to meet your new system requirements. SYSGEN allows the user to create a 'new' version of DOS/65 which includes the following:

1. User specified memory size.
2. User specified SIM length.
3. User created BOOT.
4. User created SIM.

SYSGEN does not load the newly created system to memory when execution terminates. The updated version of the DOS must be BOOTed in with LOADER (versus doing a warm boot). The chapter walks the user through a typical SYSGEN execution, and provides a few helping hints. I did not attempt an execution of SYSGEN.

ASM

The DOS/65 method of creating a machine language program goes something like this:

1. Use the EDITor to create and/or modify an assembly source of type .ASM.

2. Assemble the source to create a hexadecimal code file of type .KIM using the ASseMbler (.HEX file under CP/M).

3. If desired, execute the .KIM file using the DEBUGging tool. Make the necessary patches, and then. . .

4. Generate an executable load module of type .COM using the LOADer.

Not quite as neat as OS65D or HEXDOS, but powerful.

The assembler supports standard MOS Technology conventions and has:
- a few useful conventions:

1. The prefixes > and < signify the high and low bytes of the 16-bit value of the operand.

2. Upper-case and lower-case mnemonics are treated as the same, while no upper-case conversion is done to strings.

- and a quirk or two with respect to extensions:

1. Horizontal tabs are a valid field separator (as with OS65D), but may behave strangely within strings. To tab within a string, the user should enter the hex code for a mod 8 tab ($9) in between the portions of the string to be separated.

2. ASCII characters may be used as operands in an expression, and if used in an expression (e.g., 'A'+3) must be surrounded by quote marks, instead of just on the leading edge. I don't know of a use for either convention, though.

The assembly source listing may be directed either to disk as a type .PRN file, to the video console (and with the contl-P function onto the printer), or deleted. The fate of the various files used and created by ASM are directed by key letters appended to the source file name on the command line. The beauty of how DOS/65 allocates and handles files can be appreciated during an assembly, as the

THE DATA SYSTEM

- 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.

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.

TIME & TASK PLANNER

30 DAY FREE TRIAL — IF NOT SATISFIED, FULL REFUND UPON RETURN

- "Daily Appointment Schedule"
- "Future Planning List" - sorted
- "To Do List" - by rank or date
- Work Sheets for all Aspects
- Year & Month Printed Calendar
- Transfers to Daily Schedule

A SIMPLE BUT POWERFUL TOOL FOR SUCCESS

HARDWARE: 48K OSI, 8" floppy or hard disk, serial terminal system, OS-65U v. 1.3 or later.

PRICE: $300.00 (User Manual, $25.00, credited toward TTP purchase). Michigan residents add 4% sales tax.

FINANCIAL PLANNER

- Loan/Annuity Analysis
- Annuity 'Due' Analysis
- Present/Future Value Analysis
- Sinking Fund Analysis
- Amortization Schedules
- Interest Conversions

HARDWARE REQUIREMENTS: 48K OSI, 8" floppy or hard disk, serial terminal system, OS-65U v. 1.2 or later.

PRICE: $300.00 (User Manual, $25.00, credited toward Planner purchase). Michigan residents add 4% sales tax.

DEALERS: Your Inquiries Most Welcome

GANDER SOFTWARE

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

"It Flies"

FROM THE FOLKS WHO BROUGHT YOU:
All This
THERE IS MORE COMING SOON:
Program Generator for TDS
Proposal Planner
Time and Billing A/R

.PRN and .KIM files may physically interweave each other on the disk as they are built, but appear logically contiguous to the user when accessed. No pre-allocation, no worries about files bumping into each other, and no worries about where the file is actually located. DOS/65 handles it all, without skipping a clockbeat!!

The standard assembler supports source lines up to 8Ø characters long. Labels may be up to six characters long, although versions of the assembler do exist which support longer lengths. As stated previously, upper and lower-case variables are treated as the same (except within strings). Of course, reserved words cannot be used for valid label names. The standard 56 opcodes of the 65Ø2 are supported. Those who have 'upgraded' by installing one of the super 65Ø2 chips (65CØ2) cannot take advantage of the additional CPU horsepower (i.e., op codes) with this assembler (although the HEXDOS assembler is stated to have the extra mnemonics). Another nice feature of this assembler is that it works from disk files, not from memory (as 65D), so your source can be as long as you can fit on a disk. Just don't ask me to debug it!

Line numbers are optional, and constants may be entered as binary (%), octal (@), decimal, or hexadecimal ($). The assembler supports a symbol table (sorted alphabetically (horizontally)), and can be directed to either generate or not generate the .PRN and .KIM files, errors (why?), symbol table, and print strings. The chapter provides some cautions to be wary of in producing your own machine language routines. These deal with page zero definitions, and page zero usage. Although not applicable to the standard ASM, a caveat emptor is provided to those users anxious to obtain n-length label versions. The assembler spits out a text error message - not a number.

TO BE CONTINUED NEXT MONTH.

65Ø2 ASSEMBLY LANGUAGE PROGRAMMING CLASS

By: Richard L. Tretheway
Systems Operator for the
OSI SIG on CompuServe

In beginning this class, I have noted that at least two other SIGs on CompuServe have also held classes on this subject. The first is SIG*-ATARI, and the second is the Programmer's SIG. I would encourage you to check into these two SIGs after one or two classes here in PEEK. Now let's get to work.

The goal of this project is to get you into assembly language programming as quickly as possible. My approach will probably be similar to some of the books on this subject, but I will skip a lot of the background math in order to speed things up. In addition, some of the information provided at first may be slightly incorrect and will appear to be in conflict with any textbook. These conflicts will be resolved as needed, but not until they are needed.

To begin, let's take a look at the 65Ø2 microprocessor chip which is the brain in your OSI computer. I'm sure you've heard that the 65Ø2 is an "8" bit microprocessor, and that's true, but do you know what that means? A "bit" is the smallest piece of information the 65Ø2 can deal with. A bit is much like an on/off switch in that it is always either a 1 or a Ø. When we say that the 65Ø2 is an 8-bit machine, we mean to say that it performs all of its operations on groups of 8 bits. Each group of 8 bits is called a byte. The Ø/1 characteristic of bits leads us to the concept of binary, or base two, math. Let's take a look at a byte:

```
+---+---+---+---+---+---+---+---+
: 7 : 6 : 5 : 4 : 3 : 2 : 1 : Ø :
+---+---+---+---+---+---+---+---+
```

You'll notice that I numbered each bit from Ø through 7, instead of 1 through 8. There is a reason for this. The bit number shown refers to the exponent of 2 that each bit represents within the byte. Wow, that's a mouthful. Let me explain further. Bit number zero represents 2 to the Øth power or 1. That means that if bit zero is a one, the value of the byte is increased by one. Further, bit number one represents 2 to the 1st power or two. Thus, if bit one is one, then the value of the byte is increased by two. This process holds for all eight bits, giving rise to the following table:

Bit Number	Value
Ø	2^Ø or 1
1	2^1 or 2
2	2^2 or 4
3	2^3 or 8
4	2^4 or 16
5	2^5 or 32
6	2^6 or 64
7	2^7 or 128

	255
	maximum value

So, for example, a byte holding "Ø111Ø1Ø1" would equal:

```
1 times 2^Ø or  1
Ø times 2^1 or  Ø
1 times 2^2 or  4
Ø times 2^3 or  Ø
1 times 2^4 or 16
1 times 2^5 or 32
1 times 2^6 or 64
Ø times 2^7 or  Ø
               ---
               117
 total in decimal
```

Now, for most people, the number "Ø111Ø1Ø1" is meaningless. We're accustomed to dealing with the base ten or decimal number system. Enter HEXADECIMAL or base 16. Trust me, hex makes things simpler even though it too will be unfamiliar to you at first. In any programming task, no matter how complicated, the trick is to break the problem down into it's component parts and solve each part one by one. We use the same technique with the binary 8-bit value by splitting it into two 4-bit values. A 4-bit binary number has a range of possible values that we can easily comprehend, to wit: from Ø to 15, and which can be represented by a single digit in base 16 (HEX). By further expansion to cover all 8 bits, you'll see that we can represent the entire range of values with a 2 digit hexadecimal mumber. Let's see how our original example is represented in hex.

```
         Ø 1 1 1    Ø 1 Ø 1
         -------    -------
Decimal   4+2+1      4+Ø+1 = 117
         -------    -------
Hex         7          5   or $75

      # of 16's   # of 1's
```

Test this representation:
7*16 = 112 + 5*1 = 117

When referring to these numbers, there are some commonly used terms. Each group of 4 bits is called a "nybble"(pronounced "nibble"). Again, 8 bits is called a

TRY US!

WE MEAN BUSINESS

SC5/80-1 Computer System

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

$6990⁰⁰

For Each Add'l Multi Processor User Add $1,000⁰⁰

- **Includes Super Utility Package**

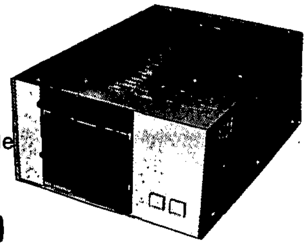
Plus our incredible new super system data base manager. With brand new Disk Tech One, 2 year warranty on internal hard disk components. Beautiful hand finished oak cabinet over steel frame. Instant access to all components. Available with casters or plastic feet

BEAUTIFUL AS WELL AS FUNCTIONAL

CLOSE OUT! C3-OEM

2 MHz/56K*
With 6502 and Z80 Processors

With Centronics Parallel Printer Interface



While they Last!

$1499⁰⁰

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

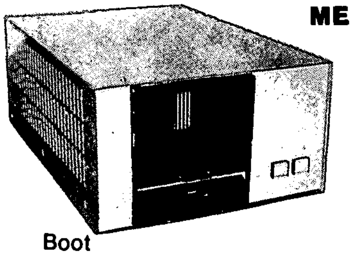
- Buy for a Spare
- Add to a Spare H.D.
- Develop Programs

Spare 510CPU Board with Z80+6502 $169⁰⁰

MULTI PROCESSOR!

COMPLETE 10 MEGABYTE SYSTEM

$3999⁰⁰



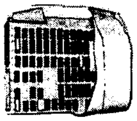
Boot From Hard or Floppy!

- 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 $400⁰⁰

NEW CONTROLLER/INTERFACE!



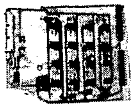
$599⁰⁰
Quantity 1

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.

9592
Quantity 1

$269⁰⁰



12 SLOT BUS

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

$79⁰⁰

10 meg. Subsystem WOW!

$1999⁰⁰

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 $699⁰⁰

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

byte. When dealing with 16 bit values we need a four digit hex number. The leftmost 2 digits of this number refers to the byte called the "most significant byte" or "MSB". The rightmost two digits refer to the byte called the "least significant byte" or "LSB".

If you consider the LSB and the MSB is terms of their values individually, you'll find that they represent what might be seen as being in base 256 with the LSB referring to 256^Ø, and the MSB referring to 256^1. The 6502 relies heavily on the concept of groups of 256 bytes. Each group of 256 bytes is called a "page". The memory addresses from $ØØØØ through $ØØFF are referred to as "page zero". Memory addresses from $Ø1ØØ through $Ø1FF is called "page one" (or the stack - see below).

Now that we know what kind of numbers the 65Ø2 can use, we need to examine how it uses them. For all of the things we get our computers to do, the 65Ø2 is essentially capable of only two mathematical operations - adding and subtracting. Furthermore, it can only add or subtract using 8 bit numbers. The one place where the 6502 can use a number larger than 8 bits long is when it is dealing with memory locations. Then it uses a 16 bit value. A 16 bit value has a range of Ø to 65,535, which is actually 65,536 individual memory locations. 65,536 = 64*1Ø24 or 64K. For exercise, try computing and totalling the values of 1x(2^Ø) through 1x(2^15).

Internally, the 65Ø2 always holds 6 numbers. As mentioned above, 5 of these are 8 bit values and one is a 16 bit value. these numbers are held in what are called registers. Registers are like memory locations in that they hold numbers and can be effectively "PEEKed" or "POKEd" with specific commands, but they are inside the 65Ø2 chip itself and are not addressed in memory. To illustrate this, if we were using BASIC, there is no POKE that would set any of the 65Ø2's registers. Only the machine code instructions can affect these registers. The registers are:

The Accumulator

The accumulator is an 8 bit register and is the primary register in the 65Ø2. Only the accumulator can perform the math operations and some of the other special functions.

The Y Register

The Y register is an 8 bit value. It cannot add or subtract, but it can be incremented or decremented with a single command (a function which is unavailable in the accumulator), thus making it useful as a simple counter, but its contents can also be used when the 65Ø2 refers to memory.

The X Register

The X register is almost identical to the Y register and has similar commands available to it. It can be incremented or decremented and used in all of the same ways as the Y register.

The Program Counter

The Program Counter (or "PC") is the only 16-bit register inside of the 65Ø2. The Program Counter holds the memory address of the instruction that the 65Ø2 is currently executing.

The Stack Pointer

The 65Ø2 uses subroutines much like you are used to using in BASIC. An area of memory called the "stack" is used to hold the memory addresses from the PC whenever the 65Ø2 executes a subroutine. The stack occupies memory addresses from $Ø1ØØ through $Ø1FF. The address stored on the stack is where the 65Ø2 "returns" to when the subroutine is completed. Since this return address is a 16-bit value, each "entry" in the stack requires 2 bytes. The Stack Pointer points to the next location on the stack that is free to hold a new return address. The Stack Pointer is only an 8 bit register, but as you can see, since the stack itself is only 255 bytes long, it is capable of handling the task.

The Status Register

The Status Register is an 8 bit register. Seven of the eight bits are used as indicators or flags. Those bits that are used as indicators give the programmer information about the results of previous instructions. Those bits that are used as flags alter the way in which the 65Ø2 operates. The eighth bit is unused in the standard 65Ø2. It may be used in some of the newer versions of the chip.

The 65Ø2 uses a specific set of instructions, or commands. Instead of words like "PRINT" or "INPUT" that we are used to using in BASIC, the 65Ø2 reads its commands as 8 bit numbers. For example, let's say the PC is at memory location $4ØØØ. In location $4ØØØ is the value $C8 or 2ØØ. The 65Ø2 understands the number 2ØØ as being the command to increment the Y register. Other values are used for other commands. This set of instructions is the native language of the 65Ø2 and is also called "machine language" or "machine code". Each instruction is called an "opcode" which stands for operation code. It is possible to use a list of these instructions and, using the monitor ROM in your system, entering each instruction you want to have executed directly to directly program the 65Ø2. Many BASIC programs that use machine code subroutines use the POKE command to put such instructions into memory. But if the desired program is very big at all, this process is extremely tedious. That is where an assembler comes in. An assembler takes a list of commands in text form and converts it into machine code. Each command in the 65Ø2's instruction set has a name that refers to its effect, much like PEEK or POKE in BASIC refer to what they do. However, these names are in an abbreviated form and are not directly pronouncable. You've probably seen an assembly language program and thought it was all in Greek. Not to worry, once you see a list of the commands and what they mean, you'll see that they are actually quite natural abbreviations and are soon memorized. These abbreviations are called "mnemonics" since they are constructed from the words that describe the command. You can find such a list in the SIG Access area in a file called "OPCODE.TXT".

I am fond of saying that assembly language programming is simple, painfully simple. By that I mean that the 65Ø2 has to be told to do everything, nothing can be assumed. This is true for programming in any language, but it is carried to an extreme in assembly language. That's the bad news. The good news is that for all of the commands available to the 65Ø2, most of them can be thought of in terms of 4 commands from BASIC. Those commands are "+" and "-" as I implied before with the accumulator, and PEEK and POKE. So forget any impression you may have had that assembly language programming is going to be hard to learn because you already know about a third of the whole thing.

Alright, we know that the 65Ø2 reads memory locations to get its instructions, but how does it know where to begin? You know that when you turn your system off you lose whatever was in memory. The secret is built into the 65Ø2. When you reset the computer by pressing the <BREAK> key, the 65Ø2 clears itself and looks at a specific memory location to find out where it's to set the PC and begin running. That location(s) is $FFFC (and $FFFD - don't forget the PC needs two bytes) and is called the Reset Vector. In OSI systems, the Monitor ROM contains memory addresses from $FDØØ through $FFFF and thus contains not only the Reset Vector but also the first program your computer runs when it is reset (that's where the "H/D/M" comes from). The Reset Vector holds the address $FEØØ. You can see this by entering the following command at the "OK" prompt in BASIC:

```
  PRINT PEEK(65532),
        PEEK(65533)
```

The computer will respond with:

```
 Ø          254
```

You'll note that the first byte is the LSB and the second byte is the MSB. This convention ot putting the LSB in the lower memory location and the MSB in the next consecutively higher location holds true tor all of the 65Ø2's operations that use 16 bit values.

Before beginning to write an assembly language program, a prime consideration is where to put the program in memory (i.e. the memory address). In all 65Ø2 based computers, some areas ot memory are reserved for special purposes. With our OSI computers, we also need to be concerned with addresses that are dedicated to certain hardware uses. Disk-based systems need to be careful to avoid those memory addresses used by the disk operating system (DOS) whether OS-65D or OS-65U. The object of all ot these considerations is to locate a range of memory addresses that is (1) populated with memory chips, (2) sufficiently large to hold our program, and (3) unused for other purposes. Such areas are often referred to as "free RAM". The following table illustrates the memory map of OSI disk systems:

$ØØØØ through $ØØFF	65Ø2 Page Zero
$Ø1ØØ through $Ø1FF	65Ø2 Stack
$Ø2ØØ through $22FF	Transient Language Area for OS-65D
$23ØØ through $3178	OS-65D V3.2 (8" version)
$3278	End of OS-65D V3.2 (mini-floppy version)
$3279	Start of workspace for mini-floppy 65D V3.2
$3A79	Start of workspace for OS-65D V3.3 all sizes
Start ot workspace	Free RAM
through $5FFF	End of 24K systems
$7FFF	End of 32K systems
$BFFF	End of 48K systems
$CØØØ through $CFFF	Dedicated to disk interface and other hardwrare.
$DØØØ through $D7FF	Black and White Video Memory
$D8ØØ through $DFFF	Various Hardware Purposes
$EØØØ through $E7FF	Color Video Memory
$E8ØØ through $FCFF	Various Hardware Purposes
$FDØØ through $FFFF	Monitor ROM

While it may appear at first glance that there is precious little free RAM available, once you begin programming you'll find that there is really a lot of room.

I think this is enough for you to digest at one time. For next time, study the OPCODES and the commands for the OSI Assembler Editor. Next time we'll let you write your first assembly language program.

BEGINNER'S CORNER PRINTER/MODEM RS-232

By: PEEK(65) staff

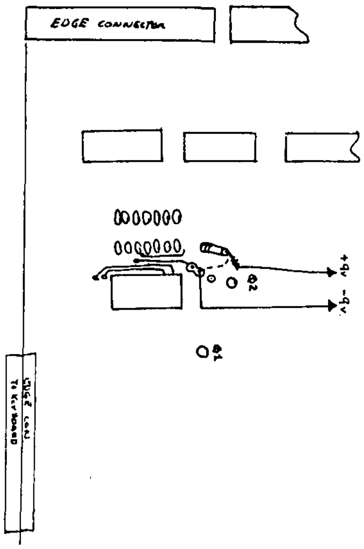
Now that printers and modems are falling in price, terminal programs are available and services like CompuServe are more in fashion each day, more and more OSIers are finding out the hard way that their "P" machines don't have a true RS-232 output at the printer or modem ports.

In simple terms, RS-232 voltage should swing both plus and minus, passing zero. OSI's ports are only zero to plus. While this may be acceptable to some I/O devices, it certainly is not for all.

So, what we have to do is to supply the negative voltage by other means. The options are several: Steal 9 v. from the floppy drive, running still more wires between the boxes. Build a 9 v. supply to fit in the CPU box (a lot of bother). Or simply install a 9 v. battery with a pigtale to the exterior and try to remember to unclip it when not using the outputs. It will drain down, slowly, even though the machine is turned off.

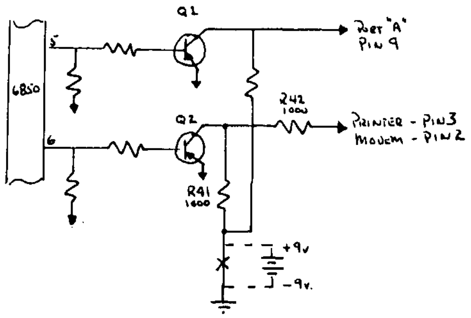
In any of the above cases, the approach and placement is the same, but let's assume the battery approach. Go to the store and get a 9 v. battery and a battery clip. Go home and clip 18", or so, off the nearest lamp cord (lighter guage is better or single strand), get out your hex wrenches, screw driver, and soldering gear. Open up the box and locate the 5Ø5 board (with the machine now up-side-down and the keyboard facing you, it should be on top), the one with the little wires coming out of the front and going to the keyboard. Unplug them after marking one end of the plug and socket with a felt-tip pen, so you will know how to put it back from whence it came. Unscrew the screw in the back-right corner of the boards, being careful not to lose the little spacers that keep the boards separated. Now, carefully, but forcefully, remove the board; pulling first one end a bit, then the other end. Don't try to cheat and use a screw driver to separate the board from the back-plane. You may damage the connectors!

With the aid of the diagrams, locate Resistor R41 (1ØØØ Ohms). The end nearest to you should be unsoldered from the board by applying your soldering iron to the bottom of the board, just long enough to melt the solder and remove the wire. Put one of your new lamp wires back in the hole and solder. The other lamp



wire will be soldered to the end of the resistor, now sticking up.

Replace the board on the backplane, secure the board spacers, connect the keyboard plug and route the new wires out of a convenient hole in the back of the cabinet. Solder the 9 v. battery clip to the new wires; making sure that the plus terminal of the battery is connected to the resistor.



Close her up and you are in business - assuming the printer cabling and hand-shake are correct. More on that subject next time.

★

PRINTER PORTS AND OSI DISK BASIC

By: L. Z. Jankowski
Otaio Rdl Timaru
New Zealand

The OSI memory map indicates $C000-$C7FF as being reserved for I/O. But the printer ports are scattered elsewhere. This article indicates how and where changes can be made and how to implement a PIA as a parallel printer interface.

Serial Port.

The port at $FC00 is moved to, say, $C700.

For OS65D 3.2 & 3.3, make DV#1 a serial printer port at $C700. $C7 hex equals 199 decimal. Add this line to BEXEC*,

```
1 X=199: POKE 9424,X: POKE 9432,X: REM & POKE 9436,X is optional.
```

To add Input (say tape) add,

```
2 POKE 9464,X: POKE 9473,X
```

For OS65U DV#1, use

```
1 X=199: POKE 14928,X: POKE 14933,X: POKE 14364,X: POKE 14370,X
```

Parallel Port

PIA output to printer input is implemented as follows:

1) connect pins 2 to 9 on the PIA (port A) to pins 2 to 9 on a 36 way Centronics printer plug (but only if it fits your printer!)

2) pins 10 & 11 of the PIA (port B) go to pins 1 & 11 of the printer plug.

3) for the printer plug, pins 19 to 27 & 30 go to earth. Pin 14 to earth will force an

extra line-feed (unnecessary with the MX-8Ø). It's that simple!

Here's the software for OS65D 3.2 & 3.3, DV#4.

```
10 FOR X=9375 TO 939Ø: READ A: POKE X,A: NEXT
15 P=49156: REM PIA base address P, is $CØØ4

2Ø POKE P+1,Ø: POKE P,Ø: POKE P+3,Ø: POKE P+2,Ø
3Ø POKE P,255: POKE P+1,4: POKE P+2,1: POKE P+3,4: POKE P+2,1
4Ø DATA 141,4,192,2Ø6,6,192,238,6,192,173,6,192,41,2,2Ø8,249
```

The printer driver is now at $249F. Omit line 2Ø if the PIA has a hardware reset. The final POKE is line 3Ø sends the strobe line high (as for MX-8Ø).

For OS65U DV#5, use this program:

```
3Ø X=15879: REM $3EØ7
4Ø P= 49156: REM $CØØ4
5Ø FOR C=X TO X+25: READ A: POKE C,A: NEXT
7Ø POKE P,255: POKE P+1,4: POKE P+2,1: POKE P+3,4: POKE P+2,1
8Ø DATA 234,234,234,234,234
9Ø DATA 173,182,56,41,127,141,4,192,2Ø6,6,192,238,6,192
1ØØ DATA 173,6,192,41,2,2Ø8,249
```

If the above program does not work for versions 1.4 & higher try

```
3Ø X=15884: REM $3EØC
4Ø P=49156: REM $CØØ4
5Ø FOR C=X TO X+23: READ A: POKE C,A: NEXT
6Ø POKE P,255: POKE P+1,4: POKE P+2,1: POKE P+3,4: POKE P+2,1
7Ø DATA 173,182,56,41,127,141,4,192,2Ø6,6,192,238,6,192
8Ø DATA 173,6,192,41,2,2Ø8,249,76,43,59
```

The number 192 decimal, in the above DATA statements is, of course, the $CØ of $CØØ4. Change the number to suit your PIA base address. The most practical way to use the programs is to make them part of BEXEC*.

★ ★

ED:

In the June issue of PEEK(65), Carl King asked how to change the default page numbering prompt in WP-65Ø2. Unfortunately, Mr. King neglected to state whether he was running the cassette or the disk version.

In the cassette version, the location to be changed is $Ø237. The normal value is $Ø1. This should be changed to $ØØ to eliminate all page numbers. After the change, the revised copy should be saved to cassette in order to make the change permanent.

If the diskette version is being used, the problem is a little tougher. I do not have the disk version of WP-65Ø2, but I believe that all versions share common code. When WP-65Ø2 is Warm started, a number of values are moved from page two to page zero. John Roecker could probably provide a list of all of these values. I disassembled the code starting at location $Ø222. When using the disassembler, tables of data become rather obvious. They usually will not disassemble to meaningful code and if they do the addresses are often outside of the normal system memory space.

I would suggest that disk users load the Extended Monitor, CALL the initial code of WP-65Ø2 into memory at some available workspace address and start disassembling the first portion of the code. If the disassembly produces a lot of question marks or nonsense code, you have found the Warm Start table! Now look through the table for bytes with a value of $Ø1. Now try changing the value of each byte from $Ø1 to $ØØ and observe the result on the 'View' menu. I only found three and the second one changed the prompt for page numbering.

Hope this will be useful to some of the WP-65Ø2 users.

Harry B. Pye
Lansdale, PA 19446

* * * * *

ED:

In reviewing my Tax Preparation program which you published in the April 1984 issue, I came across a glitch that I had not previously noticed in my program.

It would occur in situations where you had computed your tax and had a situation where you had overpayed your tax to that date. If you then stored the file and later recalled and updated it such that you now had an underpayed condition, the program did not zero out lines 65 and 66 Overpayed and Refund to You.

I have rewritten the code in error to fix this problem, and have enclosed the lines to be changed in this letter.

entered
su

```
1085 FORA=59TO65:I(66)=I(66)+I
     (A):NEXT:A=I(58)-I(66)
1Ø9Ø IFA<ØTHENI(67)=ABS(A):I
     (68)=I(67)-I(69):I(7Ø)=Ø:
     RETURN
11ØØ I(67)=Ø:I(68)=Ø:I(69)=Ø:I
     (7Ø)=Ø
1105 IFA>ØTHENI(7Ø)=A
111Ø RETURN
```

REPLACE <u>ALL</u> LINES BETWEEN 1Ø85 AND 111Ø WITH THE ONES HERE.

Robert S. Baldassano
San Jose, CA 95124

* * * * *

ED:

Seeing my stuff in print brings out the ham!

Here is another goodie. Have you ever wanted to merge a file with a long field into a shorter field by chopping off the right side of the field that is excess? Line 1Ø55 does this. The question is asked in Line 4Ø2; program line changes are lines 4Ø1 to 4Ø6 - 1ØØ5 - 1Ø45 - 1Ø5Ø - 1Ø55.

STANDARD DMS MERGE MOD.

```
4ØØ PRINT:INPUT"SEQUENTIAL MERGE (S) OR COUPLED MERGE (C) ";TM$
4Ø1 IFTM$<>"S"GOTO4Ø6
4Ø2 PRINT"SHORTEN TO FIELDS TO FIT SHORTER FROM FIELDS IF NECESSARY";
4Ø3 PRINT" (Y OR N)":INPUTQQ$
4Ø4 IFQQ$<>"Y"ANDQQ$<>"N"THENPRINT"ANSWER (Y OR N) !":PRINT:GOTO4ØØ
4Ø5 GOTO43Ø
4Ø6 IFTM$<>"C"THENPRINT:PRINT"WHAT!":GOTO4ØØ
4Ø7 IFBODF(K2)<EODF(K2)GOTO41Ø
4Ø8 PRINT:PRINT"IT IS IMPOSSIBLE TO COUPLE MERGE, THE";NF$;"IS EMPTY!":
41Ø PRINT:PRINT"<<<NOTE>>> DATA WILL STILL BE MERGED INTO THE";NF$

999 REM  EXECUTION BLOCK FOR SEQ MERGE
1ØØØ IFLR$="Y"THENV1=(BO-K1)*RL(K1):V2=(BN-K1)*RL(K2):CR=BO-K1
1ØØ5 L1=1Ø1Ø:L2=1Ø6Ø:L3=1Ø8Ø:L4=1Ø55
1Ø1Ø RP=RP+K1:INDEX<K1>=BODF(K1)+((RP-K1)*RL(K1))+V1:RT=INDEX(K1)
1Ø15 INDEX<K2>=BODF(K2)+((RP-K1)*RL(K2))+V2:TT=INDEX(K2)
1Ø25 PRINT"RECORD";RP+CR:PRINT
1Ø3Ø DEVDV$(K1):FORX=K1TORC:INDEX<K1>=RT+FP(X):INPUT%K1,D$(X)
1Ø4Ø IFLEN(D$(X))<=LF(X)GOTOL2
1Ø45 IFQQ$="Y"GOTOL4
1Ø5Ø IFQQ$="N"THENPRINT:PRINT"CHOSE TO EXIT":GOTO4ØØØØ
1Ø55 D$(X)=LEFT$(D$(X),LF(X)):NEXT:GOTOL3
1Ø6Ø D$(X)=RIGHT$(S1$+D$(X),LF(X)):NEXT
1Ø8Ø DEVDV$(K2):FORY=K1TORC:INDEX<K2>=TT+PF(Y):PRINT%K2,D$(Y):NEXT
1Ø9Ø IFLR$<>"Y"ANDRP<E1GOTOL1
1Ø95 IFLR$="Y"THENIFRP<E1ANDRP<(BE-BO)+K1GOTOL1
11ØØ IFINDEX(K2)<=EODF(K2)GOTO112Ø
111Ø EODF(K2)=INDEX(K2):INDEX<K2>=K9:PRINT%K2,EODF(K2)
112Ø GOTO54Ø45
```

Richard E. Orgel
Memphis, TN 38118

* *

ED:

Regarding Scottie Cantrell's questions using a Shugart SA-400 drive with his OSI, I have been using an MPI B51 and a Shugart for approximately 2 years with no problems. My machine is a C4 with D & N's memory expander and floppy controller boards and disk cable running OSD 3.3. According to some old notes probably taken from a PEEK(65) article, the following changes need to be made. $2769 to $34, $26A3 to $20, $26CA to $35, $2779 to $34, $2DA7 to $34 and $2DB7 to $34. According to what I could determine, the first four locations are floppy drivers and the last two are from the operating system kernal.

Booting up using the MPI and then poking 9891,32 enables both drives to be used either together or separately. The other changes are necessary to allow the use of 35 rather than 40 tracks. If one is careful, things will work out without making the other changes to the system with no #5 errors.

Right after reading last month's PEEK, I misplaced it, and I wanted to respond to someone who was requesting a bowling program. I have one I have been using for three years that maintains averages, prints standing sheets, sorts bowler weekly scores, and some other things. It is not the best but it has been working for me. I only have six teams in my league and 32 K and it just about fills memory, although, it is set up so that just a couple of changes should be necessary for more teams if the machine has more memory. We have 5 men's teams. If that person would write me, maybe we could work something out.

And now for my question. Recently, almost every time I use my machine for an hour or more, it locks up, usually when printing. I suspect that it is because of the heat from the power supplies but do not know for sure. When, after it locks up, I try to go to the kernal by pressing break ,M,.2547G. Sometimes I do get back to the kernal, others I don't. If I do get back to the kernal and try to go back to BASIC with RE BA, the BASIC prompt returns but the machine is still locked up. Does anyone know what may be causing this, or has anyone had this problem before? Not being too versed in the ways of the inner workings of these machines, I have nowhere to turn and any input would be appreciated.

Steve Rydgig
#4 Frontenac R.R. #2
Collinsville, IL 62234

continued

Steve:

The following is not really a "fix", but may bail you out - for the moment. Heat and dirt are two devious culprits. Hence a devious answer. Heat makes things move and dirt makes for "flaky" contacts. Chances are that if you open her up, remove the boards and press the chips firmly down in their sockets, you may have solved the problem. If not, let us know. We'll be scratching our heads.

P.S. Hope that you will be submitting your program for the software listing!

AD$

Send for free catalog, Aurora Software, 37 South Mitchell, Arlington Heights, IL 6ØØØ5.

* * * * *

SCHOOL STARTER SYSTEM: Challenger 8P ($2ØØ), dual disk drive ($1ØØ), U8Ø Microline Printer ($1ØØ), networked to eight 48K Challenger 1s ($1ØØ each). Entire system $1ØØØ, all system and student manuals, assorted disks included (or best offer on system or components.) Some repairs needed. Inspect by appointment. Contact Business Office, Madeira School, Greenway, VA 22Ø67, (7Ø3)-893-42ØØ.

* * * * *

OSI-23ØE, Hazeltine 142Ø Terminal, and Tech 25 Printer, must sell. Contact L. Hagerty, 1187 Piermont, South Euclid, OH 44121, 216-381-736Ø.

* * * * *

C4P set up for dual 5 1/4". Socketed for 4ØK 24K installed. 65D, all manuals, newsletters, etc. No drive, needs ext. power supply. Asking $35Ø. Mark Episcopo, (215) 441-3661 day. (215) 489-28Ø3 evenings.

* * * * *

C8D-DF for sale. 48K, Siemens drives. OSI 65D3.2, 65D3.3, ED2, BD2, MDMS, MDMS Aux-1, AARDVARK Mini-Pros WP, Games Disk 1, extra disks, disk cases, documentation, tutorials. Excellent condition, original cartons. $5ØØ or best offer. D. Gillis, Pomeroy School District, PO Box 95Ø, Pomeroy, WA 99347, (5Ø9) 483-3393.

* * * * *

OSI 23Ø E, 8MB hard disk with CP/M, OSU software, never used, $3,ØØØ or best offer. I. Kaufman 7Ø3-522-14Ø7.

FREE PEEK(65)
SOFTWARE LISTING FORM

PROGRAM NAME ______________________________

BASIC - VERSION /_/_/_/_/

/D/ OS65-D /U/ OS65-U /C/ CP/M

CATEGORY

/G/ GAME /B/ BUSINESS /U/ UTILITY /O/ OTHER

MINIMUM COMPUTER

/1/ SB,SBII, C1P,C2/4P /4/ C4P /8/ C8P /O/ C2/3OEM

/D/ C2/3D /2/ C200,C3A/B /3/ C300

MINIMUM STORAGE REQUIREMENTS

/C/_/ CASSETTE /5/_/ 5 1/4" /8/_/ 8" FD /7/_/ CD-7

/2/_/ CD23/28/36/74 (USE 2ND BOX FOR NO. UNITS REQUIRED)

TERMINAL SYS. /S/ SERIAL /V/ VIDEO /B/ BOTH

SYSTEMS SUPPORTED - MAX 2 CODES - RECORD LOCK ASSUMES MULTI-USER

/S/ ONE USER /M/ MULTI-USER /H/ HARD DISK /R/ RECORD LOCK

SOFTWARE SUPPORTED BY:

/D/ DEALER /P/ PHONE /M/ MODEM /N/ NONE

/O/ OTHER

SOLD BY

/A/ AUTHOR /D/ DEALER /M/ MAIL ORDER /O/ OTHER

COPIES IN CIRCULATION (# / 10); 1 = <11, 11 = 100-110 /_/_/

PRICE, RETAIL, PLUS TAX & HANDLING OR SASE $/_/_/_/_/NO CENTS

NAME AND ADDRESSES

AUTHOR	SELLER (or SAME)
NAME ______________	NAME ______________
STREET ______________	STREET ______________
CITY ______________	CITY ______________
STATE /_/_/ ZIP ________	STATE /_/_/ ZIP ________
PHONE (___) ___-____	PHONE (___) ___-____

DESCRIPTION - MAXIMUM 10 LINES OF 30 CHARACTERS

1. ______________________________
2. ______________________________
3. ______________________________
4. ______________________________
5. ______________________________
6. ______________________________
7. ______________________________
8. ______________________________
9. ______________________________
1Ø. ______________________________

PEEK (65)
The Unofficial OSI Users Journal

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

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 0S-65U. Complete disk sort and merge, documentation shows you how to call from any BASIC program on any disk and return it or any other BASIC program on any disk, floppy or hard. Most versatile disk sort yet. Will run under LEVEL I, II, or III. It should cost more but Sanders says, "...sell it for just..."	$89.00	$ ________
( )	**KYUTIL** - The ultimate OS-DMS keyfile utility package. This implementation of Sander's SORT/MERGE creates, loads and sorts multiple-field, conditionally loaded keyfiles. KYUTIL will load and sort a keyfile of over 15000 ZIP codes in under three hours. Never sort another Master File.	$100.00	$ ________
	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