PEEK (65)

The Unofficial OSI Users Journal

★★ $1.75 ★★

Editor: - Al Peabody
Vol. 2, | No.10, Oct.1981

P.O. Box 347
Owings Mills, Md. 21117
(301) 363-3267

INSIDE:

Column One

The way I write this column is, I take a look at the latest issue of PEEK(65), hanging up on the wall, and see just what seems to be the emphasis of the issue. Then I write about that, since this column is mostly a report on the status of things OSI, from the vantage point of this desk.

This month, I am not too happy with what I see. Sure, there is plenty of good, serious information for the OSI user. J.B. Boardman is modifying his machine extensively, to use CP/M and 65D easily and automatically, and soon will have it modified to use any disk format automatically. There are columns and letters for the cassette and disk user, the polled keyboard man and the serial terminal user, and programmers of all levels of sophistication, from the beginner at business Basic to the very sophisticated programmer. The information in this month's issue can even help you get started in FORTH, PASCAL or Assembler or a new and powerful Basic which can call machine language routines from any location, including the system or ROM.

So what's to dissatisfy? It's simply that this entire issue, like most issues of PEEK(65), assumes a great deal of knowledge on the part of our readers. We write for all sorts of machines, but very little for the novice. And let's face it: as computer use expands, and more and more hobbyists and small businessmen buy microcomputers, there are more and more novices out there. We aim to reach the <u>whole</u> OSI community, not just the most sophisticated 20%!

One reason for PEEK(65)'s existence is the lousy coverage given OSI gear by all the large national computer magazines. Believe me, I don't want to see us start publishing the same sort of trivia we see by the ream in certain mags which will remain nameless... you know the sort of article I mean... "Six Handy Basic Subroutines to Add 2+2"... but, on the other hand, we are doing no one a favor if our magazine can only be understood by the select few.

So, here it is, another

CALL FOR ARTICLES

for PEEK(65). Sure, we still want good technical information, and programming tricks, and modifications to boards. But we <u>also</u> want good introductory level articles for hobbyists and especially small business users. The kind of article which will tickle the imagination of the fellow who has never done more than run the programs he bought with his computer. Tell him just what that machine he owns can really do, easily, without a genius programmer and super hardware hack. Remember, PEEK(65) pays cash money for articles, so start 'em coming in. I am tired of writing 30% of the magazine! Push those keys! Lift that bale! Get to work!

In much this same vein, perhaps a word or two is in order about CP/M, and maybe even about the Beginning Assembler series which concludes (for the time being at least) in this issue.

A couple of people have commented that the emphasis on CP/M did not serve the whole OSI community well, since most OSI owners don't have C3 machines, and therefore can't use CP/M. One friend even said something like "Since PEEK(65) has switched to CP/M..." Heaven forbid!! PEEK(65) has <u>not</u> switched to CP/M. Nor to cassette. Nor to 65U.

But the point is well taken. We certainly have published a lot about CP/M, revealing our (maybe just my) excitement about CP/M 2.2. It is just possible we have given a wee bit too much ink to CP/M. Don't worry, faithful friends, we will balance the ledger in future issues. Stick with PEEK(65) for a whole year and you will get more solid information for your machine, no matter what it is (so long as it says Ohio Scientific on the front) than you will find in all the other computer magazines combined. And that's a promise.

Speaking of good information: hold your breath for the December issue. If you plan to move in the next couple of months, let us know well in advance -- be sure you don't miss December's PEEK(65). It will be a real Christmas present to our friends.

al

OPERATING SYSTEM CALLS FROM FBASIC

by John Fuller
14211 Apple Tree
Houston, TX 77079

The FBASIC Compiler has several unique features which allow you to call the various routines within the OS-65D Operating System directly. Coupled with FBASIC's great speed advantage over OSI BASIC they can be quite handy.

FBASIC's GOSUB and GOTO statements have a new twist which allows an absolute memory address to be specified in place of the usual line-number. This is signaled by preceding the number with an exclamation point (!). Thus, the statement GOTO !$2A51, would cause a jump to the operating system command loop which is located at $2A51. The dollar-sign ($) denotes a hexadecimal constant, and may be used anywhere a constant is applicable. The statements: GOTO !$2A51 and GOTO !10833 are equivalent.

Another handy feature for our purposes is the dot (.) operator which allows direct access to the 6502 registers. As an example, OS-65D has a routine that will print the value of the accumulator to the console. This is quite useful, but hard to use from OSI BASIC. With FBASIC two statements are sufficient:

```
10 .A=NUM
20 GOSUB !$2D92
```

The statement in line 10 uses this special operator to place the value stored in the variable named NUM into the 6502's accumulator or A register. Then the OS-65D routine to print this value in hexadecimal is called.

PEEK (65) is published monthly by DBMS, Inc., Owings Mills, MD 21117. Editor: Al Peabody.

Effective July 1, 1981
Subscription Rates

US (surface)	$15
Canada & Mexico (1st class)	$23
So. & Cen. Amer. (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

You should note that since the 6502's registers are only eight bits wide, the higher bits in the variable (NUM) are ignored. Also be advised that a value loaded into one of the processor registers usually does not stay there long. The only statements which are guaranteed not to change the registers are: GOTO, GOSUB, RETURN, REM, and a simple load or store of a different register.

As an example of the last case: .A=NUM followed by .Y=N2 will not destroy the value loaded into the A register (NUM). But, if the second statement contains more than a simple variable, as in: .Y=N2+8, then A register will probably be used to do the addition. Therefore, in this case, the Y register should be assigned first.

To print a 16 bit value in hex use the following code:

```
10 .A=NUM/256    : REM print
   the high byte first
20 GOSUB !$2D92
30 .A=NUM        : REM then the
   low byte
40 GOSUB !$2D92
```

Now, with the basics out of the way, on to some useful OS-65D routines. Here's how to read and write sectors to disk with more or less total control:

```
100 POKE 9826,TN          : REM
    track number
120 POKE 9822,1           : REM
    sector number
130 POKE 9823,11          : REM
    number of pages
140 POKE 9824,BUFF/256    : REM
    address of buffer high
150 POKE 9825,buff        : REM
    address of buffer low
160 GOSUB !$26A6          : REM
    seek track
170 GOSUB !$2754          : REM
    load head
180 IF RW THEN GOSUB !$27D7 :
    GOTO 200 : REM write
190 GOSUB !$295D          : REM
    read from disk
200 GOSUB !$2761          : REM
    unload head
210 RETURN
```

To read from disk RW is set to 0, to write it should be set to non-zero. TN should contain the track you wish to use, and BUFF should be set to the memory address you wish to use. If the FBASIC program you are writing is not to be called from OSI BASIC then you can use that memory for your buffer (from $0200 to $2200). If several tracks are to be accessed in quick succession then the disk head may be left loaded until the last access is made. This will speed the process slightly, and will help to keep the disk noise down.

And how do you find a disk file? Easy, just use this little routine:

```
400 POKE $E2,$2E :POKE $E1,$1E
410 GOSUB !$2C9B :REM input to
    kernel buffer
420 POKE $2CE5,0 :REM zero
    buffer pointer
430 POKE $2CED,17:REM set
    buffer length
440 GOSUB !$2DA6 :REM search
    directory for match
450 TL=.A        :REM first
    track returned in A
460 TL=TL AND 255:REM make
    sure high byte is zero
470 TH=PEEK($E5) :REM last
    track of file
480 TL=TL/16*10+(TL AND 15)
    :REM convert BCD to binary
490 TH=TH/16*10+(TH AND 15)
500 RETURN
```

Line 410 is a call to the routine which takes input from the console and stores it in the DOS kernel buffer. If desired you can instead set up the pointer in line 400 to point to a file name in memory somewhere. The name should be terminated by a carriage-return ($0D). If the file is not found, the usual error C will be reported, and the program will be aborted. If the file is found the routine will return with the number of the first track in TL and the last track in TH.

A quick and dirty way of gaining access to a data file is to open it with OSI BASIC and then jump to your FBASIC program to read from or write to it. To do this, the buffer should first be moved to the top of memory, or some other out of the way place. I've always considered OSI's placement of disk buffers below the BASIC text area as none too smart (putting it kindly).

So for a 48K system, and assuming the use of standard $C (12) page data file tracks, the following OSI BASIC program would be appropriate to start things off:

```
10 POKE 8998,0 : POKE 8999,180
   : REM start of buffer
20 POKE 9000,0 : POKE 9001,192
   : REM end of buff +1
30 INPUT "File name"; F$
40 DISK OPEN,6,F$
50 DISK!"GO 317E"      :REM
   call FBASIC program
```

In lines 10 and 20 the modifications are made to the operating system to move the

disk buffer for device 6. Then the file is opened and control transferred to the FBASIC program. If a different size track is to be used simply change the pointers accordingly, the Operating System will do the rest.

With FBASIC, files can be accessed a character at a time:

```
100 POKE 9826,TN  :REM track
    number
110 GOSUB !$2336   : REM
    standard input without
    echo
120 CH=.A          : REM get
    char from A register
200 POKE $2322,$20  : REM set
    output to device 6
210 .A=CH          : REM get
    char into A register
220 GOSUB !$2343   : REM
    standard output char
```

The locations $2321 and $2322 are the input and output device flags. A $20 placed here enables device 6, a $40 enables device 7. More information on disk file pointers can be found in the OS-65D manual on page 58, and page 8 in the appendix.

FBASIC's INPUT and PRINT statements may be directed to files by POKEing the OS-65D input/output distributors with the appropriate values. A $20 for device 6 and $40 for device 7. The PRINT# and INPUT# are not supported by FBASIC because they are somewhat limiting. Using direct access to the input /output distributors instead, several devices may be addressed simultaneously. With INPUTing from a file the output distributor should be temporarily set to zero. Otherwise the input will be echoed to the console.

```
100 POKE $2321,$20 : REM input
    from device 6
110 POKE $2322,0   : REM kill
    echo
120 INPUT A
130 POKE $2321,1   : REM
    restore I/O
140 POKE $2322,1
```

For more goodies take a look at Software Consultants OS-65D Disassembly manual; it provides some good ideas on other routines which you can incorporate into your own programs.

I've been using FBASIC for almost four months now, and I'm constantly amazed at its speed and versatility. At first I thought the $150 price-tag was a bit high. But the people at Pegasus have put a lot into it. And I've gotten a lot out of it.

FBASIC is available from Pegasus Software, P.O. Box 10014-P, Honolulu, HI 96816, (808) 735-5013.

★

RND(X) CLARIFIED

by Arnie Penaloza
9105 Cherry Avenue
Morton Grove, IL 60053

Problem:

The RND(X) function was found to have a cycle of 1861 numbers. This is true when the RND function is not RANDOMIZED at the beginning of the program. RANDOMIZE means, to cause the generation of a new series of random numbers for the RND function each time the program is run.

Solution:

How do we RANDOMIZE? Well, Mr. E. Carlson, in his book ALL ABOUT... BASIC IN ROM, gives us the method. Carlson's method is to use a RND(X) call, where X is any negative number, to RANDOMIZE the RND function. For example,

```
10 GARBAGE = RND(-1) : REM
   RANDOMIZE RND FUNCTION
```

might be at the beginning of a program.

Caution:

Two interesting properties show up.

1. Any negative argument used will produce its own unique series of random numbers. Try the following program to convince yourself.

```
10 X=-1:REM CHANGE TO ANY
   NEGATIVE NUMBER
20 GARBAGE=RND(X):REM
   RANDOMIZE RND FUNCTION
30 REM GARBAGE IS USELESS AS A
   RANDOM NUMBER
40 FOR I = 1 TO 5:REM JUST
   LOOK AT FIRST FIVE NUMBERS
50 PRINT RND(I):REM THE
   ARGUMENT CAN BE ANY
   POSITIVE
55 REM NUMBER, THE RND
   FUNCTION DOES NOT CARE.
60 NEXT I
```

Run a few times changing X in line 10 to any negative number.

2. These unique series are not cyclic. Unless, the same negative argument is used to re-RANDOMIZE, in which case the series repeats itself. Try the following to convince yourself that negative arguments are not cyclic, while positive arguments are.

```
 5 I=0:REM CYCLE FINDER
10 X=-1:REM CHANGE TO ANY
   POSITIVE OR NEGATIVE NUMBER
20 GARBAGE = RND(X):REM
   RANDOMIZE
30 REM SAVE FIRST THREE
   NUMBERS IN THE SERIES
40 A=RND(1):B=RND(2):C=RND(3)
45 REM THE RND FUNCTION DOES
   NOT PAY ATTENTION TO
46 REM ANY POSITIVE ARGUMENT
50 PRINT CHR$(13);I; :REM
   PRINT HOW MANY NUMBERS
   CALLED
55 REM COMPARE 3 SAVED NUMBERS
   WITH EACH RANDOM
56 REM NUMBER CALLED, IF ALL 3
   MATCH THEN CYCLE FOUND.
60 I=I+1:IF A=RND(1) THEN 80
70 GOTO 50
80 I=I+1:IF B=RND(2) THEN 100
90 GOTO 50
100 I=I+1:IF C=RND(3) THEN 120
110 GOTO 50
120 PRINT:PRINT"THE CYCLE
    IS";I:END
```

Summary:

Therefore, to properly use the RND function, we must RANDOMIZE with any negative argument at the beginning of a program. If this is done, we will have a unique series of noncyclic random numbers to work with.

★

THE 65U GOODIE BOX

by
Ken Holt
H/B Computers
217 E. Main Street
Charlottesville, VA
(804) 295-1975

I've noticed that PEEK(65) has been leaning pretty heavily toward hobbyist systems lately. What about us people with business systems? We need help, too. This column will be largely for business systems: those using OS-65U with serial terminals, and, in many cases, hard disks. A lot of you business system people must have a lot of good stuff to offer, so let's share and make life easier for all of us. If you don't have the time or inclination to get your goodies into shape for an article, send me your raw notes and I'll put them in this column. Either way, let's show Al that we're out here.

For this month, we'll look at a problem which has been the

digital technology

BUS-II LEVEL I BOOKKEEPING & ACCOUNTING SYSTEM
The BUS-II turn-key multi-client accounting package is the leading OSI business software package. BUS-II Version 3.1 includes five principle modules:

	Inst. Price	Ref. Price
GENERAL LEDGER	$1200	$599
ACCOUNTS RECEIVABLE (a)	1000	599
ACCOUNTS PAYABLE (a)	1000	599
ORDER ENTRY W/ INVENTORY (a)(b)	1000	599
PAYROLL	1200	799

The Accounts Receivable, Accounts Payable, and Order Entry W/Inventory are completely interactive with the BUS-II General Ledger. Two optional specialized packages (completely interactive) are also available.

CPA EXTENSIONS (see below)
POINT-OF-SALE TERMINAL W/INVENTORY (see below)

The BUS-II CPA EXTENSIONS Package provides special features for accountants and bookkeepers. The POS-1 Point-of-Sale Terminal package enables the operator to use the computer system's video terminal as an on-line "electronic cash register".

Note: BUS-II V 3.1 operates on floppy-disk or hard disk-based systems running the OS-65U V 1.2 operating system (LEVEL I, II, or III). Multi-client use can accommodate any number of client companies on floppy disk systems or hard disk systems with H/D/E (required for hard disk use). BUS-II LEVEL I files are limited in size for floppy disk back-up; floppy disk operation continues in case of hard disk failure.

BUS-II LEVEL II (EXPANSION TO BUS-II LEVEL I)
BUS-II LEVEL II is designed for much larger businesses. Expanded file size and special operations allow virtually unlimited numbers of accounts and transactions. BUS-II LEVEL II requires BUS-II LEVEL I. Minimal back-up is data cassette (tape) or floppies—although multiple Winchester disk operation is recommended (provides ability to continue computerized bookkeeping functions in case of hard disk failure). H/D/E Hard Disk Executive is required.

	Inst. Price	Ref. Price
GENERAL LEDGER	$ 600	$ 399
ACCOUNTS RECEIVABLE	600	399
ACCOUNTS PAYABLE	600	399
ORDER ENTRY W/ INVENTORY	600	399

POS-1 POINT-OF-SALE TERMINAL (a)(b)
POS-1 is an on-line multi-store point-of-sale terminal program with integrated inventory designed for cash register emulation. POS-1 controls cash drawer and ticket printer (or system printer). Automates taxable or nontaxable sales, cash transactions, and credit sales (with verification operations). POS-1 is interactive with the BUS-II V 3.1 BOOKKEEPING & ACCOUNTING SYSTEM.

POS-1 Inst. Price $2400 Ref. Price $1199

CPA EXTENSIONS PACKAGE (a)
CPA EXTENSIONS is designed for public accounting firms. A number of special operations are provided: a "bankers" Balance Sheet and Profit and Loss statement with summarization and consolidation options, Statement of Changes in Financial Position, Statement of Changes in Components of Working Capital, Cash Flow Analysis, Departmentalized Sales Analysis, Comparative Income Statement, Budgetary Analysis, Asset Depreciation Schedule (compatible with TAXMAN-1040), and Loan Amortization Schedule. CPA EXTENSIONS is interactive with BUS-II V 3.1 BOOKKEEPING & ACCOUNTING SYSTEM.

CPA Extensions Inst. Price $ 3600 Ref. Price $1599

TAXMAN-1040 PERSONAL INCOME TAX PREPARATION
TAXMAN-1040 is designed for tax practitioners and public accountants. TAXMAN-1040 is the leading tax package for OSI microcomputers—the package has been installed on OSI, Hewlett-Packard, DEC and IBM systems. Designed and supported by CPA tax experts. This package automatically prepares FORM 1040 and 28 schedules. Individual state tax option available. Support includes annual forms and tax table revisions. Purchasers of 1980 TAXMAN will automatically receive 1981 revisions at no extra charge.

TAXMAN-1040 Inst. Price $3600 Ref. Price $2399

TAXMAN-1120 CORPORATE INCOME TAX PREPARATION (a)
TAXMAN-1120 (under development) is a corporate tax preparation package designed to work in conjunction with TAXMAN-1040 to provide full-service tax accounting functions. TAXMAN-1120 requires BUS-II G/L. Individual state tax option available; support includes annual forms and tax table revisions. Purchasers of 1980 TAXMAN will automatically receive 1981 revisions at no extra charge.

TAXMAN-1120 Inst. Price $3600 Ref. Price $2399

OS-DMX DATABASE MANAGEMENT SYSTEM
Command-oriented OS-DMS compatible database management system. OS-DMX operates under the OS-65U V 1.2 operating system (LEVEL I, II, or III). Features such as control files, extensive operating commands, and the innovative HELP feature, in addition to Digital Technology's exclusive on-line documentation, make this one of the most usable—as well as powerful—systems available for microcomputers. OS-DMX may be used instead of, or in addition to, OS-DMS Nucleus, Query, Sort; OS-DMX will replace virtually all of the specialized OS-DMS modules—and in most applications will provide greatly improved performance.

OS-DMX Inst. Price $2000 Ref. Price $1199

ECR-1(P) ELECTRONIC CASH REGISTER POLLING (c)
ECR-1(P) provides cash register polling and control (including cash register reprogramming) in conjunction with OSI microcomputers. Cash register polling is an alternative to on-line operation which allows the use of regular preset-total style electronic cash registers (with RS-232 communications). Versions are currently available for MKD BANTAM II and certain NCR cash register systems.

ECR-1(P) Inst. Price $1600 Ref. Price $799

ECR-1(C) DATA CASSETTE POLLING (c)
ECR-1(C) provides data cassette polling, allowing multi-store cash register polling. ECR-1(C) is recommended when diverse store locations make telephone line communications prohibitively expensive.

ECR-1(C) Inst. Price $1600 Ref. Price $799

SALES-1 SALES ANALYSIS (c)
SALES-1 is an OS-DMX-based sales analysis package for use in conjunction with OS-DMX, ECR-1(P), or ECR-1(C). Breakdown is provided by key-hit, family group, etc., indicating totals and percentages of sales. OS-DMX is required; ECR-1 is recommended; manual stand-alone operation is optional.

SALES-1 Inst. Price $1600 Ref. Price $799

INV-1 RESTAURANT INVENTORY & MENU EXPLOSION (c)(d)
INV-1, used in conjunction with OS-DMX, ECR-1 and SALES-1, provides a detailed breakdown of sales by family group and menu components. Provides managerial information detailing waste, pilferage, menu costs, stock levels, reorder levels, percentage-of-sales and percentage-of-cost from menu explosion. OS-DMX required; ECR-1 and SALES-1 recommended; manual stand-alone operation optional.

INV-1 Inst. Price $1600 Ref. Price $799

H/D/E HARD DISK EXECUTIVE
Digital Technology's implementation of H/D/E is the answer to AMCAP's HDM. Digital Technology's H/D/E provides user functions not found on HDM or similar products: ability to copy from any user "system" to another; automatic recovery in case of "back-up to floppy" or "restore from floppy" utility failures, allowing the user 3 options: (1) ignore error, (2) abort to menu, (3) try again; use of both "A" and "B" floppy drives to back-up hard disk files; and automatic back-up diskette initialization. H/D/E operates on any OSI Winchester disk system from 7 - 80 megabytes. Re-use of hard disk space is provided. Superior to AMCAP's hard disk manager in every respect (and Digital Technology software does not self-destruct). NOTE: H/D/E is required when installing any Digital Technology business applications packages on OSI hard disk systems.

H/D/E Inst. Price $800 Ref. Price $499

H/D/M MULTI-USER MANAGER (g)
H/D/M (under development) is Digital Technology's multi-user extensions to OS-65U. Replaces T-MUM by AMCAP. Need we say more?

H/D/M Inst. Price $1200 Ref. Price $499

DIGITAL TECHNOLOGY, INC. software is sold through OSI Dealers worldwide. For detailed product information call (714)270-2000. For the name of your nearest OSI Dealer call (toll free) OSI's "hot line" 1-800-321-6850.

digital technology
inc.
P.O. BOX 178580
SAN DIEGO, CA 92117
(714) 270-2000

REQUIREMENTS
(a) BUS-II LEVEL I G/L req'd
(b) BUS-II LEVEL I A/R req'd
(c) OS-DMX req'd
(d) ECR-1 recommended
(e) C3 CPU W/56K RAM & OS-CP/M or Lifeboat Associates CP/M req'd
(f) SYNCHRONOUS INTERFACE ASSY req'd
(g) H/D/E Required

bane of nearly all OSI business application programmers: screen formatting. An approach which has been used all too frequently is to hard-code the terminal-handling logic right into the program. It's easy to do, but Heaven help you if you need to make it work on a terminal different than the one the program was designed for.

With a little more effort at the program design stage, a lot of time and grief can be spared later. Instead of hard-coding terminal-dependent logic all through the program, isolate it to one subroutine. Then, if the terminal type changes, there's no need to comb through the code to find all the places that need to be revised; just change the subroutine, and you're done!

You should take extra care at the beginning to properly design the subroutine. Think of what sorts of things you will need to do to the screen, and how they should be specified. Make these specifications general enough that they will apply to nearly any terminal, not just a Hazeltine 1420 (or whatever you are using).

Listed is an example of such a subroutine. It handles four different screen-formating functions which are available on most recent-model terminals. To perform the function, set the variable FU equal to 1, 2, 3, or 4 according to the function desired, set X and Y to indicate the desired coordinates on the screen, then GOSUB 19900.

```
19900 REM
19901 REM  SCREEN FORMATTING
      ROUTINES MICRO-TERM
      ACT-5A
19902 REM
19903 REM  CALL WITH FU, X,
      AND Y SET
19904 REM
19905 REM  FU=1: GO TO X,Y
      AND CLEAR TO EOL
19906 REM  FU=2: GO TO X,Y
      ONLY
19907 REM  FU=3: CLEAR SCREEN
      AND GO TO X,Y
19908 REM  FU=4: GO TO X,Y
      AND CLEAR TO EOS
19909 REM
19910 ON FU GOTO 19920,19930,
      19940,19950: STOP
19920 GOSUB 19930: PRINT
      CHR$(30);: RETURN
19930 IF X<0 OR X>79 OR Y<0
      OR Y>23 THEN STOP
19935 PRINT CHR$(20);CHR$(Y);
      CHR$(X);CHR$(0);:RETURN
19940 PRINT CHR$(12);CHR$(0);
      :GOTO 19930
19950 GOSUB 19930: PRINT
      CHR$(31);CHR$(0);:
      RETURN
```

The routine as shown is for a Micro-Term ACT-5A (my favorite terminal). If you have a different terminal, you'll need to re-write the routine to make it work for whatever you have. If you deal with several different terminal types, invest a little time now and write the routine for each terminal type. They will come in very handy later.

One additional note: because different terminals use different terminology to refer to points on the screen, it is important to determine a standard coordinate system and then have the subroutine do the conversion to fit the particular terminal. For instance, the manuals for several terminals refer to the first column as 1, others refer to it as 0. The subroutine must use one or the other, and make allowances for terminals that use the other standard. In the subroutine given, both X(horizontal) and Y(vertical) coordinates are assumed to start at zero. Thus, X=0,Y=0 is the upper left corner and X=79,Y=23 is the lower right corner.

Next month, I'll show you a program which uses the above subroutine to make it possible to design and implement a screen "form" in 5 minutes.

OS65D3 #2 IN A SERIES

D.R. "STRETCH" Manley
5664 East Evans Creek Road
Rogue River, OR 97537

"Secrets" and Tricks

There are a few useful things about OS65D3 and O.S.I.'s microsoft BASIC that aren't generally known. Here's some that I've discovered.

How many times have you wished you could add a file buffer to a program? Say you have a program that you wrote originally with one disk file (#6), and now you find that adding a file would be advantageous. Sure, you have heard about POKEing a new #7 file buffer location in your program, but that seems like a lot of trouble. Well, it may or may not be. I use that idea myself, and I'm glad to know the trick. However, there is another way.

Say your program has #6 used, and you want to add #7. First, create a scratch file on disk, at least as large as your program, maybe a little larger. We'll call it "TEMP" in this case. Now load your program with a 'DISK!"LOAD program"'. Now watch carefully. Still in the immediate mode, we type "DISK OPEN,6, "TEMP"', then "LIST #6". If your program is a big one, you may hear the disk drive writing as the buffer fills up. When you see the "OK" from BASIC, type 'PRINT#6, "OK"' and "DISK CLOSE, 6" to write the last partial buffer to disk. Guess what? You've just done a cassette-type save of your program to disk! Your program is on disk in the file "TEMP" in ASCII form, not tokenized as it's normally stored. Also, there are no disk buffers saved with it.

Now to add the extra file buffer. RUN "CHANGE" and set up for 2 file buffers. Then type 'DISK OPEN, 6, "TEMP"', and "POKE 8993, 32". What you will see is your program being read in from disk and echoed to the screen, just as if you had typed it, but a lot faster, I'll wager! When the input routine sees the "OK" that you printed at the end of the file, it will give you a "SN" error, and reset the device number in location 8993(DEC.) to the default device number. Now you can save your program with a 'DISK!"PUT program"', and you're done. Two file buffers in the normal position and not too much pain.

There are two details to be careful of, however. If you have entered any program lines with a "?" in place of "PRINT", to pack more on the lines, you'll have to re-enter those lines by hand. Since we are going into BASIC through it's normal 72 character input buffer, even in this little trick, we are limited as to line length. Also, the program in memory when you do the 'DISK OPEN,6,"TEMP"' has to have at least 1 file buffer defined, #6. If you want to use this procedure with file - less programs, then you must reset buffer #6 to high memory. Listing #1 is a short program I use when I run into that problem. Be sure to reset buffer #6 to the proper location when you are done with it by booting up again.

Neat trick, huh? If you stop and mull this one over, you can see that it's not only neat, it's really useful. For instance, if you want to merge two programs, renumber one to have line numbers above the

other, then use this operation to merge them. First list one (say the renumbered one) to "TEMP", as above, then load the other one with a 'DISK!"LOAD program2"'. Type 'DISK OPEN, 6, "TEMP"' and "POKE 8993, 32". When you get the "SN" error, the merge is done.

I use the ASCII listing of the program in "TEMP" for input to my text editor, to edit the program. Then I have the text editor list it back out in ASCII form, and use that to input back to BASIC to get the tokenized form for "PUT"ting and "RUN"ning.

Note that the "OK" is used to cause the "SN" error on purpose. You can do other things. If you print "RUN", as soon as the program has loaded, it will run, and the keyboard will be dead, unless you have something like this in your program:

```
10 POKE 8993,2
```

This brings up some more ideas. How about one program that will run another, and pass parameters to it, with no intervention from you? How about a whole chain of programs that will load and run, perhaps conditionally, without any intervention? Try this:

The maximum memory size is set in the BASIC interpreter when it's loaded by the operating system. It's checked on every 'RUN', but not the operating systems value, only BASIC's own copy. This is stored in locations 133(DEC), the high byte, and 132(DEC), the low byte. POKE a new maximum memory size to these locations and then set device #5 to the start of your reserved memory area.

An example is in order: I have 48K of memory, so my normal top is $BFFF, with the $BF portion stored in 133(DEC.) as 191(DEC) and the $FF portion in 132(DEC.) and as 255(DEC.). This line saves me a page (256 bytes, DEC.) for my own use, protected from BASIC:

```
10 A=PEEK(133): A=A-1:
   POKE133,A: RUN20
20 REM  REST OF PROGRAM
```

THE "RUN 20" assures that the change takes effect. Now at the end of the program, have it do this:

```
10000 DISK!"MEM BF00,BF00":
      REM BF00 IS FOR 48K
      SYSTEMS
10002 REM USE 7F00 FOR 32K
      SYSTEMS
10004 REM USE 5F00 FOR 24K
      SYSTEMS
10010 PRINT#5,"INPUT FOR THE
      NEXT PROGRAM"
10020 PRINT#5,"MORE INPUT FOR
      THE NEXT PROGRAM"
10030 DISK!"IO 10": RUN
      "NEXT.PROGRAM"
```

When "next.program" runs, any input that is normally from the default device, via an "INPUT A$", or such, will come from device #5, the stuff we printed at the top of memory. There is little serendipity about this whole thing. Any fatal errors will cause the BASIC interpreter to reset the input device to the default device and return to the immediate mode. Don't worry about inputs from specific devices, such as "INPUT #6, T$". They will work normally. Also, since we didn't change the output device with our "IO" command, all output is normal.

I use this idea with a mailing list edit program. When I'm done editing, the edit program always runs a general sort program to insure the mailing list file is kept in alphabetical order, and all deleted records are removed. "SORT" wants all kinds of input, such as name of the file to sort, which drive it's on, what field to sort on, where to put the sorted file, etc. This is all written to high memory by the calling program, and read by the sort program just as if

it were entered from the keyboard. The only line added to "SORT" is:

```
20000 A = PEEK(8960): POKE
      133,A: DISK!"IO 02,02":
      RUN 20010
20010 REM REGULAR END OF
      PROGRAM
```

Location 8960(DEC.) has the number of pages of memory found by the boot routine. This is the value normally in 133(DEC.). The "IO" clears the input device to the default device, and the "RUN" makes sure BASIC gets its high memory reset. If "SORT" had been run from the keyboard, this line would have no effect, even though it is executed, since the default devices were already 02, and location 133(DEC.) was already the same as location 8960(DEC.).

This "MEMORY KEYBOARD" can do any legal immediate mode command or function, such as:

'IF PEEK(location) = literal.number THEN RUN "PROGC"' OR IF variable.name = 56 THEN RUN "BEXEC*

If you use variable.name instead of literal.number, you had better be sure variable name exists, and in the proper form. You can use a file for the keyboard, if all the programs have a file buffer defined. Just print the input for the "next.program" to #6, and then have "next.program" do this:

```
10 IF (PEEK(8993) AND 32)=32
   THEN DISK OPEN, 6,"CMDFIL"
```

Then the input will come from the file, instead of the keyboard. You can even do a "GET" on the file on output or input, to use only a portion of it, or a different portion if certain conditions are met. You can see that this idea, and a real time clock, will allow you to run the computer when you aren't even there! I can dream up a lot of uses for this "FILE KEYBOARD" idea.

You can also use the "IO" command to substitute or add other devices for the keyboard or terminal. Don't believe advertisements that say the assembler/editor and/or the extended monitor can't output to the printer. I do it all the time. I have a polled keyboard and T.V. terminal (device #2). and a Heath H14 for a printer, hooked up to a 550 board as device #8. To get an assembly listing, I type "!IO,82" before I type "A" or "P", and as the listing prints on the screen, it also prints on the printer. If you have #2, and #4 for a printer, do a "!IO,0A". If you have #1 and #4, do "!IO,09". For #1 and #8, I type "!IO,81". When the listing is done, I type "!IO,02" and the output is back to only the screen.

I hope I've stirred your imagination with these ideas of mine. They have given me a lot of amusement, and a system that's a lot more useful.

By the way, I put my name at the beginning of this column so that anyone who wants to comment will know where to send letters. If anyone out there is modifying OS65D3, which is what I am involved in with my machine, please drop me a line. We may be able to save each other a lot of work. I hate re-inventing the wheel!

Listing #1

Program "HIBUFF"

```
10 REM THIS PROGRAM WILL RESET
   BUFFER #6 TO THE TOP OF
20 REM USER RAM, AND RESET THE
   MAXIMUM MEMORY TO A POINT
30 REM BELOW THE FILE BUFFER.
40 BM=PEEK(133)*256+PEEK(132)
45 REM VM = BYTES OF MEMORY
   BASIC KNOWS ABOUT
50 RAM =(PEEK(8960)+1)*256:
   RA M=RAM-1
55 REM RAM = BYTES OF MEMORY
   IN MACHINE
60 OFF=RAM-BM
65 REM OFF = NUMBER OF BYTES
   SAVED ALREADY (OFFSET)
70 BM=RAM - (OFF + 3072)
75 REM 3072 = BYTES IN A
   NORMAL FILE BUFFER (12
   PAGES)
80 BHM=INT(BM/256):BLM=BM-
   (BHM*256)
85 REM BHM & BLM ARE BM IN
   TWO BYTE FORM
90 POKE 133, BHM: POKE 132,
   BLM: RUN 100
95 REM THE RUN MAKES THE
   MEMORY CHANGE EFFECTIVE
100 BM=PEEK(133)*256 +
    PEEK(132)
105 REM WE MUST RECONSTRUCT
    VARIABLES, BECAUSE THE
106 REM "RUN" TRASHED THEM ALL
110 BUFF6+BM+1
115 REM BUFF6 STARTS AT NEXT
    BYTE AFTER BASIC'S MEMORY
120 BH6=INT(BUFF6/256):
    BL6=BUFF6 - (BH6*256)
130 POKE 8999,BH6: POKE
    8998,BL6
135 REM SET START OF BUFFER
    #6
140 BUFF6 = BUFF6 + 3072
145 REM BUMP BUFF6 TO LAST
    BYTE +1
150 BH6=INT(BUFF6/256):
    BL6=BUFF6 - (BH6*256)
160 POKE9001, BH6: POKE
    9000, BL6
165 REM SET END OF BUFFER #6
170 PRINT"FILE #6 BUFFER IS
    NOW SET TO THE TOP OF
    USER RAM"
180 NEW
```

The Beginning Assembler

Part IV

By Al Peabody

In previous columns in this series, I have described my adventures with assembly- language programming. Patient readers have struggled with me as I became noddingly familiar with the 6502 instruction set, learned to hand-assemble programs (an agonizingly slow but necessary process), fought with OSI's assembler and particularly with its documentation, and finally learned to move machine-language routines from 65D into 65U so that they could be called as USR routines from Basic.

Last month, I even presented a program which did something useful, a "dumb terminal" program. I promised that I would continue to develop this program and incorporate parts of the OSI communications protocol as I went along. However, life has gotten ahead of me. A few days ago, I received a review copy of Jim Sanders' new OS-65U terminal program, and realized I was trying to reinvent the wheel. This terrific program does everything I was trying to get my program to do, does it now, does it smoothly and well, and comes with full source code so that the user can modify it if he wants to, all for $27.50 including a free floppy disk! I see no reason to go on working on my program. Standardization is one of the most sorely needed features of communications programs; I suggest all 65U users standardize on Jim's brilliant and inexpensive program. In the next Issue of PEEK(65) we will present an analysis of this program, designed to assist developers of communications programs for other systems to make their programs compatible.

This month, I would like to make a few comments about a word processor designed for assembly-language programming, then share with you the dynam-

ics of my progress as a programmer.

The word processor in question is OSI's WP-1B. I first heard of WP-1B in OSI literature, and ordered it from my dealer when I bought my computer. However, WP-1B is no longer supported by OSI.

So how is WP-1B? Readers familiar with WP-2 already know. WP-1B is simply a marriage of WP-2 (a fairly early incarnation) with OSI's Assembler. The whole works runs under 65D and provides automatic line numbering (increments of 10 only, but beginning with any number you desire), global search, global search and replace, block move (by line number range and with or without deletion of the source lines), printing of selected line numbers or entire listings to console or to both console and printer, just as WP-2 does, plus assembly, generation of object code, assembly and insertion into the destination RAM location specified in the "origin" statement which all 6502 programs must have and, with the Extended Monitor, the ability to run programs after they are assembled.

This means that the (reasonable) power of the Word Processor can be used to type in source code, with the ability to edit lines, move subroutines around, find all mentions of any string, replace them with another string, etc. Then the source file can be saved to disk, assembled into RAM and run, without changing disks as is necessary if you are using assembler and Word Processor on different disks. Full screen editing with cursor control it ain't, but it is much handier than writing in the assembler, which allows lines to be changed only by retyping them completely! Now if there were only an assembler which would run with 65U...

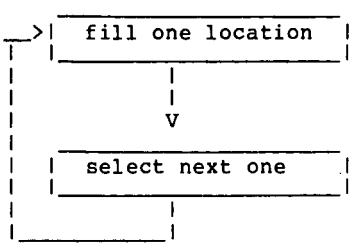
Dynamics. I promised to comment on the dynamics of development of a beginning assembler. Actually, so far it looks something like this:

```
          +
         ++
        +++
   ++++
```

...which is supposed to look like a curve rising very slowly at first, then much faster as time passes.

It means, the first time I sat down and tried to translate the meaningless heiroglyphics of the 6502 instruction set into anything useful, it just didn't gel. All I wanted to do was write a little routine which would fill certain memory locations with a number. After ten minutes of staring at the instruction set, then the blank piece of paper onto which I was supposed to write a flow chart of the operation, my eyes glazed over and I heard faint snoring. Mine.

Then I forced myself to write the stupidest flow-chart known to man:



No, I didn't leave out the test to see if all the blocks were filled yet. That is to say, I didn't leave it out in this recreation of my flow chart. It wasn't in the original. When I said simple, I meant SIMPLE.

After a couple of hours, I managed to get the code to run this little routine written and hand assembled. Then I ran it, from the monitor, and checked to see if it had filled the locations. It had, for 256 bytes. How did the program end? It didn't. It just went on, filling the same page of RAM with $FF over and over again, until I hit the reset button.

It seemed to me that I went along at about this rate for some time. It became somewhat simpler to write subroutines, and, by getting out cheat sheets where I had written stuff down, I could even reproduce some things without a great deal of thought; but the larger, conceptual problem continued to be beyond me. I could not look at a problem and have the foggiest idea how that set of opcodes could be any help in solving it.

The good news is, a few weeks ago a breakthrough occured. I was working on a fairly complex program, a game of LIFE, and realized that, first of all, my flow-charts were more sensible, easier to convert into meaningful code, and secondly, in most cases the very basic process of staring at a problem and getting an idea as to a method to solve it was actually flowing right along!

I can figure out no reason for this transformation; no single or several insights which have come to me; no secret bits or bags of knowledge which have smoothed the still rocky road. I guess if you spend enough hours at this process, you just begin thinking something like a 6502.

...which means that this series, the Beginning Assembler, is now at its end. No, my friends, I am not now an Assembly Language Programmer. But neither am I the sort of absolutely rank beginner I was a few months ago when we started it all. Of course, I will keep you posted as to new developments, achievements, disasters, programming aids etc. And, as always, await your comments and hints!

★

LOGIC CONTEST ENTRY

```
 1 CLOSE:POKE2888,0:GOTO 10
 5 REM HEX TO DECIMAL
   CONVERTER
 8 REM Convert Hex Ascii
   Character in X to Decimal
10 DEF FNH(X)=FNV(((47-X)*
   (X<58)+(X-54)*(X>64)*
   (X<71))-1)
14 REM Returns Large negative
   if X is not valid Hex
16 DEF FNV(Y)=1E9*(Y<0)+Y
20 INPUT"HEX NUMBER";H$
25 IF LEN(H$)=0 THEN END
31 REM Calculate Hex to
   Decimal
40 H=0:FORI=1TOLEN(H$):H=H*
   16+FNH(ASC(MID$(H$,I,1)))
   :NEXTI
46 REM Display result or END
   if bad Hex number
50 IF H<0 THEN PRINT"Hex not
   valid": END
60 PRINT TAB(20); H
70 GOTO 20
130 REM Determine the MINIMUM
    of two values A1 and X
140 REM Define A1 before
    calling the function
160 DEF FNM(X)=-X*(X<A1)-A1
    *(X>A1)-A1*(X=A1)
180 REM Determine the MAXIMUM
    of two values A1 and X
190 REM Define A1 before
    calling the function
210 DEF FNX(X)=-X*(X>A1)-A1*
    (X<A1)-A1*(X=A1)
250 thus endeth my contri-
    bution to the great
    logical function contest
260 by
270 J. Sanders
```

OSI AARDVARK OSI
NOW MEANS BUSINESS!

WORD PROCESSING THE EASY WAY– WITH MAXI-PROS

This is a line-oriented word processor designed for the office that doesn't want to send every new girl out for training in how to type a letter.

It has automatic right and left margin justification and lets you vary the width and margins during printing. It has automatic pagination and automatic page numbering. It will print any text single, double or triple spaced and has text centering commands. It will make any number of multiple copies or chain files together to print an entire disk of data at one time.

MAXI-PROS has both global and line edit capability and the polled keyboard versions contain a corrected keyboard routine that make the OSI keyboard decode as a standard typewriter keyboard.

MAXI-PROS also has sophisticated file capabibilities. It can access a file for names and addresses, stop for inputs, and print form letters. It has file merging capabilities so that it can store and combine paragraphs and pages in any order.

Best of all, it is in BASIC (OS65D 51/4" or 8" disk) so that it can be easily adapted to any printer or printing job and so that it can be sold for a measly price.

MAXI-PROS – $39.95

NEW-NEW–NEW TINY COMPILER

The easy way to speed in your programs. The tiny compiler lets you write and debug your program in Basic and then automatically compiles a Machine Code version that runs from 50-150 times faster. The tiny compiler generates relocatable, native, transportable machine code that can be run on any 6502 system.

It does have some limitations. It is memory hungry – 8K is the minimum sized system that can run the Compiler. It also handles only a limited subset of Basic – about 20 keywords including FOR, NEXT, IF THEN, GOSUB, GOTO, RETURN, END, STOP, USR(X), PEEK, POKE, -,=,*,/, (,), <>, Variable names A-Z, and Integer Numbers from 0-64K.

TINY COMPILER is written in Basic. It can be modified and augmented by the user. It comes with a 20 page manual.

TINY COMPILER – $19.95 on tape or disk

THE AARDVARK JOURNAL

FOR OSI USERS – This is a bi-monthly tutorial journal running only articles about OSI systems. Every issue contains programs customized for OSI, tutorials on how to use and modify the system, and reviews of OSI related products. In the last two years we have run articles like these!

1) A tutorial on Machine Code for BASIC programmers.

2) Complete listings of two word processors for BASIC IN ROM machines.

3) Moving the Directory off track 12.

4) Listings for 20 game programs for the OSI.

5) How to write high speed BASIC – and lots more –

Vol. 1 (1980) 6 back issues - $9.00
Vol. 2 (1981) 2 back issues and subscription for 4 additional issues - $9.00.

ACCOUNTS RECEIVABLE – This program will handle up to 420 open accounts. It will age accounts, print invoices (including payment reminders) and give account totals. It can add automatic interest charges and warnings on late accounts, and can automatically provide and calculate volume discounts.

24K and OS65D required, dual disks recommended. Specify system.
Accounts Receivable. $99.95

*** * * SPECIAL DEAL – NO LESS! * * ***

A complete business package for OSI small systems – (C1, C2, C4 or C8). Includes MAXI-PROS, GENERAL LEDGER, INVENTORY, PAYROLL AND ACCOUNTS RECEIVABLE – ALL THE PROGRAMS THE SMALL BUSINESS MAN NEEDS. $299.95

P.S. We're so confident of the quality of these programs that the documentation contains the programmer's home phone number!

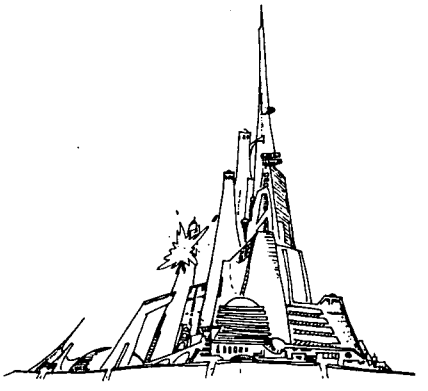
SUPERDISK II

This disk contains a new BEXEC* that boots up with a numbered directory and which allows creation, deletion and renaming of files without calling other programs. It also contains a slight modification to BASIC to allow 14 character file names.

The disk contains a disk manager that contains a disk packer, a hex/dec calculator and several other utilities.

It also has a full screen editor (in machine code on C2P/C4)) that makes corrections a snap. We'll also toss in renumbering and program search programs – and sell the whole thing for – SUPERDISK II $29.95 (5 1/4") $34.95 (8").

AND FUN, TOO!



BOOKKEEPING THE EASY WAY – WITH BUSINESS I

Our business package 1 is a set of programs designed for the small businessman who does not have and does not need a full time accountant on his payroll.

This package is built around a **GENERAL LEDGER** program which records all transactions and which provides monthly, quarterly, annual, and year-to-date PROFIT AND LOSS statements. GENERAL LEDGER also provides for cash account balancing, provides a BALANCE SHEET and has modules for DEPRECIATION and LOAN ACCOUNT computation.
GENERAL LEDGER (and MODULES) $129.95.

PAYROLL is designed to interface with the GENERAL LEDGER. It will handle annual records on 30 employees with as many as 6 deductions per employee.
PAYROLL - $49.95.

INVENTORY is also designed to interface with the general ledger. This one will provide instant information on suppliers, initial cost and current value of your inventory. It also keeps track of the order points and date of last shipment.
INVENTORY - $59.95.

GAMES FOR ALL SYSTEMS

GALAXIAN - 4K - One of the fastest and finest arcade games ever written for the OSI, this one features rows of hard-hitting evasive dogfighting aliens thirsty for your blood. For those who loved (and tired of) Alien Invaders. Specify system – A bargain at $9.95

NEW – NEW – NEW

LABYRINTH - 8K - This has a display background similar to MINOS as the action takes place in a realistic maze seen from ground level. This is, however, a real time monster hunt as you track down and shoot mobile monsters on foot. Checking out and testing this one was the most fun I've had in years! – $13.95.

NIGHT RIDER - You've seen similar games in the arcades. You see a winding twisting road ahead as you try to make time and stay on the road. NIGHT RIDER uses machine code to generate excellent high speed graphics - by the same author as MINOS.
NIGHT RIDER – $12.95 cassette only

THIEF - Another machine code goody for the C1P cassette only. You must use mobile cannon to protect the valuable jewels in the middle of the screen from increasingly nasty and trigger happy thiefs. Fast action and fun for one or two players. THIEF $13.95 on C1 cassette only!

SUPPORT ROMS FOR BASIC IN ROM MACHINES – C1S/C2S. This ROM adds line edit functions, software selectable scroll windows, bell support, choice of OSI or standard keyboard routines, two callable screen clears, and software support for 32-64 characters per line video. Has one character command to switch model 2 C1P from 24 to 48 character line. When installed in C2 or C4 (C2S) requires installation of additional chip. C1P requires only a jumper change. – $39.95
C1E/C2E similar to above but with extended machine code monitor. – $59.95

Please specify system on all orders

This is only a partial listing of what we have to offer. We now offer over 100 programs, data sheets, ROMS, and boards for OSI systems. Our $1.00 catalog lists it all and contains free program listings and programming hints to boot.

VISA

OSI

AARDVARK TECHNICAL SERVICES, LTD.
2352 S. Commerce, Walled Lake, MI 48088
(313) 669-3110

OSI

LETTERS

ED:

Having completed a 520 RAM board, bringing my memory up to 49K, I decided to order CP/M from LIFEBOAT. I was encouraged in this decision by the letters in Peek(65) from readers who had been using CP/M. I was anxious to try the software available through the CP/M users group CPMUG. This is an exciting collection, including several languages, with each disk costing only eight dollars.

On receiving CP/M, I was annoyed to discover I needed 8K more memory. Not only did I need additional memory but it had to be placed at $D000 where my 540B video board lived. After stewing about this for a while trying to decide if I wanted to spend $150. to $200. for an 8K board, I realized my 527 RAM board had an unused 8K block of memory available if it could be readdressed. This turned out to be a practical solution easy to implement. If you think this would be of interest, I could provide a detailed description.

[Of course!...Al]

With the 527 board modified and the 8K additional memory chips installed, I was ready to load CP/M. You can imagine my joy when after rattling my "A" disk drive for some milliseconds the CP/M message and prompt came on my screen. Suddenly a whole new world was opened up for me. All of the Z-80 - 8080 software was now within my reach. I no longer had to skip over the assembler listings in the magazines with their odd yet fascinating mnemonics. I spent the rest of the evening pouring over back issues, trying to get a feel for the universe I was capable of entering.

Next I had to attend to the problem of the 540 video board which was now removed from the system. Switching boards by plugging and unplugging them is not at all convenient. I wanted the process to be automatic. If I were using CP/M I wanted the 8K memory available, if using OS65D, I wanted the 540 board available.

The 510C CPU has a 6820 PERIPHERAL INTERFACE ADAPTER(PIA) which is accessed at $F7XX. This very useful device has sixteen ports that can be either input or output. Four of these are committed to switching the three processors. One is labeled HIS(?), four are connected to the inputs of 8T95 buffers and seven are unused. On reset, all of the ports become inputs. But since it requires 1.6 milli-amp to pull these to ground they will look like a logic one to any device reading them. In addition these ports are latched, that is when used as outputs they remain either a one or a zero until changed.

What this means is, I had a software controllable signal with the following properties:

1. Automatically high when the computer is reset.
2. Physically capable of driving the address bus.
3. Capable of becoming low by the same command that switches from the 6502 to the Z-80.

If I now could modify the two boards to respond appropriately, I had an automatic switching mechanism that required no thought or effort to use. If I reset and used OS65D I had a video board available. If I loaded CP/M the required 8K memory took the place of the video.

The 527 memory was simple to modify since I had made use of the bank memory switching circuitry when I readdressed the upper 8K memory. Bringing a signal from B19, the bus pin I used for the address bit A19, through an inverter to the decoding "And" gate was all that was neccesary. With this change a negative signal on B19 was required before the upper 8K could be read or written.

The 540B video board was more complicated since it had no bank switching circuitry available. Briefly, it involved cutting an input foil to the "And" gate decoding the color memory at $E00 & connecting this input to B19. Then making use of the two input "And" gates and inverter used for the AC control to combine A19 and A15. The output of the inverter is then routed to the former "And" gate input of A15. If you are interested, I can provide a more detailed description.

[Again, of course!...AL]

Please note that the 540B board is not a modified 540 board. The two boards are quite different and should not have been given the same numerical designation.

The only remaining task was to change the CP/M initializing routine. This was quite simple to do. On "Reset" track zero of CP/M2.23 is loaded to $2200 as is normal with all of OSI floppy disk software. Track zero, in this case, contains a short section of 6502 code. This code first puts a jump vector at $0000 for the Z-80 and then changes the PBX ports of the PIA at $F7XX to outputs. It then loads this output register with the code for switching from 6502 to Z-80. The code then has a small loop permitting the 6502 to wait for the Z-80 to take over. All that is necessary is to change one bit in the word that does the switching, to change PA0 from one to zero. This can be accomplished using the Track Zero Read/Write routine.

I realize that the above paragraphs are both sketchy and cabalistic. What I am trying to establish is that the project is both realistic and practical. Although it took a day or so to figure out exactly what to do and how to do it, only a morning was consumed with the actual conversion including testing.

At this point, I sat back quite pleased with myself. I had the best of all possible worlds. I could go out and conquer the S-100 world without giving up one bit of my 6502 kingdom. With this glow of self-satisfaction I read Thomas L. Robb's letter in the July Peek (65).

Let me quote Mr. Robb's letter so you may appreciate the magnitude of the shock I received.

"The OSI disk controller requires and creates a unique disk format. CP/M software disks billed a "standard 8 inch, soft-sectored" won't work on OSI hardware. Most other major hardware vendors offering CP/M use the standard format - but not OSI. ....."

Well what to do now? My S-100 universe had shrunk to one vendor. LIFEBOAT has certainly done a good job of offering CP/M for OSI and their selection of software is very good. Nevertheless, the situation is not what I had in mind. I sat down and studied the schematic for the 470 floppy disk interface for some time. OSI had been quite frank in their manual. They had used a 6850 ACIA instead of a 6852 SSDA. The difference between them is small but quite crucial. The ACIA uses at last one stop bit in its word format, the SSDA uses none. Perhaps though, the situation is not as depressing as I thought.

Almost every pin on a SSDA can be paralleled with the ACIA. Indeed, the OSI instructions for the 470 board provide for replacing the ACIA with the SSDA. There is plenty of prototyping area on the 470 board for adding as SSDA and any other logic required.

Of course the software interface would have to be modified. The SSDA has a larger number of registers to initialize. My most urgent need is for information. What is the "standard disk" format? I need to know the two byte sychronizing word required by the SSDA.

Despite the uncertainties involved, I believe adding a 6852 SSDA to the 470 board to permit reading the "standard" format, is a practical solution to the problem. The physical part, adding the components to the board and wiring them, is not difficult. Of course, the addressing of the new part must be arranged. But that would be necessary in any case.

This capability would, I believe, make the OSI C3 the most attractive and useful machine available today. What about APPLE software? I understand it uses a soft-sectored format.

I am grateful to John Fijalkowski for calling attention to this problem and to Thomas Robb for pointing a way to a possible solution.

J.B. Boardman
Stamford, CT

J.B.:

If you can solve this problem, you will have a product which will be tremendously valuable. I applaud the good work you have done so far, and wish you luck with the rest of it.

I suggest you try Phillip Woellhoff of Westico, Inc., 25 Van Zant St., Norwalk, CT 06855 and/or Jeff Beamsley of The Software Federation, Inc., 44 University Dr., Arlington Hgts., IL 60004 for the information you need. And DO keep us posted!

AL

* * * * *

ED:

My jaw hurts from gnawing on the walnut side panels of this C4P-cassette system along side of me.

One of the problems is loading programs into the machine. As suggested by various articles, I have been up and down the volume scale, tried numerous types of recorders (including a Sony reel to reel stereo tape deck), and all kinds of tape; I have only succeeded in shattering my crystal nerves.

Being a novice in both computing and circuit building, I will describe the symptoms as best as possible.

Programs having as few as 10 statements will have at least 3 statements in error.

The AC3 monitor displays what I believe is called "swimming" so badly that I must choke down 3 dramamine tablets and keep a stomach distress bag handy.

Swimming occurs everywhere in the building the system was tried; AC3 on top or 6 feet away from the computer.

The machine is grounded to cold water pipe directly.

I have considered calling a priest in to spritz the system, now in my office, with a dose of "H"-H20; I reconsidered when a picture flashed of water droplets streaming through the keyboard.

If anyone recognizes the problem, or can offer some suggestion, I will be very grateful.

I am thinking about using an Exatron Stringy-Floppy for storage in the near future.

Exatron will offer an OSI compatable S-F sometime in the fall. They have available now a RS 232 S-F; is anyone utilizing one with a C4P?

This brings me to my next question; can I activate the RS 232 port myself, or must I send the machine to the factory for wiring?

I will make the rest of my questions and comments brief because there are quite a few.

Is there any way possible to get an 8 bit parallel I/O out of the C4P?

What boards from Mittendorf, Orion, Aardvark, etc. are adaptable to the C4P?

Has anyone experimented with the DAC? Will you submit your programs to Peek (65) so that we all can learn to use it?

Next year I would like to add a used Selectric to the system; can the inexpensive Kilobaud Klassroom Komputer (a 6802) with one ACIA and one PIA be used as an intelligent interface instead of a KIM-1? I understand the 6820 and 6520 PIAs are virtually the same. Could the ACIAs communicate?

Can someone explain how the active ports of the C4P function electrically; what parameters are being measured? With this information, others may come up with experimental hardware. Joystick ports must have potential use.

I have an idea for a 16 bit output port, but it is slow going without equipment and circuit savvy.

We need a good hardware honcho, as Steve Ciarcia is to TRS-80, for OSI. Let's seek him out.

Two projects I want to undertake right away are to interface an "Electric Mouth" (Netronics), and an AY-3-8910 PSG to the C4P; is there any chance the Apple 11 "Electric Mouth" can be mated to our machines? The 16 bit port mentioned above is a possibility if it works out. Has anyone been successful matching an EM to a C4P?

Can Peek (65) reproduce camera ready 2X scale P.C. layouts to actual size (1X)? If not, what is the best way to submit this kind of artwork? What about schematics?

Any other OSI users in my area?

I have more questions, comments, and ideas (homebuilt plotter), but I'll reserve these for the next letter.

M.J. Petyak
Wilkes-Barre, PA

Dear M.J.:

Wow! Submit all artwork 1.43X (for 70% reduction to 1X). Where is our Steve Ciarcia?

AL

* * * * *

M.J.:

Your cassette problem could be one of several problems. If you have difficulty in loading tapes that you did not record, but can load tapes that you recorded then the problem would appear to be in the read/write clock. However, if you cannot load tapes that you recorded and or other tapes, then the problem might be that the tape pulse duration is out of adjustment. Either of these adjustments require an oscilloscope to be done accurately. The procedure and wave forms are contained in Sams Manual on the C4P.

I had a monitor that displayed your swimming symptoms. The problem turned out to be loss of synch. Again there is an internal adjustment to help correct this. I suggest that you either return the unit to your dealer for service or obtain the Sams Manual (available from DBMS, Inc.) and oscilloscope before proceding further.

About activating the RS-232 port. Not knowing which version of C4P you have, there may not be an extra RS232 port. If you have the 502 CPU then the RS 232 port is used for the cassette interface. The 505 CPU board has an extra RS232 port that is ready to go, all you have to do is write software for it.

As for an 8 bit parallel port there is one on the 505 CPU board, but not on the 502 CPU board. Currently, OSI has available several different kinds of parallel I/O. The 470 configured as a centronics

interface is a 16 bit parallel I/O board which uses 6821 PIA. The 555 network board has provisions for two PIAs giving a total of 32 bits parallel I/O. There is also a 48 bit parallel board available. I guess I should inject here that the C4P series of OSI computers uses their 48 pin bus, so any OSI board can be put in the C4P series computers. So how about C4P hard disk. If you have any other questions, please write to us, here at PEEK(65). Good Luck!

Brian

* * * * *

ED:

I was just reading Jim Sander's column in the June issue of Peek (65). In the last paragraph he mentioned a method of clearing a file out. Here is a program to do exactly what he mentioned. I wrote this because I wanted a way to empty out files without going through the DELETE - REPACK - CREATE procedure. At first I simply printed spaces into the file, but being a freak about these things, I wanted NULLS. By reversing the patch in TECH LETTER #4, NULLS go to the file. To change the Op-Sys RUN "CHANGE" - MODE=H - OFFSET=C00 - ADDRESS=282F. Change 282F to 0A and 2830 to F0. At 2831 enter an "X".

Because of the OS-65U Op-Sys change, this program is kept on a separate disk along with BEXEC*, DIR, and FLIST (the new version of FDUMP). When I need to empty out files, I reboot the system with the CLEANER disk in "A". After the selected files have been wiped clean, reboot the system with a normal version of OS-65U . I have an OS-65U CD-36 F.D. operating system on this disk so I can clear out files on the hard disk also.

I put in a lot of error capturing routines because I am not the world's greatest typist.

Good luck!

```
10 REM FILE CLEANOUT PROGRAM
20 REM REQUIRES 65U OP-SYS
   CHANGE AT HEX 282F (0A) &
   2830 (F0)
30 REM
40 REM BY JAMES S. ISABELLA
   VER 1.2  12/11/80
50 REM Computer Applications
   Company, Inc.
60 REM 3004 Center Rd /
   Stewart Hill Plaza
70 REM Poland, OH  44514
   PH:(216) 757-3711
80 REM
90 REM
100 REM   HEADER INFO
110 REM
120 CL$=CHR$(27)+CHR$(28)
    :POKE2888,0
130 A$=CHR$(0):FORX=1TO7
    :A$=A$+A$:NEXTX:REM
    SETS A$ TO 128 NULLS
135 STOP
140 PRINTCL$:PRINT"FILE
    CLEANOUT PROGRAM"
    :PRINT:PRINT
150 REM
160 REM   INPUT DATA AND CHECK
    TO SEE IF CORRECT
170 REM
180 INPUT"ENTER WHICH DEVICE
    FILE IS LOCATED ON
    (A-E)";DV$
190 IF LEN(DV$)<>1 THEN
    PRINT:GOTO 220
200 IF ASC(DV$)<65 OR
    ASC(DV$)>69 THEN GOTO 220
210 GOTO 230
220 PRINT"IMPROPER DEVICE....
    ........":PRINT:GOTO 180
230 CY=0:IFDV$="E" THEN INPUT
    "ENTER CYLINDER ADDRESS OF
    SYSTEM";CY$
240 IF CY$="" THEN CY$="0"
250 CY=VAL(CY$):IFCY<0ORCY>
    2560RCY<>INT(CY)THENPRINT
    :PRINT"WHAT!":PRINT
    :GOTO230
260 INPUT"ENTER NAME OF
    FILE";NM$
270 INPUT"ENTER PASSWORD";PW$
    :IFPW$="."THENPW$="ANAN"
280 INPUT"ENTER TOTAL LENGTH
    OF FILE";L$:IFL$=""THENL$
    ="0"
290 L+VAL(L$):IFL<10RL<>INT(L)
    THENPRINT:PRINT"WHAT!"
    :PRINT:GOTO 280
300 INPUT"ENTER STARTING INDEX
     -  <CR>=0";SI$:IF SI$=
    ""THENSI$="0"
310 SI=VAL(SI$):IFSI<0ORSI>
    LORSI<>INT(SI)THENPRINT
    :PRINT "WHAT!":PRINT:GOTO
    300
320 INPUT"ENTER ENDING INDEX
    - <CR>=EOF";EI$:IFEI$=
    ""THENEI$=L$
330 EI=VAL(EI$):IFEI<SIOREI>
    LOREI<>INT(EI)THENPRINT
    :PRINT"WHAT!":PRINT
    :GOTO320
340 PRINT CL$
350 PRINT"ON DEV ";DV$:PRINT
    :IFDV$="E"THENPRINT"
    CYLINDER ADDRESS";CY:PRINT
360 PRINT:FILENAME ";NM$:PRINT
    :PRINT"PASSWORD";PW$:PRINT
370 PRINT"TOTAL LENGTH OF FILE
    ";L:PRINT
380 PRINT"BEGINNING INDEX OF
    FILE CLEANOUT ";SI:PRINT
390 PRINT"ENDING INDEX OF FILE
    CLEANOUT ";EI:PRINT:
400 PRINT"TOTAL NUMBER OF
    BYTES TO BE NULLED
    ";(EI-SI):PRINT
410 INPUT"IS EVERYTHING
    CORRECT";QA$:IFQA$=""THEN
    QA$="Y"
```

```
420 IF LEFT$(QA,1)<>"Y"THEN
    GOTO 140
430 REM
440 REM      CLEAN OUT FILE IN
    128 BYTE SECTIONS
450 REM
460 IF DV$="E" THEN
    POKE13314,CY
470 DEV DV$:OPEN NM$,PW$,1
    :INDEX<1>=SI
480 IF (EI-SI)<128 THEN GOTO
    560
490 C=INT((EI-SI)/128)
500 FOR X=1 TO C:PRINT%1,A$;
   :PRINT"SECTION ";X;"
    NULLED":NEXT X
530 REM
540 REM CLEAN OUT REMAINING
    FRACTION OF 128 BYTE
    SECTION & END PROGRAM
550 REM
560 D=INT((EI-SI)-(C*128))
570 IF D=0 THEN GOTO 640
580 A$=RIGHT$(A$,D)
590 PRINT%1,A$
640 IF CY>0 THEN POKE13314,0
650 PRINT: PRINT: PRINT
    "ALL DONE.....": CLOSE
    :DEV"A":END
```

Jim Isabella
Poland, OH

* * * * *

ED:

I see from "Column One" that you've discovered logical evaluation in BASIC. As you suspected, this can be a very powerful (and fast) technique. I use it a lot in writing video games which not only have to run as fast as possible, but also must fit in 8K (as a general rule). By the way, the expression (D>9) if true will evaluate to -1, not 1. This usually means having to take the ABS of the final result, or adjusting it in some way.

As an example of one way I've used logical evaluation, here's an excerpt from my highly modified Aardvark Minipros word processor (which by the way is what I'm writing this letter on). This routine allows you to determine where in a list of nonconsecutive items a particular entry occurs.

When writing a menu-driven program (like a word processor), it's nice (user-friendly) to be able to enter the command (or just the first letter of the command), instead of being forced to enter 1 for the first command, 2 for the second, and so on. What this usually results in is a program something like this:

```
100 INPUT"YOUR COMMAND";A$:A$=
    LEFT$(A$,1):REM TO GET THE
    FIRST LETTER
110 IF A$="D" THEN GOTO AAA
115 REM (DELETE)
120 IF A$="E" THEN GOTO BBB
125 REM (EDIT)
130 IF A$="G" THEN GOTO CCC
135 REM (GET)
140 IF A$="I" THEN GOTO DDD
145 REM (INSERT)
150 IF A$="L" THEN GOTO EEE
155 REM (LIST)
160 REM... AND SO ON FOR 10
    DIFFERENT COMMANDS!
```

In my version of the minipros, I had to conserve as much memory as possible as I only have 8K in my C1P. To do any practical word processing in this little memory, the processor program had to take up as little space as possible. By using logical evaluation to tell the program which command you've chosen, the above routine looks like this:

```
100 INPUT"YOUR COMMAND";A$:A$=
    ASC(A$)
110 I=(A>68)+(A>69)+(A>71)+
    (A>73)+(A>76)+(A>77)+
    (A>78)+ (A>80)+(A>82)
120 I=I+(A>83):I=I-I:ONI GOTO
    160,315,195,175,50,40,220,
    70,140,115,45
```

This looks a lot more complicated than it really is. What it does is to add up all the expressions which are true. If the command is L (ist), the ASCII value of L is 76. Then A is greater than 68, so I=-1. A is greater than 69 too,so I=I+ -1, so I is now -2. The same happens until it reaches (A>76). At this point I=-4. A is not greater than 76 since A=76, so that expression evaluates to zero, thus I=I+0. All the rest of the way down the line, A is not greater than the value in parens, so 0 is added each time. When the end of the evaluation is reached, I=-4. By subtracting I from 1, we then have the number of the command which was entered (LIST is the fifth command), then the on GOTO takes us to the proper routine for action.

I hope that's clear enough to follow. I've found this an effective method of determining the order of non-consecutive items, whether commands in a menu or target character codes in a video game. (P.S. I didn't mean to imply that the Aardvark Minipros is unfriendly! I just had to do it this way to save memory.)

Bob Retelle
Ypsilanti, MI

Bob:

Thanks for the programming lesson. The way I count it; a list of 10 command choices takes 193 bytes with 1 LEFT$(A$,1) statement and 10 IF...THEN statements, only 140 bytes your way. How about:

```
100 INPUT "CMD.";C$
110 GOTO 100*ASC(C$)
```

Then place the routine for a command whose first letter is A at line 6500, etc.

I count 29 bytes total (including 6 bytes "overhead" per BASIC line. Who can reduce it still further?

AL

* * * * *

ED:

(In reference to an article entitled BASIC PLUS, in the July 1981 issue of Peek (65).

Ohmygod... I did it again... (heh, heh).

I should just blame the editor for "typos"; but I cannot tell a lie-- once again, I sent in corrections to a previous item and failed to get it all together.

So, in commemoration of waking up this morning and smelling the coffee, I hereby attach the fixes.

Line #

```
  6 For X equals 560 to 667
 10 Read A : Poke X, A: Next
 14 REM 560-589 is Control
    Output S/Q
 15 Data 72, 169, 246, 141, 0,
    223, 169, 192, 44, 0
```

```
20 Data 223,208,12, 169, 252,
   141, 0, 223, 169, 192
30 Data 44, 0, 223, 208, 244,
   104, 76, 105, 255, 0 :
   REM 589 REM S/Q listings
70 Data 32, 186, 255, 201,
   12, 208, 3,32,139,255,201
   :REM 12 is L/Load
80 Data 26, 208, 3, 32, 150,
   255, 201, 127, 208, 3,
   76: REM 26 is Z/Save
90 Data 134,2,201,2, 208, 3,
   76, 181,164: REM 620 REM
   127 is Rub Out use "2"
   for"Cntl B"
95 Data 201, 27, 208, 3, 76,
   216, 0: REM 27 is ESC Port
   Audit
96 Data 201, 4, 208,3, 76,
   158, 2 REM 4 is D/Down
   scroll.
100 Data 201, 18, 208, 6,32,
    119, 164,32, 194, 165,
    96:REM 18 is R/Run
110 REM Lines 120 - 130 for
    Rubout Key Screen Clear
    routine
120 Data 72, 169, 32, 162, 0,
    157, 0, 208,
    157, 0, 209, 157, 0, 210
130 Data 157, 0, 211, 232,
    208, 241, 104, 96
140 REM Lines 150 - 160 Set
    Input and Output vectors.
150 Poke 11, 78: Poke 12,2 :
    Poke 536, 78: Poke 537, 2
160 Poke 538, 48 : Poke 539, 2
165 REM Pokes 216-235 is the
    Port Auditing Routine
170 For X equals 216 to 235:
    Read Y: Poke X, Y : Next
180 Data 169, 255, 141,
    3,2,44, 3,2,16, 9, 32,
    186,255,32,45,191,24
190 Data 144,242,96
200 For X equals 670 to 720:
    Read A: Poke X, A: Next
210 Data 169, 223, 133, 80,
    169, 255, 133, 82, 169,
    211, 133, 81, 133, 83,
    162,  REM 684
220 Data 4, 160, 0, 177, 80,
    145, 82, 136, 208, 249,
    198, 81, 198, 83, 202 REM
    First data "4" is number
    of screen pages.
230 Data 16, 242, 162, 32,
    169, 32, 157, 0, 208,
    202, 208, 250, 162, 32,
    157,  REM First instance
    of Data "32" is screen
    width.  Poke 703, xx
    to widen or narrow screen
    as desired.  Second in-
    stance of Data "32" is
    (poke with blanks).  Poke
    705,xx to fill screen
    with other characters de-
    sired.
240 Data 96, 211, 202, 208,
    250, 96
250 New REM BASIC program
    erases itself, leaves
    poked up machine code in
    pages zero and two.
```

I have subscribed to Peek (65) since issue #1, although in those early days it came to my neighbor who sold me my C1P about a year ago. When I tried to run programs I found, if they would not work, I and my neighbor Kritchevsky, would jointly curse you, Al, and wonder,"...why doesn't the editor get it right???"; now in all humility and humbleness, with eyes bulging and red from typing data statements, I say, "I am sorry".

Patrick Townson
Chicago, IL

* * * * *

ED:

With respect to your question: How does OS65D read that RUN"BEXEC* in the line buffer? On booting OS65D, after ports, etc. are set up, control is transferred via a jump at $22C4 to $2AE6. This latter subroutine is the BASIC Command in OS65D. (By changing the jump at $22C4, to $2A51, OS65D alone will boot (it overlays the last page of BASIC from $2200-$22FF, crucial for IO).) When BASIC boots, it cold starts at $20E4. This cold start sets up BASIC, calls a 1 page disk sector (named BASIC OVERLAY by OSI) to $20E4 covering up the cold start, then commences to a warm start of BASIC.

Upon warm start, BASIC is ready for input. Here is where OSI got very clever. On boot up, the Output Flag at $2321 is set to 0; no output. The Input flag is set to $10; memory input. The memory input pointer is set to $2E25, which just happens to be the 'R' in RUN"BEXEC* appearing at the end of the OS65D line buffer. A carriage return is needed at the end of this buffer. Thus these letters are jammed up against the end of the buffer to take advantage of the return.

The first line BASIC reads is from memory and is RUN"BEXEC*. Further, the first BASIC program must reset the IO flags, accounting for the obligatory first line of BEXEC*.

OS65D's error routine resets IO flags to default values. My method of booting OS65D causes this routine to be called via a Syntax Error. Both OS65D and BASIC call this routine upon receiving an error. The Assembler and WP-2 generally do not. The character at the end of the Indirect File (used by Jack) probably causes a BASIC Syntax Error resetting the flags, allowing the file to be used. At the end you should see two left brackets (not just one).

If you do not, the file is closed with the wrong character.

You may still use the Indirect File by being too clever. To terminate a file, you want to EXIT WP-2 and create an OS65D Syntax Error. That may be done as follows. List a BASIC program into the Indirect File but do not close the file at the end. Type EXIT then return. Now you are in OS65D and still feeding the Indirect File. Next type 1 and return. This will cause a Syntax Error closing the Indirect File. Now bring the Indirect File into WP-2. At the end it will encounter EXIT and the Syntax Error again terminating the Indirect File. You will be in OS65D and will have to re-enter WP-2. Clean up the end of your WP-2 file. Now you have your file in WP-2.

OS65D is almost an operating system. To close this small gap, one must really understand it to make best use of it. Most of those who have written me for help have made useful suggestions. I have always encouraged them to write their suggestions to Peek (65). Unfortunately, few do this. Letters from readers are still, far and away, the best part of Peek (65).

Be careful how far you wander into CP/M. Most of your readers do not own C3's and cannot hope to run CP/M. It is not a panacea. I find the wait painfully slow. In WordStar, I have often typed a line only to find the disk busy taking only one letter from that line. I suggest you look at Memorite if you want to see an 'even better' word processor.

Tom Berger
Coon Rapids, MN

* * * * *

EDITOR'S NOTE: These comments relate (I think) to WP6502 V1.2 and 1.3 for OS65D.

ED:

Jeff Krause asked in a notice on the Peek (65) CBBS, for advice about word processing software for OSI and expressed concern about not being able to exploit his printer if he chose WP6502, which is in machine language. My advice would be to buy WP6502 (cheap version) unless he does an awful lot of writing. There may be better word processors but they cost much more!! As to exploiting his printer, there are two ways to do it.

WP6502 allows you to print unprintable characters (I am talking ASCII, now). You won't see them on your CRT or video screen but if they mean something to your printer, it will react accordingly. WP6502 uses "embedded commands" to format text at print time; one of these allows you to print ASCII code, including the control characters. I won't explain it here but it is simple. I am actually thinking about trying graphics with WP6502.

The other way to exploit your printer and use WP6502 is to write a program in BASIC. WP6502 stores text (on disk, anyway) as a stream of ASCII code, embedded commands and all. The first track of the file has a five byte header (I think this is OSI standard). Subsequent tracks do not. If you put a track at a time into memory, you can read it out to wherever you wish with PEEKS to get the ASCII code, and PRINT CHR$(N) to print, translating the embedded commands as you go. I use the disk buffer area for memory because my terminal control program comes up with the disk buffer set up. In fact, I have written this notice on WP6502 and am transmitting it with my modification of an Aardvark Terminal Control Program.

A final comment. WP6502 does not take up any workspace. I think it uses RAM that is normally occupied by BASIC. If memory is a problem, and isn't it usually, WP6502 would have a definite advantage over any word processor written in BASIC, no matter how good the latter was.

Mike Mahoney
Peek (65) CBBS user #2004

* * * * *

ED:

Do you know of a source of documentation for OSI's Extended Monitor? I have the C2/4P cassette version of it, but the dealer lost the documentation.

Also, I have been without a computer for nearly two months. My C2/4P died of an overloaded power supply. Although the supply was sufficient to power a 500 card and a 540 video board, the addition of 527-24K memory board eventually drove it to the grave.

My computer is now dismembered and scattered around the basement waiting for parts. I'm playing with the idea of building an external power supply. Then I may use the area in the case normally occupied by the supply for a floppy disk controller (homebrew), an RS-232 interface, and other projects. If it works, I'll give you an article on it.

Jeff Jensen
Omaha, NE

Jeff:

See Kerry Lourash's article in September PEEK(65) for ExMon. information.

Of course, we would love to get an article from you about your proposed modifications.

AL

* * * * *

ED:

Your reply to Mr. Yasuo Morishita on page 17 of the July issue of Peek (65) puzzles me. On page 10 of the April issue he asked about the Aardvark Cegmon (C2E) monitor failure to correct the string "bug" as clearly stated in the Aardvark ad in the October, November and December issues. You advised him to contact Aardvark directly and let you know what happened. He appar-

Z-FORTH IN ROM by Tom Zimmer
5 to 10 times faster than Basic. Once you use it, you'll never go back to BASIC! $ 75.00
source listing add $ 20.00

OSI FIG-FORTH True fig FORTH model for 0S65D with fig editor named files, string package & much more $ 45.00

TINY PASCAL Operates in fig-FORTH, an exceptional value when purchased with forth.
TINY PASCAL & documentation $ 45.00
FORTH & TINY PASCAL $ 65.00

SPACE INVADERS 100% machine code for all systems with 64 chr. video. Full color & sound on C2, 4P & 8P systems. The fastest arcade program available. $ 14.95

PROGRAMMABLE CHARACTER GENERATOR $ 99.95
Use OSI's graphics or make a complete set of your own! Easy to use, comes assembled & tested.
2 Mhz. boards $109.95

PROGRAMMABLE SOUND BOARD $ 74.95
Complete sound system featuring the AY-3-8910 sound chip.
Bare boards available. $ 29.95

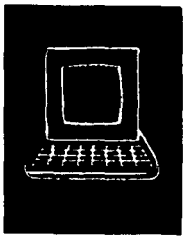
32/64 CHARACTER VIDEO MODIFICATION $ 39.95
Oldest and most popular video mod. True 32 chr. C1P, or 32/64 chr. C4P video display. Also adds many other options.

ROMS!!!
Augment Video Mod with our Roms. Full screen editing, print at,selectable scroll, disk support and many more features.
Basic 4 & Monitor $ 49.95
Basic 3 $ 18.95
All 3 for $ 65.00

65D DISASSEMBLY MANUAL. by Software Consultants
First class throughout. A must for any 65D user. $ 24.95

NUMEROUS BASIC PROGRAMS, UTILITY PROGRAMS AND GAMES ALONG WITH HARDWARE PROJECTS. ALL PRICES ARE U S FUNDS.
Send for our $1.50 catalogue with free program (hardcopy) Memory Map and Auto Load Routine.

CHARGEX VISA master charge



OSI Software & Hardware

3336 Avondale Court
Windsor, Ontario, Canada N9E 1X6
(519) 969-2500

3281 Countryside Circle
Pontiac Township, Michigan 48057
(313) 373-0468

progressive computing

ently did contact them by phone and was offered a BASIC ROM 3 fix at a bargain price, but the original question about the Aardvark ad went unanswered. He then wrote you to report what happened, as you had requested and you compounded the confusion by misreading the October ad.

Al, I have selfish interest in the disposition of this problem since I, too, am disappointed that my Cegmon monitor did not contain a string handling "fix". But more important is the impression of advertising practices left with Mr. Morishita as a result of this confusion. I sincerely hope there is an explanation to the problem, and that Mr. Morishita continues to be a frequent contributor to your pages.

S.C. Dodd
Alamogordo, NM

S.C.:

Yasuo called me on the phone and said perhaps it was his poor English, but he still thought the October Aardvark ad said the Cegmon would fix the string bug. I pulled my October issue to set him straight, and boy, was my face red when I read the ad more carefully! The ad clearly says it will! The ad was later corrected, but I agree that anyone who bought a Cegmon before it was corrected should at least be able to get a refund by returning the part!

AL

* * * * *

ED:

In recent issues of PEEK(65), several articles or notes, mention the replacement of OSI char. gen. ROM with a self-designed EPROM. I wonder if anyone is willing to sell these in one form or another for us non-hardware types. Also, has anyone out there implemented a programmable char. gen. for C4P MF, which would be the best solution for disk users. I imagine vast character sets that could be called at will for a given program needs from disk. I know of the product available from Progressive Computing for the C1P. If anyone out there can help, please do. Thanks.

Don Colin
DeFuniak Springs, FL

* * * * *

ED:

I was just re-reading the August '81 issue of PEEK (65) when I discovered an error in the drawing accompanying my letter to Mike Carroll (p. 17). There should be an inverter in the line coming from the 7402, going to the 8196/G2 not.

The output of the 7402 is HIGH when the address is $DB01. But the /G2 not input must be LOW.

Bruce Showalter
Abilene, TX

* * * * *

ED:

Please forgive the errors in the presentation of the Super II expansion.

Two letters in this same issue, (August 1981), one by Pete Hitt and the other by Bruce Showalter, cover Data Buffers select lines very well.

For the record "Data Direction" goes low when R/W, 02, and one of four "8-K BYTE" (y1, y2, y3, y4) selectors are all high.

This same circuit is found on page 25 of the Sams' schematics for the Challenger Series. The RAMs are 2114-1, 300 nanosecond, of different manufacture.

I've begun a new I/O board which includes the floppy disk controller found in the May & June issues of Byte.

Alex Jackson
Towson, MD

* * * * *

****
NEWS RELEASE
****

BROWN/COLINSON ASSOC. Software House of Lake Oswego, Oregon, has added a Level3 utility to their software tools for OSI programmers. Their latest is called MONITR, a system monitor for Level3 systems.

"MONITR is a valuable tool for programmers and operators", the spokesman said, "as it shows what is going on...while it is running. It also enables the user to debug multi-user partitions and programs."

* * * * * * * * * * * * * * * *

* * *RESEQ 5.2 BUG NOTICE* * *

A bug has been found in RESEQ. RESEQ will cause problems if a program containing a line with a colon as the first character. For a complete fix, send a SASE to PEEK (65).

* * * * *

ADS

C1P Mini Floppy owners add computer power to your machine with "BIG BAG". The biggest bag (5 1/4" disk) of software you can buy for the bucks) "BIG BAG" contains:

1. Single disk copier
2. Terminal emulator - makes a terminal out of your computer
3. High memory disk buffers
4. Data management system (not OSI)
5. Disk head load unload for all disk IO
6. Spool to disk (better than indirect memory)
7. VTOC fixer
8. Menu of files at boot time
9. File transmission program via phone lines with error checking and recovery, and a personal word processor.

BIGBAG is only $25. and WP is $15. with BIGBAG or $20. sold seperately. Detailed documentation is available for $5. deductable when software is purchased. Computer Power, 3223 Suffolk Lane, Fallston, MD 21047, PH. 301-692-6538.

FOR SALE: OSI C3-S1 computer 56K static RAM, parallel printer board, fully checked out and disks aligned $2800 and I pay shipping also 16K static RAM memory boards (520 REV B) never used fully checked out $250 ea. or all 3 for $675. Call after 6 PM 904-596-3641

FOR SALE: Pre-owned OSI blue cases, 1 for C2-4P or Superboard, 1 for 5" floppy, w/most hrd. both for $59.00 - Pair of Atari joysticks, set up for OSI, $16.00 - 3M Anti-glare filter for 12 in. video, $10.00. Al Casper 414-272-0920 or 414-675-6946, 3632 CTH I, Saukville, WI 53080.

DEALER CLOSE OUT - C20EM $2000., C4PDMF48K $1800., HAZ 1420 $750., BEEHIVE 150 $250., DMS each $50., SPINWRITER $2000., AMCAP I $500., OKIDATA 125 lpm $2500., CP/M $450., MX-80 $500., MISC BOARDS save. COMPUTER CONCEPTS, INC., 6565 Gateway Ave., Sarasota, FL 33581-5895, PH. 813-921-6444.

POKE

AROUND! YOU'LL SEE OUR SOFTWARE IS THE BEST

That's right. The kind you get from SOFTWARE CONSULTANTS... it's efficient, cost-effective, and flexible. Next time you dump some old programs, remember to load up with the hardworking software you get from SOFTWARE CONSULTANTS.

We're an independent vendor dedicated to producing top quality software just for you , the OSI users, at the most reasonable prices around. As we've told you before, most of our products are modifications and/or extensions of existing OSI packages. Some of our items completely replace OSI's, and all of our products are easily extensible for your custom applications.

The many responses we've recently received have shown us just how badly many of you needed a quality source for OSI software. So, to keep up with your requests for more "good OSI stuff," we've been working overtime on lots of new products. We'll be telling you about each one of them as soon as they're ready, so be sure to keep us in mind for all of your upcoming projects.

We just don't have enough room in these ads to fully describe our products to you, so please call or write us for the latest copy of the free SOFTWARE CONSULTANTS product catalog. It'll give you all the facts on our current line and full details as each new product is introduced.

Remember, for fine OSI software at the most reasonable prices around, it's SOFTWARE CONSULTANTS. Take a quick look at this list and ask yourself...can you really afford to keep using any one else's OSI software?

1. **OS-65D V3.2 DISASSEMBLY MANUAL**
   A super-complete manual that has it all -- 50 pgs. of dissassembly listings, complete and clear comments on most every line, 10 pgs. of computer generated cross reference listings, and more! Praised by many OSIers who couldn't believe it 'til they bought one. A deal at $25.95.

2. **REF COMMAND UNDER BASIC**
   A complete, cross reference utility that'll find and list any BASIC line.number, variable or numeric constant. It's available under 65D or 65U and comes on 5¼" or 8" floppies. This one will save your sanity, and cut out hours of wasted time. Yours for $31.95.

3. **SPOOLER/DESPOOLER UTILITY**
   A useful utility that feeds backed-up data to your printer for normal output, and leaves your screen free for other work by intercepting data bound for your printer and temporarily storing it on hard disk. Written in super fast machine language. Interfaces with serial and parallel printers.

4. **FIG FORTH UNDER OS-65U**
   So far, the only one that's running under 65U at a price within reach. You get all the pluses, like terminal oriented editor, lots of printer and terminal tools, and more. Under multi-user, runs BASIC simultaneously, too. Unbelievable at $89.95.

5. **VIDEO ROUTINE**
   This convenient program really spices up your video system with little niceties like 24 separate control codes, horizontal and vertical plotting, and many variable screen parameters. Software extensions are available to connect this with the graphics resolution booster. The routine alone is $25.95, with extensions, $29.95.

6. **GRAPHICS RESOLUTION BOOSTER**
   An ingenous piece of hardware that'll increase the graphic resolution per character by 8 times. With this device, your circles will be rounder and your curves smoother. A slick addition to your video-based system. Priced at $49.95, or, with the video routine and extensions, $79.95.

Unlike the majority of other software vendors, we offer our customers copies of source code (on floppies) for any of our products they've purchased. For a nominal fee of $12, covering our material, postage, and handling, we'll send you the source code you choose. Simply write it in as you fill out the order coupon. Of course, dealer inquiries are invited for all of our products.

SOFTWARE CONSULTANTS
HARDWORKING *software*

Enclose your check with coupon and mail to: SOFTWARE CONSULTANTS, 7053 Rose Trail, Memphis, TN 38134 USA. TN residents add 6% sales tax. Phone us at 901/377-3503 for rush orders.

Dear Software Consultants:
Sounds great. Here's my order. I want:

_____DISASSEMBLY MANUAL(S) @ $25.95 ea.
_____CROSS REFERENCE UTILITY(S) @ $31.95 ea.
Disk size 5¼"_____or 8"_____
OS-65D_____or OS-65U_____
_____SPOOLER/DESPOOLER UTILITY(S) @ $69.95.
Std. parallel interface_____
Serial interface w/430 board or equivalent_____
Serial interface w/CA10X board_____
_____FIG FORTH(S) @ $89.95 ea.
_____VIDEO ROUTINE(S) @ $25.95 ea.
With software extensions @ $29.95 ea.
_____SOURCE CODE @ $12.00 ea.
For_____
_____GRAPHIC RESOLUTION BOOSTER @ $49.95 ea.
With Video Routine and extensions @ $79.95 ea.
_____SOFTWARE CONSULTANTS Product Catalog Free

NAME _____

ADDRESS _____

FIRM _____

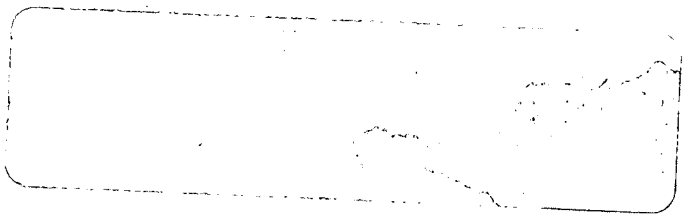
CITY/STATE _____ **POSTAL CODE** _____

PEEK (65)
The Unofficial OSI Users Journal

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

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

DELIVER TO:



PLEASE SEND PEEK (65) FOR ONE YEAR (12 ISSUES).

All rates quoted in and should be submitted in U.S. dollars.

( ) $15.00 Enclosed. U.S. (Maryland residents add 5% sales tax.)
( ) $23.00 Enclosed. Canada and Mexico, 1st class, surface.
( ) $35.00 Enclosed. South and Central America. Air Mail.
( ) $35.00 Enclosed. Europe. Air Mail.
( ) $40.00 Enclosed. All other. Air Mail.

NAME_____ STREET _____

CITY_____ STATE _____

ZIP CODE_____ COUNTRY_____

Please send the following back issues. I enclose:

( ) $2.00 ea. U.S. Surface. (Maryland residents add 5% sales tax.)
( ) $2.50 ea. Canada and Mexico. Surface.
( ) $3.00 ea. South and Central America. Surface.
( ) $3.00 ea. Europe. Surface.
( ) $3.50 ea. All other. Surface.

Vol 1. 1980

( ) Jan #1, "Welcome"
( ) Feb #2, "A month ago"
( ) Mar #3, "Peek continues"
( ) Apr #4, "We are OSI fans"
( ) May #5, "The continued growth" (mistakenly labled #4)
( ) Jun #6, "This column should"
( ) Jul #7, "Several times recently"
( ) Aug #8, "A few minutes ago"
( ) Sep #9, "Of course."
( ) Oct #10, "Publishing PEEK(65) is"
( ) Nov #11, "As you can see"
( ) Dec #12, "This issue marks"

Vol 2. 1981

( ) Jan #1, "First,"
( ) Feb #2, "This month's Peek"
( ) Mar #3, "Last month"
( ) Apr #4, "First things"
( ) May #5, "A careful"
( ) Jun #6, "A letter to"
( ) Jul #7, "The dam is"
( ) Aug #8, "In this month's"
( ) Sep #9, "It was one of"