

PEEK (65)

The Unofficial OSI Users Journal

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

\$1.75

MAY 1983

Vol.4, No.5

INSIDE

| | |
|--------------------------|----|
| BASIC INTERNAL FORMAT | 2 |
| INDIRECT FILES | 8 |
| BASIC EXTENSION PROC. | 9 |
| FINANCIAL PLANNER | 12 |
| ETX/ACK FOR CP/M | 13 |
| DATA SEP. FOR SASI DRIVE | 15 |

Column One

First, two items of business.

1. Fortunately, my terminal screen is a "non-glare" type which doesn't reflect very well, so I don't have to see how red my face is. Repeat after me, Al, 10 times:

Ken Wortz
Ken Wortz
Ken Wortz...

Sorry for misspelling your name last month, Ken!

Second, another

CALL FOR ARTICLES

As usual, our readers are supplying us with great material to print, but also as usual, we want more. Specifically, we want articles on business uses of OSI's computers, especially the new 300 series (but don't hesitate to write something about ANY OSI computer being used for ANY purpose!)

Someone recently asked if the new series 300 computers are based on the OSI 48-pin bus. The answer is, Yes, but...

The 300 series computers run CP/M, on a Z-80 chip, writing and reading standard IBM 3741 format floppy disks. This means any board you have will plug right into the bus, but many of them won't work right. Here's why:

The CA-10 X board, for example, is addressed at CF00. This means in order to send a

character out through the CA-10, you have to "store" the ASCII code for that character in "memory location" CF00. The OSI 470 floppy disk controller is also memory mapped, addressed into a RAM area.

On a 300 series machine running standard CP/M, those areas are true RAM. Store a character at CF00 and all you have done is change memory location CF00. Nothing goes out any I/O. The same is true for the 470 board.

Also, many OSI machines use rather slow dynamic RAM, whereas a 4 MHz Z80 requires faster static RAM. So the RAM boards from your C2OEM won't work (but the RAM from a 2 MHz C3 will work).

So the bottom line is, some RAM boards will work, some won't. Virtually none of the various I/O boards will work (though you could write a routine to drive a CA-10 or 430 board at FB00 without eating into your RAM area too greatly...but then you would have to make sure you didn't also have RAM addressed at FB00...probably wouldn't be worth it.

This issue contains another in the series of articles by Steve Hendrix on OSI's version of Microsoft Basic. We pondered a bit whether to print

this article, because it is highly technical. However, looking over the past few month's PEEK(65)'s, we noted that MANY of our articles of late have been highly technical.

Now, reading the article pasted up and ready to be sent to the printer, I am glad we did it. It is certainly interesting, and must have taken Steve a tremendous amount of work to compile and write. We seem to have become (largely by default) the OSI technical forum for Basic-in-ROM machines. With this I have no problem.

What I do have a problem with is the severe lack of business-oriented material we receive. As noted in the "Call for Articles" above, we do want more business articles, and will pay for them. So if you think Steve's stuff is too technical, too hobby-oriented, don't gripe -- write something. After all, it is you the readers and writers of this journal who determine what we print.

al

**THE INTERNAL FORMAT
USED BY MICROSOFT BASIC**

by: Steven P. Hendrix
Route 8, Box 81E
New Braunfels, TX 78130

In this article I will show how Microsoft BASIC handles its internal storage on the Ohio Scientific Challenger 1P. Though I will address only this specific machine, most other systems using Microsoft BASIC use similar schemes. By knowing something of how program lines and data are stored internally, you can choose among alternate ways of doing a job to optimize memory use and/or speed. If you're really ambitious, you can take advantage of this knowledge by fooling BASIC into taking some shortcuts. You can blend machine language with BASIC to gain the speed of machine language and still have the power of BASIC available. If nothing else, you can satisfy a little of that urge to find out what makes it tick.

The key to BASIC's memory usage lies in a table in page zero. This table contains pointers to different segments of memory, and BASIC adjusts these pointers as necessary to allow each segment to grow as necessary. Each pointer points to the beginning of the associated area, and the area includes a contiguous block of memory up to but not including the location pointed to by the next pointer. The various pointers are shown in Table 1.

Table 1
Memory Pointers

| | |
|---------------|--------------------------|
| \$0079-\$007A | Program |
| \$007B-\$007C | Simple Variables |
| \$007D-\$007E | Arrays |
| \$007F-\$0080 | Free space |
| \$0081-\$0082 | Strings |
| \$0083-\$0084 | Scratch pointer |
| \$0085-\$0086 | End of read-write memory |

These pointers are initialized during a cold start by a routine in read-only memory (ROM). On this system, most 16-bit numbers are stored with the high-order byte in the higher-numbered memory location, consistent with the way the processor (a 6502) deals with addresses. On other systems, this sequence may differ according to the processor's protocol, and the exact location of this table will be different, but the sequence of entries usually remains the same. I will now describe the format used in each section individually.

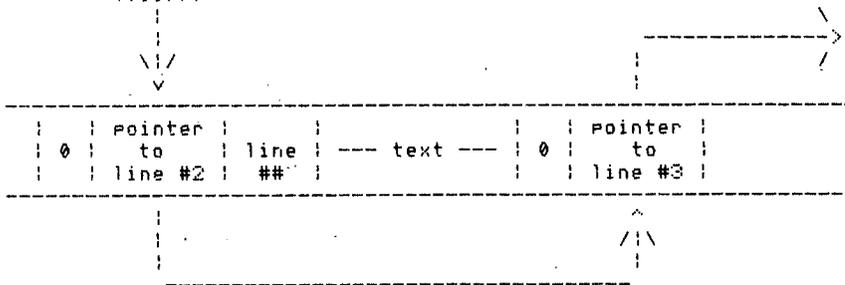
PROGRAM

Program lines are stored in order by line number, as you would expect. When you insert a new line, BASIC searches for the appropriate place in the program for the new line, moves all lines beyond it up in memory far enough to allow room for the new line, and inserts it in its assigned place. If you have a very long program loaded, you can see both the delay caused by the line search and the delay caused by moving many lines around in memory. It takes about a 32K system to allow a

long enough program to see these effects. When you enter a line to be placed in such a long program, the system is "locked" until the line is actually entered (it will not respond to a keyboard entry). To see the delay in finding the line, enter a line near the end of the program and press another key immediately after pressing RETURN. You will see that it takes a noticeable amount of time before BASIC accepts the keystroke. To see the delay of moving higher numbered lines, enter a line near the beginning of the program similarly.

The internal format of a program line is shown byte-by-byte in Figure 1. The first line starts at the byte pointed to by the "Program" pointer. The byte immediately preceding this byte must contain a zero. The first two bytes are a pointer to the beginning of the succeeding line, for use in rapid searches for a specific line. The next two bytes contain the line number in binary. Next comes the compressed text of the line, and finally a zero byte to indicate the end of the line.

Figure 1
Program Line



The text in each line is compressed to a form which saves a little bit of space, but primarily is easier for BASIC to interpret. Each reserved word is analyzed and compressed to a single-byte token before the line is entered in the program. All other text is left stored as ASCII characters. Bit 7 (the

high-order bit) of each byte indicates whether that byte is ASCII or a token for a reserved word. Thus, bytes representing ASCII will have values less than 128 (decimal), and reserved words will have values of 128 or greater. Table 2 lists the reserved words and the token values for each.

Table 2
Reserved Words

| Token Value | Word | Token Value | Word | Token Value | Word | Token Value | Word |
|-------------|-------|-------------|---------|-------------|-------|-------------|-------|
| 80 128 | END | 88 136 | GOTO | 90 144 | ON | 98 152 | CONT |
| 81 129 | FOR | 137 | RUN | 145 | NULL | 153 | LIST |
| 130 | NEXT | 8A 138 | IF | 146 | WAIT | 154 | CLEAR |
| 131 | DATA | 8B 139 | RESTORE | 147 | LOAD | 155 | NEW |
| 132 | INPUT | 140 | GOSUB | 94 148 | SAVE | 156 | TAB(|
| 133 | DIM | 141 | RETURN | 149 | DEF | 157 | TO |
| 134 | READ | 142 | REM | 150 | POKE | 158 | FN |
| 87 135 | LET | 8F 143 | STOP | 97 151 | PRINT | 9F 159 | SPC(|

Copyright © 1983 by PEEK (65) Inc. All Rights Reserved.
published monthly
Editor - Al Peabody
Technical Editor - Brian Hartson
Circulation & Advertising Mgr. - Karin Q. Gieske
Production Dept. - A. Füsselbaugh, Ginny Mays
Subscription Rates
US (surface) \$15
Canada & Mexico (1st class) \$23
So. & Cen. America (Air) \$35
Europe (Air) \$35
Other Foreign (Air) \$40
All subscriptions are for 1 year and are payable in advance in US Dollars.
For back issues, subscriptions, change of address or other information, write to:
PEEK (65)
P.O. Box 347
Owings Mills, MD 21117
Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes endorsements of the product or products by this magazine or the publisher.

| | Token Value | Word | Token Value | Word |
|----|-------------|---------|-------------|---------|
| A0 | 160 | THEN \$ | 178 | POS |
| | 161 | NOT | 179 | SQR |
| | 162 | STEP | B4180 | RND |
| | 163 | + | 181 | LOG |
| A4 | 164 | - | 182 | EXP |
| | 165 | * | 183 | COS |
| | 166 | / | B8184 | SIN |
| | 167 | ^ | 185 | TAN |
| A8 | 168 | AND | 186 | ATN |
| | 169 | OR | B0187 | PEEK |
| | 170 | > | 188 | LEN |
| | 171 | = | 189 | STR\$ |
| AC | 172 | < | 190 | VAL |
| | 173 | SGN | B6191 | ASC |
| | 174 | INT | C0192 | CHR\$ |
| AF | 175 | ABS | 193 | LEFT\$ |
| B0 | 176 | USR | 194 | RIGHT\$ |
| | 177 | FRE | C3195 | MID\$ |

The pointer field is used to indicate the end of the program. The last line of the program points to the two bytes which would be the pointer field in the next line. That pointer field instead consists of two zero bytes, which are included in the program space. Most routines which test for the end of the program simply test the high-order byte for a zero. The variable pointer points to the byte immediately after this end-of-program mark.

VARIABLES

All non-subscripted variables are stored in this area. This includes numerical variables, string variables, and user defined functions (DEF FNxx ...). Each entry consists of 6 bytes; the first two bytes contain the name of the variable and its type, and the other four variables contain the appropriate type of information.

BASIC considers only the first two characters of a variable name. The "\$" indicating a string variable is not counted in these two characters. The ASCII values of the first two characters are stored in the first two bytes of the variable. If a variable name is a single letter, the second is made zero. If it is a string variable, bit 7 of the second character is set to a 1, effectively adding 128 to that value. If the variable is a function identifier, bit 7 of the first byte is similarly set. Since this system prohibits user-defined string functions, bit 7 of both characters may not be set.

The value of a numeric variable is stored in floating-point format in the four-byte data field of the variable. The format in variables differs slightly from the format

used in the "accumulators", where BASIC does its arithmetic. The main accumulator is at \$00AC-\$00B0. The byte at \$00AC is the base-2 exponent, with 128 added to insure a positive value. Thus, negative exponents are represented by values of zero thru 127, with zero being the most negative, and positive exponents are represented by values from 128 thru 255, with 255 being the most positive. The mantissa appears in \$00AD-\$00AF, with the most significant byte in \$00AD (contrary to the standard of high byte in the higher-numbered memory). The binary point is assumed to appear just prior to the first bit of the mantissa. Thus, the number 1141 (decimal) is converted to a mantissa and exponent in base 2, .10001110101 and 1011, respectively. The sign is stored in bit 7 of \$00B0, with a zero meaning positive and a 1 meaning negative. Thus, 1141 would appear in the accumulator as:

```

$AC      $AD      $AE
10001011 10001110 10100000

$AF      $B0
00000000 00000000

```

For another example, let's look at a fraction. Choosing .0703125 will keep the binary representation simple. This would be represented in binary as .00010010. Converting this to the normalized form (with the first 1 appearing just after the binary point) results in a mantissa of .1001 and an exponent of -11 (binary). Adding 128 (decimal) gives an exponent of 01111101. Thus the internal representation would be:

```

$AC      $AD      $AE
01111101 10010000 00000000

$AF      $B0
00000000 00000000

```

Either of these numbers may be negated simply by replacing bit 7 of \$00B0 with a 1 since negative numbers are given as a sign and magnitude.

Since bit 7 of \$AD is always a 1 in the normalized form, we need not actually store that bit in variables. If we replace bit 7 of \$AD with bit 7 of \$B0, we need only store 4 bytes for each complete floating point number, and this is the actual format used. Numbers are expanded to the full 5-byte format when loading them to the accumulator simply for ease in manipulating them. There is also a second accumulator at

\$00B3-\$00B7, using the same format. All two-operand functions such as +, -, * and / use this second accumulator to save one operand while analyzing the second operand, and operate on the two numbers directly from the accumulators, leaving the result in the accumulator at \$00AC-\$00B0.

Strings are stored using three bytes of the four byte field as a descriptor of the string. The actual text of the string is placed elsewhere in memory. If the string is a literal string appearing in a program, the text is left in the program and referenced from there. Otherwise, the required number of bytes are allocated from the high end of the free space and added to the string space. If B\$ is 6 characters long, a simple statement like A\$=B\$ or even B\$=B\$ will cause 6 bytes to be removed from the free space and added to the string space. The string space which was used by the old string in A\$ or B\$, respectively, is simply discarded. It remains part of the string space, unavailable to BASIC. Ultimately, this process will use up all available memory if a program does many string operations. When this happens, a routine commonly known as a "garbage collector" is called to determine what memory is still in use, and move the active strings back to the high end of memory, returning the unused space (the "garbage") back to the system as free memory. There does not seem to be a clever way of doing this; most systems use a rather brute-force approach which takes a significant amount of time. This explains occasional long pauses in a program, during which you will be unable to stop execution with a ctrl-C (called BREAK or STOP on most other systems). If you can design your program to minimize the number of string assignments, you can speed them up quite a bit. The garbage collector on this system has a small bug, causing it to crash with some strange effects when using string arrays. The problem is that the garbage collector expects entries in a string array to be only 3 bytes long but they are actually 4 bytes. Various companies are marketing replacement BASIC 3 ROMs which contain a corrected garbage collector.

Now that the string is actually stored in memory, the descriptor in the string

variable must point to it. The first byte is the length of the string in bytes. Since the largest number which can be stored in one byte is 255, string length is limited to a maximum of 255 bytes. The

```
Variable
Pointer
$007B-7C $0304
```

\\
v

| | | | | | |
|------|------|------|------|------|------|
| \$41 | \$80 | \$04 | \$FC | \$7F | \$00 |
|------|------|------|------|------|------|

NAME LENGTH POINTER

\\
v

| | | | |
|------|------|------|------|
| \$41 | \$42 | \$43 | \$44 |
|------|------|------|------|

A B C D

When strings are operated on, two bytes in the accumulator (\$00AE and \$00AF) point to the string descriptor. If the string is an immediate string or a processed string which is not yet stored as a string variable, a descriptor is built at \$0068-\$006A.

For a function, bit 7 of the first character byte is set to a 1. The first two bytes of the value field point to the text just after the equals sign in the function definition. Since functions are only allowed in a program (not in the immediate mode) under this interpreter, this pointer will always point within the program space.

The other two bytes of the value field contain the first two characters of the name of

the function's dummy variable (the variable appearing inside the parenthesis in the definition). When the function is called, the value in the parenthesis is analyzed. The value of the dummy variable is saved on the stack, and then the dummy variable is assigned the value which actually appeared in the parenthesis. The expression appearing after the equals sign in the function definition is then analyzed as an arithmetic expression and the value left in the accumulator for processing as the value of the function. Lastly, the original value of the dummy variable is popped from the stack and restored. In the following example, I will define a function S which will return the square of a number, using the dummy variable X.

```
DEF FN S(X) = X * X
```

The entry in the variable table would appear something like this:

| | | | | | |
|------|------|------|------|------|------|
| \$D3 | \$00 | \$0C | \$03 | \$58 | \$00 |
|------|------|------|------|------|------|

FN S null pointer X null

\\
v

| | | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|------|
| \$95 | \$9E | \$53 | \$28 | \$58 | \$29 | \$AB | \$58 | \$A5 | \$58 | \$00 |
|------|------|------|------|------|------|------|------|------|------|------|

```
DEF FN S ( X ) = X * X
```

ARRAYS

Storage of arrays is in many ways similar to simple variables, with a few changes as necessary to allow for subscripts. The first array is stored at the beginning of the array area, and the name is stored just like a normal variable. String arrays are flagged by bit 7 of the second character, just as in simple variables. The next two bytes give the total number of bytes allocated to this array in bytes. Since arrays are of variable length, this is necessary for searching through the array area to find a specific array name. If the first array is not the desired one, simply skip over the given number of bytes to find the beginning of the next array name. The number of bytes given is the total number of bytes including the name and these two bytes themselves.

The next byte is the number of dimensions in this array. Arrays with 255 dimensions (subscripts) may be stored, but this is restricted in practice by the fact that this interpreter limits lines to 71 characters. In practice, then, arrays with some 30 subscripts may be declared, but a program using arrays with more than about 3 subscripts is rare. The interpreter detects that a variable designates an array by the left parenthesis. Thus, it is possible to have a simple variable X and an array X with no conflict. Arrays are further distinguished by the number of subscripts, so it is also possible to have an array X(A,B) and an array X(A,B,C) with no conflicts. If an array is referenced without being dimensioned, this and most other Microsoft interpreters will automatically dimension it with the number of dimensions given, with a maximum subscript of 10 in each dimension. Thus, if you reference an array with the wrong number of dimensions, you will create an entirely separate array with the new number of dimensions.

Next come a set of byte pairs giving the size of the array in each dimension. These are given with the high byte in lower memory, unlike most other two-byte items. The number given is the actual number of elements in that dimension, so if an array is dimensioned 10, the number given will be 11 (there is a 0th element). You can also

OHIO SCIENTIFIC, Inc.

With our new management team, OSI is proud to announce the addition of the **KeyFamily 300** series —

MULTI-PROCESSING BUSINESS SYSTEMS

to our complete line of 200 series timesharing business computers. Utilizing state-of-the-art microprocessor technology OSI now offers the highest performance microprocessor based business system available. Each user has his own Z80A 4MHZ CPU, 64K memory, 4 channel DMA and two serial ports. A system master processor with a separate CPU, 56K of memory, 4 channel DMA and 2 serial ports handles all disk and system I/O tasks. Our separate, proprietary, 8 Megabit inter-processor communications bus provides nearly instantaneous inter-processor data transfers. Running OSI's proprietary version of the KeyOperator-1 Multi-processing operating system allows most of the over 3000 CP/M based packages to run together with OSI's ...

KEYBASIC Version 2.0

KeyBasic 2.0 is the 65U BASIC version 1.43 compatible SUPER-BASIC language, the culmination of **your** input on 65U extensions and has many, many features unavailable in any other language. These include;

- Enhanced Extended Input
- Character oriented Disk I/O
- FIND command with limit
- CRT Command
- SWAP
- WHILE WEND
- KILL MultiByte to MultiByte input translation
- Semaphore WAIT FOR with time limit
- Enhanced Extended Output
- Key Map
- RANDOMIZE
- TIMER
- Selectable Dynamic File Allocation
- RESUME
- Invisible SPOOLING on 1 to 16 Queues onto 1 to 16 printers
- **Record Locking**
- Extended EDITOR
- 4 types of Program Chaining with COMMON Verb
- Up to 15 Disk Channels with individual buffers
- Subroutine CALL
- SuperTrace
- TIME
- DATE
- RENAME
- INSTR\$
- Delete, Resequence and Renumber In Basic
- PRINT USING
- ON TIMER GOTO
- ! and !! editor commands
- ON ERROR GOTO
- ERASE (delete file)
- OPEN (creates file)
- FIX
- 16 Digit Precision
- DEV\$

The KeyFamily 300 series will initially be available in 4 models, the 10MB 330E and 40MB 330I (up to 4 users) and the 350J/JJ (up to 8 users). These systems will include **KeyOperator-1**, **KeyWord** Word Processing System and **KeyBasic**.

ORDER YOUR SYSTEMS NOW!!!

from your dealer or

OHIO SCIENTIFIC, Inc.
6515 Main Street
Trumbull, CT 06611
(203) 268-3116

think of this as the lowest number which is not a valid subscript in this dimension. There is one pair of bytes for each subscript, given in reverse order of the subscripts.

The actual elements of the array come next, four bytes per element. In a numeric array, the values are stored as in a numeric variable, and in a string array, they are stored as in string variables. The order is most easily visualized as follows: holding all subscripts except the first constant at zero, run through all possible values of the first subscript from zero to the maximum value. Then increment the next subscript and again run through all possible values of the first subscript. When the second subscript reaches its maximum value, reset it to zero and increment the next subscript, and so on. Another way of describing it would be to equate a two-dimensional array to a table of numbers in row-column format. If the rows are stored as blocks, the first subscript is the column number and the second is the row number. Of course, you may visualize arrays in your programs in any convenient manner with no effect on how your program operates. The order of storage for an array dimensioned as DIM A(2,3) would be A(0,0), A(1,0), A(0,1), A(1,1), A(0,2), A(1,2).

STACK ENTRIES

At times, BASIC makes use of the hardware stack of the processor. In particular, GOSUBs and FOR-NEXT loops require saving information in such a way that the most recent is recovered first. The stack on this system is 256 bytes long but some other items share a portion of the stack space, limiting BASIC to about 205 bytes. Thus, if you do too many GOSUBs before RETURNing, or nest FOR loops too deeply, you will cause an error. Unfortunately, they chose to use the OM error (Out of Memory) to signal this, rather than assigning another error code. This is why some programs get this error even though there is still ample free memory on the system.

The entry for a GOSUB is fairly simple. The items pushed onto the stack are just those necessary to return and resume processing on the line containing the GOSUB. First the address of the byte after the GOSUB is pushed onto the

stack, high byte first. Then the line number of that line is pushed on, also high byte first. This is necessary so that any error message caused in the line with the GOSUB after returning can include the correct line number. Lastly, the GOSUB token itself

(\$8C) is pushed onto the stack. In returning from a GOSUB, the interpreter must strip any subsequent FOR loop entries from the stack in reaching the GOSUB entry, closing any loops which were opened in that subroutine.

Figure 2.
GOSUB Stack Entry

```

High byte \ Address of byte after GOSUB
Low byte / (1st char of subroutine line #)

High byte \ Line number of the line
Low byte / containing the GOSUB

$8C Stack marker (GOSUB token)

----- <-- Stack pointer

```

The stack entry for a FOR loop is, as you might expect, a good deal more involved. First, the address of the end of the FOR statement is pushed on the stack, high byte first. This is where the processor will loop back to upon reaching a NEXT statement. Next, the line number of the line containing the FOR statement is pushed, high byte first. The reason for this is the same as in the case of GOSUB. The loop limit (limiting value of the control variable) goes on the stack next, in floating point form just as it would appear in a variable. The byte which would appear highest in memory goes on the stack first, followed by the next lower bytes in order. The next item is a single byte giving the direction of stepping of the

control variable. It is \$FF for a negative step, \$00 for a zero step, and \$01 for a positive step. This is considered both in determining whether to add or subtract and also in determining if the loop variable has passed its limit. Next is four bytes giving the step size, as if it were stored in the accumulator and the byte at \$AF were pushed first, followed in sequence by \$AE, \$AD, and \$AC. The last real entry is the address of the loop control variable, pushed onto the stack high byte first. This is the address of the value field of the appropriate variable, so it makes no difference to the FOR loop logic whether this is a simple variable or an element of an array. The entry is finished out with a FOR token (\$81) to mark this as a FOR loop entry.

Figure 3.
FOR Loop Stack Entry

```

Low byte \ Address of the end
High byte / of the FOR statement

High byte \ Line number containing
Low byte / the FOR statement

$00AF \
$00AE \ Loop
$00AD / Limit
$00AC /

Step direction

Mantissa \
Mantissa \ Step size
Mantissa / (Absolute value)
Exponent /

High byte \ Address of the loop
Low byte / control variable

$81 Stack marker (FOR token)

----- <-- Stack pointer

```

Note that throughout the stack entry, parameters are entered in exactly the form that they will be used during execution of the loop. The address of the FOR allows a direct jump back to that point, without requiring a search for the FOR or, as in the case of a GOTO, a complete scan of the program to find a specific line number. For this reason, a FOR loop runs faster than an equivalent loop set up with an IF statement, especially in a long program. The loop limit will be compared against the value stored in a variable, so it is stored in exactly the same form as the value in a variable. The step size will be loaded into the accumulator for arithmetic, so it is stored in that format (note that you can treat bit 7 of the step direction as its sign).

An Example

For one example of a way you can speed up your program by understanding the internal storage, take the FOR loop. The FOR alerts the interpreter that the program will later be returning here, allowing it to save the location on the stack as above. It is possible to make the FOR loop mimic other types of loop logic, such as a loop which will exit based on a variable reaching a specified value. Take this program segment as an example:

```
120 INPUT A$
130 IF A$ <> "QUIT" THEN 120
```

This example looks trivial, but the basic logic matches a pattern which appears frequently in BASIC programs. Where the jump is to a preceding line, this tends to be very slow, because the interpreter must first recognize that the line number is prior to the current line, and then must start from the beginning of the program and search for the appropriate line. This can take a great deal of time in a long program.

Now consider the following alternate way of coding the same logic:

```
110 FOR BV = 0 TO 0 STEP 0
120 INPUT A$
130 BV = (A$ = "QUIT")
140 NEXT
```

Since this loop includes a keyboard input, that will determine overall loop execution time, rather than the speed of the interpretation. The basic principle remains valid for many

other types of loop which will terminate upon a particular condition being met, but which do not obviously lend themselves to the FOR loop structure.

The first example is straight forward. If the user types in the word "QUIT" the loop will terminate; otherwise, it will just continue asking for more inputs. The second example contains the same logic in a disguised version. Line 110 sets up the loop and presets BV to zero. Line 120 receives the user's input as before. Line 130 is the first "tricky" part. The expression (A\$ = "QUIT") is a boolean expression which can be assigned a numerical value. Different interpreters use different numbers to represent "TRUE", with -1 and 1 being the most common. Virtually all BASIC interpreters, however, use zero for "FALSE". Thus, if the user inputs anything other than "QUIT", BV is set to zero; if he types "QUIT", it is set to 1.

Now consider the task of the NEXT statement in line 140. It must retrieve the value of the loop variable (BV), add or subtract the appropriate step size, and compare the result with the loop limit. Since the step size was zero, BV being exactly equal to the limit of 0 will cause the loop to repeat, while any other value will cause the loop to terminate. Therefore, if the input string is a "QUIT", BV has some non zero value, and the loop terminates; if not, BV will be zero and the loop repeats. Compare this logic to the first example. This way of coding does require more actual code and is more difficult to read, but balancing this is the fact that it runs faster and does not depend on line numbers, which permits easier renumbering of programs.

CONCLUSION

By now I hope to have you curious enough to poke (or PEEK) around a bit in your own system. Many of the popular personal computers use an internal format almost identical to this, and those which differ are usually close enough to be recognizable. Aside from the satisfaction of better understanding your system, you can use this information to devise ways of doing things better or faster on your system, such as inserting one string into the middle of another without a lot of MID\$ operations, or

DISK DRIVE RECONDITIONING

| FLAT RATES | Parts & Labor Included (Missing parts extra) |
|---------------------------|---|
| 8" Double Sided Siemens | \$170.00 |
| 8" Single Sided Siemens | \$150.00 |
| 8" Double Sided Remex | \$225.00 |
| 8" Single Sided Shugart | \$190.00 |
| 8" Double Sided Shugart | \$250.00 |
| 5 1/4 M.P.I. Single Sided | \$100.00 |

Specific models & other rates upon request.

ONE WEEK TURN AROUND TYPICAL

You'll be notified of —

1. The date we received your drive.
2. Any delays & estimated completion date.
3. Date drive was shipped from our plant.
4. Repairs performed on your drive.
5. Parts used (#and description).

90 day warranty —

Write or call for detailed brochure

We sell emergency parts

Phone: (417) 485-2501



FESSENDEN COMPUTERS
116 N. 3RD STREET
OZARK, MO 65721

OSI—AFFORDABLE DATA BASE MANAGER

B&W FILE MASTER runs under OS65D V3.3, (video only). Single or dual drive. Requires 48K RAM.

FEATURES: User and/or pre defined files with coding options, formatted screen viewing and inputting, find, edit, update, delete & page. 'Screen', 'quick' and 'format' dump. Manual included. only \$55.00

Manual only (price applied towards purchase) \$10.00

ADD ON FEATURES:

Label print option \$45.00

Report generator \$45.00

SPECIAL INTRODUCTORY OFFER!

Expires June 15th, 1983.

B&W File Master & Report Generator \$80.00

B&W File Master & Label Print Option \$80.00

B&W File Master, Report Generator & Label Print Option \$105.00

Above prices include manual.

For more information contact:

BUNIN & WARD COMPUTER SERVICES
P.O. BOX 895 CHURCH STREET STA.
NEW YORK, NY 10008
(212) 434-5760

even making one array overlap another! The information on the internal storage of variables and program is particularly useful to those of us who program frequently in a combination of BASIC and machine language. I can't recall writing a program of any real size without such a mixture in a long time. Good luck finding out what makes your system tick!



INDIRECT FILES IN 65D

by: Charles Stewart
3033 Marvin Dr.
Adrian, MI 49221

This is one of the more valuable but difficult commands to understand and master for the new disk user. The purpose of the indirect file is to allow the user to partition memory in RAM i.e. to have more than one program in memory thus allowing you to merge programs together. This all sounds rather confusing, allow me to try and simplify.

Consider the computer like a file cabinet. We know we can put things into or take items out. Under normal operation when we have a program loaded from the disk, it starts at HEX location \$327E and continues until the requested program is loaded. This is fine if all we want to do is run one program, but suppose you would like to run some of the various utilities available such as the Variable Table Maker (vtm), line renumber, etc., and others available from the major software houses. Most of these are basic programs which are designed with line numbers starting at 60000 and above, and requires the user to load this program on the top of the subject program. You can have your utilities on cassette and load with the IO command, which is fine but there is a simpler way!

The indirect file command solves the problem!

In my analogy of the file cabinet, you add whatever you desire into your file, but on the computer we must determine where to put the program, see if there is enough free space for it, etc. The following is a step by step example of how to utilize the indirect file commands.

1. Load the first program from the disk in the normal

manner. i.e. DISK! "LOAD EX-AMPLE"

2. Peek the location 133 and note the contents, i.e. PRINT (PEEK(133)). Location 133 contains the highest page available in memory found during boot up. We will call this HP in our calculation.

3. Determine the number of unused pages in memory -- this can be obtained by the command PRINT INT(FRE(X)/256) we will call this UN for unused.

4. The starting page of the work space is 51 on 5 1/4" Disks 3.0 to 3.2 operating systems.

5. Determine the number of pages used by the program to be moved to the indirect file.

Pages used=(HP)-51-(UN)

If (UN) is greater than the pages left, then there isn't sufficient room to move programs to the indirect file.

6. Once we have determined pages used by the program, we are ready to set up the vectors for the indirect file move. You must poke locations 9554 and 9368 with the page number that you wish to start the transfer. Determine this by the following formula.

Page Number=(HP)-(UN)

Poke locations 9554 and 9368 with the page number in the immediate mode. For those who haven't discovered it, the immediate mode is a command recognized by the basic interpreter without a line number typed in from the keyboard followed by hitting the return key. RUN is an immediate mode command.

7. Type LIST but do not hit return key yet!

8. With the shift key held down, press the K key (Shift K).

9. Press RETURN and wait for the listing to end.

10. With the SHIFT key depressed type M, a double bracket will appear.

11. With the SHIFT key still depressed, strike the P, release the shift and press the RETURN, i.e. shift M K RETURN.

12. You now have the program in the indirect file. Clear the workspace with the 'NEW' command and load the program

marmen
MARMEN COMPUTING, INC.

Fire Department Software
• DISPATCH •

A COMPLETE DISPATCHING SYSTEM
FOR OSI MULTIUSER SYSTEMS.
COMPLETE DOCUMENTATION
AND OPERATING INSTRUCTIONS

• Record Keeping •

UNIFORMED FIRE INCIDENT
REPORTING SYSTEM (UFIRS)
PREPARES UFIRS REPORTS
COMPLETE LOCAL DATA BASE

DEALER INQUIRIES WANTED
CONTACT
Bob Tidmore
MARMEN
125 Sixth Avenue
Menominee, Michigan 49858
906-863-2611

"Computer Business Software"

"CBS"

• INTEGRATED
BUSINESS SYSTEM

— FEATURING —

- Accounts Receivable
- Inventory Control
- Order Entry/Invoicing
- Accounts Payable
- General Ledger
- Payroll

• BUSI-CALC
"An electronic worksheet"

— FEATURING —

- Local and General Formatting
- Replication
- Variable Column Widths
- Editing
- Insertion/Deletion of Rows and Columns
- Protected Entries
- Help Screen
- Flexible Printing
- Complete User Manual

MICRO SOFTWARE
INTERNATIONAL
3300 South Madelyn □ Sioux Falls, SD 57108
1-800-843-9838

you wish to merge in the normal manner.

13. When the 2nd program is loaded execute a CONTROL X
Continued on page 19

**A BASIC EXTENSION PROCESSOR
FOR BASIC IN ROM (BEP)**

By: Gerdt Vilholm
Prinsessegade 4 B, St.
DK 1422 Copenhagen K
Denmark

This program makes it easier to make extensions to BASIC in ROM through a patch into the Parser routine at addr. 00BC. It has, however, provisions to avoid the problems earlier described by Ed Carlson and Michael Mahoney in MICRO.

I have moved the screen driver to the MONITOR-EPROM, and therefore, the space above Coldstart (on my Superboard) can be used for BEP. If you have one of the monitors with its own screen driver, you can put BEP into Basic No. 4 EPROM, and change the JMP at BF2D for your own screen-driver.

The code at BCFF is the patch, which on Coldstart will be placed in the Parser at 00CD. At 00D0 will be placed a JMP BF30 - a vector which allows further extensions to be added.

BEP will recognize the extension-operators !, #, %, & and ' when they appear in a BASIC line or in immediate mode.

The processor status will then be saved, and the return address will be pulled off the stack and saved:

00D8-D9 Return address (R)
00DA Accumulator
00DB X-reg.
00DC Y-reg.
00DD P-reg.

R will also appear in X(LO)

BCFF 4CF5BE JMP BEF5 Patch in.
BD02 4C3CBF JMP BF30 00BC routine
BD0A 90C0D0 32768 in FLOAT
BD0D 00

Basic Extension Processor. Enter at BEF5

| | | | | |
|------|--------|-------|------|-----------------------|
| BECE | A411 | LDYZ | 11 | Convert binary in |
| BED0 | A512 | LDAZ | 12 | 11-12 to FLOAT |
| BED2 | 48 | PHA | | in FPA 1 |
| BED3 | 297F | ANDIM | 7F | |
| BED5 | 20C1AF | JSP | AFCM | make FLOAT |
| BED8 | 63 | PLA | | If HI bit not set |
| BED9 | 10D1 | BPL | BEAC | then RTS |
| BEDB | A90A | LDAIM | CA | If HI bit set, |
| BEDD | A0BD | LDYIM | BD | then add 32768 |
| BEDF | 4C6CE4 | JMP | B46C | |
| BEE2 | A512 | LDAZ | 12 | Convert hexstring |
| BEE4 | 48 | PHA | | pointed to by C3-4 |
| BEE5 | A511 | LDAZ | 11 | into FLOAT in FPA 1 |
| BEE7 | 48 | PHA | | |
| BEE8 | 2064BE | JSP | BE64 | hex to bin. |
| BEEB | 20CEBE | JSP | BEEC | bin to float |
| BEEE | 63 | PLA | | |
| BEEF | 9511 | STAZ | 11 | restore 11-12 |
| BEF1 | 68 | PLA | | |
| BEF2 | 9512 | STAZ | 12 | |
| BEF4 | 6C | RTS | | |
| BEF5 | 38 | SEC | | BEP Enter |
| BEF6 | E930 | SECIM | 30 | Do number test |
| BEF8 | 38 | SEC | | |
| BEF9 | E9D0 | SECIM | D0 | |
| BEFB | 9043 | BCC | BF40 | RTS - it is number |
| BEFD | F041 | BEG | BF40 | RTS - end of st.mnt |
| BEFF | 03 | PHP | | |
| BF00 | C928 | CMFIM | 28 | |
| BF02 | B03E | BCS | BF3F | it is between 28-2F |
| BF04 | C924 | CMFIM | 24 | |
| BF06 | F037 | BEG | BF3F | it is \$ |
| BF08 | C922 | CMFIM | 22 | |
| BF0A | F033 | BEG | BF3F | it is " |
| BF0C | 35DA | STAZ | DA | it is !, #, %, & or ' |
| BF0E | 36DB | STXZ | DE | save entire proces- |
| BF10 | 34DC | STYZ | DC | sor-status |
| BF12 | 68 | PLA | | |
| BF13 | 35DD | STAZ | DD | save P-reg. |
| BF15 | 68 | PLA | | |
| BF16 | 85D3 | STAZ | D3 | save Return addr. |
| BF18 | AA | TAX | | put also R into |
| BF19 | 68 | PLA | | X (LO) and Y (HI) |
| BF1A | 85D9 | STAZ | D9 | |
| BF1C | A8 | TAY | | |
| BF1D | ASDA | LDAZ | DA | |

Continued

MnM Software Technologies, Inc.

416 Hungerford Drive, Suite 216
Rockville, Maryland 20850

**INTRODUCING OUR
NEW PRODUCT LINE**

The missing tools for the OS-65U system. Our products are written in 6502 native code and are compatible with 65U, single, time-share or network modes. Floppy or hard disk systems.

Ky. ASM V1.1-ASSEMBLER (Virtual source files, superfast, many extra features including a label table) ...\$129 (manual \$25)(50 pgs.)

Ky. COM V1.5-COMPIER (Configures itself to V1.2 or 1.42, dynamic variables and arrays-DIM A (N), supports machine language routines at hex6000, last 2 pages in high memory accessible, debug with interpreter and compile in 2-3 minutes. Protect your valuable source routines, gain as much as 2-10 times on average programs in execution speed. Supports INPUT and PRINT on the 1.42 system.\$395 (manual \$25)(110 pgs.)

Ky. DEV 1-ASSEMBLER AND COMPILER TOGETHER...\$474(manual \$40)

KEYMASTER I V1.0-The word processing missing link for OS-65U based systems. KEYMASTER I is screen oriented, menu driven, simple to use yet highly advanced. KEYMASTER I contains most of the best features only found in dedicated word processing systems. Ask for the features you have been looking for and the answer will most likely be "YES!" To be released in February...Introductory price \$475 (Manual \$25)

All software comes with license agreement, registration card, manual, binder, diskette holder and 8" diskette.

Manuals are available by themselves and are deductible from full purchase price of software within 60 days after purchase.

Foreign orders must be paid in U.S. dollars and drawn on a U.S. bank or international money order.

ALLOW 2 WEEKS FOR DELIVERY AFTER RECEIPT OF CHECK OR MONEY ORDER

CALL 301/279-2225

High Resolution Color Graphics

Finally, low-cost high-resolution color graphics is available for your OSI computer. With Color-Plus from Generic Computer Products, you can have the following features:

- Color — 15 unique colors plus transparent
- High Resolution — 256 × 192 with 2 different colors in each group of 8 horizontal dots
- Medium Resolution — 48 × 64
- Text — 24 × 40 characters
- Sprites — 32 programmable animation patterns that move smoothly across the screen without disturbing the background
- Joystick interface — Supports up to 2 joysticks or 4 game paddles with 8-bit resolution
- Software — Extensions for OS65D which provide a superset of Apple][graphics instructions
- Video switch — Software selects the Color-Plus or standard 540 video display

Color-Plus does not need user memory, leaving the full 48K memory space available for user programs.

Two versions are available:

CP-16 — Connects to C4P and C8P systems with the 16-pin bus or to any system equipped with the OSI CA20 board. Comes in ABS plastic case.

CP-48 — Connects to the standard 48-pin bus.

Cost: \$279

Low Power Memory Board

Our popular MEM+ board is ideal for:

- Partitions for multi-user systems
- 64K CP/M systems when combined with the D&N-80 CPU board
- Upgrading systems where backplane space, low power consumption, and/or low heat dissipation is required

Options include:

- OSI compatible floppy disk controller — protects against disk crashes caused by power failures
- Real time clock/calendar — Date and time with battery backup
- Centronics parallel printer interface — Supported by software that automatically patches OS65D and OS65U
- One year warranty

MEM+ includes the following features:

- Low power consumption — A 48K board draws about 1/4 amp. A fully populated board draws about 3/4 amp
- Accepts 2K × 8-bit memory chips — Compatible with 2716-type EPROMs
- High reliability — All memory chips in machine-screw sockets
- Versatile addressing — Divided into 3 16K blocks and 2 individually addressable 4K or 8K blocks

| | | | |
|------|-------|----------------------|------|
| Bare | \$100 | | |
| 16K | \$275 | Disk controller | \$95 |
| 24K | \$325 | | |
| 32K | \$370 | Real time clock | \$65 |
| 40K | \$410 | | |
| 48K | \$450 | Centronics interface | \$45 |
| 56K | \$490 | | |
| 64K | \$530 | | |

VISA, MasterCard, personal checks and C.O.D.s all accepted. Add \$5 per board for shipping and handling.

To order, or for more information, contact:

Fial Computer
11266 SE 21st Avenue
Milwaukie, Oregon 97222
(503) 654-9574



Generic Computer Products

5740 S.E. 18th Ave. Portland, OR 97202

and Y(HI) and A contains the operator. The Parser is called from many different places in BASIC, and therefore, R is checked to see if the call came from a place which is allowed to call the extensions. If no match is found in EPROM, a JMP is made to the vector at 00D0. You can point this vector to your own extensions, only remember that when a match to R is not found, a JMP BF30 should be executed. This will restore registers and return to BASIC.

Some extensions are built into BEP to allow hexadecimal arithmetic. These extensions will use the hex to binary conversion I have enclosed in the Coldstart-routine in my last article.

The new functions are:

1. Arithmetic expression

A=&ABFF+&7F
POKE &02FE,&AA
X=PEEK(&F000) AND &0F
and many others.

2. PRINT&X will print variable X as a 4-character hexnumber. PRINT <X will print only the 2 least significant hexdigits. You can use tabulation, commas, and semicolons as usual.

3. X=VAL(M\$) will yield the correct value if M\$ contains a valid hexnumber preceded by & such as "&FOAB".

4. A\$=STR\$(X) will yield a 4-character hexstring with the value of variable X. A\$=STR\$(X) does the same, but only the 2 LSD just as in PRINT. & is used instead of \$ to avoid confusion with string variable names.

When you make a patch into the Parser, you may expect a slight slowdown of execution. I have, however, made the patch at a late point, where tokens and alphabets already have left. They will, therefore, not be delayed. A test showed, that a program containing lots of numbers would be delayed 1.3 percent. But then I substituted hex for all the decimal numbers, and now the test program would run 30 percent faster!

The checking of R allows you to interfere almost anywhere in BASIC, but some guidelines can be given for making your own extensions: If you want an extension operator to work as a command on its own, R should read A5F8.

If you want an initial keyword to do a special task, such

| | | | | |
|------|--------|-------|------|---------------------|
| BF1F | C926 | CMFIM | 26 | is ch. & |
| BF21 | F01E | BEG | BF41 | then enter hexrout. |
| BF23 | 4CDEBF | JMP | BFDE | else enter Vector |
| BF26 | 20E2BE | JSR | BEE2 | VAL, hex to Float |
| BF29 | 4CEEB3 | JMP | B3EE | Re-enter VALroutine |
| BF2C | EA | NOF | | |
| BF2D | 4C71F8 | JMP | FB71 | Patch for new CRT |
| BF30 | A6DB | LDXZ | DB | Restore Processor- |
| BF32 | A4DC | LDYZ | DC | registers if search |
| BF34 | A5D9 | LDAZ | D9 | has failed |
| BF36 | 48 | PHA | | |
| BF37 | A5D8 | LDAZ | D3 | |
| BF39 | 48 | PHA | | |
| BF3A | A5DD | LDAZ | DD | |
| BF3C | 48 | PHA | | |
| BF3D | A5DA | LDAZ | DA | |
| BF3F | 23 | PLP | | And go back |
| BF40 | 60 | RTS | | to Basic |
| BF41 | E0A6 | CPXIM | A6 | & found, search R |
| BF43 | D004 | BNE | BF49 | |
| BF45 | C0AB | CPYIM | AB | R=ABA6? |
| BF47 | F099 | BEG | BEE2 | Arith. Expression |
| BF49 | C0AB | CPYIM | A3 | |
| BF4B | D008 | BNE | BF55 | |
| BF4D | E02E | CPXIM | 2E | R=A82E? |
| BF4F | F04F | BEG | BFA0 | PRINT& |
| BF51 | E0BF | CPXIM | BF | R=A8BF? |
| BF53 | F04E | BEG | BFA0 | PRINT& |
| BF55 | E02C | CPXIM | 2C | |
| BF57 | D004 | BNE | BF5D | |
| BF59 | C0AC | CPYIM | AC | R=AC2C? |
| BF5B | F05E | BEG | BFBB | STR\$& |
| BF5D | E0EA | CPXIM | EA | |
| BF5F | D004 | BNE | BF65 | |
| BF61 | C0B3 | CPYIM | B3 | R=B3EA? |
| BF63 | F0C1 | BEG | BF26 | VAL hex |
| BF65 | 4CDEBF | JMP | BFDE | Else Vector |
| BF68 | C9C2 | CMFIM | C2 | Make hexstring |
| BF6A | F005 | BEG | BF71 | from binary |
| BF6C | A512 | LDAZ | 12 | in 11-12 |
| BF6E | 2073BF | JSR | BF73 | if Ac=02, string |
| BF71 | A511 | LDAZ | 11 | will be 2 ch. |
| BF73 | 48 | PHA | | else 4ch. |
| BF74 | 4A | LSRA | | |
| BF75 | 4A | LSRA | | |
| BF76 | 4A | LSRA | | |
| BF77 | 4A | LSRA | | |
| BF78 | 207EBF | JSR | BF7E | |
| BF7B | 63 | PLA | | |
| BF7C | 290F | ANDIM | CF | |
| BF7E | 18 | CLC | | |
| BF7F | 693C | ADCIM | 3C | |
| BF81 | C93A | CMFIM | 3A | |
| BF83 | 9002 | BCC | BF37 | |
| BF85 | 6906 | ADCIM | 06 | |
| BF87 | 91AD | STAIY | AD | AD-E is indirect- |
| BF89 | C3 | INY | | pointer to string |
| BF8A | 60 | RTS | | |
| BF8E | 20BC00 | JSR | 00BC | If next ch. is < |
| BF8E | C9AC | CMFIM | AC | then put 02 into |
| BF90 | D007 | BNE | BF99 | 00FF |
| BF92 | A902 | LDAIM | 02 | |
| BF94 | 85FF | STAZ | FF | |
| BF96 | 4CBC00 | JMP | 0CBC | |
| BF99 | A904 | LDAIM | 04 | else put 04 there |
| BF9B | 85FF | STAZ | FF | |
| BF9D | 4CC200 | JMP | 0CC2 | RTS via 00C2 |
| BFA0 | 208BBF | JSR | BF8B | PRINT& |
| BFA3 | 20ADAA | JSR | AAAD | Evaluate expr. |
| BFA6 | 2008E4 | JSR | B408 | FLOAT to BIN. |
| BFA9 | A0C1 | LDYIM | 01 | Set pointer to |
| BFAB | 84AE | STYZ | AE | 0100 and Y=00 |
| BFAD | 88 | DEY | | |
| BFAE | 84AD | STYZ | AD | |
| BF80 | A5FF | LDAZ | FF | Get length 2 or 4 |
| BF82 | 2068BF | JSR | BF68 | make hexstring |
| BF85 | 203CBA | JSR | BA8C | end string 00 |
| BF88 | 4C4DA3 | JMP | A84D | Re-enter PRINT |
| BF8B | 68 | PLA | | STR\$& |
| BF8C | C97A | CMFIM | 7A | Check token |
| BF8E | D018 | BNE | BFDE | Exit if different |

cont, page 13

SOFTWARE REVIEW

GANDER SOFTWARE'S FINANCIAL PLANNER

by: Gary L. Gesmundo

A new software house has been heard from, and if this package is any indication, OSI users can feel pleased.

Gander Software's Financial Planner consists of 6 basic programs that provide analysis of "Ordinary" loans and annuities, "Annuity Due" transactions, present and future values, and sinking funds. Also included is an interest conversion program that allows conversions of nominal rates to effective rates of return based on the usual compounding periods (semi-annual to continuous).

When in use, the program resembles many other spread sheets. However, it's not intended to provide the same kind of utility, nor is the user required to learn any math or "language" to use it.

Rather than require user input of formulae, definition of "cells", etc., this program provides the formatting and math for the types of problems identified.

In all but the interest conversion and amortization programs, the user can play "What If".

When a "What If" program is entered, the user first "loads" one problem. This is a dummy, serving to initialize the arrays used for "What If". For example, in the Loan/Annuity Analysis, the user first enters information for the compounding periods, loan amount, term, and interest rate, after which the payment is calculated by the program.

Once the first problem has been entered, the user can go to the "What If" mode/menu.

In "What If", the user can change any variable, and have the program re-solve for any other variable (the only exception being that re-solve for compounding periods is not allowed, for obvious reasons relating to the real world, though the compounding periods can be changed and any other variable re-solved).

When in the "What-If" mode, the original problem is preserved at the top of the screen, and each time the user asks for another "What-If,"

the most recent problem is duplicated, up to 10 versions on the screen at one time.

You can play "What-If" as many times as you like within one problem, but the program's great value comes from duplicating the problems, changing your variables successively, and then comparing the answers.

The programmer has built in some nice "extras" that render these programs very useful. For example, in the Loan/Annuity Program, for a simple key-stroke, you can get the Future value of any of the "What-If" problems on the screen, thus telling you what you could earn with your dollars if you invested instead of borrowed. You can compare any two "What-Ifs" on the screen and get the net difference; and, in the loan program, generate an instant amortization schedule, i.e., fetch an instant pay-off for a date certain.

Finally, in all the "What-If" programs, the user can save any specified "What-Ifs" to disk, and each program provides a print-out of the "What-Ifs" on the screen, and other useful information.

The Amortization Schedule is, frankly, about the best I've seen. It is driven by records created in the Loan/Annuity program and allows remarkable flexibility. For instance, it copes with balloon payments specified in the Loan program, but will also create its own if you tell it there is a payoff required at a date certain. Extra features include specification of calendar or fiscal years, the ability to specify increased payments of either a known percentage or dollar amount, to specify a title for the schedule, and YTD and Totals-to-Date summaries of interest and principal paid. All page breaks come at a year's end, and a print-out of the revised payment schedule, if any, is provided.

One of the most unusual features of the schedule is that the entire history is tracked in months and years, not just by payment numbers.

The interest conversion program previously noted is not a "What-If." It provides the answers, up to 10 at a time, allows comparisons, and dumps to the printer a history of the problems on the screen, plus the net return on

\$1,000.00 for one year at the rates in those problems. Included is a math calculator you can access without disturbing screen contents.

The conversion program also allows the creation, with printer, of hard copy interest rate tables. There is an "auto-increment" mode that allows the user to specify a base interest rate, and an increment. The program then solves those, up to 10, displays them all, and lets you print. In approximately 2 minutes I had created a hard copy table, for example, of the various effective rates between 12.00% and 12.90%. Pretty slick.

The hardware required is a 88 K OSI dual floppy, with serial terminal. When you get your program, you get just a program disk. It contains the utilities necessary to create your own data files to back up disks, etc. The program appears to use a modified BEXEC* which supports most non-ANSI terminals, and includes a set up routine for others.

Within the menu, which the user must date on the way in, is a routine that accurately calculates the day of the week (can be re-set), and prints to screen or printer a calendar for the month and year in the menu. One can also specify the printer port enabled.

The user manual is concise, but thorough, and well laid out, done in the "walk through" style, includes many copies of screen displays, and even contains a glossary of financial and computerese terms. Unfortunately, it does not include an index, something I hope software documenters soon get around to providing.

The general feel of the programs, and documentation, is professional. The screens are attractive, well mannered, and error trapping is very good.

As indicated earlier this package does not have the power of many spread sheets. It doesn't do everything, but what it does it does very well. For the banker, accountant, investor, and business man who does not want to learn to use a spreadsheet, but just wants answers, these programs provide them, quickly and accurately.

My overall impression is that this package, though not cheap at \$400.00 retail, will help dealers sell computers.

continued from page 11

as SAVE!, R should read the address minus one of the keywords routine. If you want a secondary keyword to do something special, such as X=INT%, R should read AC2C. Then pull from the stack the token shifted one bit left - see the code at BFBB. This token can also be found in 00DB. Don't forget to push the token back on stack, if the match fails.

I hope to come back with some more extensions in a later article.

| | | | | |
|------|--------|-------|------|----------------------|
| EFCC | 208BEF | JSR | BF3B | Check for < |
| EFCC | 20F5AE | JSR | ABF5 | Evaluate (expr.) |
| EFCC | 20B0AA | JSR | AAB0 | Check datatype |
| EFCC | 2008B4 | JSR | B4C3 | FPA 1 to bin |
| EFCC | A5FF | LDAZ | FF | get length 2 or 4 |
| BFCE | 20A4E0 | JSR | E0A4 | allocate string |
| EFD1 | A000 | LDYIM | CC | |
| EFD3 | A5FF | LDAZ | FF | length |
| EFD5 | 2063BF | JSR | BF63 | make string |
| EFD8 | 4CEDE0 | JMP | B0ED | check string-RTS |
| EFDE | 43 | PHA | | token on stack again |
| BFDC | A5DA | LDAZ | DA | restore Ac |
| EFDE | 4CD0C0 | JMP | 00DC | Enter RAMvector |



ETX/ACK FOR CP/M

By Al Peabody

Computers are fast. Printers are slow. This creates a problem. If you just let your computer spew out data to the printer as fast as it can, most of it will be lost. But if you slow it down to the printer's slowest speed as it is doing a carriage return, you will grow old waiting for things to print.

As a result, and particularly for letter-quality printers, which vary greatly in speed

depending on what they are doing, a "protocol" must be established to allow the printer to move as rapidly as it can while not losing data.

About the simplest way is to use a "hardware handshake," meaning that the printer, when turned on and ready to run, outputs a voltage on one of its connector pins, saying, "I'm ready, send me something." Then, when the computer sends the printer enough data that the printer's buffer memory is almost full, the voltage is turned off. Later, when most of the data in the

buffer has been printed, the "handshake line" is turned back on, and the computer can start sending more data.

This simple arrangement has several shortcomings. First of all, the computer has to have the appropriate circuitry to refrain from sending characters to the printer unless the "ready" signal is hot. Secondly, certain programs work faster when they can detect a positive signal from the printer and interrupt whatever else they are doing to act on it, rather than detecting a voltage. Word-

From Gander Software

A New Standard of Excellence

FINANCIAL PLANNER

Get "What If" answers for up to 10 displayed problems in:

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

HARDWARE REQUIREMENTS: 48K OSi, dual 8" floppy, serial terminal system.

FEATURES: package allows configuration to almost all non-ANSI terminals, AND user specification of printer port.

PRICE: \$400.00 (User Manual, \$25.00, credited toward Planner purchase). Michigan residents add 4% sales tax.

COMING SOON: Hard Disk version.

DEALERS: This program, of great value to lawyers, bankers, insurance people, and real estate people, will help you sell hardware. Inquiries invited.

A POWERFUL TOOL FOR EVALUATING ALTERNATIVES!

The first four programs all: allow you to solve a named variable after changing another variable, let you net the difference between any displayed problems, provide selective saves to disk, give you very informative printouts based on the problems solved, and much, much more.

The "Amortization Schedules" program provides more flexibility than any other schedule known to GANDER. It lets you deal with balloon payments, early pay-offs, annual payment increases (by percentages or dollars), keeps a running total of your entire transaction to pay off, schedules payments by both month and year, and reports YTD totals based on user selected calendar OR fiscal years.

"Interest Conversions" lets you key in any nominal rate and reports the true effective rate for compounding semi-annually, quarterly, monthly, daily, and continuously, and allows the print out of interest tables (your choice of rate and increments). It also includes a simple calculator, which can be used without disturbing other problems displayed, and which contains three separate user addressable memories.

Finally, to aid planning, the Menu program will generate a calendar for any month/year between 1901 and 2399, and accurately accounts for leap years!

GANDER SOFTWARE

3223 Gross Road
"The Ponds"
Hastings, MI 49058



"It Flies"

Star, for instance, works so much faster when it uses its "Port Driver" rather than a simple handshake that it becomes genuinely possible to edit one file while another is busily printing, even in a single-user system.

The problem is that WordStar, and other programs which use a "Port Driver," require that the printer return a signal to tell them when it is ready for more. Then they can send more quickly, and get back to whatever else they were doing.

In my own case, I recently purchased a Diablo printer. I hooked it up to my OSI + D&N-80 CP/M computer, and liked the pretty output. But I was unable to take full advantage of some of my programs because they used the "list" device driver in CP/M to send each character to the printer, and that driver has no provision for inputting information from the printer to determine when it is ready for another character -- so the sophisticated "software protocol" I wanted to use was impossible.

Being a man who doesn't know the meaning of the word "impossible" (it has more than three syllables), I determined

to rewrite the list device driver to incorporate ETX/ACK protocol.

The list device driver is part of the BIOS, the hardware dependent part of CP/M written by Lifeboat, or OSI, or D&N, or whoever produced your CP/M, not by Digital Research. Everyone's will, therefore, be somewhat different, but in a way it will be the same: it has to do the same thing, so the code must be similar. To find the list driver in your CP/M, you must DDT the system, saved as "CPM56.COM."

You know the address of your printer output board: perhaps FB00 if you have a CA-10, B0 if you have a D&N 1600, or whatever. Look through CP/M's code and disassemble it until you find driver which outputs to that address in a small loop, by inputting the status port, checking it against a constant, then outputting to the data port. CAUTION: do this whole process ONLY on a backup disk you can afford to bomb!

Once you have found the area you are looking for, you are 1/3 of the way there. You must now find another area of code which will never be called, to insert your ETX/ACK

routine. I used the device #8 driver, which outputs characters to the CA-10 board at CF00H. I have no such board, so it won't be used.

Now you must insert, right AFTER the instruction which sends a byte out to your printer, a subroutine CALL instruction pointing to the ETX/ACK routine you will install. I found I only had to move one byte of code down to make room, since the next routine in my BIOS was the DEV#8 driver I wanted to replace. The only instruction I had to move was the RET (return) instruction following the "byte output" instruction. I considered it unlikely that any other code (outside the routine I was modifying) would jump to a return, since a jump instruction takes more room than another RET would!

Okay, where are we? We have found the output routine we want to modify and inserted a subroutine call, so that each time CP/M outputs a character to the printer, we will jump to our subroutine. So what must the subroutine do? This:

1. Increment a counter keeping track of how many bytes we have sent out;

WHAT ARE THE USERS SAYING???

About Multi-Processing with the Denver Board

“ . . . The easiest OSI enhancement we have ever installed!”

Bruce Sexton
Southwest Data Systems
Liberal, KS

“ . . . No more waiting. In the past I had to wait for my secretary to finish her work . . . not with the Denver Boards.”

Chuck Nix
School Administrator
Sterling, CO

“ . . . Five user system . . . No slow-down, you can't tell if anyone else is on the machine. We were amazed how few program changes were necessary . . . and support has been great.”

Dave Kessler
Computer Center
Tyler, TX

IF . . . you have an OSI system with two or more users
THEN . . . you should have the Denver Board.

Call or write:

DBi, inc.

p.o. box 7276
denver, co 80207
(303) 364-6987

Dealer Inquires Invited

2. When that counter reaches a certain value (depending on the size of your printer's buffer),

- a. output an ETX character (CHR\$(3));
- b. wait for the printer to return an ACK (CHR\$(6));

3. Then return to the calling point.

Some CP/M's have BIOSes written in 8080 assembler, some in Z80 assembler, so I can't give you source code which you can plug right in.

However, you already have routines (look for them!) which check the status of the port and output a character to the port. Now all you need to do is be sure you use them, and be sure you DON'T write your routine so that it outputs a byte (the ETX character) by calling the routine which you have modified to jump right back to the routine you are writing, or you might get in an endless loop, with the routine calling itself until the stack overflows and the computer crashes...

My Diablo now prints happily, either in WordStar or by using the LST: device.



HOW TO BUILD A DATA SEPARATOR AND USE IT TO INTERFACE A CIP WITH SASI-COMPATIBLE DRIVES

By: Jim McConkey
7304 Centennial Rd.
Rockville, MD 20855

Like Cassette Corner, I have recently gone disk. However, in my quest to upgrade at the lowest possible cost, I didn't use OSI's 610 board or an MPI drive.

I am presently running with a CIP rev D with Progressive Computing's true 32 mod (which also lets me use 16 x 64), a BMC green phosphor monitor, an IDS-225 graphics printer and a homebrew expansion cage. The expansion consists of another power supply, address and control buffering, a real time clock, a hardware random number generator, 24K of RAM, a floppy disk controller and drive and still more yet to come.

The floppy controller was built from scratch with minor modifications from the SAMS manual for OSI's 610 board. I scratched the 610's memory

section and instead designed a new memory expansion based on the fairly new TMM2016 2Kx8 RAMs, which are about half as expensive as the usual 2114s. 24K was chosen for two reasons. First, that's all that would fit on a card and second, the last 8K will be used in a 256x256 dot-addressable graphics board, which will give me a total of 40K.

As you may know, the IBM PC is supplied with two SS/DD drives (Tandon TM100-1). Many PC buyers (especially in my area) replace the SS/DD drives with DS/DD drives and sell the old drives cheap. I picked up an ex-PC drive for just over \$100. A couple quick calls to my local OSI suppliers showed that none knew how to use a Tandon drive or where I could find the required data separator. Not to be deterred, I designed my own data separator and figured out how to connect the Tandon.

The Tandon TM100-1 uses the Shugart Associates Standard Interface (SASI). Needless to say, the Shugart SA-400 (SS/SD) also uses the SASI, along with several other manufactureres. I have tried the connections presented here with both the TM100-1 and the SA400 and both work fine.

Tandon and Shugart drives, like the MPI, do not separate read data from the read clock, so a data separator is required. The schematic for my data separator is shown in figure 1, and the associated logic diagram is shown in figure 2.

The read data signal from the drives consists of a string of negative-going pulses about 1 usec wide. A zero is represented by one pulse in an 8 usec (the data clock rate is 125KHz) bit cell while a one is represented by two pulses evenly spaced in a bit cell.

Flip-flop IC1a is configured rather unusually to function as an inverter. One shot IC2 is set for 6 usec and is non-retriggerable. This lets it ignore the extra pulses which indicate ones and occur 4 usec apart. This recovers the clock.

Flip-flop IC1b recovers the data. Since the rising edge of the inverted input pulses now is centered in the low portion of the separated clock, a string of pulses every 8usec, indicating zeros, clocks zeros through this flip-flop. If a pulse follows

another by 4usec, indicating a one, the separated clock is still high and a one gets clocked through the flip-flop. The separated clock and data then go to their respective pins on the 610 board.

Calibrating the separator is very much the same as calibrating the 610 board. Connect the separator's input to pin 9 of J3 on the 610 board and connect a scope to the separated clock output line. Adjust the trimmer for a 6usec positive-going pulse.

Now that we have a data separator, the rest is easy, just a simple matter of making a cable to connect the 610 board's J3 with the disk drive's 34 pin edge connector. Your local OSI dealer can probably supply you with a Molex connector to plug onto J3. A SASI-compatible 34 pin edge connector can be had at your local Radio Shack (part #276-1564) for about \$5 each. In addition, you will also need a suitable length of 24 conductor ribbon cable, preferably shielded if the cable will be long.

Figure 3 shows the proper connections for two single-sided drives (TM100-1, SA400 or similar). If you just wish to use one drive, ignore the connections for the second drive. Figure 4 shows the connections for a single double-sided drive (TM100-2, SA450 or similar). The switch shown in the connections turns the motor(s) on when closed. The drives have a socket for jumpers, which may or may not have a shunt dip or dip switch installed. There should be a jumper in the proper drive select position. If you're using an SA4X0, a jumper in the MH position (marked on the PC board) will cause the head to be loaded when the motor is turned on via the switch. Due to its advanced head design, the TM100 head is ALWAYS loaded when the door is closed and there is no provision for raising it. All other shunts should be open.

The last consideration is power. The TM100-1 and SA400 both require +5v at about 600mA and +12v at about 900mA. The power connection is made via a 4 pin polarized Molex connector. With a little work, the Radio Shack #274-234 (\$1.09) may be pressed into service. The pins are a little large and need to be crimped a little. You will also need to shave a corner off. The connections are shown in fig. 5.

HOW TO BUILD A DATA SEPARATOR by JIM McCONKEY

FIGURE 1.
DATA SEPARATOR SCHEMATIC

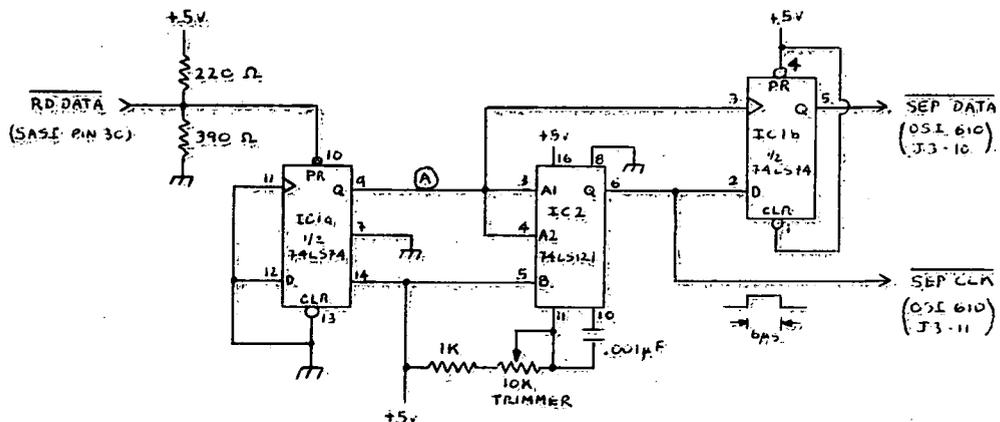


FIGURE 2.
DATA SEPARATOR TIMING

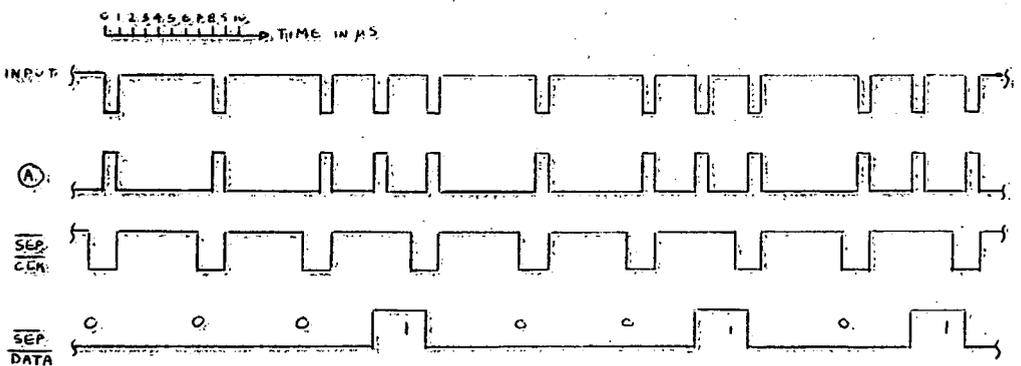
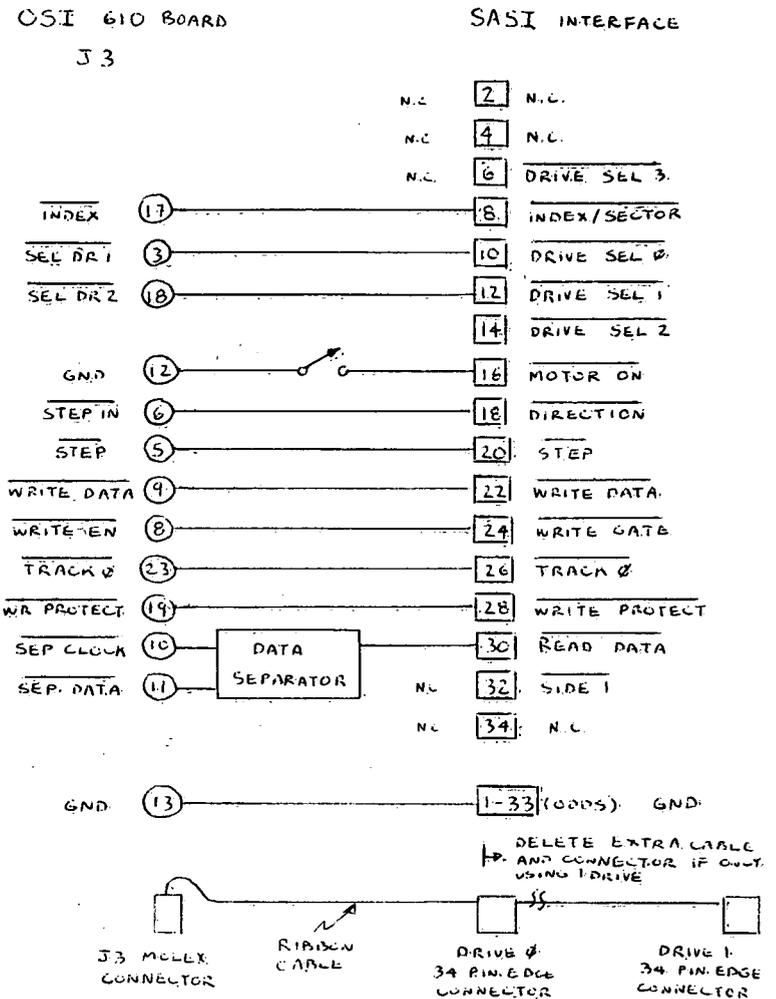


FIGURE 3.
SINGLE SIDED DRIVE CONNECTIONS



Continued on page 18



3 USERS-80 Mega Bytes — \$8990⁰⁰*

INTRODUCTORY SPECIAL

**WITH DUAL FLOPPIES
BRAND NEW —
1 YEAR WARRANTY ON HARD DISK!
REGULAR \$10,990.⁰⁰**

- 90 Days on Power Supply, Floppy Drives — Circuit Boards.
- Configured for Time-Share @ 2 MHZ
- Includes: 2 Serial Printer Ports with Handshake, Improved Cooling, and Ball Bearing Roller Chassis Rails

**ALSO AVAILABLE WITH
3 MULTI-PROCESSOR**

Denver Boards with 64K each user and
Centronics Parallel Printer Port at
\$9990.⁰⁰

*DEALER DISCOUNTS AVAILABLE

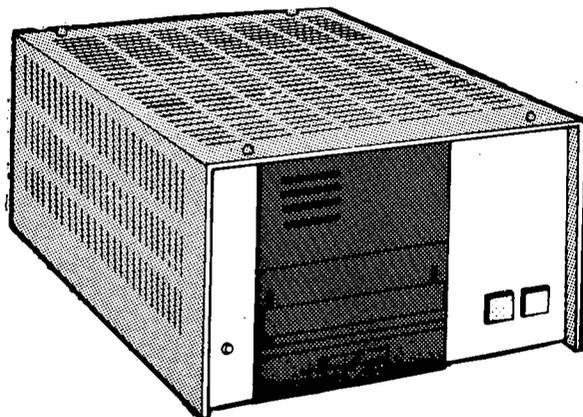
**8" HARD
DISK SYSTEMS**

SINGLE BOX TABLE TOP WITH IMPROVED COOLING 10 M/B HARD DISK
AND 8" FLOPPY DISK 2 USERS AND 2 SERIAL PRINTER PORTS
\$5990.⁰⁰

AS ABOVE WITH 2 MULTI-PROCESSOR 64K DENVER BOARDS
PLUS CENTRONIC PARALLEL INTERFACE **\$6990.⁰⁰**

OR INSTALLED IN CABINET AS ABOVE
WITH DUAL FLOPPIES PLUS 10 M/B.

| | | |
|--------------------------------|--|-----------------------------|
| STD. BOARD TYPE | 1 USER w/ Centronics Printer Port | \$6490.⁰⁰ |
| | 2 USER w/ 2 Serial Printer Ports | \$6990.⁰⁰ |
| DBI MULTI PROC. | 2 USER w/Centronics Printer Port | \$7790.⁰⁰ |
| | 3 USER w/Centronics & Serial Printer Ports | \$8990.⁰⁰ |



**MULTI-PROCESSOR
DEVELOPMENT SYSTEM
SPECIAL**

- 5 M/B Hard Disk-1 8" Floppy
- 1 Centronics Parallel Printer Port
- 1 Serial Printer Port, 1 Modem Port
- 2 DB-1 Multi-Processors
- Complete Programmer Manual and Software Overlays

**SPECIAL
ONLY \$5990.⁰⁰**

CLOSE OUTS

C4P—\$350.⁰⁰, C4PMF—\$699.⁰⁰, C4P DMF 48K—\$1199.⁰⁰, C8P-DF—\$1499.⁰⁰ C2-OEM—\$1799.⁰⁰, CM-2—\$69.⁰⁰, CM-10—\$89.⁰⁰, CM-11—\$499.⁰⁰, CA-9—\$129.⁰⁰, CA-10-1—\$149.⁰⁰, 510 CPU—\$299.⁰⁰, OSI C4P Disk Programs, Regular \$29-\$49 — NOW \$9.⁰⁰ each.

Call or Write for Bargains List!

WHERE WE STILL LOVE OS-65U — AND SUPPORT IT!

Space-Com International

22991 LA CADENA DRIVE, LAGUNA HILLS, CALIFORNIA 92653

ORDER TODAY

(714) **951-4648**

SOME QUANTITIES LIMITED

FIGURE 4.
DOUBLE SIDED DRIVE CONNECTIONS

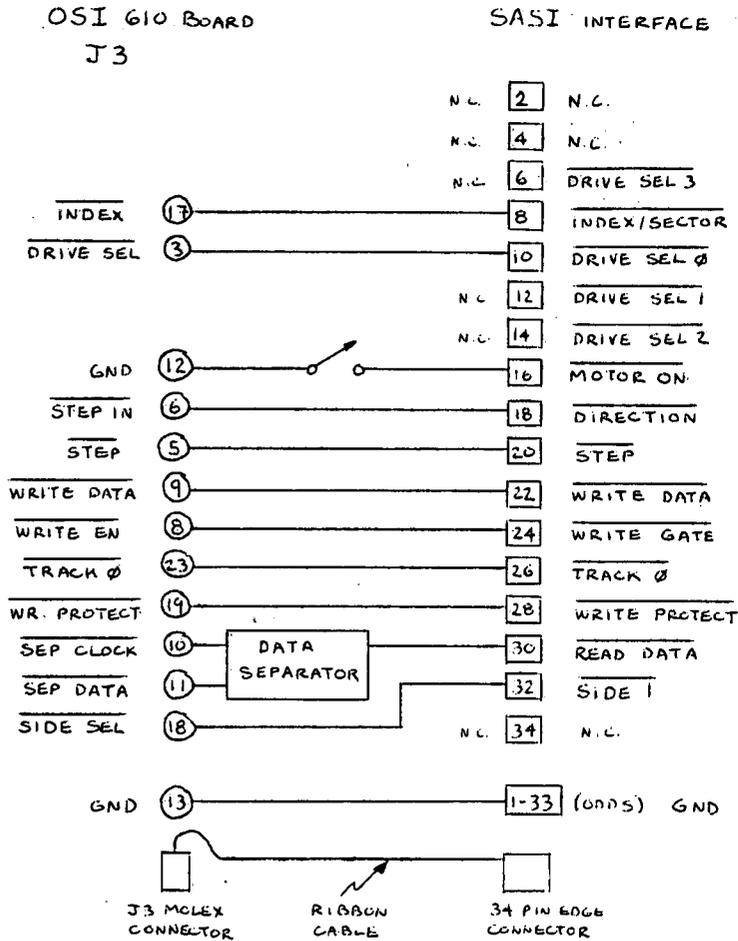
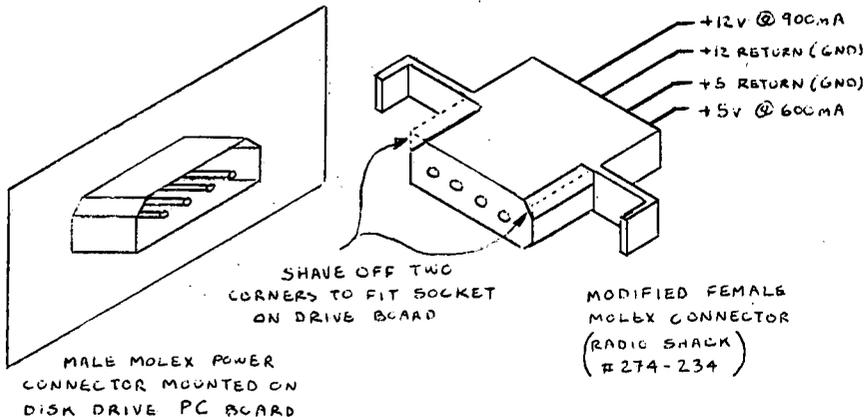


FIGURE 5.
POWER CONNECTIONS



That's it. Don't worry about all the unused pins on J3. The disk boot routine ignores them and I assume OS-65D does also. I have been using HEXDOS and it also seems to ignore the unused lines. You don't have to worry about the unused SASI lines either. They are pulled up on the

drive board. Mount the data separator wherever convenient.

Hopefully, I have saved several people from pulling out their hair over trying to use Tandon or Shugart drives with a C1P or just trying to find a data separator.



OSI-FORTH

OSI-FORTH 3.0 is a full implementation of the FORTH Interest Group FORTH, for disk-based OSI systems (C1, C2, C3, C4, C8) Running under OS65D3, it includes a resident text editor and 6502 assembler. Over 150 pages of documentation and a handy reference card are provided. Requires 24K (20K C1P). Eight-inch or mini disk \$79.95. Manual only, \$9.95. "OSI-FORTH Letters" software support newsletter \$4.00/year.

Other Software for
Ohio Scientific Computers:

VIDEO EDITOR

Video Editor is a powerful full screen editor for disk-based C2, C4, C8 systems with the polled keyboard and color video boards (b&w monitor ok). Allows full cursor-control with insertion, deletion and duplication of source for BASIC or OSI's Assembler/Editor. Unlike versions written in BASIC, this machine-code editor is co-resident with BASIC (or the Assembler), autoloading into the highest three pages of RAM upon boot. Video Editor also provides single-keystroke control of sound, screen format, color and background color. Eight-inch or mini disk: \$14.95. Specify amount of RAM.

SOFT FRONT PANEL

Soft Front Panel is a software single-stepper, slow-stepper and debugger-emulator that permits easy development of 6502 machine code. SFP is a fantastic monitor, simultaneously displaying all registers, flags, the stack and more. Address traps, opcode traps, traps on memory content and on port and stack activity are all supported. This is for disk systems with polled keyboard and color (b&w monitor ok). Uses sound and color capabilities of OSI C2/C4/C8 systems (not for C1P). Eight-inch or mini disk \$24.95. Specify amount of RAM. Manual only, \$4.95 (May be later credited toward software purchase). Six page brochure available free upon request.

TERMINAL CONTROL PROGRAM

OSI-TCP is a sophisticated Terminal Control Program for editing OS-65D3 files, and for uploading and downloading these files to other computers through the CPU board's serial port on OSI C2, C4, and C8 disk-based systems with polled keyboards. Thirteen editor commands allow full editing of files, including commands for sending any text out the terminal port and saving whatever text comes back. INDUTL utility included for converting between BASIC source and TCP file text. Eight-inch or mini disk \$39.95. Manual only, \$2.95.

WRITE FOR FREE CATALOG!

Prices shown are postpaid.

Specify computer model & RAM.

NEW ADDRESS

Technical Products Company

P.O. BOX 9053

Boone, NC 28608

Continued from page 8

(with the CTRL key depressed press the X key) and you will see the program in the indirect file loading just as if you typed it in from the keyboard, but a lot faster I'll bet. Please note that any line numbers that are the same will be replaced by the line from the indirect file program, so if you wish to merge 2 programs that may overlap with line numbers, you should first renumber one of them.

14. Save or run the program.

LETTERS

ED:

I have an earlier version of WP6502 (copyright 1979, serial #815522) that I have been happy with, but with some reservations. If I updated to the latest version I would want to be sure it didn't have the same problems (of course, mine was only \$75). I recall the disclaimer in the manual that said it will work only with the standard out-of-the-box variety machine. But that's exactly what I have: a 32K C8P-DF, with a C.Itoh 8510a matrix printer. The only modifications I've made have been to hardware to permit direct audio and video connection to my 12" Sony, including a new power switch with an extra pole that permits me to ground the controlling relay in the Sony. That way, I can leave the computer connected to the television. If the computer is off, I get regular TV. If the computer is on, I have a monitor. Anyway, since my modifications had been limited to hardware, and not the 65D operating system, I was a little annoyed with the bugs that appeared.

The most spectacular of these bugs was the tendency on occasion to write continuous garbage over and over on the same line, with no possible recovery. This could happen whether or not I was outputting to the screen or to the printer. I could sometimes fix this by rearranging some of the text or imbedded commands, but never to the point where I could spot a pattern.

The other problems were easier. It took some experimentation to discover that it takes two 'y's to activate the

hold-at-the-top-of-a-page command. Nothing in the manual suggested that. In fact, the manual says quite the opposite. That, if I want to hold the printer, I should simply press return. If not, I should press N.

Aside from the above, which I've had to live with, I've really enjoyed using WP6502 for correspondence. But of course, I'd like to know if there are any fixes for the garbage out problem. It seems unlikely I'm the only one who's had the experience.

I've enjoyed PEEK(65) and look forward to receiving it. Keep up the good work. Us OSiers need all the help we can get.

Don L. Heimbach
Fullerton, CA 92633

* * * * *

ED:

My hardware consists of OSI C4P w/48K, Dual 5" Disks and Epson MX-80 Dot-Matrix printer. My software consists of OS65D3 Version 3.2 (I don't like V3.3), word processor WP6502 V1.2 (I don't like V1.3), DQ Secretary and DQ Justify.

I have learned to get a great deal of service from my system. The DQ Secretary is particularly convenient for commanding the disks and loading and saving programs and text.

I have made two hardware modifications. I have installed a Disk Switch (DSK-SW) from D&N Micro Products, Inc., and I have installed a SYNKEY ROM from Micro-Interface. Both are installed inside my computer and out of sight. The DSK-SW turns off the disk drives except when I am reading or writing from or to a disk. This saves wear on the disks and drives. The SYNKEY ROM normalizes my keyboard, so I can write conveniently with lower case characters.

Now I can keep a disk in each of my drives A & B, and I only need to have the operating system, OS65D3, on the "A" disk for "booting up". This leaves me tracks 01 to 11 free for programs and text storage on the "B" disk. (Of course, I must maintain track 12 for the disk directory.) This is all very convenient. For my routine work I can leave my disks in the drives continuously.

I have only one problem that

v3.3 TEXT PROCESSING

User friendliness is the key feature of this OS65D v3.3 text processing system--so simple, complete training takes less than two hours! FEATURES INCLUDE:

- Line Orientation
- Insert, Delete, Replace, Move and Swap Editing
- Right Justification on demand
- Auto Centering
- Document Preparation with
- Auto Numbering and Paging
- Unique 'Progressive Merge' Block Manipulation
- Easy to read manual
- Plus more!

\$49⁹⁵

Manual only (Applies towards purchase) \$10.00
C2-8", C4-5 1/4". Video v3.3 preferred. Serial and v3.2 versions available (please specify).
Check or Money Order accepted and satisfaction guaranteed. Postage included.
Authors phone number is included for support.

MMSOFT

1100 W. HIWAY 40
VERNAL, UTAH 84078
(801) 789-0525 ask for Mark

OSI Software House is selling the following equipment:

OSI C3A Standing Processor
Hazeltime 1420 Terminal

The following newly developed software:

- Inventory
- Purchasing Payroll
- Building Materials
- Quotation
- Accounts Receivable
- Estimation
- Accounts Payable
- Education
- General Ledger

For more information, contact
Michael Guidry at (318) 988-1300
during office hours.

bugs me. So far, I can't use track 00 on the "B" disk! The track 00 doesn't have an address or leader or whatever is required for the disk operating system to recognize it. I have "created" an entry in the directory for it. I call it TRACKO, and it is printed whenever I call for the directory. The system seems to respond satisfactorily when I command SAVE, TRACKO, but when I command LOAD, TRACKO, the DQ-Secretary responds "NOT FOUND".

Now my question is: Is there some way to write a header or initialize track 00, so that I can read data from it?

Carl M. King
Sarasota, FL 33579

CARL:

As far as I know there is no way to initialize track 00 so that data can be read from it. Maybe a reader can help!

Dick McGuire

* * * * *

ED:

We are relatively new on the Challenger II, and we are still in the save by tape stage.

I am trying to do a program of accounts maintenance for tape. I know this is a relatively slow method using tape. However, I believe in starting at the ground level and working up to disk.

My problems are, of course, the GC problem, and therefore, I am interested in the different poke routines to change addresses to fix the problems encountered.

I hope to expand my machine using the SEB 3S4 by Orion Software, but I am a little skeptical in that I won't be able to use software I already have, or does this make any difference?

Donald W. Leith
Hawthorne, WI 54842

Donald:

Account maintenance with a cassette based system?! I am frequently amazed at what people get their computers to do!

Who has experience with the Orion SEB 3S4?

Al.

* * * * *

ED:

I recently purchased a copy of OS65D 3.3 and ran into the same problem described by Tim Lowe in the January issue. That is, that the modem routine just doesn't work on version 3.3. If version 3.2 or earlier (3.2 is supplied on tutorial disk 2) is booted you can load and run this program just fine. Incidentally, the section from line 500 - 990 just sets up the 48 characters line, so if you've already got that, deleting these lines and changing 3000 to a Rem will make it leave the display alone.

Brick Rule
Sarasota, FL 33582

* * * * *

ED:

I would like to know the location of the subroutine which monitors the keyboard in V3.3 and the location of the value of the key depressed in memory. In V3.2 the subroutine begins at 252B and the memory location of the ASCII value is 9815.

A. J. Smith
Amherst, OH 44001

* * * * *

OSI LIVES!

and gets FULL SUPPORT at Community Computers

Keywriter - New Word Processor

Compatible with Single User, Multi-User and Network Systems!

Keywriter incorporates standard commands with powerful features like:

- Mail Merge, DMS Compatible
- Menu Driven
- Full Screen Editing • User Friendly
- On Screen Help and Prompts and Formatting
- Linked Print-out of up to Nine Files
- Compatible with latest OS-65U Version
- Requires 8" Floppy or Hard Disk System

Keywriter offers a true full screen editor, with four way cursor control at all times.

Keywriter documentation includes a 60 page Self Teaching Manual. **\$300**

Compiler for 65U

A true native code compiler. Supports all OS-65U features, except common variables. 2-10x improvement in speed. Compatible with latest version of OS-65U. **\$395**

Editor-ROM

Most powerful Editor-ROM available for OSI machines. Full four way cursor movement; windows; keystroke control of special features. Also has communications software for level 1 multi-station systems.

For all C1P, C2, C4, C8P Basic-in-ROM systems, except 400 and 500 Rev A, B, C, CPU's. Requires some cuts and jumpers **\$30**

- Full Support for OSI
- Custom Hardware & Software
- Service Contracts Available

Cluster System Software

Connect up to 16, or more, C1, C2, C4, or C8 systems to any OSI 8" floppy system. Fast, simple disk/printer share system.

Ideal for schools. **\$500**

DMS-X

DMS compatible database management system with full screen file editor; definable reports with specifications editing; powerful report formatter; fast machine code keyfile sort; flexible create and recreate utilities; more.

System is fully driven menu.

\$300 + DMS license

OSI / IBM Double Density Floppy Controller

- Replaces 470 board
- Fully compatible with OSI format and IBM single density format.
- Double density, too. Up to 2.4 meg storage on standard floppy drives.
- 5 1/4" Drive capability, software selectable.
- Phase-locked loop insures data integrity.
- Special introductory price. **\$500**



Community Computers

Since 1977

(703) 527-4600
2704 N. Pershing Dr.
Arlington, Va 22201

Dealer Inquiries Invited

ED:

Here is a generalized file sort routine for OS65-D for MDMS data files. The sort is entirely done in memory and the size of the file is only limited by memory and disk space. The sort is based on the shell algorithm and is of order N. LOG₂ N.

OS65-D MDMS

```
10 REM FISORT - WRITTEN BY DAVID W. HANABURGH
20 REM YALE CORDAGE
30 REM YARMOUTH, ME 04096
70 POKE173,96:POKE2893,28:POKE2888,0:POKE8722,0
80 POKE2972,13:POKE2976,13
90 DEFFNX(X)=VAL(EN$(X))
100 DEFFNA(X)=X*NB+1
110 PRINT"THIS ROUTINE SORTS ANY FILE ON ANY SPECIFIED FIELD
120 PRINT"THE SORT KEY MUST BE A NUMERIC STRING AND THE FILE
130 PRINT"STRUCTURE MUST BE AN MDMS DATA FILE
140 FORI=OTO6:PRINT:NEXT
150 INPUT"FILE TO BE SORTED";F$
160 INPUT"SORTED ON WHICH FIELD";FD
170 INPUT"IGNORE O'S";AN$
180 IFLEFT$(AN$,1)="Y"THENSK=-1
200 PRINT:PRINT:PRINT"PLACE DISK WITH FILE IN DRIVE A - PLACE
BACKUP IN DRIVE B
210 PRINT"SORTED FILE WILL BE ON DRIVE B
220 INPUT"PRESS ENTER";AN$
230 GOSUB1000:REM OPEN FILE
240 DIMKY(EN-1),IT(EN-1),EN$(NF)
250 FORI=1TOEN-1
260 IT(I)=I
270 DISK GET,FNA(I)
275 GOSUB2100:REM READ RECORD
280 KY(I)=FNX(FD)
290 NEXT
300 GOSUB3000
305 K=1
310 FORI=1TOEN-1
320 IFSKANDKY(I)=OTHENNEXT
330 DISK GET,FNA(IT(I))
340 GOSUB2100:REM READ RECORD
350 DISK!"SE B
360 DISK GET,FNA(K)
370 GOSUB2200:REM WRITE RECORD
380 K=K+1
390 DISK!"SE A
400 NEXT
410 IFNOTSKTHENEND
420 GOSUB1000
430 PRINT#6,FIS$:PRINT#6,NB:PRINT#6,NF
440 PRINT#6,PH:PRINT#6,K
450 DISK PUT
999 END
1000 REM OPEN FILE - GET PARAMETERS
1010 DISK!"SE A
1015 DISK OPEN,6,F$
1030 POKE12076,6:POKE12042,32
1040 INPUT#6,FIS$:INPUT#6,NB:INPUT#6,NF
1050 INPUT#6,PH:INPUT#6,EN
1060 RETURN
2100 REM READ RECORD
2110 FORJ=1TONF
2120 INPUT#6,EN$(J)
2130 NEXT
2140 RETURN
2200 REM WRITE RECORD
2210 FORJ=1TONF
2220 PRINT#6,EN$(J)
2230 NEXT
2240 DISK PUT
2250 RETURN
3000 REM SORT ROUTINE
3010 GAP=EN
3020 IFGAP<=1THENRETURN
3030 GAP=INT(GAP/2)
3040 PRINT"SORTING
3045 RX=EN-1
3050 HE=RX-GAP:SW=-1
3070 FORI=OTOHE
3080 PT=I+GAP
```

3090 IFKY(PT)<KY(I)THENGOSUB
3500

```
3110 NEXT
3120 IFSWTHEN3020
3130 GOTO3050
3500 A=KY(I):KY(I)=KY(PT):
KY(PT)=A
3510 A=IT(I):IT(I)=IT(PT):
IT(PT)=A
2520 SW=0
3530 RETURN
```

David W. Hanaburgh
Yarmouth, ME 04096

* * * * *

ED:

About one year ago, I wrote to PEEK(65) asking a few questions. I was hoping to get a reply by mail, because I had not yet subscribed. A reply was never received! None of my questions were answered.

About a month ago came a flyer advertising PEEK(65). A check for a subscription and all the back issues went out. Only five days later, all of what I ordered was at my door. Great service!

It took me a while to read through a number of them, but I finally came to the May 1982 issue. In one of the letters to the editor, someone described a system exactly like mine! Upon reading on, I must have turned twelve shades of red. It was my above mentioned letter that I thought I never got a reply to.

It's been a while since that first letter, and a lot of learning later. The Aardvark Journal was a lot of help, but since it's demise, PEEK(65) will now be my main source of OSI information. In the May 1982 issue, I mentioned that I may be able to contribute to your (our) magazine. The following information should be of some help.

In the March 1983 PEEK(65), readers needed help using Radio Shack printers. I bought (and regret) a R.S. DMP-100 on sale. I had the same trouble as other OSI users had with double spaced lines. The cure when using OS65D3.0 & OS65D3.3 was to modify the machine code I/O routine. The program included will do this.

It's been about six months since I wrote it, and my memory of it will not allow a lot of detail. Essentially, I interrupted the cassette/printer I/O routine, and jumped to this little routine placed in some free space just

in front of the I/O routine. When the machine sends a character to the I/O routine, it compares what is sent to a hex \$0A (LF). The machine doesn't send a character if a LF is encountered but instead it now performs a RTS from the I/O routine. The computer will continue on and then send another character to the I/O routine and the whole above mentioned sequence will repeat itself.

The program has worked for quite a while and I haven't had any problem with it. Others can write to me and let me know how it has worked for them. I can see it now, a new EPROM that will do this automatically. Naa.

My present project is trying to interface my Heathkit H-89 to my ClPMF as a terminal. My ClP will then have a 80 x 24 character display! Wow! When life exists, the possibilities are endless...

```
10 REM PROGRAM TO REMOVE
   LINEFEEDS FROM I/O
   ROUTINE OF
20 REM OS65D3.3 OR 3.0
30 REM BY DAVID L. KUHN
40 REM 109 SHAW AVENUE
50 REM LEWISTOWN, PA. 17044
60 REM
70 FOR X=9394 TO 9404
80 READ A
```

```
90 POKE X,A
100 NEXT
110 FOR X=9430 TO 9432
120 READ A
130 POKE X,A
140 NEXT:END
150 DATA 201,10
160 DATA 208,1,96
170 DATA 141,01,240
180 DATA 76,217,36
190 DATA 76,178,36
```

David L. Kuhn
Lewistown, PA 17044

ED:

Our company is a structural steel and miscellaneous iron contractor in New Jersey and we own both a C2 and C3 OEM, which we use to prepare invoices, estimate, and do project bookkeeping.

We would be highly interested in corresponding with any general contractor or subcontractor currently using OS65U for business use.

Martin King
3-25 Dorothy St.
Fair Lawn, NJ 07410

Martin:

Why not write an article about some of those things and send

it to Peek?

Al

ED:

I had the same problem as Tim Lowe had in the January issue. When I called CompuServe I got back a bunch of letters and symbols. The symbol would be that of the letters' ASCII number plus 128. For example, a space (32) would come back as a filled in square (161). Maybe the machine code people can make something of that.

My system is a SBII series 2, 32K, 5 1/4 Disk, and a Radio Shack Modem I. Way back in PEEK, I found a simple terminal program for cassette and that would work fine. Then, by accident, I discovered I could use OS65D3.2 and make the OSI Modem program work. 3.3 wouldn't work.

When I phone, both CIS and OSI said the other was at fault and needed to change their system.

If you use the 12 x 48 screen, you need to change line 60 and add 62:

```
60 DISK!''CA 25A0=11,1'':
   POKE55296,1:FORI=1TO32:PRINT:
   NEXT
```

OSI Disk Users

Double your disk storage capacity Without adding disk drives

Now you can more than double your usable floppy disk storage capacity—for a fraction of the cost of additional disk drives. Modular Systems' DiskDoubler™ is a double-density adapter that doubles the storage capacity of each disk track. The DiskDoubler plugs directly into an OSI disk interface board. No changes to hardware or software are required.

The DiskDoubler increases total disk space under OS-65U to 550K; under OS-

65D to 473K for 8-inch floppies, to 163K for mini-floppies. With the DiskDoubler, each drive does the work of two. You can have more and larger programs, related files, and disk utilities on the same disk—for easier operation without constant disk changes.

Your OSI system is an investment in computing power. Get the full value from the disk hardware and software that you already own. Just write to us, and we'll send you the full story on the DiskDoubler, along with the rest of our growing family of products for OSI disk systems.

Modular Systems

Post Office Box 16 D
Oradell, NJ 07649.0016
Telephone 201 262.0093

™DiskDoubler is a trademark of Modular Systems.

62 PRINT'' MODEM IS READY''

I can't explain this, but maybe it will help someone until an explanation comes along.

Jack Vaughn
Beaumont, TX 77707

ED:

Help, please! Is there a feasible and economical way to increase the RAM on the Ohio Scientific C3-OEM? Mine is the optional 56K machine. Most of my applications use CP/M 2.2 which was configured at 49K by Lifeboat. I want to add a spreadsheet program. I tried CalcStar but it runs out of memory quickly. I tried ScratchPad which can store out-of-memory portions of the matrix to diskette, but my machine does not have enough memory to install the program.

I read that CP/M 3.0 is available for banked systems with greater than 64K and I assume it could be configured to handle a separate additional RAM, if there is a way to add it. Or am I barking up the wrong tree?

Mitchell McNabb
Pascagoula, MS 39567

MITCHELL:

An interesting problem which many of us have been fighting for some time. There are several solutions, none completely satisfactory:

The reason for the problem is that the old OSI I/O boards (470 disk controller, CA-10 serial I/O, etc.) were memory mapped into locations below 64K, so that those locations could not be used for true RAM. Lifeboat cleverly wrote their CP/M BIOS in a section of high memory not used by the boards, so that they could use the full 48K low RAM available. However, as you note, some programs such as spreadsheets really need more.

The radical solution is to buy a D&N-80 board (or a new OSI computer!) which uses the entire 64K, and also reads and writes standard CP/M 8" disk format.

In your particular case, load a COPY of your Scratchpad disk into your A drive, and type:

A>ERA SP.COM
A>REN SP.COM=SPSMALL.COM

"SPSMALL" is a special version

of scratchpad supplied for computers with memory shortage problems. This program will do virtually everything regular Scratchpad will do, but uses less RAM. I have seen it work fine on an OSI with 56K.

Al

ED:

There are a large number of micro-computers in use in New Zealand, in schools, homes and small businesses, with a substantial portion being OSI systems.

We have recently been asked for a word processing package for the ClPMF Series II system, which allows database word processing.

To date, we have been unable to locate such a software package despite keenly reading PEEK(65) and other micro-computer journals.

I would sincerely appreciate being contacted by any OSI user who knows of a word processing program with the capabilities, which we could buy.

R. I. McLean
P.O. BOX 492
Wellington, New Zealand

MR McLEAN:

It is my understanding that WP6502 will work on a ClPMF. However, I am not sure what you mean by "database word processing." If you mean merging of names and variables from a database into form letters created by a word processor, I believe WP6502 will do this as well, working with MDMS.

Readers, who knows whether it will for sure?

Al

WANTED

Regression package (or social sciences statistical package that includes a regression package) for use on the OSI OS-65U V1.3 (floppy disk) operating system. Contact Peek (65).

Wanted: A copy of Edward H. Carlson's book, "ALL ABOUT OSI BASIC IN ROM", second edition.

Please contact: Don Cwynar.
3900 Royena Avenue, Reading,
PA 19605.

USER GROUP NOTES:

Central Pennsylvania Ohio Scientific Users Group Forming.
Contact: Dave Fisher, 610 S. 20th St., Harrisburg, PA 17104 or call (717) 236-0479.

ADS

USED OSI - BUY SELL SERVICE.
C3-B 6K. Dale King, P. O. Box 5412, Arlington, TX 76011 (817) 265-3760.

FOR SALE: OSI 48K Challenger, C8PDF, Polled keyboard, Leedex Monitor, 65D 3.3, 65U V1.2, Manuals and more. \$2,000 or best offer. Gary Johnson, 421 First Street, Breckenridge, MI 48615, (517) 842-3478.

HELP ME

Forced to Sell at loss!!! Guaranteed Excellent Condition - C2-OEM-02 w/OS-65U, OS-65D, WP-1A, WP-1B, WP-2, Assembler, Editor, Extended Monitor, Smart-term modem program, Memory test package, 2 mhz 6502, 48k NEC memory, 3 ms 8" Siemens disk drives, which have never missed a lick!!! CA-10-2 board with printer and modem port. Plus every piece of OSI documentation. (50) floppy disks and 2+1 programs, games and utilities in my place. It got me past college, but now I've got to start making payments and live. Please help me out!!! I paid over \$4,100.00 for the complete system, but I've got to sell for at least \$1,600.00.***Soroc IQ 120-good condition - just scratched. w/C2-OEM-02 \$200.00, w/o \$425.00 *** Epson MX-80 w/graftrax like new w/C2-OEM-02 \$325.00, w/o \$375.00 *** Microbuffer 8k Serial for Epson \$105.00 *** Brand New Anchor Signalman modem w/adapter - never plugged in \$75.00 *** Entire package with C2-OEM-02, Soroc IQ 120, Epson MX-80 w/Graftrax, Microbuffer and Anchor Modem w/adapter for only \$2,305.00 *** You pay shipping and insurance. I'll pay off the bank. Keep Calling --- Bob Duffett, 110 North Woods Rd, Watkinsville, GA 30677, (404) 549-7343 (404) 769-7689.

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:

76:8

GOODIES for OSI Users!

PEEK (65)
The Unofficial OSI Users Journal

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

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

Cash enclosed Master Charge VISA
Account No. _____ Expiration Date _____
Signature _____
Name _____
Street _____
City _____ State _____ Zip _____

TOTAL \$ _____
MD Residents add 5% Tax \$ _____
C.O.D. orders add \$1.65 \$ _____
Postage & Handling \$ 3.50
TOTAL DUE \$ _____
POSTAGE MAY VARY FOR OVERSEAS