PEEK (65)

The Unofficial OSI Users Journal

$1.75

DECEMBER 1984
VOL.6, NO.12

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

INSIDE

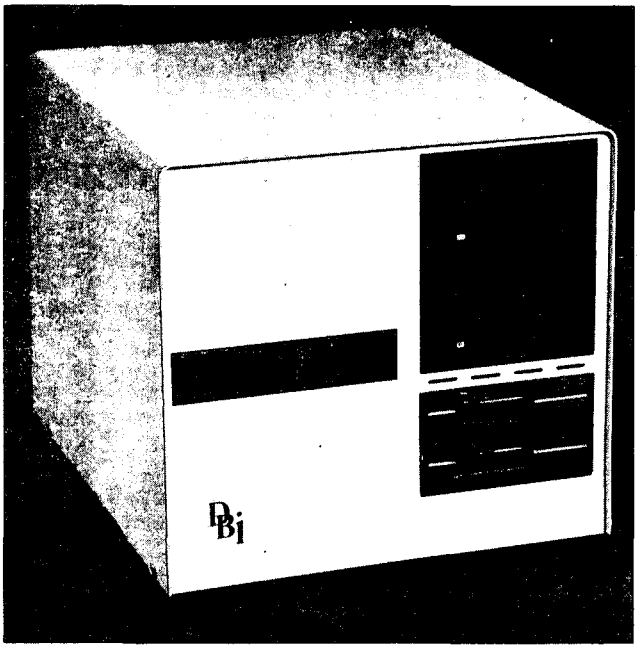
Column One

The biggest single piece of news should be obvious! But it is not green! That's just our Christmas touch. The machine is gray and white. As promised last month, you will find that we have given the DBI machine a pretty thorough going-over. Even so, I am sure that you will still have other questions. Just send them our way!

The other promised article involves the OSI 235 machine. It is not as lengthy an article, but then you are familiar with the 230 box, OS-65U, etc. What has changed is the hardware. With OSI's cooperation we do plan a future article on OSI, "Inception to Present" in a forthcoming issue. Certainly enough of you have asked for it.

On the OSI subject, you will note in the 235 article on page 17, that Gander Software's "The Data System" now comes with OSI's WORKSYSTEM machines or is available directly from Gander. We must have picked a good one to review in the August issue.

In your excitement to bone up on the new machines, don't forget to check in on the Assembly Language class for this month. There is even something for you experts in the Beginners Corner this time - binary searching - and "Wazzat" should make you think again about your input routines.



Finally, I want to remind you to fill in and send us the Reader Survey in last month's issue. We want to make some New Years resolutions and we need your INPUT. When it is all in, we will publish the results. At that time, we will know how many of you have responded and submitted articles. So, send something in with the survey and be counted among the PEEK supporters.

Give PEEK(65) to a friend for Christmas! That way you will know where your copy is when you want it!

This, our fifth year of serving the community, has been a year of getting to know more of you and a year of rebuilding. ISOTRON, now joined by DBI in the manufacturing arena plus other upcoming gems, should provide for an exciting new year.

The rest of the PEEK crew joins me in sending you our best wishes for a joyous Christmas Season.

Eddie

MULTI-PROCESSING 6502 MACHINE LAUNCHED BY DBI

By: Editor

The idea of letting multiple users utilize one CPU was a neat and clever idea. If nothing else, it allowed a number of people to use a common set of resources like hard disks and printers. The problem is that businesses tend to grow and thus need additional terminals. As users are added, the work load on that lonely CPU gets heavier and heavier until users start complaining about the slowness of their screens.

That's exactly what happened in Denver to Chuck, the owner of a photo lab. He had purchased an OSI multi-user machine from Carl who worked for Tri-Com, the local OSI dealer. It had served well, until it suffered from the load. Chuck called Art in to see if he could program his way around the problem. Art went one better. He came up with a Star Network that ran at high speed with a series of peripheral processors for printers, etc. to ease the load on the 6-user system. A little while, and a lot of head scratching later, Art came up with what is now the DBI multi-processor system.

They showed it to Carl, he liked it and pretty soon there were orders - one of the first was a cash offer for 40 boards. That kind of offer is hard to refuse. So Art and wife built the first 50, by hand, in their basement. Before those were delivered, they had orders for 100 more. All that on a $200 investment back in October of 1982. Since then these DBI boards have been used to upgrade over 300 OSI Series 200 computers. Now that is the stuff American business is made of!

About a year ago, Mike and Pat, who both worked for Linatel/Allied, also in Denver, met the group while visiting the board manufacturer that they both were using. They liked what they saw and joined the group to add direction and systems work to the mix. Together, they decided to make the company something more than a board manufacturer, hence the new DBM-1 machine.

On the basis of the reputation of the three existing boards, it's no surprise that they decided to build their own box. Because they didn't want to enfringe on OSI (the whole thing never would have happened were it not for OSI) they got a license from Microsoft and proceeded to create their DB-DOS operating system. In structure, it is quite foreign to OS-U, but to those familiar with CP/M and/or Unix, it will be quite similar. Interestingly, it is still compatible with OS-U (no PEEKs, POKEs or ML). With DB-DOS you will, however, get a list of replacement code for the obvious and necessary, and a list of the new reserved words. Incidentally, most of the Flags are replaced with reserved words. That doesn't sound like much, but just add 45 more reserved words to OS-U's current 67 words. The object is to make it as "transparent" as possible. You might just want to run PASCAL, FORTH, or some other language as well. On top of that "queuing" or multi-tasking is possible! It is all still quite new, with the first Beta sights installed in October of this year.

At this point, the effects of using multi-processing in an environment with 6 or 8 users is pretty well defined. To put it another way, the slow downs are rare, if noticeable at all. The way they look at it is, in a 14 user office environment system, a user normally accesses the disk less than 25% of the time and rarely uses it for longer than 1.2 seconds at a time. At 60K bytes/sec, the disk transfer is no slouch. What has not been considered in the user world is what to do when we want more users. How many? While the practical limit for the DBI configuration is about 75 users, the Beta test sight is configured with 33 users. This, of course, means a network to allow processors to talk and data to be transferred. Although the release version of the network is not finished, it is nearing completion.

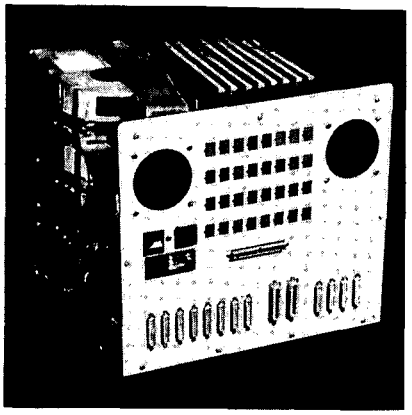
One of the nice things about the Denver boards (DB-1) is that it can be rebooted by the user. Not only does this mean that one user cannot lock up the entire machine, but because of the auto-boot capabilities, a user can be remotely booted. Although DBI does not directly support this, it does mean that a user could be dedicated to, say, despooling, and still be brought up and controlled by another user.

In reconstructing BASIC and the operating system they found a number of things that OSIers were not aware of in BASIC. One can only guess, that way back in the early days, some BASIC items would have been pitfalls and were thus eliminated. But they are all there in DB-DOS to give the user a more powerful language to work with. That in no way is to say anything bad about 65-U or Microsoft, -- they have both been "the best".

What else does this BASIC give you? In addition to Ext. Input, that you expected, there is PRINT USING, a 200 character input type-a-head buffer and more. If you are familiar with M-80 BASIC, you will have a good idea. Add to that a "real" clock calendar that is on board - leap year too.

Under the DB-DOS system, BASIC and the OP System are separate and are put together using the Unix concepts, which is to say that one boots into the DOS, adds shells for the I/O, language and then the program. The net result is that speed is about the same as OS-U even though some operations are faster. The plus is that many of the utilities are part of BASIC. In the DOS, there are eight 512 byte disk buffers that are dynamically allocated, thus allowing for byte level lock-outs. Let's look at it this way. If I am in a file, the beginning may be in one buffer, the middle in another and the end in a third, but I can look at any of the three without a disk read. User two comes along and wants the same buffer. The DOS says, he can have it, but not until I dump it to disk first. This operation is still controlled by semaphores, thus giving the programmer control. All in all, it is in disk access that DB-DOS gains its speed.

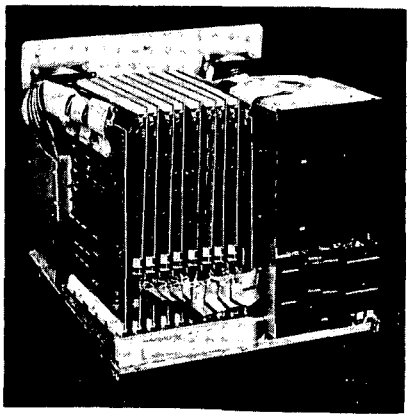
Why all the Unix type structure? There are several reasons. Transportability (though it is somewhat limited) is a programmer's delight. Utilities on line without program loss would be another. Then there is also the possibility of running some "other source" Unix programs too. But certainly these folks must have their eyes on the next generation machine too. Whatever the next generation machine will be, it will more than likely be of the 68000 family running something like Unix, and will definitely be a multi-processor machine and support the existing software. They put it another way, "Never walk away from a market and never stop supporting a market."



Of course, one of the major diversions from the OSI world is the SCSI buss. Amongst other things, this buss means that a whole raft of "off the shelf" disks and peripherals become available. One of the neatest things offered by DBI is their Bernoulli removable 10 MB floppy disk. Of special mention is the fact that the machine thinks of it as a regular hard disk. It just happens that you can change disks and thus use it for a swift back-up media at a reasonable price. The SCSI controller in each machine will support up to two floppies and two hard disks - that could be two 140 MB 5.25" disks. If that still is not enough, add up to 6 more controllers. By my math that's 1960 MB. One of the specialties of this arrangement is that if you run out of room on your first disk, you just add another and tell the machine that it is part of the first, and the files just stream on like one big disk. But you could still have separate devices. Thus, you see, the limit is more physical than machine bound. It is little known, but OS-U is capable of addressing up to 4.3 Giga-bytes.

The secret of DBI's process is in buss arbitration. In CP/M, the kernal always has a memory map to the disk, which requires a dedicated processor (master processor) which means that all of the users (slaves) must talk to the same master for I/O control. Therein lies the bottleneck. At DBI, each processor is complete, i.e., DOS and language. The problem is who gets to use the buss and how to pass semaphores and disk parameters to the other users. Fortunately, there are a couple of unused lines on the 48 pin buss that can be used to allow a user to talk to another user, or all users. Since the semaphore and disk allocation tables are located on each user's board, a user is able to tell the others what he is up to and where he is on the disk - yet he still thinks that he is the only guy out there. When he is finished, everyone knows where the disk was left. If you like, you can think of it as a master - master system. As to who gets the buss, there is a multi-level checking system that is simple, quick, and accurate. In actual fact, user #1 will probably get about a 5% priority over other users, but that is the limit of preference.

As the folks at DBI say, "Software sells machines." They have found lots, and a lot of it has already had the minor fixes made to it to run under DB-DOS, but they are not talking specifics as they are not planning to market it. That's where the dealers and authors get together with DBI as the match-maker. They do become involved with the utilities and programmers tools. Shortly, they will have the



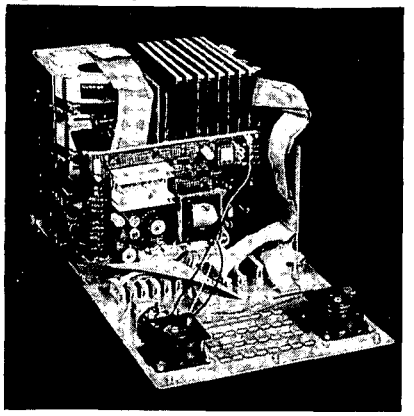
finishing touches on their "bundled" software which will include: word processing, data base, assembler, compiler, and a whole bunch of pure utilities. How about a COPY program that creates and copies from submit files and doesn't care if the disk is hard or floppy and a CREATE that knows if a disk is standard or back-up and can copy systems and initialize disks under multi-user and squashes the files back to minimum size? That is possible because, although disk allocation is dynamic, it can appear to the user to be sequential and has forward/backward linking.

A lot has been said about what can go on inside, but not a word about the box itself. Its diminutive size (13hx15x15) and whisper quietness make it, and the operators, comfortable in any office. In fact, it is so small, the whole thing is UPSable. That alone says something about service. To get eight users and 280 MB of hard disks in a box of this size obviously took planning, but the beauty is that, although there is no wasted space, it is neat and tidy and has a minimum of cables. In fact, when stripped of boards and disk, it looks like Mil. Spec., complete with gold plated pins.

One thing you may have noticed in the photos are the 5.25" floppies. Just remember that DEV A-D are not used (normally). That means that you still can stick in an OSI 470 8" floppy controller and drive and proceed to copy from DEV A-D to either DBI's 5.25" floppies or to the hard disks. That should solve any problem with moving your software. As for the other standard software that comes with the machine, it is all on their 5.25" floppies.

So there you have it. A new guy on the block - well not that new. A little box with lots of power and flexibility that is geared to the small and medium size business, with emphasis on the medium. And it is all backed by a devoted and tight knit team of originators who believe in both upward and downward compatibility. On top of that, they are downright anxious to work with their dealers; whether it be a software search or a translate program for an OS-U program. When it comes to the real nitty gritty, I think that the dealer will like the terms too, not to mention that the basic machine retails for $4995.

We at PEEK welcome the DBI gang on board along with all of their users. There is obviously much more to the total picture and we will do our best to keep you informed as the saga unfolds. Likewise, we hope that those of you with experience will share that knowledge with other PEEKers by keeping our mailbox full.



KPS BUSINESS SYSTEMS CONCLUSION

By: Russell D. Daugherty
P. O. Box 719
Parkersburg, WV 26101

Anyone starting to automate their business system should be aware of one important point. Management MUST COMPLETELY SUPPORT the program with 150% effort. Without TOTAL COMMITMENT their investment will be wasted. I heard this statement 15 years ago and I can now attest to its accuracy. People being human resist change, regardless of pre-planning and pre-education. We spent over a year talking, analyzing possible benefits and effort that would be required, but when reality struck, interest faded. When you impose substantial extra work on already busy people, the going gets tough. To get the tough going, hard nose single minded determination is required from someone and that must be top management. I am sure that if we had hesitated anywhere in the program, nothing would have been accomplished.

One of the greatest benefits, management information, is also the cause of resistance. When people realize that individual accountability, without emotional opinions, is now possible, they naturally become uneasy. While we have tried to assure everyone that the information is to be used for constructive purposes, sales improvements, inventory management, reduction of routine work, training, individual income and personal growth, resistance and distrust still exist. We have gone through useless exercises to convince a few that software is not the problem, garbage in will cause garbage out. Since we have been very careful in using individual information, we are beginning to see an attitude change and more effort to be accurate. People are beginning to notice that boring detail work has disappeared, outages are less frequent and special orders are received quicker. They are starting to use the information generated to improve sales and they are thinking of new ways to improve productivity.

Price changes are handled two ways. As described in invoice processing, when costs change, prices are automatically recalculated, edited and files updated. The second way is manual input when prices are changed with prior notice. This method requires substantial effort from an operator in a concentrated time period and is error prone. Since we use our own product numbering system (manfacturer's catalog numbers are not unique) we must first find our corresponding number which is not easy, fast or accurate by any method. "Search" takes time and is boring. Scanning an inventory list is just as bad when dozens of items have to be changed. After our numbers are located we enter the product number, bring up the proper fields and enter new cost. To reduce input, the first entry is assigned to all cost fields and new selling prices are calculated. Then we change only those that need to be changed. Selling price is edited to get rid of odd cents. Product number and new data is stored in a change file until the effective date. A report is printed showing old prices, new prices and quantity on hand at each location. Prior to the effective date, files are updated and new disks are distributed to each location. Prices for stock on hand and inventory records are updated. This seems easy, but invariably one or more items are wrong. The safety valve is restocking. When a new shipment arrives and our order cost does not match the invoice, trust the law of averages, it covers your mistakes.

We provided the capability of printing price tags but have not yet started this procedure. This should reduce errors by guaranteeing correct price and product number on the sticker, but of course, will not assure correct application.

Inventory list shows stock on hand at all locations. While list are printed only once per month, they have been a big help to branch stores locating low volume merchandise.

In summary, this business system has made us painfully aware of our past inefficiency and gives renewed optimism for the future. We will gain a competitive advantage in aggressive but profitabe price restructuring, inventory distribution and availability, targeting of advertising and promotions, improved productivity and general operating efficiency. Our skill requirements for labor will change. In the past, we had to put emphasis on general ability to do a lot of tasks well, using judgment. Now we can concentrate more on sales and number ability. In effect the labor pool has now increased in size and labor cost versus sales will improve.

Management time is now more productive. Time formerly spent on clerical work can now be channeled to supervision, training and planning.

We have been asked several times, knowing what we know now, would we tackle such a project again? The answer is, faced with the same conditions, YES. However, the conditions are not the same, KPS Business Systems has a computer system that exactly meets our specifications and we firmly believe it is on the leading edge of software technology. This system is a total package. It starts with point of sale and finishes with condition statements, never duplicating human effort. Before anyone starts they should be prepared to give up 4-5 years, working 80-90 hours per week.

Several more modules are planned to provide more analysis of our data base and more programs strictly peculiar to Photo Retail. We want to track processing orders, our customers and measure specific promotions. Then to completely automate we will track repairs, print invoices or sales tapes, collection letters and direct mail advertising to very specific customer's groups. Photo dealers who use this package will receive the advantage of preloaded files thereby substantially reducing overall cost.

On the technical side, we originally budgeted $5,000 per store for hardware which consisted of a double floppy drive computer, terminal and printer. We have since revised this budget to $7,500 to get more efficiency from a hard disk and added a Modem. Unfortunately, the US Mail can't deliver a first class package 100 miles overnight.

Our 80 MB disk is divided into three systems. The work system contains all programs and data files. The second system, called Backup, contains only data files. The third system, EOM backup, contains all files that are closed or zeroed at end of month. A modified copy program selects data or program files from the directory to upload.

We use DBI processor boards to

From Gander Software, Ltd.

The Ultimate Data Base Manager

THE DATA SYSTEM

AFTER 2½ YEARS OF DEVELOPMENT, THE MUCH RUMORED & SOUGHT DBM IS HERE!

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

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

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

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

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

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

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

THE DATA SYSTEM IS AVAILABLE TO DEALERS FROM ISOTRON, INC., OR GANDER SOFTWARE, LTD.

GANDER SOFTWARE, LTD.

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

"It Flies"

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

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

enable us to run more terminals than the OSI configuration would allow. Actually, we don't need more terminals, just more processors. We are currently using 2 terminals for sales, 2 term's for stock manipulation and ordering, and 1 for executive use. We also have several "slave" processors, without terminals, to do dedicated tasks. Thanks to DBI, we have auto-boot on all processor boards and depending on processor number, auto run to the normal function; that being master directory on the executive terminal, point of sale screen on sales terminals, inventory adjustment menu on warehouse screens, or a monitor and dispatch program on slave processors. A control program can be run on any terminal to assign a task to a slave processor. Since as a rule, the slaves do not have a terminal attached, the task must not need a terminal for input or display. Although this seems to limit their usefulness, all our software is written on the basis of providing error reports to be acted upon at a later time. Because of this, a terminal is not necessary most of the time. This method also allows someone other than a computer operator to have final approval of any exceptional conditions or notification of any corrections to be made.

One of the specific functions performed by slave processors is statement printing. Multiple locations can be run at the same time using slave processors and print spooling. The statement print updates files and assesses service charges as necessary. Although printing is the most time consuming part, at least the files are updated and the rest of the end-of-month processing can be started. Slave processors may also be used to run multiple printers if desired. Future projects include multiple processor sort utilities that will use any available slave processors in the system. The DBI multiprocessor system is not the total solution however. Much care must be taken in software design to keep one processor from interfering with another.

A key part of our data integrity plan is continual backup of hard disk data. We are currently running two complete systems: a C3-OEM and a 250-J. We have one slave processor on each system whose sole purpose is to transfer data between the two systems via a 500kbps. line. The software transfers the most volatile files on a continual basis during the day. The result is that even with massive computer failure (hard disk and processors) all that is required is to switch the cables to the backup computer, and the data is at most 20 min. old and in many cases less than 5 min. We have never had a failure of this magnitude but this also allows for minor problems as well. It is great to have a spare of everything on site.

Another valuable effect of auto-boot is the chance to use remote start-up of distant computers. With an auto-answer modem and a strict security system, it makes possible the updating of a remote store's files at night, when processing is done and phone rates are lower. Software patches, new programs and technical help are transmitted in this way to provide the most accurate support available. Service charge information is transmitted at end-of-month to insure service charges are assessed and credit risk flagged on the 1st of the month.

Numerous OSI utility programs have been changed to increase efficiency or run automatically for specific purposes. All application programs run compiled which gives a substantial edge in processing speed over IBM compatibles.

★

6502 ASSEMBLY LANGUAGE PROGRAMMING CLASS

Part VI

By: Richard L. Trethewey
Systems Operator of the
OSI SIG on CompuServe

In the last lesson, I presented a table of opcodes and the flags in the Status register that are affected by them. We have seen that the various conditional branching instructions make their decisions based upon these flags. One of the most common tasks facing the assembly language programmer is comparing two bytes and determining which is greater than or less than the other. The following table shows the result of a sample test of this sort (namely "CMP #$nn") and how to evaluate the results of this comparison:

ACC. vs	#$nn	If TRUE then...
ACC <	$nn	BCC will branch
ACC =	$nn	BEQ will branch BCS will branch
ACC >	$nn	BCS will branch
ACC <>	$nn	BNE will branch

As you can see, in some cases more than one branching instruction will be true for a given condition. Therefore, you must be careful about the order in which you insert the conditional branching instructions in your programs so that you eventually isolate the condition you are testing for.

So far in our discussion of the assembler, we have limited ourselves to the 6502 instruction set. Assemblers also allow us a specialized set of commands which aid in the development of programs. These commands are called "assembler directives". We have used two such directives already, namely the Origin command (i.e. *=$xxxx) and labels.

Both the OSI Assembler/Editor and my DEBUG Assembler allow the use of another type of directive that is used to designate memory locations to hold data. There are three such directives and they are:

.BYTE

.BYTE designates single memory locations as data and has the form:

```
label .BYTE operand(,operand..)
```

where the "operand(s)" hold an 8-bit value. Multiple values can be defined in a single statement by separating them with commas. Additionally, strings of ASCII text can be defined with the .BYTE directive by entering the string within single quotes as in:

```
label   .BYTE 'SAMPLE ASCII
        STRING'
```

.WORD

.WORD is similar to .BYTE, but .WORD designated two consecutive memory locations to hold a 16-bit value. .WORD has the form:

```
label  .WORD $D000
```

.WORD stores the 16-bit values in LSB/MSB order as is required by the indirect addressing mode. Thus, in the above example, the memory location at "label" would hold $00 and the location at "label+1" would hold $D0.

.DBYTE

.DBYTE is similar to .WORD in that it designates two consecutive memory locations to hold a 16-bit value, but .DBYTE stored the values in MSB/LSB order. Thus, in the example for .WORD, location "label" would hold $D0 and "label+1" would hold $00.

Many assemblers, like DEBUG, provide additional assembler directives. Consult the manual for the assembler that you use.

We are almost ready to begin programming. I think it is important to point out now that many of the essential routines (such as keyboard, display, and disk drivers) are provided by OS-65D. Indeed, not only are these routines provided for you, but most of them begin at the same address for all versions of that operating system. The file LABELS.65D in section 1 of the #SIG Access Reference Library holds the labels and addresses of the major routines in OS-65D V3.3. I will often use those labels in the examples to come, therefore, it is suggested that you download LABELS.65D and save it for reference. As these routines are used, I will attempt to document the function of each one.

When we began this series, we noted that the 6502 is an 8-bit microprocessor and is thus limited to dealing with values (numbers) between 0 and 255. Additionally, each of the memory locations available to us has the same limitation on the values they are able to hold. Since our programs must deal with the real world, we need to develop a strategy for handling values beyond this range. Fortunately, the 6502 provides us with the tools we need.

Let's take a look at the simplest of the mathematical functions, addition. For example, let's say we want to add 12 and 4. Within the 6502, this operation is handled as individual bits. Remember, we're really using base two mathematics, so that 0+0=0, 0+1=1, 1+1=10, as in:

```
      0 0 0 0 1 1 0 0     (12)
            +
      0 0 0 0 0 1 0 0     ( 4)
      -----------------   ----

      0 0 0 1 0 0 0 0      16
```

So far, so good. But 12 and 4 are very small values. Let's take a look at what happens when we try to add two larger values, such as 100 and 200:

```
      0 1 1 0 0 1 0 0     (100)
            +
      1 1 0 0 1 0 0 0     ($C8 or 200)
    ------------------
???   0 0 1 0 1 1 0 0  =  ($2C or 44????)
```

In the same way we say "carry the 1" when we're adding on paper in decimal, the 6502 uses the Carry flag to indicate an overflow here. This is where the meaning of "Add with Carry" for "ADC" comes from. When the 6502 executes the ADC instruction, the value of the Carry flag is added to the least significant bit of the result. In order to compute values over 255, we must use more than one byte to hold our totals. Let's reconstruct the addition of 100 and 200 in an actual assembly language program:

```
10  LDA #$00      ; INITIALIZE ACCUMULATOR
20  STA TOTAL     ; CLEAR TOTAL LSB
30  STA TOTAL+1   ; CLEAR TOTAL MSB
40  LDA #$64      ; LOAD 100 INTO THE ACC.
50  CLC           ; CLEAR THE CARRY FLAG!
60  ADC #$C8      ; ADD 200
70  STA TOTAL     ; SAVE THE LSB RESULT
80  LDA TOTAL+1   ; FETCH TOTAL MSB
90  ADC #$00      ; ADD ONLY THE CARRY (IF ANY)
100 STA TOTAL+1   ; SAVE RESULT
110 RTS           ; RETURN OR QUIT
120;
130 TOTAL .WORD $0000 ; 16-BIT TOTAL
```

After execution, "TOTAL" would hold the $2C result we calculated above, but the second and Most Significant Byte;

TOTAL+1, would hold $01 to denote 256. 256+44=300, and thus the correct total is preserved. The principle of multiple bytes can be extended to provide as great a range of values as needed. As long as the Carry flag was cleared when the addition was begun, and not cleared until the last addition is performed, the result will be correct if the Carry flag is clear.

Subtraction on the 6502 is handled slightly differently than addition. The principle of multiple bytes for values over 255 is still valid, but the Carry flag takes on a new role of a borrow marker. Before executing the SBC instruction, the Carry flag must be SET, rather than cleared, so that it is available when a larger byte is subtracted from a smaller byte. Should the Carry flag be clear when SBC is executed, the result is decremented. In this way, if the subtraction of the LSB of a 16-bit value (for example) requires a borrow (i.e. the partial result is negative) the Carry flag is used to obtain the borrow and the correct LSB result is preserved. However, this same operation clears the Carry flag so that the result of the subtraction of the MSBs is also corrected by decrementing it, to reflect the borrow.

Try to remember these two things when you begin to develop your own programs to handle addition and subtraction. First, before either operation is executed for the LSB, the Carry flag must be properly conditioned - Clear it for addition and Set it for subtraction. Second, in order to better visualize the operation, try writing it down on paper (use HEX notation so that you will better see the values of the individual bytes) as if you were adding or subtracting by hand.

We have only dealt with positive integral values here. There are several methods of handling negative values. For signed integral math, the method used is called "2's compliment". The books on 6502 Assembly Language programming cover this topic in depth and I suggest you look to those books for information in this area. It is important that you know about 2's Compliment math to become fully proficient in your programming, but my purpose here is to get you into programming your OSI system and so I will not be covering this topic.

★

BEGINNER'S CORNER

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

WILD CARDS, THE SEARCH IS ON!

The subject of 'searching' is vast and complex. Many techniques abound, from 'Binary Search' to 'Signature Screening' techniques. For small BASIC programs a simple, common sense approach will do.

Listing 1 is an excerpt from the 'Otaio Mailing List', (see June '84 issue). The program in Listing 1 illustrates how a search program can be written to be both fast and comprehensive.

This is what the program does. Imagine a list of 100 names, each name beginning with 'Mr'. It would be advantageous not to have to type 'Mr' every time a search is made. So when using the program to find the name 'Mr Nagear', it is only necessary to type the one word, 'Nagear'. But there is more. Why type the whole name? The search will be twice as fast if only 'Nag' is typed. There is a slight disadvantage to doing this though. The search will also find all other names beginning with 'Nag'. Okay, 'ear' is the solution to that problem.

A wild card search is for those occasions when the proper spelling is unknown. Was the name 'Ragear' or was it 'Bagear', or ....? No problem. Go for a wild card search and type '?agear'. The question mark is the wild card and stands in as any letter. If all six-letter names are wanted from the file, then just type six question marks thus, '??????'. Voila!

It would be wasteful to search every field of every record so line 610 produces the 'FIND' menu of fields from which a choice can be made. Records can be searched on any one of 'P' fields. Alternatively, a record can be accessed immediately by its record number. Variable 'M' contains the number of the field on which the file will be searched. The search always begins with the first record, so it does not matter if the records are not sorted. Wild card searches must be made right through the whole file, so starting with the first record is probably a good idea!

The string to be searched for, is input in line 690 - and the search is on!

In line 720, variable 'Q' holds the number of the current record to be accessed and 'M' is the field to be searched. Line 730 is the only complicated part. Imagine that T$="OP" and that Y$="HOP". The search is for "OP" in "HOP". The first comparison will be with "HO" in Y$, and the next, successful, comparison will be with "OP" in Y$. Two comparisons must be made. In fact, the number of comparisons made is always one more than the difference in length between the two strings T$ and Y$. (This is very useful if T$ and Y$ are the same length!) And that is what determines the value of R. What if 'R' is less than 1? In that case the FOR ... NEXT loop (in line 730) runs once and makes one unsuccessful comparison - there are more characters in T$ than in Y$! A good way to understand how the search works is to visualize two strips as in the diagram. The shorter strip moves to the right, relative to the longer strip. This continues until a match is found.

```
----
word
----
------------
verylongword
------------
```

If a wild card search is required, flag 'K' is set to '-1' in line 670. This flag is picked up in line 720 where a branch can be taken to the wild card code beginning in line 830.

It works like this. Imagine that the file is to be searched for the string 'Pterydactyl', and that there is uncertainty about its spelling. If this is the case, then substitute four wild cards and search for '????ydactyl'. Each time a question mark is found in 'T$', the code in lines 830-850 substitutes, in 'Y$', a question mark for a character. By this process the string 'Pterydactyl' (in Y$) would be converted to '????-ydactyl' (as in T$). Once the conversion has taken place the match to T$ can be easily made - T$ and Y$ are now both the same.

Records can be found by record number and this is begun in line 800. Line 810 is a dummy FOR ... NEXT loop - it does nothing but is required because of the 'NEXT Q' in line 790. Line 790 is also used by

BECOME A SUCCESSFUL

OHIO SCIENTIFIC VAR

WITH

POINT-OF-SALE SOFTWARE

COMPLETELY INTEGRATED SOFTWARE INCLUDES:

- POS INTERACTIVE WITH INVENTORY, A/R & GENERAL LEDGER
- CASH TICKETS OR COMPLETE INVOICES
- HANDLES CASH, CHARGES, LAY AWAYS, SPECIAL ORDERS & TRADE-INS
- PRICE LOOKUP, SALES ANALYSIS, PASSWORD PROTECTED
- COMPATIBLE WITH BAR CODE WAND

Now! You Can . . .

✓ **SELL TO THE NO. 1 VERTICAL MARKET - RETAIL BUSINESS**
Literally thousands of retail stores need inventory control with complete integrated accounting. Checkpoint-Of-Sale allows you to sell to the following retail stores: **HARDWARE, SPORTING GOODS, AUTO PARTS, SHOE, HOBBY AND TOY, FURNITURE, CLOTHING** and **GIFT SHOPS.** All these applications from the same software package.

✓ **UNDERSELL COMPETITION BY THOUSANDS OF DOLLARS**
Sell complete turnkey multi-terminal systems including hardware and software for under $15,000 and make over 50% profit. (Try that on an **IBM** [or compatible], **DEC, TRIAD** or **DATA GENERAL.**)

✓ **DELIVER PROVEN, TIME-TESTED SOFTWARE - FULLY SUPPORTED**
Your customer will be buying software that has been installed and running for 4 years, not just promises.

✓ **PROFIT BY SELLING THE SAME SYSTEM OVER AND OVER**
CHECKPOINT-OF-SALE is not customized, it's standard. However, if you choose to customize, the code's open.

✓ **PURCHASE UNLIMITED LICENSE FOR $6995 ($7995 VER. 3)**
License the software to as many end-users as you have time to sell.

DISTRIBUTORS: USE THE SOFTWARE FOR YOUR OWN BUSINESS AT NO EXTRA CHARGE.

FOR INFORMATION ON DISTRIBUTORSHIPS
WRITE OR CALL

CHECKPOINT OF SALE Software by

SILEO, INC.

381 SO. BROADWAY
DENVER, COLORADO 80209 (303) 777-3222

POS "ON LINE" COMPUTING POWER

```
560 REM FIND A RECORD
570 PRINT !(28):Q$="?":F=0:K=0:PRINT "Print Records to Device # ? ";
580 GOSUB 310: V=Y: IF V=0 THEN V=2
590 :
600 PRINT V: PRINT : PRINT "   FIND MENU": PRINT " -------": PRINT
610 FORC=1TOP+1:PRINT STR$(C)"> by "N$(C): NEXT C: PRINT " -> EXIT"
620 PRINT : PRINT "Choice ? ";: GOSUB 310: PRINT C$;: M=Y
630 IF Y$="-" THEN 190
640 IF M=P+1 THEN 800
650 IF M=0 OR M>P+1 THEN 570
660 :
670 PRINT"Wild Card Search ? ";:GOSUB310:PRINT C$;:IFA=121THEN K=-1
680 :
690 PRINT "Which  * "N$(M);: INPUT " ";T$: IF T$=H$ OR T$="" THEN 570
700 I=LEN(T$): PRINT !(28): PRINT TAB( 20)"* SEARCHING *"
710 :
720 FOR Q=1 TO Z: Y$=D$(Q,M): IF K THEN GOSUB 830
730 R=LEN(Y$)-I+1:FOR X=1TOR:IFT$=MID$(Y$,X,I)THEN X=R:NEXTX:GOTO 780
740 NEXT X
750 IFQ=Z AND F=0THEN PRINT:PRINTT$;"<< Not found Bub! >>":PRINT:PRINT
760 GOTO 790
770 :
780 PRINT : PRINT "Record "Q"of"Z: PRINT : GOSUB 1880: F=-1
790 NEXT Q:PRINT "Ready ? ";: GOSUB 310: GOTO 570
800 PRINT !(28): INPUT "* Record #";Q: IF Q<1 OR Q>Z THEN 800
810 FOR Q=Q TO Q: GOTO 780
820 :
830 B$="":R=LEN(Y$):FORY=1TOR:R$=MID$(T$,Y,1):IFR$=Q$THENR$=Q$:GOTO850
840 R$=MID$(Y$,Y,1)
850 B$=B$+R$: NEXT Y: Y$=B$: RETURN
OK
```

the general search code which DOES use a FOR ... NEXT loop. The dummy loop in line 810 is just a ploy to save duplicating lines 780-790 (without the 'NEXT Q'), for use by the 'search by record number' code.

POINTS ARISING

A flag is a marker and it is set either 'TRUE' or 'FALSE'. In OSI BASIC the mysteries of binary arithmetic dictate that the value '-1' means 'TRUE' and '0' means 'FALSE'. In fact, any non-zero value will be interpreted as 'TRUE'. The following two programs are equivalent.

```
10 K=-1
20 :
30 IF K=-1 THEN PRINT "FLAG IS ON"
40 PRINT
50 :
60 K=0
70 IF K=0 THEN PRINT "FLAG IS OFF DEAR"
```

And,

```
10 K$ = "TRUE"
20 :
30 IF K$="TRUE" THEN PRINT FLAG IS ON"
40 PRINT
50 :
60 K$="FALSE"
70 IF K$="FALSE"THEN PRINT "FLAG IS OFF"
```

In the first program change line 30 to:

```
30 IF K THEN PRINT"FLAG IS ON"
```

What does 'IF K' mean? BASIC interprets this as, 'if K is not equal to zero'. Is 'IF K$' equivalent to 'IF K'? Test this out in the second program in line 30. The following expressions are all equivalent to each other:

IF K, IF K<>0, IF K=-1.

Line 830 of the OML has some redundant code. The final two statements in line 830 should be changed to:

```
IF R$=Q$ THEN 850
```

But the mistake is instructive. The equals sign in the statement 'IF R$=Q$' means, 'compare for equality'. And in 'R$=Q$' means, 'assign to R$ the string in Q$'. It is evident that the equals sign is leading a double life - sometimes it means 'compare' and sometimes it means 'assign'. Other languages use ':=' for assignment and reserve '=' for what it actually means.

★

WAZZAT CORNER!

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

The BASIC command 'INPUT' allows the user to type up to 71 characters in response to the input query. This response must be followed by a <RETURN>. When large amounts of data have to be input, sometimes by inexperienced users, a more efficient method is desirable. This is particularly true for numbers.

The first improvement would be to do away with the need to use the <RETURN> key to 'send' the data to BASIC. Secondly, it is desirable that the typed input be automatically limited to the required length. Thirdly, it should be possible to filter out certain characters. For example, characters that are not numbers. OS65U 1.4 has the utility 'Extended Input' which goes some way to meeting these needs, but I'm not too thrilled with it. OS-65D 3.3 has nothing in this line at all.

Listings 1 and 2 illustrate a couple of ideas which could be tailored to specific needs.

Readers will be familiar with the single character input, halting get-key subroutine of OSI BASIC. This subroutine is at $FD00 for ROM BASIC, at $252B for DOS 3.2, at $2336 for DOS 3.3, and at $0587 for OS65U 1.3. The subroutine is the basis of the programs. ROM BASIC users will need one of the new monitors, e.g., CEGMON, if the programs are to work. DOS 3.2 users change '9059' to '9815', in line 1000. Leave '2336' as is; do not change it to '252B'.

The first program, Listing 1, will accept a string of seven characters, maximum. If lower case only (with numbers) is required, add 'OR 32' to 'PEEK(9059)'. Thus, 'Y=PEEK-(9059)OR32'. If upper case only (without numbers) is

```
5 REM LISTING 1
6 :
10 PRINT!(28): L$=CHR$(8): U$="-------": L=LEN(U$)
20 PRINT "TYPE WORD " + U$;:FOR C=1 TO L: PRINT L$;: NEXT
30 D$="": FOR C=1 TO L: GOSUB 1000: PRINT Y$;
50 :            NEXT C: PRINT: PRINT: PRINT"WORD IS " D$: END
90 :
1000 DISK!"GO 2336": Y=PEEK(9059): Y$=CHR$(Y) :N=VAL(Y$): D$=D$+Y$
1010 RETURN
```

```
5 REM LISTING 2
6 :
10 PRINT!(28): L$=CHR$(8): A=10000: U$="----.--": L=LEN(U$)
20 PRINT "TYPE NUMBER " + U$;:FOR C=1 TO L: PRINT L$;: NEXT
30 L=L-1: D=0: FOR C=1 TO L: A=A/10: GOSUB 1000: PRINT Y$;
40 :              IF C=4 THEN PRINT".";
50 :            NEXT C: PRINT: PRINT: PRINT"Amount=" D: END
90 :
1000 DISK!"GO 2336": Y=PEEK(9059): Y$=CHR$(Y) :N=VAL(Y$): D=D+A*N
1010 RETURN
```

```
5 REM LISTING 3
6 :
10 PRINT!(28): L$=CHR$(8): U$="-------": L=LEN(U$)
20 PRINT "TYPE WORD " + U$;:FOR C=1 TO L: PRINT L$;: NEXT
30 D$="": FOR C=1 TO L: GOSUB 1000: PRINT Y$;
50 :            NEXT C: PRINT: PRINT: PRINT"WORD IS " D$: END
90 :
1000 DISK!"GO 2336": Y=PEEK(9059)OR32:Y=Y-32:Y$=CHR$(Y)
1005 IF Y<65 THEN 1000
1010 IF Y<>95 THEN D$=D$+Y$: K=K+1: RETURN
1020 C=C-2: IFC<1 THEN C=0
1030 IF LEN(D$)<2 THEN D$="": GOTO1050
1040 D$=LEFT$(D$,LEN(D$)-1)
1050 IF K THEN PRINT L$"-"L$;:K=K-1
```

wanted, then use 'EOR 32'. Oops! OSI BASIC does not support Exclusive OR. Never mind, insert the 'OR32' as shown above, and then follow with this insertion, ':Y=Y-32'. Zounds, it works!

The program in Listing 2 will accept numbers and convert all other characters to zero.

Listing 3 is a tidy up of Listing 1, firmly rejecting all numbers and converting lower case characters to upper case. Backspace and strike-over is permissible - use the 'Rubout' key.

A major problem with utility programs written in BASIC is that they usually require extensive string operations. These operations generate 'garbage' which soon has to be 'collected'. When this happens, even a pause of a few seconds can be irritating. The answer is to write the utility in machine-code!

★

GENERIC'S COLOR PLUS

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

Those of us with video systems, have always enjoyed the ease with which you could program graphics on an OSI machine with all the predefined graphics characters, but I am sure that we all have been a little bit envious of all the high resolution and sprite driven graphics available in newer machines, some at the few hundred dollar level.

For a long time there was not much available in true high res and color for the OSI, unless you were lucky to get one of the Hi Res versions of the 4PMF before OSI sold out video owners. Even with that, there was not much software written for it, if any at all.

Well, you don't have to look at your Apple owner friends with envy any longer, because for more than a year there has been a board on the market that not only gives you Hi Res (256 x 192), but it gives you 15 colors plus transparent! Apple owners will be the ones green with envy when you tell them that the plotting commands for Color Plus are for the most part compatible with Apple Basic plot commands but that shape tables are not required and up to 32 programmable Sprites can be used.

Now you will say, "This is great, but I don't have a lot of memory." Or, you might say "But who cares, I have a C3 and can't do graphics anyway!!"

Well, all the more reason you should hear about this board. First, it has its own graphics memory (16k bytes on board) so it won't use up any of the OSI memory. Second, it comes in two versions--a 16 pin version for the C4 computer and a 48 pin version for the C8 and C3, so even C3 owners can dabble in graphics, although the software supplied currently only supports 65DV3.3 and V3.2. Gen-

eric says they are working on 65U software for the board.

All this must be expensive you say. I paid $195 for the CP-8 version a year ago, but the price has dropped $50 for each version. Now it's really a bargain.

Let's talk a little more about the board. First of all it comes completely assembled (sorry bare boards are no longer available for do it yourselfers), and as I said earlier, with software to support both 65DV3.3 and V3.2. The heart of the system is a TI9918A video display processor, and the documentation includes a complete TI manual on this chip!

This chip supports 35 planes; 32 Sprite planes, a pattern plane, and a backdrop plane. The last plane is the external VDP plane. The Color Plus board includes a software switch to allow display of an alternate source such as the OSI 540 video board, so you can have it both ways.

All this fancy circuitry allows, in addition to the Hi Res mode I already mentioned, a 64 x 48 Low Res mode also with 15 colors and transparent and a 24 x 32 character mode in the Hi Res mode. The board will also support 4 game paddles or 2 joysticks, but they must be Apple II (R) compatible --- OSI joysticks won't work!

When I got my board, it came with the TI manual, a 13 page Color Plus manual which includes the schematic, software including demo programs, and an adapter to connect to the 540 board if it is installed.

Installation takes longer to describe than it took, and consists of plugging the board next to the 540 board in my case and in any slot in case of a C3. The 12 pin molex connector from video out is removed from the 540 board and is inserted in video out on the Color Plus board, then the adapter is installed in the vacant 540 connector and then to the video in on Color Plus. You should also connect paddles and joysticks at this time to the supplied sockets. For C2 or C3 owners it is even easier, plug in the board and put the video connector in video out on the Color Plus board. That's all there is to the hardware --- honest!

To create a running system, you only have to transfer two files on the disk supplied to a system diskette, a new BEXEC*, and a PLOT file. The only drawback to the system is that the PLOT file resides at the start of workspace and uses 12 sectors. When a program you create is saved, the plot routines are saved with it, thus taking up more disk space.

The manual supplied also contains information for Assembly Language programmers, and details the video memory map. The board resides from $C900 - $C902.

The minor differences between Apple commands and Color Plus commands are spelled out in detail, so that Apple programs can be easily converted. One friend of mine has taken Apple graphics books and coded them directly and they worked the first time!

A while back in this article I mentioned demo software. This may be worth the cost of the board itself, and it is fun to watch it go through its paces. Included on the disk are the following programs:

Bomb--A bomber program that needs joysticks.

BTLSHP--Battleships - also needs joysticks.

QMAZE-A Hi Res 3D maze program.

MAGIC-A Hi Res demo that draws moving patterns on the screen in color.

PATERN-A Lo Res kaleidoscope.

MOVIES-This is a very interesting demo of Color Plus with Assembly Language with 11 Hi Res picture files. This has to be seen to be believed!

CHRSET-The character set for the text mode.

When you run these demos you will be impressed like I was with the jitter free displays as the figures move about at high speed, but certainly the most impressive display is the MOVIES demo. This is not one of those plodding plot demos that you often see, but you will see displays drawn in one to two seconds!!!

So to sum up, Color Plus is a board that will give both serial and video OSI owners another Hi Res option for their computer at an affordable price with plenty of software available for it in the public domain. So let's all get a Color Plus and add to that software and our fun too.

★

MATRIX INVERTER

By: Joseph Ennis
212 20 Street
Niceville, FL 32578

THINK SCIENTIFIC

Not only can OSI users be embarrassed by their game playing friends with their $50 TV games, embarrassment can also come from one's scientific friends. The first thing that each of my "scientific" friends ask, when they come to visit my OSI for the first time, is "let's see it invert a matrix." I have gotten so tired of explaining that my version of BASIC does not have any of the Matrix Functions in it (although the original Dartmouth BASIC does) that I have written the following program and offer it to other OSI users.

This method is called the sweep method, or Gauss-Jordan and is faster and more compact than the more widely taught Cramer's Rule. This method also has the advantage of both solving the Determinant and Inverting the Matrix at the same time. The solution vector will be stored in the highest column of the Array and Inverse will be the rest of the Array.

USERS INSTRUCTIONS

LOAD and RUN normally, the program prompts you for its needs. If you are only inverting a matrix rather than solving a simultaneous set of linear equations, put zeros in for the last coefficient asked for in each equation. The model assumes the form of the input data to be:

$$\begin{vmatrix} a_{00}X_0 & a_{01}X_1 \cdots & a_{0n}X_n \\ a_{10}X_0 & a_{11}X_1 \cdots & a_{1n}X_n \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & a_{nn}X_n \end{vmatrix} = \begin{vmatrix} C_0 \\ C_1 \\ \cdot \\ \cdot \\ C_n \end{vmatrix}$$

and for instance if you tell the model that you have three equations, then the model will ask for four coefficients, the first corresponding to a_{00} and the fourth corresponding to C_0. The solution vector

will be of the form:

$X_0 \quad X_1 \quad X_2 \ldots . X_n$

The inverse will be of the form:

$$\begin{matrix} a_{00}^{-1} & a_{01}^{-1} & a_{02}^{-1} & a_{03}^{-1} & \ldots\ldots & a_{0n}^{-1} \\ a_{10}^{-1} & a_{11}^{-1} & a_{12}^{-1} & a_{13}^{-1} & \ldots\ldots & a_{1n}^{-1} \end{matrix}$$

For very large arrays, this will cause a display problem. Display control is in the subroutine on line 20.

There is no limit on the number of equations other than your memory size. Program runs fast, a 10 x 10 done in 5 seconds. Enjoy.

If you get the PIVIT=0 diagnostic in the middle of a large run it either means that it is a coincidence that the principal diagonal value for this equation has become zero as the other equations were swept, or that your data set of equations are not truly independent (most likely). The model will automatically put you in the data check mode. Use this mode to view this equation's coefficients and those of the adjacent rows. If it becomes obvious that a small fix-up is possible, then this mode will allow you to change values in the array and return to processing rather than having to restart and enter all that data again.

However, the usual problem is that the rank of the matrix is less than the number of the equations. That is, one equation is equal to another equation times a constant, or that one equation is merely the sum of two or more other equations in the matrix. In each case the indication of there being a rank problem is that a whole row in the inverse array will have all its terms going to zero or numbers several orders of magnitude less than those in the other rows. When this happens, your data set is bad! The solution is to fix the data set. Review the bad equation and find out why it is not independent or unique.

Possible solutions are run another experiment, take another observation, or drop your least correlating independent variable and run without this equation.

AN OBSERVATION

For very large arrays the accumulated rounding errors will become significant enough to affect the answer to the extent that they will yield wrong conclusions in interpreting the resulting solution. It has been noted that this algorithm run on 32 bit machines produces different (and better) solutions than when it is run on a 12 bit machine. The only cause was found to be the change in the variable values caused by the truncation of trailing decimal fractions during many multiplication and divisions. A check of this algorithm (to detect when the rounding errors become significant for your application) is to take the resulting inverse matrix and enter it as the input coefficients for a second run. An inverse of an inverse should return the original coefficient matrix.

PROGRAMMERS INSTRUCTIONS

The main program starts at line 100. Lines 120 to 160 merely collect the input data and load it in the array and could be changed to read from cards, tape, etc..

Lines 180 to 220 allows the

```
1 REM******************
2 REM**  MATRIX   *****
3 REM**  INVERTER *****
4 REM**    BY     *****
5 REM**  J ENNIS  *****
6 REM**904 6782624*****
7 REM******************
8 PRINTCHR$(3):GOTO100
10 IF LEFT$(A$,1)="Y"THENONNGOSUB17,15,18,19
11 RETURN
13 PRINT:INPUT"CHANGE VALUE";A$:N=2:GOSUB10:RETURN
14 PRINT:FORL=0TOI-1:PRINTAR(K,L);:NEXTL:PRINTAR(K,I):RETURN
15 PRINT:INPUT"INPUT CHANGE ROW,COLUMN,VALUE";K,L,M
16 AR(K,L)=M:GOSUB14:GOSUB13:RETURN
17 FORK=0TOJ:FORL=0TOJ-1:PRINTAR(K,L);:NEXTL:PRINTAR(K,J):NEXTK:RETURN
18 PRINT:FORK=0TOJ:PRINTAR(K,I);:NEXTK:RETURN
19 FORK=0TOJ:FORL=0TOI:A(L)=A(L)+AR(K,L):NEXTL,K
20 PRINT:FORL=0TOI-1:PRINTA(L);:NEXTL:PRINTA(I):RETURN
100 N=0:M=0:L=0:K=0:INPUT"ENTER NUMBER OF EQUATIONS";I:J=I-1
110 DIMARRAY(J,I),A(I):PRINT
120 FORK=0TOJ
130 PRINT:PRINT"FOR EQUATION"K+1:PRINT"ENTER COEFFCIENTS":PRINT
140 FORL=0TOI:PRINT"COEFFICIENT "L+1;:INPUTAR(K,L):NEXTL,K:PRINT:PRINT
150 REM INPUT DATA IS NOW IN ARRAY(J,J) AND CAN BE CHECKED FOR ERRORS
180 INPUT"IS INPUT SATISFACTORY";A$:IFLEFT$(A$,1)="Y"THEN230
190 INPUT"DO YOU WISH COLUMN SUMS";A$:N=4:GOSUB10
200 PRINT:INPUT"DO YOU WISH TO REVIEW ARRAY";A$:N=1:GOSUB10
210 PRINT:INPUT"DO YOU WISH TO CHANGE ARRAY";A$:N=2:GOSUB10
220 PRINT:INPUT"IS INPUT SATISFACTORY";A$:IFLEFT$(A$,1)<>"Y"THEN190
230 REM THE FOLLOWING IS THE DETERMINANT SOLVING AND ARRAY INVERTER
240 FORK=0TOJ
250 IFAR(K,K)=0THENPRINT"PIVIT=0 IN EQUATION "J:GOSUB14:GOSUB13
260 PIVIT=AR(K,K)
270 FORL=0TOI:AR(K,L)=AR(K,L)/PI:NEXTL
280 AR(K,K)=1/PI
290 FORL=0TOJ
300 IFL=KTHEN350
310 IFAR(L,K)=0THEN350
320 PKEY=AR(L,K)
330 FORM=0TOI:AR(L,M)=AR(L,M)-PKEY*AR(K,M):NEXTM
340 AR(L,K)=-PKEY/PI
350 NEXTL,K
360 REM SOLUTION VECTOR IN LAST COLUMN REST OF ARRAY IS INVERSE
370 INPUT"DO YOU WISH SOLUTION VECTOR";A$:N=3:GOSUB10
380 PRINT:PRINT:INPUT"DO YOU WISH INVERSE";A$:N=1:GOSUB10
390 PRINT:PRINT"a GOTO180 command will allow changes without retyping";
400 PRINT" all the values in again":END
```

input data to be checked for input errors, and can be ignored by the user if he is using the program only to entertain and impress visitors.

Lines 240 to 340 do the work. Roughly speaking what they do is form an inverse by taking the value of the principal diagonal and using this to simultaneously calculate the determinant, transpose, and multiply the original coefficient matrix with its inverse producing the solution vector.

Line 10 is the entry point for the subroutines. To be compact and run fast, everything that could be put in a subroutine was. Each subroutine call sets a vector in N. If the test in line 10 for YES is TRUE the vector N is used to compute the jump. Subroutines are ranked so that the most often used have the lowest line numbers.

★

ASSEMBLER/EDITOR HOLD PATCH

By: Harry B. Pye
2406 Hillock Court
Lansdale, PA 19446

Here is a dandy little utility that can be added as a permanent part of the Assembler/Editor of OS-65D V3.X on C2/C4 systems. It allows the output of the Assembler to be halted and restarted at any time. If you don't like what you see, the assembly or listing can be aborted. It has served as a useful tool for me. Saves making printouts that are reviewed to diagnose a few errors and then discarded.

If there are problems when executing an "Errors Only" assembly, you only see the offending lines. In a full assembly, the errors often roll off the screen so quickly that you miss even the line number. Once this utility is installed, just tap the "Escape" key and the output is stopped. This gives you a chance to review the code both before and after the error and make notes. To continue, just press the "Repeat" key. If you decide that the error should be corrected before proceeding, press the "Control" key and you are back to the Assembler's prompt.

The standard Assembler code contains a JMP $2343 at address $0BF9. This is a jump to the character output routine in the operating system. Line 290 of the utility changes this to a jump to the patch. In the patch the contents of the accumulator (rA) are saved temporarily on the stack. Row zero of the keyboard matrix is tested for the "Repeat" key depressed. If the key is not down, the character is retrieved from the stack, printed and control returned to the Assembler. If the "Escape" key is pressed, the patch goes into a loop. The BIT instruction in line 380 allows testing for the presence of either the "Repeat" or "Control" key. If the "Repeat" key is detected, the patch exits just as before. If the "Control" key is pressed the character on the stack is discarded and the program jumps to $0200.

Installing the patch is simple. Create a file called HLDFIX. My commented version takes one track on an eight inch disk. Key in the source code and save it to this file. Run an "Errors Only" (A1) assembly and correct all of the indicated errors. Of course, you should save this back on the disk. When everything seems correct, execute an assembly to memory(A3). Now run a normal (A) assembly and test the operation. If it doesn't work, edit the source file and try again. It may be necessary to reload the Assembler if it got clobbered.

```
10 ; **********************************
20 ; **                              **
30 ; ** ASSEMBLER/EDITOR HOLD PATCH **
40 ; **                              **
50 ; **  for use with OS-65D V3.X    **
60 ; **                              **
70 ; **   Harry B. Pye  02/17/84     **
80 ; **                              **
90 ; **********************************
100 ;
110 ; "ESC" Halts the Output of an
120 ;       Assembly or Print.
130 ;
140 ; "REPEAT" Continue Output
150 ;          after a Halt.
160 ;
170 ; "CTRL" Aborts an Assembly or
180 ;        Print. Returns to Prompt.
190 ;
200 ; EXTERNAL EQUATES
210 ;
220         SYSOUT=$2343    ($FFEE on cassette)
230         KEYBRD=$DF00
240         ABORT =$0200    ($1160 on cassette)
250         START =$0BF9    ($1333 on cassette)
260         NXTCHR=START+3 (Omit on cassette)
270 ;
280         *=START
290         JMP PATCH    Install Jump to the Patch
300 ;
310         *=$1600         ($0222 on cassette)
320 PATCH   PHA          Save Character
330         LDA #$01     Strobe Keyboard Row #0
340         STA KEYBRD     for Keystroke
350         LDA KEYBRD   Test for Key depressed
360         CMP #$21     Is it the "ESC" key ?
370         BNE PRINT    No, Bail out & Print Character
380 LOOP    BIT KEYBRD   Yes, Keep checking keyboard
390         BMI PRINT    "REPEAT" key pressed, exit
400         BVC LOOP     No key, keep testing
410         PLA          Cleanup the 'Stack'
420         JMP ABORT    "CTRL" key, abort processing
430 PRINT   PLA          Retrieve Character
440         JSR SYSOUT  Output character (Change JSR to
445 ;                                    JMP FOR tape)
450         JMP NXTCHR     and continue processing
455 ;                      (Omit line 450 on cassette)
460       .END
```

If all goes well, save the revised Assembler/Editor and Extended Monitor back to the disk. On an eight inch drive system this is:

```
!SA 05,1=0200/B
!SA 06,1=0D00/C
!SA 07,1=1900/7
```

Now reboot the system, reload an assembly source file and verify that the changes have been made permanent.

A few final notes. Users with C1 systems will have to modify line 330 to change the keyboard strobe as well as lines 390 & 400 to modify the tests for the "Repeat" and "Control" keys. C2/C4 users with cassette systems can also modify the patch to run on their systems. The original version was written to reside from $0222 to $023D, an area that is generally unused on the C2/C4 cassette systems. The comments in the listing show the necessary changes for cassette users.

★

PIACOM OPERATING NOTES

By: Steve Donachie
6811 S.W. 81st Terrace
Miami, FL 33143

Well, it's finally happening. After four years of use, my company is about to replace

our venerable C4P-DF with some sort of MS-DOS system. As a fond farewell to PEEK(65) and all those participants who provided vital support to us over the years, I am making this contribution to public domain software for anyone who needs it.

The enclosed program and notes allow data and programs to be transferred between 5" and 8" disks by connecting the seldom used parallel ports on the A-15 board at the rear of the C-4/8 series of computers. Obviously, two systems are required, but anyone facing this need is likely to be upgrading to a larger 8" disk system, as I was.

Thank you all for the years of info!

PIACOM is a file transfer program designed to exchange programs and data between 5" and 8" disks. The program runs simultaneously on two OSI C-4/8 computers which have incompatible disk sizes. The machines are connected through one of the parallel ports on the A-15 board (connect pins 1 through 8 only).

Data can be transferred as is. If there is no end of file indicator at the end of your sequential or text file, or if you don't know how many random records in your file, use any non-ASCII code that will not appear in a text file, or use the maximum number of records for the size of the file. Then, when you see the end of data go by and garbage follow, you can terminate the program manually with a CTRL-C followed by DISK CLOSE,6. This will in no way compromise the integrity of the data already transferred.

To send a program, first list to disk as follows:

Supply a buffer for the program if it does not already have one.

With the program loaded, enter DISK OPEN,6,"filename"

Enter LIST#6

then PRINT#6,CHR$(end of file indicator) which can be any non-ASCII character except 192 decimal, which is utilized for another purpose by the program.

Enter DISK CLOSE,6

Next run PIACOM to send the file. After transfer, load the transferred text file into

```
1 REM PIACOM PROGRAM--PUBLIC DOMAIN COURTESY OF STEVE DONACHIE
2 GOTO 500
10 POKE CTRL,0:POKE PIA,255:POKE CTRL,4:RETURN:REM OUT
20 POKE CTRL,0:POKE PIA,0:POKE CTRL,4:RETURN:REM IN
100 REM SEND
110 GOSUB 10
120 POKE PIA,192:IF PEEK(PIA)<>192 THEN 120
125 POKE PIA,255:RETURN
150 POKE PIA,255:IF PEEK(PIA)<>255 THEN 150
160 POKE PIA,C
180 POKE PIA,255:RETURN
200 REM RECEIVE
210 DISK!"IO  ,22":GOSUB 20
220 IF PEEK(PIA)<>192 THEN 220
230 GOSUB 10:RETURN
250 GOSUB 20
260 C=PEEK(PIA):IF C=255 THEN 260
270 GOSUB 10:RETURN
500 INPUT"PORT A OR B";A$
505 IF A$="A"THEN PIA=63232:CTRL=63233:GOTO 520
510 IF A$="B"THEN PIA=63234:CTRL=63235:GOTO 520
515 GOTO 500
520 RT=0:SP=24:BY=3072:REM SP=16 AND BY=2048 FOR 5 IN ***
522 REM INSERT A 'GOTO 800' HERE FOR TEST FEATURE
525 INPUT"DISK FILE TO USE";F$
530 INPUT"IS THIS A RANDOM ACCESS FILE";R$:R$=LEFT$(R$,1)
535 IF R$<>"Y"THEN 550
537 Z=192
540 INPUT"HOW MANY RECORDS IN FILE";RT
545 INPUT"HOW MANY BYTES PER RECORD (128 STD.)";BY:GOTO 575
550 PRINT"THEN ENTER A CHARACTER, OR ASC VALUE OF A CHARACTER"
560 INPUT"TO USE AS AN END OF FILE MARKER";Z$
565 Z=ASC(Z$):IF Z$="00"THEN Z=0
566 IF TN>TC THEN PRINT"FILE "F$" TOO SMALL":GOTO 520
570 V=VAL(Z$):IF V>0 THEN Z=V
575 DISK OPEN,6,F$
580 TC=PEEK(9003)-PEEK(9002)+1:BU=PEEK(8998)+256*PEEK(8999)
585 EB=BU+BY:TN=INT(RT*BY/BY):FZ=INT((RT+1)*BY-TN*BY)
586 IF FZ>0 THEN TN=TN+1
587 IF R$<>"Y"THEN TN=TC
590 INPUT"(1)SEND OR (2)RECEIVE";FU:ON FU GOTO 591,595
591 PRINT"START RECEIVER FIRST...PRESS ANY KEY WHEN READY"
592 DISK!"GO 252B":GOTO 600:REM 'GO 2336' FOR 65D 3.3
595 PRINT"WAITING"
600 ON FU GOSUB 100,200
610 FOR R=0 TO TN*24 STEP SP:DISK GET,R:K=BU
620 IF FU=2 THEN GOSUB 250:GOTO 640
630 C=PEEK(K):IF C=255 THEN C=0
635 GOSUB 150
640 PRINTCHR$(C);:IF C=13 THEN PRINT#2
650 IF C=Z THEN 690
655 IF R$="Y"AND TN=R/SP+1 AND K=FZ+BU THEN 690
660 K=K+1:IF K<EB THEN 620
670 IF FU=2 THEN DISK PUT
680 NEXT R:C=Z: GOTO 635
690 PRINT:DISK CLOSE,6:DISK!"IO  ,02"
695 PRINT:PRINT"TRANSFER COMPLETE":END

799 REM THIS SECTION FOR TESTING ONLY--MAY BE DELETED
800 INPUT"COMMAND";A$:V=VAL(A$)
805 IF LEN(A$)>1 AND V=0 THEN 800
807 IF V>0 THEN 850
810 IF A$="P"THEN PRINT"PIA ="PEEK(PIA)
820 IF A$="I"THEN GOSUB 20:PRINT"SET IN...PIA ="PEEK(PIA)
825 IF A$="O"THEN GOSUB 10:PRINT"SET OUT...PIA ="PEEK(PIA)
830 GOTO800
850 POKE PIA,V:PRINT"SENT"V"...PIA ="PEEK(PIA)
860 GOTO 800
```

program form like this:

Establish a disk buffer in memory.

Enter DISK OPEN,6,"filename"

Enter POKE 8993,32 (to reset input flag)

The file will scroll on screen as it loads. The end of file indicator will cause a syntax error, which is normal. The program is then ready to SAVE in RUNable form--use a different disk file in case of errors so you will still have the original text intact.

The handshaking principle used by PIACOM exploits a feature of the programmable bidirectional parallel ports used on the A15 board. (User of home-brewed ports beware!) When the sending computer transmits a character to the receiver and then PEEKS its own port, the contents will appear altered if the receiving computer was not configured to receive at the time of transmission; but if the receiver WAS configured to receive, the port contents will match the data character being sent.

The dialogue works like this:

SENDER	RECEIVER
set for output	set for output
(ready to send)	(not ready yet)
send character	--
peek the port	--
contents don't	set for input
match	(ready now)
--	check for char.
--	none; check again
resend char.	received
match; was received	set for output
send next char	(busy signal)
--	store character
etc.	get next one;etc.

Actually, it's slightly more complicated than that. A non-ASCII control code is sent

first to determine if the receiver is ready, then the data follows. The memory locations to watch are the port addresses, identified in the manual as the "data and control registers." Other parameters in the program adjust for the number of pages per track used by the 5" and 8" systems (assumes 8 and 12).

Note that the program requires one disk buffer as device#6. The listing is for the 8" version. Make the indicated changes for the 5" one. Both run under OS65D3.2. They should also work under version 3.3 if the keyboard poll is changed as marked in line 592.

A testing feature is implemented in lines 800-860 to facilitate diagnosing problems if you have a non-standard port. This allows you to set the port for input or output, send characters, and examine the contents of the port. Commands I and O do the setting, P displays the contents. Any numeric entry will represent the ASCII value of a character to be sent.

★

OSI's 235 COMPUTER

By: Edward T. Gieske, Jr., Editor

The 235 is OSI's latest in small business machines that hosts the new 515 "do all" board and 5.25" hard disk.

What is a 235? Well, it looks like most any other 200 series computer from the outside, but there the comparison ends. Basically, it is a computer-on-a-board, a hard disk controller and the disk itself. That in itself is quite a change from the innards of most older OSI machines. Do not take it the wrong way, but the old machines were a ganglion of wires and cables, almost to the point that it ought to choke itself.

So, now here it is, all neatly on one 515 board. To me that means three things: 1) the cost goes down, 2) reliability goes up and 3) it might just be that we will see a smaller box - lets face it, the box is almost empty!

Let's see just what they have packed onto this wonder-board. For starters, the usual 6502 running at 1.94 MHz. Then, get this - 4 user ports! That's right, it is a 4 user time-share computer with 4 blocks of 48K RAM plus the 8K for multi-user, a total of 196K, all in 8 x 10 inches. Of course there is also a RS-232 port and a Centronics parallel port for printers and a network port so you can tie it to the big guy, and a floppy disk controller so you can mount the hard disk and make backups. If you don't see the need for 4-users, there is also a 2-user version with 100K of RAM.

Perhaps another way of looking at things is to think of what the 515 replaces. Let's throw out the CPU, 4+ memory boards, a CA-10 or maybe a 555 jungle board, and certainly the 470 floppy controller.

One added attraction is that for about $240 you can send your 2-user board back to OSI for the 4-user upgrade. On the other hand, if you have some CM-20 memory boards lying about, you could use them instead.

Who wants a 5.25" hard disk, they are slow compared with the 14" jobs - or are they? The new controller for these disks is "intelligent" which means that parity checking is done on the controller board, not by the 6502, and thus it can use the high speed buss to communicate with the CPU. The net result is that, in most instances, the 5.25" is just as fast as its big brothers. They are not as big in capacity as the 14"ers, but 18.4 and 29 MB, formatted, should be big enough for a great many applications. Another nice feature particularly for those who move machines, is that these drives are double self-locking on shut-down. So much for size and simplicity. How about the cost? Remember the 230I single user? Well the 235/2 is only about $300 more. Would you not rather have a 2-user machine for the extra change?

There is one other possibility. These machines are also available as the WORKSYSTEM 235/4, which means that for about $300 more you get an Esprit terminal, Citho FP-1500 letter quality printer, the TP-2 word processor and The Data System (data base manager) along with all associated cables.

LETTERS

ED:

Re: Char. Get / Char. Got Page Zero Routine PEEK(65) Readers and Carl King

I am sorry that you think that the POKE and information covered in my letter of April 84 in PEEK(65), Pg 17, is worthless and doesn't work. Really it does work, you see with this POKE enabled, space characters are not legal, that is the reason that OSI put this routine in their BASIC Interpreter in the first place. Without this routine your BASIC programs will not RUN if they contain any spaces. You must reset the POKE back to normal if you intend to have BASIC Interpret any BASIC Code with spaces contained in the BASIC Code from either the Direct or Program Mode. Or, if you wish to execute a PRINTPEEK(X) command from the immediate mode, you must type it without spaces, leading or otherwise.

With this space compression routine POKE set to the off

state, BASIC now sees space characters as illegal characters and causes a SN ERROR as any illegal character, statement, or command would. If this routine was not in page zero on OSI Computers, it would have a similar working routine elsewhere in the BASIC Interpreter, in much slower memory, the end result would be OSI Computers would run quite a bit slower. Also, this is one of the same reasons that the INPUT, TOKENIZING, and EXECUTION BUFFER is in page zero, for speed in loading Indirect Files and Cassette Tapes, etc.

J. L. Pottier, had a similar POKE, probably for Pico-DOS, which explains the slight page zero address difference, that appeared in January 84 in PEEK(65), Page 22. I am not the only one to use this idea!

Carl, it is true, spaces do take up a fair amount of memory (workspace). Some programmers like to indent blocks of lines and other BASIC code, and then use a program packer utility for their Final "Run-Time" Version, leaving the original version of the program with the "Pretty Print" Format and Remarks for later use in changing the program or trying to remember what the program was supposed to do. Sometimes using this "Pretty Print" Format makes the program source code look a lot nicer as well as making it a lot easier to read, if carefully done.

I do a fair amount of program coding in PASCAL and ASSEMBLER on my OSI C-8P. I also do a fair share of PASCAL and COBOL coding on a DEC PDP 11 and other Mini-computers at an area college, where this "Pretty Print" Format is almost a necessity to be able to read my original source code.

The main reason I made this POKE available to PEEK(65) was to show other readers a way to input leading spaces into a string input variable under program control. Here is a short program to demonstrate this. (Note: the first space after the line number is not seen by the BASIC Interpreter; all other BASIC program code after the first POKE, until the space compression is reset by the POKE in line 40, MUST NOT CONTAIN SPACES except inside double quotes and after Remark statements.)

The decimal number 36 POKEd in to the proper location, depending on the system in use, will turn off BASIC's space compression on all input buffer operations, even when entering lines of BASIC CODE with line numbers.

This POKE location is in the page zero character-get routine in OSI's BASIC. The routine checks for a BASIC statement separator (colon), and space characters. If the routine finds a space it skips it and continues to look at the line of code until it finds something other than a space.

If you intend to use this space compression routine in the disabled mode(POKEx,36), from the BASIC direct command or immediate mode (at the Ok prompt), you must type all commands, for example LIST, LIST#4, or PRINTFRE(X), with no spaces anywhere in the line. As I said before spaces are not illegal characters with this POKE set like this: (POKEx,36).

This type of POKE modifies a routine that was never intended to be used by a beginner or novice, as with a lot of useful POKEs (and FLAGs, too in 65U). There are lots of POKEs & FLAGs like this in 65D and 65U, that must be reset immediately after use. Also, while the POKE or FLAG is in use, care must be taken to insure that the proper conditions have been met and that they will not change state once the operation has been started.

I never claimed that this POKE did not have side affects. Most POKEs and/or FLAGs you will use do have side affects. They are altering pre-set conditions, and must be used with GREAT CARE.

Readers, some of this may have been my fault. In my earlier letter, not explaining all of what could happen and giving more information on side affects of this POKE. If this is the case, I'm sorry about not giving enough details. I guess I thought most everyone would be smart enough, and careful too, when working with this POKE. My motto on this has always been "Be Careful With All POKEs and FLAGs". Remember, I did say to use great caution when changing this page zero character get-routine in my first letter, this should have warned most of the erudite readers to be careful in the use of this POKE, as I am with any POKE or FLAG.

Sorry it took so long to reply concerning this situation. I have been extremely busy, and I'll be busy again in about two weeks. I am going back to college to work on program coding in COBOL 77, among other course work.

Carl, I also found out that you would like to change your WP6502 word processor to have the page number default to zero. This is very easy to do. I have knowledge of the 8-inch diskette version. Minidisk versions may be a bit different.

Boot up a standard OS65D BASIC operating disk, exit to the BASIC prompt, now remove your OS65D disk, and put your WP6502 disk in the selected drive and close the door. Then carefully type: DISK!"LO WP6502<RETURN>, now list the program to line 70(LIST-70), insert a line like this somewhere earlier on in the program. 65 POKE 12696,0. The PEEK of this should be 1 to start with. Then save it back on the disk by typing: DISK!"PUT WP6502, if anything gets screwed up, there is probably a file on your WP6502 disk called WP6503 that should be a back-up of this same program. There is a better way to change this, I can tell you later if you want. Hope this helps. Sorry about any misunderstanding.

Al Adams
Midland, MI 48640

* * * * *

ED:

Recently, a friend of mine asked for help in solving what

```
10 POKE x, 36:REM space crunching off
20 INPUTA$:REM input string
30 PRINT" Spaces Ok In Double Quotes ":REM Spaces Ok Here Too!
40 POKEx,240:REM reset space gobbler back on. <Normal>
50 PRINT A$:REM spacing ok here
```

Where x equals the location depending on your machine type, from this list:

VERSION of BASIC	VALUE OF x
OS65D v3.2	x=207 These Values are for
OS65D v3.3	x=203 8 Inch Disks.
BASIC-IN-ROM USERS	x=203 Should work for all BASIC-IN-ROMS.

The POKEs for OS65D should work for the 5-inch disks, too.

OSI HACKERS BONANZA

(for knowledgeable users only)

FANTASTIC PRICES ON

DEMO COMPUTERS, BOARDS, AND ACCESSORIES FOR
Ohio Scientific Challenger Series

Computer Systems

Superboard II	$ 50
Superboard III (32k)	100
C2 Net	500
C3 OEM	1200
C3A	2000

Populated Boards (less cables)

CA26(538) E Prom	$ 50
CA60 (620) Expansion Board	20
CM-2 (527)4k	50
CM-9 (527)24k	100
CM-10(527)8k	75
543 10 Key Pad	10

Partially Populated Boards[1]
(all $10 each)

505	555	582
512	567	594
522	570	600
538	571	610
540	572	620
541	574	630
542	580	A15

[1]Number of components on these boards varies widely.

Accessories (less cables)

Diablo Parallel WP Printer	$ 800
Joysticks (2) 8 Axis	5
Epson MX80 Serial Card	25
Print Heads	10
Ribbons	2
Cat Modem 300 Baud	50
Power Supplies	
12V@1A/9V@1.2A/5V@7A	40
12V@1.5A	10
5V@3A	10
10VCT@5A AC Transformer	2
9.2V@3A AC Transformer	1
Floppy 5-1/4" Diskettes/(box of 10)	12

Bare Boards (all $5 each)

505	567	580
520	570	582
527	571	600
535	572	610
538	573	A15
543	574	A12
555	575	A13

Ordering Information

* Orders are COD freight collect except:
* Orders over $500 require pre-payment
* Minimum orders $50
* All items are for Domestic use only (no foreign orders)
* Quantities are limited-orders subject to availability
* 10 day return for refund on computer systems, other items as is (no warranty)

TO ORDER: Call (216) 562-4136 8 A.M.-12 P.M. 1 P.M.-5 P.M. EST

ISOTRON

Manufacturer of Ohio Scientific Computers
1333 S. Chillicothe Road Aurora, Ohio 44202

appeared to be a growing performance problem with a data entry program. In the application, a numeric code was entered as part of the input data. This code was tested against an array containing valid codes to determine if an error condition existed. Over time, more and more codes had been added to the table; and, periodically, the data entry program appeared to take a long pause when certain codes were entered.

Examination of the array and the program disclosed that the array had grown quite large and that a FOR ... NEXT loop was being used for searching the array (table lookup). I replaced the sequential search my friend had been using with a binary search. The binary search produced a significant improvement in the performance of the program. Because of the results achieved, I thought the binary search method might be of interest and possibly of value to some PEEK(65) readers.

A FOR ... NEXT loop can always be used to search a table, but this can be a slow process if the table is large. The binary search technique permits a table lookup to be performed a great deal faster because of the way it segments the table, rapidly isolating the portion of the table in which the desired entry is most likely to be. My experience has been that the binary search technique is probably preferable to a FOR ... NEXT loop when a table has eight or more entries. With fewer than eight entries, the binary search doesn't offer an advantage. The larger the table, the more dramatic the effect of the binary search will be.

When a binary search is used, the table must be in sequence (either ascending or descending) for the technique to work. In the routine to follow, the table is considered to be in ascending sequence. The sample code is also applied to a numeric array. Changes can be easily made so that it will work with a string array.

The binary search begins by dividing the table (array) into two segments and comparing the search argument against the table entry dividing the two segments (the common bound) to determine the segment (upper or lower) within whose range the search argument falls. This test "throws out" half of the table entries (one of the segments). The chosen segment is divided in half and the testing is repeated. Following each test, one of the segments is discarded, and with it, half of the table entries remaining from the previous test. (Of course, at any time a match may be found, in which case the search stops.) The process continues until there are only one or two entries left in the table to be tested. At this point a FOR ... NEXT loop is used for the final searching.

So much for a confusing description of the binary search technique. Let's look at the program code:

David A. Weigle
Morton, IL 61550

* * * * *

ED:

In regard to Mr. Goodglass's letter in the August issue of PEEK(65), I have researched the problem he is having with his OKIDATA Microline 82A printer. Why that particular printer doesn't work when hooked up to the RS-232 interface on his ClP stems from another one of those non-standard standards. To give the topic full consideration, let me introduce a little history and background information.

The RS-232 interface was originally developed back in the early days of dumb (and I mean DUMB) modems for remote printers (see the July 2 issue of INFOWORLD). Besides the obvious two lines and a ground needed to send serial data back and forth, all kinds of lines were added for status information and control functions. Hence, the 25 pin connector.

The TTL voltages in use at the time (which are the same as the ones inside your computer) run from 0 to .8 volts (.35 volts typical) for a Binary 0 and from 2.0 to 5 volts (3.4 volts typical) for a Binary 1. This representation is only good for a few hundred feet because it doesn't take much resistance in the wire to drop say 3.4 volts down below 2 volts. Then it becomes confusing as to whether the signal is a Binary 1 or 0.

But in many large scale environments, the distance between the modem (telephone switch board) and the remote printer site could be considerable, even in separate buildings. So the "standard" also included special voltage levels to indicate data. These are -12 v. to -3 v. for a Binary 1 and +3 v. to +12 v. for a Binary 0. That's right, they are inverted from what you would expect, just to keep it confusing! However, this large voltage swing gives an

Definitions
DIM TBL(N) = table of N entries (N>7) in ascending order

LB = bottom table entry for a segment pair (the lower bound)

UB = top table entry for a segment pair (the upper bound)

CB = table entry between the two segments (the common bound)

SA = search argument

Program line 1000 - routine entered if no match found

Program line 2000 - routine entered if a match is found

```
 10 REM FIRST DETERMINE IF THE SEARCH ARGUMENT MAY EVEN BE
 20 REM IN THE TABLE
 30 IF SA<TBL(1) OR SA>TBL(N) GOTO 1000 : REM if outside table
 40 REM FOR THE FIRST PASS THE FIRST AND LAST TABLE ENTRIES
 50 REM ARE THE LOWER AND UPPER BOUNDS
 60 LB=1 : UB=N
 70 REM DETERMINE COMMON BOUND ENTRY BETWEEN THE TWO SEGMENTS
 80 CB=INT((UB-LB)/2)+LB
 90 REM IS THE TABLE ENTRY PROBABLY IN THE LOWER SEGMENT?
100 IF SA<TB(CB) THEN UB=CB-1 : ON 1-(UB-LB<3) GOTO 80,160
110 REM IS THE TABLE ENTRY PROBABLY IN THE UPPER SEGMENT?
120 IF SA>TBL(CB) THEN LB=CB+1 : ON 1-(UB-LB<3) GOTO 80,160
130 REM A MATCH WAS FOUND -- IT IS THE COMMON BOUND ENTRY
140 GOTO 2000
150 REM SEQUENTIAL SEARCH- LESS THAN THREE TABLE ENTRIES REMAIN
160 F=1 : REM found/not found switch
170 FOR K=LB TO UB : REM set loop iterations
180 IF SA = TBL(K) THEN F=2 : CB=K : K=UB : REM match found?
190 NEXT K
200 ON F GOTO 1000,2000 : REM route to appropriate routine
```

effective range of about 5 miles.

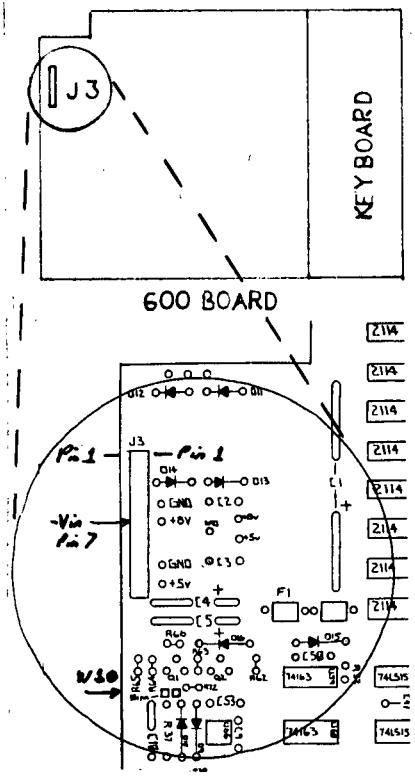
Later, as the standard was applied to more and more devices (like terminals, card-readers, etc.) many of the pins were reassigned or deleted. When finally the microcomputers arrived, the standard was already botched up. Consequently, no RS-232 interface on any microcomputer is really "RS-232 standard". But for the microcomputer, nothing could be better than to use the old RS-232 interface because many peripherals were already using it. (That is unless you are a printer manufacturer like Centronics and could develop your own, which is where the parallel interface came from.)

Finally, since long distances are not needed in the microcomputer environment and since many computers (like the C1P) do not provide negative voltages anyway, most printer manufacturers have gone back to using the TTL voltage levels. I say "most" because the 82A is one that hasn't. That is why the interface works with the EPSON printer but not with the OKIDATA.

OKIDATA's approach to the problem has been to design "Plug 'N Play" printer cables for the popular microcomputers. The cable also contains any necessary electronics to convert signals for the specific computer. Of course, there is no such cable for OSI computers.

I was not aware of this fact when I recommended the 82A to Mr. Goodglass. However, as PEEK(65) pointed out, OSI machines have the ability to be made to work with just about anything. And the 82A is a good choice for a printer (see the Oct. 1983 issue of CONSUMER REPORTS). But where is the negative power supply going to come from? Well, fortunately there is a -12.5 v. supply available at pin 4 of the RS-232 connector on the printer.

What needs to be done is as follows. Pin 7 of J3 (see diagram) is the negative voltage input for the RS-232 interface on the C1P "600" board. However, THIS INPUT IS NORMALLY GROUNDED so the interface will function in "TTL mode". THE TRACE OR JUMPER AT W10 MUST BE REMOVED TO PREVENT SHORTING THE -12.5 VOLT SUPPLY FROM THE PRINTER. Once that is done, adding another wire (defining our own RS-232 again) from pin 7 of J3 to pin 4 on the printer will provide the necessary negative voltage required by the 82A.



Jerry Travis
Olympia, WA 98503

* * * * *

ED:

The following program is very similar to one which appeared as a part of a Disk Program in PEEK(65) (can't locate it right now). It may have been missed by those on ROM machines. Useful for anyone who requires to convert Hexadecimal to Decimal and vice versa and about as short as can be for the job. Can be divided off easily if only one type of conversion is needed.

```
63910 FORQ=1TO32:PRINT:NEXT:PRINT"TOGGLES
      BETWEEN THE TWO.
63911 PRINT"HEX/DEC:DEC/HEX CONVERSION.
      NO #/DEC = 0 or";
63912 A=INT(D/16):B=D-A*16:H$=CHR$(B-7*
      (B>9)+48)+H$:D=A
63913 IFD<>0THEN63912
63914 PRINTSPC(12);"HEX = ";H$:PRINT:
      INPUT"HEX =";H$:N=0:H=1
63915 FORX=1TOLEN(H$):L=ASC(RIGHT$(H$,X))
      -48:L=L+7*(L>9)
63916 N=N+L*H:H=H*16:NEXT:PRINT;"DEC =
      "N:H$="":PRINT
63917 PRINTSPC(12):INPUT"DEC =";D:
      GOTO63912:REM <RETURN> EXITS.
```

R. N. Hislop
Porirua, New Zealand

* * * * *

ED:

I have a C4P to which I have added the D&N Mircro Products Disk Controller and Memory board. Also, I have added the Generic Color Plus board and the results are fantastic. I will be adding the second drive in November.

I'm changing over my programs from tape to disk. I was able to change all but the following programs:

1. Galaxia
2. Caterpillar
3. Minos (maze)
4. Fantastic Copy (never did work)
5. Space Debris
6. Tiny Compiler (tape version works beautifully)

As I am not familiar with machine language programming, I'm having difficulty with the above programs. I have made Disassembler listings of Galaxia and Catapiller.

I would appreciate hearing from any readers who have these programs to arrange either a program listing or a disk copy of the programs.

I am finding the articles on machine language programming very interesting.

Please keep up the good work and quality of PEEK(65).

Gary Florence
Alberta, Canada T3A 1R9

Gary:

I am not immediately familiar with all of the programs you want to convert, but except for Tiny Compiler, I believe they are all machine code. Translating machine code game programs from tape to disk will usually entail relocating the programs to reside in memory above $4000 or so to avoid BASIC and OS-65D. You're also going to have to look over the programs for references to memory locations below $4000 and change them to free areas of memory as well. If the tapes are auto-loading, I suggest you check out the Assembler/Editor - Extended Monitor Reference Manual so that you can decode the format in which these programs are stored on tape and also so that you'll be able to use the Extended Monitor to help relocate the programs in memory so that you can save them to disk.

If all this sounds complicated to you, you're absolutely right, and I really don't see

any easy solutions for the games.

There is some good news. The author of Fantastic Copy is a member of the user group OSMOSUS and I'm sure if you contact them, they'll be able to get you a working copy of Fantastic Copy. OSMOSUS' address is:

OSMOSUS
c/o Donn B. Baker
3128 Silver Lake Road
Minneapolis, MN 55418

And, if I might be allowed a small plug; the author of "Tiny Compiler," David Pitt, has given permission for the OSI version of the program to be freely distributed and it is available in OSI SIG on CompuServe. I've even touched the program up a bit to add support for the keyword "DISK" so you can issue commands to OS-65D in programs.

Hope this helps!

Rich Trethewey, Sys Operator
OSI SIG on CompuServe

PEEK(65) INDEX FOR 1984

AD$

HELP WANTED: Full and Part-time. New York authorized OSI Dealer is looking for programmers and hardware technicians. Attractive salary and good working conditions. Must have strong background in OS65U with hard disk experience. Call 212-926-7634 or send resume to Crescent Computer Systems, Inc., Box 119, New York, NY 10037

* * * * *

WANTED: Cipher 1/2" tape drive with OSI interface board, in good working condition. Also OKI 2350 printer. Call collect Ting Barrow (212) 989-1945.

* * * * *

MUST SELL. Still in original wrappings, KEYWORD CP/M Word Processor, CP/M v 2.25. Cost was $400.00 each. Will sacrifice $250.00 each, or $400.00 for set. Reply PEEK, Box K, c/o PEEK(65), P.O. Box 347, Owings Mills, MD 21117.

* * * * *

Large collection of OSI hardware/software/manuals/schematics/computers. Send S.A.S.E. for current price list to Thomas Technical Service, Rd #1, Box 135, Linden, PA 17744, phone 717-398-1893 evenings.

* * * * *

EXCELLENT condition: C2 serial, 56K, dual 8" drives, 1 parallel & 2 serial ports, OS-65D, OS-65U, WP2, WP6502v1.2, Sanders' SORT/MERGE and RESEQ, diagnostic software. Roy Harris, Rt. 2, Box 340, Charlottesville, VA 22901. No reasonable offer refused. Phone (804) 293-3605.

* * * * *

Send for free catalog, Aurora Software, 37 South Mitchell, Arlington Heights, IL 60005. Phone (312) 259-4071.

* * * * *

OSI BARGAINS! (4) C3-Bs, (2 are Dealer Demos, Never Sold), all have NEW 80-Mbyte Disks and NEW (or Reconditioned) Siemens Floppy Drives, w/terminals too, (Hazeltine 1420s, SOROC IQ-140s, or Televideo 912-Cs), from $3000 (single user) to $5000 (4-user). Up to 100 disks of free software too, (you pay for disks and manuals). (6) OSI C3-OEMs, w/ terminals, $995 each. (4) NEC Spinwriters, perfect condition $800/ea. (1) DataProducts M-200, new, $2000. (1) Alloy Engineering Tape Back-Up, never taken out of box, $2000. (1) Okidata 160 Line Printer, new, never used, $1200. (9) Okidata 22s, reconditioned as new, $900/ea. All or part. FOB San Diego, CA. Subject to prior sale. Computers West, (619) 583-1651.

N O W A V A I L A B L E
MOTOROLA 68000 HARDWARE FOR THE OSI

Now you can have the latest 16/32 BIT processor and keep your OSI.

Are you looking for a way to keep your OSI and have the latest technology? Are you a serious hardware or software hacker? Do you want to learn to program the most powerfull microprocessor available today? If so then maybe I have the answer for you.

Now with my hardware and software you can connect a Motorola 68000 processor with up to 1 MEGABYTE of memory to the OSI. This OSI-68000 system can be used to speed up BASIC, used as a RAM DISK or you can run complete programs in the 68000 with the OSI used only for I/O. A high resolution color graphics board can also be connected to the 68000 board. (See the Tandy 2000 with color graphics to see the kind of resolution that I am talking about.)

At this time the software consists of hooks to BASIC to do the math functions in the 68000, a monitor for the 68000 and several demonstration programs to show the power of the 68000. There is a program called the Hand Assemblers Helper that can be used to learn 68000 assembly language and also to help you assemble short programs.

In the near future there will be a 68000 cross assembler for the OSI. A BASIC like language is now being developed by the manufacturer of the 68000 board that will run between 10 and 100 times faster than Microsoft BASIC on a 6502.

For further information or to be placed on an information newsletter list contact:

David Livesay
ave de la Resistance 6
B4920 Embourg, Belgium

Note-for those interested in 68000 assembly language the Hand Assemblers Helper program is available by itself on either 5 1/4" or 8" disk for $25 including mailing.

published monthly

Editor - Eddie Gieske

Technical Editor - Brian Harston
Circulation & Advertising Mgr. - Karin O. Gieske
Production Dept. - A. Fusselbaugh, Ginny Mays

Subscription Rates	Air	Surface
US		$15
Canada & Mexico (1st class)		$23
So. & Cen. America	$35	$27
Europe	$35	$27
Other Foreign	$40	$27

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 (301) 363-3268

PEEK (65)
The Unofficial OSI Users Journal

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

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

DELIVER TO:

GOODIES for OSI Users!

PEEK (65)
The Unofficial OSI Users Journal

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

( )	**C1P Sams Photo-Facts Manual.** Complete schematics, scope waveforms and board photos. All you need to be a C1P or SII Wizard, just	$7.95 $ _____
( )	**C4P Sams Photo-Facts Manual.** Includes pinouts, photos, schematics for the 502, 505, 527, 540 and 542.boards. A bargain at	$15.00 $ _____
( )	**C2/C3 Sams Photo-Facts Manual.** The facts you need to repair the larger OSI computers. Fat with useful information, but just	$30.00 $ _____
( )	**OSI's Small Systems Journals.** The complete set, July 1977 through April 1978, bound and reproduced by PEEK (65). Full set only	$15.00 $ _____
( )	**Terminal Extensions Package** - lets you program like the mini-users do, with direct cursor positioning, mnemonics and a number formatting function much more powerful than a mere "print using." Requires 65U.	$50.00 $ _____
( )	**RESEQ** - BASIC program resequencer plus much more. Global changes, tables of bad references, **GOSUBs** & GOTOs, variables by line number, resequences parts of programs or entire programs, handles line 50000 trap. Best debug tool I've seen. MACHINE LANGUAGE - VERY FAST! Requires 65U. Manual & samples only, $5.00 Everything for	$50.00 $ _____
( )	**Sanders Machine Language Sort/Merge** for 0S-65U. Complete disk sort and merge, documentation shows you how to call from any BASIC program on any disk and return it or any other BASIC program on any disk, floppy or hard. Most versatile disk sort yet. Will run under LEVEL I, II, or III. It should cost more but Sanders says, "...sell it for just..."	$89.00 $ _____
( )	**KYUTIL** - The ultimate OS-DMS keyfile utility package. This implementation of Sander's SORT/MERGE creates, loads and sorts multiple-field, conditionally loaded keyfiles. KYUTIL will load and sort a keyfile of over 15000 ZIP codes in under three hours. Never sort another Master File.	$100.00 $ _____
	BOOKS AND MANUALS (while quantities last)	
( )	**65V Primer.** Introduces machine language programming.	$4.95 $ _____
( )	**C4P Introductory Manual**	$5.95 $ _____
( )	**Basic Reference Manual** — (ROM, 65D and 65U)	$5.95 $ _____
( )	**C1P, C4P, C8P Users Manuals** — ($7.95 each, please specify)	$7.95 $ _____
( )	**How to program Microcomputers.** The C-3 Series	$7.95 $ _____
( )	**Professional Computers Set Up & Operations Manual** — C2-OEM/C2-D/C3-OEM/C3-D/C3-A/C3-B/C3-C/C3-C'	$8.95 $ _____

( ) Cash enclosed ( ) Master Charge ( ) VISA

Account No. _____ Expiration Date _____

Signature _____

Name _____

Street _____

City _____ State _____ Zip _____

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

POSTAGE MAY VARY FOR OVERSEAS