PEEK (65)

The Unofficial OSI Users Journal

JANUARY 1985
VOL.6, NO.1

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

INSIDE

Column One

Happy New Year! This seems an appropriate time to say thank you to all of our subscribers and advertisers for their support over the past year and to announce our expectations for the coming year. It is the start of our sixth year and already the year is starting off with a flurry of activity on all fronts.

But first, one bit of old business. Please look back to page 14 of the November issue and find the "Reader Survey". The response has been good, but we would like to hear from every subscriber. This is your chance to get your two cents worth in and to tell us what direction you want PEEK to take in the future.

COMDEX is now over and although there was not a lot of new and exciting innovation this time, it was a show that was good to OSI. Attendance and interest were good and the hints of new and better machines (probably of the 68000 family) just around the corner this spring whetted the appetite of attendees. Best guesses are that these new speedsters will be running UNIX and still take OS-U in stride.

Continuing on the OSI front, word has it that the Source Book (of software) is being completely revamped and updated. You know the Work Systems - the packaged deal? The package just got better. Now it is your choice of extras to go with the 235 and 245 machines. The old 12 bit parallel printers are just about gone and instead you can have a Citoh 25 cps if you like. There is also a choice of Esprit terminals (T6110 and T6310), Alloy tape back-up, and P220 DataSouth printers.

You may not have heard it, but there is a board "swap" service plan in effect for dealers to both simplify and expedite repairs. Word has it that the plan may well be expanded to include hard disks too!

Another tidbit, for all users, is that there is a possibility that there might be a price reduction in the cost of OS-U and a return to a reasonable price for "up-grades". We, for one, would like to see "U" competitively priced in the market place and all users using the latest version.

A hardware change has taken place in the 235 line. The 10 MB disk has been replaced with a 13 MB unit for the same price.

In the meantime, the folks at DBI have not been sitting on their hands either. Sales are reported to be brisk with an ever increasing demand for more users on one machine. The box featured last month is rumored to be spawning a big brother with 16 slots. By my count, that probably means up to 14 users and we understand that there is a good chance that it will support up to a pair of 160MB disks too. Now that's no small time play toy!

In their spare time, we understand that the DBI people are also hot on the trail of a 68000 configuration which is a natural jump from the current DB-DOS operating system. It too, is expected to support a OS-U shell.

Now comes a piece of news you probably wish you were not getting. You have often heard us say that PEEK is a labor of love. Well, it is all too true. But no amount of love will pay the printer much less our beloved Post Office which is poised to raise its rates again in February. We have watched our costs very closely and remain intent upon giving you the greatest amount of information for the least amount of money. We have no intentions of forcing the costs of fancy glossy paper on you, but we started in November to "trim" the edges so that you can thumb the pages more easily. We thought this to be a desired improvement, although no one seems to have noticed. The last time that PEEK increased its rates was way back in July of 1981. Not much else has gone unchanged that long. So, effective January 1, 1985, the one year subscription will be $19.00 domestic and other rates as indicated in the copyright block on the inside front cover.

So the year gets started with a bang. Now two manufacturers are hard at it, in a friendly way, which prompts a parting thought for "P" users: Did you ever think about what you would have if you stuck a DB-1 or 515 board in your four-slotter?

Eddie

6502 ASSEMBLY LANGUAGE PROGRAMMING CLASS

PART VII

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

In the last lesson, I described the instructions on the 6502 which perform discreet and familiar mathematical operations, namely addition and subtraction. However, the 6502 instruction set includes additional commands which perform logical and bit manipulative operations that in turn extend the range of mathematical possibilities to the Assembly language programmer.

Let's look at the logical instructions first. There are three such instructions; AND, OR, and EOR (for Exclusive OR). The best way to describe the effect of these instructions is with what are called truth tables.

```
AND            OR             EOR
---            --             ---

0   1          0   1          0   1
+---+---+      +---+---+      +---+---+
0 ! 0 ! 0 !    0 ! 0 ! 1 !    0 ! 0 ! 1 !
+---+---+      +---+---+      +---+---+
1 ! 0 ! 1 !    1 ! 1 ! 1 !    1 ! 1 ! 0 !
+---+---+      +---+---+      +---+---+
```

I have my own ways of thinking of these instructions that you might find useful. I think of "AND" as a filter, since 1's are only allowed to pass through if they are in the same position as the 1's in the filter. I think of "OR" as a combining command in that all of the 1's from both bytes involved are reflected in the result. Finally, I think of "EOR" as an inverter because the 1's in the masking byte cause the bits in the source byte to be inverted - if they were 0's they become 1's and if they were 1's they become 0's.

Finally, we have the bit manipulating instructions of which there are four: ASL, ROL, LSR, and ROR. These commands shift the bits within a memory location or the accumulator one position to the right or left. As I will illustrate below, these instructions always cause one bit to "drop off" and one bit position within the byte being manipulated to become "open". The difference in the "shifting" and "rotating" instructions is in how the open bit is handled. Both types force the bit that dropped off into the Carry Flag of the Status Register. Let's look at the shift instructions first:

```
ASL

---   =====7=====> Carry Flag
      !    +---+---+---+---+---+---+---+---+
      +<= ! 6 ! 5 ! 4 ! 3 ! 2 ! 1 ! 0 ! x ! <- 0
           +---+---+---+---+---+---+---+---+

LSR

---                    Carry Flag <======0========
                                                 !
           +---+---+---+---+---+---+---+---+     !
   0 --> ! x ! 7 ! 6 ! 5 ! 4 ! 3 ! 2 ! 1 ! =>+
           +---+---+---+---+---+---+---+---+

ROL

---   =====7=====> Carry Flag ======>=====>===+
      !                                       !
      !    +---+---+---+---+---+---+---+---+  !
      +<= ! 6 ! 5 ! 4 ! 3 ! 2 ! 1 ! 0 ! x ! <-+
           +---+---+---+---+---+---+---+---+

ROR

---   +====<=====> Carry Flag ======0=========+
      !                                       !
      !    +---+---+---+---+---+---+---+---+  !
      +-> ! x ! 7 ! 6 ! 5 ! 4 ! 3 ! 2 ! 1 ! =>+
           +---+---+---+---+---+---+---+---+
```

You'll note that the numbers in the above tables represent the original position numbers of the bits involved, but they are shown to be in the actual position where they end up after the instruction has been executed. Note also that if you ignore the Carry Flag, the result of movements to the right decrease the value of the byte involved by two and movements to the left increase the value by two. This principle comes to light many times when indexes to a table of 16-bit values are calculated. The bit manipulative instructions are also useful in routines to multiply and divide multiple precision values.

So far in our discussion, we have dealt exclusively with binary math even though we have often relied on the hexadecimal numbering system. Of course, us poor mortals were all taught only to think in base ten or decimal. Here again, the 6502 comes to our rescue. The 6502 has a special mode of operation in which it treats individual bytes as if the high nibble (the 4 most significant bits) were tens instead of sixteens and it does this only for the instructions ADC and SBC. To enter this mode, we must set the Decimal flag in the Status register with the "SED" instruction. Since each byte can only hold a 2 digit hexadecimal number, this same byte can only represent the values 0 through 99 when in the decimal mode. Let's take a look at a quick example of the decimal mode:

```
Normal Mode        Decimal Mode

   18                  18
 + 24                + 24
   --                  --
   3C                  42
```

It is worth noting that in the decimal mode, a byte might easily contain an illegal value. That is to say that whenever either the most or least significant nibble contains a value greater than 9, that value is illegal in the decimal mode and calculations using such illegal values will be erroneous.

The decimal mode is often used in converting binary values into their decimal equivalents. It is also often used to perform accurate floating point calculations to base ten results.

The representation of values in the decimal mode is called Binary Coded Decimal or BCD. The instruction "CLD" clears the Decimal flag and restores normal operation to the 6502.

★

Copyright © 1985 PEEK (65) Inc. All Rights Reserved.
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		$19
Canada & Mexico (1st class)		$26
So. & Cen. America	$38	$30
Europe	$38	$30
Other Foreign	$43	$30

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

BEGINNER'S CORNER

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

SORTING

It's a big subject. Over three dozen sorting algorithms are known. A comprehensive reading list would span over one hundred texts!

Unfortunately, many of these texts are written in 'University English' and deal with such esoterica as 'Sorting with drum storage', not to mention "Amphisbaenic sorting' (something to do with lizards ......!). The micro user is usually left bewildered by the range and complexity of sorting algorithms. A good introduction to the subject would be, 'The Art of Computer Programming, Vol 3 by Knuth. This is not an easy book.

It is a fact that no one sorting method is 'best'. The Bubble Sort is very slow for a list size of 12 or more, unless there is parallel storage and the members of the list are less than an average of log n positions out of place. Quicksort is fast for random lists with over 500 elements. The Insertion Sort is fastest when sorting one element into an already ordered list. It is evident that sorting algorithms must be matched to specific needs.

When choosing a sorting method several factors must be considered. The two factors which affect the speed of every sorting algorithm are; how ordered the list is to begin with, and its length. As a general rule, sorts can be made more efficient but at the cost of using more computer memory. A list may be required to be sorted without actually moving members of the list. The list could be so large that a merging technique is required. The length and complexity of the sorting program could also affect sorting speed. The programming language used and even the hardware could have their influence on speed also.

The micro user must chop through this sorting jungle to find the most suitable algorithm. Fortunately, the search can be considerably narrowed by throwing out all Bubble Sorts and their like, and concentrating on short, RAM only sorts. The OML uses two sorting routines, see Listing 1, lines 1270-1460 (Otaio Mailing List, see June '84 issue). They are the Insertion Sort and the Shell Sort.

The Insertion Sort is only four lines long, 1340-1370. The Shell Sort is effectively only 6 lines long, 1400-1450. The Insertion Sort is simplest to understand. Search forward for an out of order element, then a search back through the list and insert the element in its correct position. The Shell Sort is a little more complicated. Elements compared are a specified distance apart. This distance decreases until only adjacent elements are compared. The actual distance follows the sequence of numbers ..., 511, 255, 127, 63, 31, 15, 7, 3, 1 and is calculated in lines 1390 and 1400. The numbers are chosen so that for any number, it and its predecessor, have no common factors.

```
1 REM LISTING 1
2 :
470 REM PACK FILE
480 PRINT "Is the File SORTED ? ";: GOSUB 310: PRINT Y$: IF A=121 THEN 510
490 IF Y$="" OR A=104 THEN 540
500 PRINT : PRINT "* SORT File first Bub! *": PRINT : GOTO 200
510 PRINT !(28): PRINT TAB( 20)"* PACKING *"
520 Q=Z
530 IF LEFT$(D$(Q,1),2)="ZZ" THEN Z=Z-1: FOR Y=1 TO P: D$(Q,Y)="": NEXT : GOTO 520
540 GOTO 190
550 :
1270 REM SORT FILE
1280 PRINT "SORT on which FIELD # ? ";: GOSUB 310: PRINT Y: B=Y
1290 PRINT : IF Y=0 OR Y>P THEN 1460
1300 PRINT "Is the File partially Sorted on Field"B" ? ";
1310 GOSUB 310: IF A=104 OR A=45 THEN 1460
1320 PRINT !(28): PRINT TAB( 20)"* SORTING *": IF A=110 AND Z>4 THEN 1390
1330 :
1340 FOR Q=2 TO Z: Y=Q: FOR C=1 TO P: X$(C)=D$(Y,C): NEXT C
1350 IF D$(Y-1,B)<=X$(B) THEN 1370
1360 FOR C=1 TO P: D$(Y,C)=D$(Y-1,C): NEXT C: Y=Y-1: IF Y>1 THEN 1350
1370 FOR C=1 TO P: D$(Y,C)=X$(C): NEXT C,Q: GOTO 1460
1380 :
1390 I=(2^INT(LOG(Z)/LOG(2)))-1
1400 I=INT(I/2): IF I<1 THEN 1460
1410 FOR Q=1 TO I: R=Q+I: FOR C=R TO Z STEP I: Y=C
1420 FOR K=1 TO P: X$(K)=D$(Y,K): NEXT K
1430 IF D$(Y-I,B)<=X$(B) THEN 1450
1440 FOR K=1 TO P: D$(Y,K)=D$(Y-I,K): NEXT K: Y=Y-I: IF Y>I THEN 1430
1450 FOR K=1 TO P: D$(Y,K)=X$(K): NEXT K: NEXT C,Q: GOTO 1400
1460 GOTO 190
1470 :
1480 PRINT FILE
1490 INPUT "*  # of copies of each Record ";L: PRINT : IF L<1 THEN 1680
1500 :
1510 SS=0: PRINT : PRINT "Two columns ? ";: GOSUB 310: PRINT Y$
1520 IF A=110 OR A=45 THEN 1570
1530 IF A=104 THEN PRINT : GOTO 1490
1540 PRINT : PRINT "Records must be PACKED. Hit <RETURN> if not."
1550 SS=-1: E=ST+1: TB=40
1560 :
1570 PRINT : PRINT "Device # ? ";: GOSUB 310: IF Y=0 THEN 1680
1580 PRINT Y: V=Y: PRINT : PRINT "Ready ? ";: GOSUB 310: PRINT : PRINT
1590 FOR Q=1 TO Z: FOR X=1 TO L: IF LEFT$(D$(Q,1),2)="ZZ" THEN 1670
1600 :
1610 IF SS=0 THEN 1660
1620 IF Q=E THEN E=Q+ST*2: Q=Q+ST: PRINT #V,F$
1630 IF Q>Z THEN 1680
1640 GOSUB 1920: GOTO 1670
1650 :
1660 GOSUB 1880
1670 NEXT X,Q
1680 V=2: GOTO 190
1690 :
1870 REM Print a Record
1880 PRINT #V,D$(Q,1)TAB( 32)Q: FOR C=2 TO P: PRINT #V,D$(Q,C): NEXT C
1890 PRINT #V: RETURN
1900 :
1910 REM Print a Record in 2 columns
1920 PRINT #V,D$(Q,1)TAB( 32)Q TAB( TB)D$(Q+ST,1)TAB( TB+32)Q+ST
1930 FOR C=2 TO P: PRINT #V,D$(Q,C) TAB( TB)D$(Q+ST,C): NEXT C: PRINT #V
1940 RETURN
```

Strictly speaking, there is no such thing as a 'Shell Sort'. The algorithm can be applied to any 'compare and exchange' algorithm, and is here applied to the 'Insertion Sort'. The Shell Sort is best for a random list with at least 5 elements. The Insertion Sort is best for a list with only a few out of order elements.

Looking at Listing 1 again, notice that, because an array is being used to store the list, it is easy to pin-point the field on which the list is to be sorted - see lines 1300, 1350 and 1430. During the sort, when records need to be swapped, every field of every record is swapped. This is time-consuming. The sort would run considerably faster if only an index to the list had to be sorted. However, this extra speed has a cost. More computer memory is used and the program has to be longer and more complex to cope with the insertion and deletion of records.

When choosing a sorting algorithm, the wise computerist tests it with actual data before making a final choice. Text books on the subject are best treated as guides only.

For many applications two sorting programs are required if speed is important. One program would serve random lists and the other would be for nearly-ordered lists.

PRINTING

This part of the OML is complicated by the option on two-column printout. The option was written in to provide a tidy two-column printout of databases which are not a mailing list. This option is flagged by the variable 'SS'. The number of records per column is held in 'ST' (see line 130). Line 1620 sends a form-feed to the printer after a page of records has been printed in two column format. Notice that in line 1610, before each record is printed, the 'SS' flag is checked. This is because the single and double column printout routines are mixed together. For program clarity, these two routines should be separate. The cost of doing this is a longer, slightly slower program.

PACKING

Packing means removing something unwanted from a file. For example, a BASIC program can be packed by removing all blanks and REM statements. A file of records is packed when deleted records are removed from the file. In the OML the process has three stages. First, a record is 'Deleted' (but not removed) from the file by writing 'ZZ' as the first two characters of the first field. This is useful for subscription lists. Late payers are not sent the latest mailing but remain in the file for quick 'undeletion' as soon as their sub is received. If the sub is not received the record can be removed from the file by the next pair of actions. First, records are 'Sorted' on field 1 and, Hey Presto, all deleted records appear at the end of the file - cunning those 'ZZ's'! Now 'Packing' the file merely removes all those records marked with a 'ZZ'. The file is tidy once again.

★

WAZZAT CORNER!

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

PRINT USING IN BASIC

A PRINT USING command is used to format numbers on the decimal point. In some BASICs the command can be used to format graphics. Very few versions of BASIC have a 'PRINT USING' command. OSI implemented the command in OS65D 3.3, and it is particularly easy to use. Obviously, a machine code version of PRINT USING is preferable to a BASIC program equivalent if only because string garbage collect is avoided. But a BASIC program can still be very effective, particularly if there is no other choice! Such a program is presented here.

The subroutine that does all the work is in lines 190 to 240 of the program. Numbers are right-justified on the decimal point. This can be done anywhere on a line as set by the value of 'T'. Leading and trailing zeros are correctly printed, no matter how the number is input in line 40. The sign of the number is also correctly formatted. Line 190 could obviously be made an earlier line, for example line 20.

The most efficient way to manipulate numbers in BASIC is with string functions. The first step then, in line 200, is to convert the number 'N' into string 'M$'. Multiplying 'N' by 100 guarantees trailing zeros, if they are there. Next, the sign of the number is copied into 'S$'. In line 210 the sign is deleted from 'M$'. Leading zeros are then attached, if required. Finally, in line 230, the number is printed, right justified on the decimal point. Line 240 forces a very brief garbage collect. Doing this removes the need for BASIC to do a lengthy garbage collect when string space becomes full. Several numbers could be formatted on one line by changing the value of 'T' as required. Also, a semi-colon would need to be placed at the end of line 230 and a 'PRINT' done after the printing of each line of formatted numbers.

Listing 2 illustrates a good programming practice - writing programs which are as general as possible, with input-error trapping. This 'print-using' program gives the user a choice on how many digits are to follow the decimal point. If the program is to be run in ROM BASIC a change is required in line 50. Change the number to 1 000 000, ROM BASIC arithmetic is limited to 6 significant figures. The tabbing variable is now 'P', in line 5.

See listings page 23 ★

BASIC RULES OF RESPONSIBLE PROGRAMMING

By: Luis E. Robles
P. O. Box 2036
Hato Rey, P.R. 00919

First of all, I would like to commend you on your excellent newsletter to which we have been subscribing since day one. Throughout the years it has proved to be very useful and enlightening, and most of all consistent. CONGRATULATIONS!

Puerto Rico Computer, is one of the few remaining "old guarde" distributors. We have been around since OS65U version 1.1, devoting ourselves mostly to the selling, programming, maintenance and installing of commercial applications. Besides the bread and butter applications of Billing, Inventory, Accounts Receivable, Accounts Payable, Sales Analysis, General Ledger and Payroll, we have developed also such exoteric applications as Insurance Premium Financing, Insurance Agency Accounting, Custom Brokers, Wholesale Lab.,Test Billing, Leasing Companies and Time Management Accounting and Invoicing.

In our quest of many years for the painless and easy to maintain software, our experience with packages has been nightmarish (!) to say the least, making us retrieve our old faithful software more times (and money) than I want to remember. Finally, we reached the conclusion that we must have been doing something right, and that the grass was indeed greener on our side. We have been misled by well advertised, SOMETIMES WELL DOCUMENTED, software packages, that lack what we call the basic rules of "responsible programming".

We have divided our notes in six main topics which we feel are the keys of our success in the software environment. These are:

1. File Structure and Sorting

2. Query & Creating Programs

3. Listing Programs

4. Updating Programs

5. Merges

6. Menus

FILE STRUCTURE AND SORTING

First of all, we decided to

DBi, inc.
p.o. box 21146 ● denver, co 80221
phone (303) 428-0222

SPECIAL PURCHASE on hard disk drives
SPECIAL PRICES on **DBI BUSINESS SYSTEMS**
RUNS DB—DOS & OS—65U PROGRAMS*

DBI 420SE

(4) DB-1 MULTI-PROCESSING BOARDS
★ TRUE PARALLEL/MULTI-TASKING
★ ALL USERS RUN AT 2 MEGAHERTZ
(1) DS-1 SCSI HOST ADAPTER
★ W/BATTERY BACKED-UP REAL TIME CLOCK
(1) DP-1 UNIVERSAL PRINTER BOARD
★ 4 RS-232 SERIAL INTERFACES
★ 2 CENTRONICS COMPATIBLE INTERFACES
(1) FAST 20 MEGABYTE HARD DISK
(1) 318K BYTE FLOPPY DISK
(1) INTELLIGENT SCSI CONTROLLER
★ W/ERROR CHECKING AND CORRECTION

LIST $10,490
ONLY! $6,695
SAVE! $3,795

DBI 220SE Same as 420SE except Two Users

LIST $7,900
ONLY! $6,395
SAVE! $1,505

DBI 440SE Same as 420SE With 40 Megabyte Hard Disk

LIST $13,100
ONLY! $8,895
SAVE! $4,205

DBI 240SE Same as 440SE except Two Users

LIST $10,510
ONLY! $8,595
SAVE! $1,915

* OS-65u IS A TRADEMARK OF OHIO SCIENTIFIC, INC.

QUANTITIES ARE LIMITED **PLEASE DON'T DELAY!**

stick to OSI's DMS standards, with a few changes. The only requisite to understand what follows is to be familiar with OSI DMS file structure.

We concluded that there were basically three field types, which are: string, numeric, and date. So we added an S, N, and a D to the label of each field which, for convenience, we made 12 characters long (please, keep in mind these were prehistoric times before the Extended Input Capability).

On the issue of sorting we decided that we were going to sort only key files and leave the master files untouched which turned out to be a wise decision in an era of brownouts and power failures. But instead of using type 11 key files format, the DMS standard, we left it as a type ten with one field. Then the sorting order was stringed or concatenated. For example: If we wanted to sort a file by credit limit and customer number, we constructed a key file as follows:

01 +
(CREDIT LIMIT)

23456 +
(CUSTOMER NUMBER)

12345678 +
(DISK ADDRESS, ALWAYS 8 CHARACTERS LONG)

KEY = 012345612345678

QUERY AND CREATING PROGRAMS

Our next logical step was to create a standard program that could handle these modified DMS files, and, by using S, N, and D indicators, edit the fields as the operator was entering them. So as soon as you have created the file, with DBMFCR (remember?) we could immediately provide the various options of Creating and Query, Listings, Updating, Merging, and Menus programs. Probably you are asking, "Why not use the DMS package developed by OSI?" Well, we felt these programs were good tools but needed the editing, previously mentioned (S, N, and D) and also some additional controls to work properly in an office environment, these were:

1. The first field was always going to be a string field and the "key" by which to make sure the record in question was going to be uniquely identified, i.e., Customer number, Inventory number, etc., or a control number generated by the program, if we could not identified the record "uniquely".

2. By using this first unique field whenever we are creating a record, the duplication of customers number, for instance, could be avoided by making a "find" before proceeding with the other fields.

3. Also, by using this first field, whenever quering, we could search the file and if the record already has been created, display the other fields of the record, or indicate that it doesn't exist.

4. Always make sure fields are of the proper size by padding it with right hand spaces. Also, always write in a record boundary (see formula below).

```
     /- INDEX(1) - BODF -/
INT  /  ---------------  /  =
     /- RECORD LENGTH   -/

        INDEX (1) - BODF
        ---------------
        RECORD LENGTH
```

5. Use the "^" sign to precede the first field (the key). This will simplify the find (nevertheless, check for record boundary).

6. To erase a record, substitute the "^" with a "|". By the way, do it by programming, by no means leave it to the operator.

7. By using the "|" for erased records you could easily recover them in case of emergency.

8. Whenever the file gets full (EODF=BODF), search for the erased records by searching for the "|" and reuse the space.

9. All numeric fields are going to be nine characters long. Why this waste? Well as you know, OSI is hot into hard disks so if you recommended the proper hard disk size to your customer, no problem with volume. Therefore, by defining numeric fields with a length of nine, you will be sure to reach a field overflow only when reaching OSI's maximum number handling capacity. Also you will avoid the problem that so many many packages have, of finding out you are overflowing a field at the time you are updating it. What to do then? Redefine the file, erase the old one, run a packer, and then return to the exact point where you were updating (whew!).

What then, if by using our method, you overflow a nine character long field? Truncate and give a message (a very common occurrence in the construction and leasing industries where you deal with large figures).

10. Following we will mention several screen formats or variations of the query/ create programs as we have modified them. These are:

a. Standard Query/Create, first field is the Key. Labels are posted left hand side of screen.

b. Standard Query/Create, first field is the Key. Labels are posted in two columns: Notice customer name is left on a line by itself since usually this second field (or item description) will overlap the right hand column (usually over 30 characters long).

c. Multi Screen- more than one screen

d. Batch control- first field is a control number generated by the computer to uniquely identify the record.

e. Order entry- The screen is divided into top half and bottom half. The top half includes the invoice No., customer No., date, salesman, ship to, sold to address, and "terms" line. The bottom half includes the item No. description, amount order, back order, unit price, price extension, discount, total amount, etc.

LISTING PROGRAMS

Basically you need four types of Listing Programs:

1. Dumping a file as it is.

2. Dumping a file by first referring to a sorted Key file. It is usually requested to break at totals by category (usually not more than two subtotals, i.e., subtotals by salesman, by item, and final totals).

3. Aging and Statement printing. The previous program (No. 2) could be used as the base to build these, though the key file will have to "merge" two masters files, (customer and transactions), and keep track of which file

the key points to.

4. Invoice printing: This printing program also merges two masters; the sold to, ship to, and billing information with the detail or item lines file. The aging and statement program could be the basis of this invoice printing program.

We developed a utility that works nicely to run fast, printing programs Types 1 and 2 (above). We eliminated the instructions that print the heading and the variables from programs 1 and 2. Then we made a program that asks you, one by one, each field heading and variable name. Convert these inputs to printing instructions of heading and variables. Then "list" it to a file and merge these heading and variable names (Flags 13 and 14) with the printing programs.

UPDATING PROGRAMS

There are three basic updating programs.

1. Adding transaction to a file, i.e., you want to merge a batch of payments with a transaction master file. First make sure you have enough room to accommodate the new transactions by testing the master file EODF. Second, after adding the transactions, don't forget to "Erase" the temporary file.

INDEX<1>=9: Print % 1, EODF

Try to do this EODF updating right at the end of the program to minimize a call to the programmer. In other words, if the update program is interrupted for any reason, most probably just by running it again everything will be Ok. After updating the EODF, always post the words "accumulated" and amount at the end of the batch listing. This will save you a lot of guessing if anything goes wrong.

2. Eliminating transactions from a permanent file - If for example the "permanent" transactions file needs to be cleaned up of all accounts that balance to zero, we have found out that the best way is to create a twin or image file and pass the transactions that DON'T zero balance. Then when finished interchange the file names with a modified Rename program. By the way - you will need intermediate file names to avoid hanging up in a Duplicate File name error. By using this technique you get a bonus. You have an internal backup of last month's transactions file, and this alone certainly can save the day!

3. Updating totals in a master file - As an example, quarter to date and/or year to date figures in a payroll master file. As in the previous example, create a twin or image file and copy the payroll master file to it, write a flag in the control file stating that an update is in process then proceed to update the master file. If the update is not succesfully completed "lock the system" and recopy the master file from the image file. Then proceed with the update again. This technique saves a lot of time when the update is interrupted since it avoids the painful process reconstructions and balancing totals. Also, as before, you get a bonus since you have an internal back up of the master file previous to the update.

MERGES

A simple program that either writes and/or merges "key files". Avoid writing on a "one by one" basis, i.e., read a master file record and write a key file record. That really ties up the system. Instead, dimension variables to the maximum. I feel a good guess is 3584 divided by the record length, i.e., 3584/10 = 358:DIM A$(358). Gobble up all the master records you can from one read. Construct the keys using dimensionable variables and write all the keys in one burst. You will be surprised by the speed!

MENUS

Last but not least, remember most of the time the menu is what sells. Make it look attractive and functional (though this is very subjective) avoid displaying all Query; programs together and all the listing programs together and the same applies to the Sorting programs. Even though this looks neat it tends to confuse the operator since he has to look all over the screen to find the different related program.

OSI COMPUTER & PARTS CLOSE-OUT! WOW!

ALL NEW

C3-OEM 2MHz 56K	$1299.00
C3-D w/10 MB H.D.	$2499.00
C2-OEM 1 MHz 48K	$ 999.00
CA-9 CENTRONICS PRINTER INTERFACE	$ 49.00
+ 5, + 12, − 9 POWER SUPPLY	$ 39.00
+ 24 v. 2.4 v. POWER SUPPLY	$ 29.00
+ 5 v. 3 AMP. POWER SUPPLY	$ 19.00
CM-2 4K EXEC MEMORY	$ 39.00
CM-10 8K EXEC MEMORY	$ 49.00
8" SIEMENS DISK DRIVE	$ 149.00
OUR 9592 DISK INTERFACE (Super)	$ 199.00
510 REV C CPU BOARD (CP-1) 2 MHz	$ 99.00
CM-9 24K LO POWER 2 MHz MEMORY	$ 79.00
8 SLOT BACKPLANES STD OSI	$ 39.00
C3-SAMS TECHNICAL MANUAL	$ 20.00

CLEAN—USED

C3-OEM DEMO 48K 2 MHz	$ 995.00
C3-B 80 MB 56K	$3000.00
C3-B 80 MB 104K	$3500.00
CD-74 DISK W/CONTROLLER	$2750.00
592 DISK INTERFACE BOARD	$ 150.00

AS IS

8" SIEMENS DISK DRIVES AS IS	$ 50.00
CM3-520 REV B 16K MEM AS IS	$ 20.00
CM3-520 REV C 16K MEM AS IS	$ 25.00
502 CPU PARTIALLY POPULATED	$ 10.00
ALL OSI COLOR USER MANUALS	$ 1.00
ALL OSI C1P C4P C8P SAMS TECH MAN	$ 5.00
C3-OEM CHASSIS w/POWER SUPPLIES + BUS	$ 350.00
530 CM-4 16K 1 MHz DYNAMIC MEMORY	$ 10.00

(714)951-4648

ORDER FOR SPARES WHILE YOU STILL CAN! SPACE-COM 22991 LA CADENA, LAGUNA HILLS, CA 92653

HOW TO SOLVE THE SEARCH FOR LINE NUMBER PROBLEM

(For disk based systems)

By: David Livesay
Ave de la Resistance 6
b4920 Embourg, Belgium

One of the well known problems with Microsoft BASIC is that when the interpreter finds a GOTO or GOSUB statement it starts to search for the correct line and in some cases must look at each line in the program. For example, if we have a 200 line program and at line 1 we find a GOTO 200 statement, the BASIC interpreter will search through 199 lines to find line number 200. If on line 100 we have a GOTO 1, the interpreter will look at line 101 and discover that it is greater than the line number that is being searched for. Whenever this occurs, the search goes back to the beginning of the program. The interpreter will have looked at 2 lines to find line 1. Obviously, the placement of the subroutines can make a big difference in the run time of a program. There have been many articles published that suggest putting the subroutines at the beginning of the program to speed up your programs. The following describes a technique for speeding up the search for line number routine in the BASIC interpreter.

In connection with their work on interfacing a Motorola 68000 to the Apple computer, the people at Digital Acoustics developed a solution to this problem. The solution that I show here is an adaptation that I developed while working on interfacing the Digital Acoustics 68000 board to the OSI.

The solution to this "search for line number" problem is to replace the normal search routine with one that looks up the address of the correct line in a table. The following must be done:

1. Space for a table must be provided, - this must be two bytes for each line of the program.

2. The program must be numbered, starting with 0 with steps of 2, i.e., 0, 2, 4, 6, 8 ...

3. Before the first GOTO or GOSUB is encountered, the BASIC program must call a machine language program that builds the table of addresses. This program also checks that

MEMORY DUMPS

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
3A70   F8 4C EC 2C 00 00 00 00 00 7F 3A 37 3D 01 00 AC
3A80   3A 00 00 4C 4D AB 32 35 35 3A 48 4D AB 31 38 35
3A90   3A 8E 20 24 42 39 46 46 3A 52 45 4D 20 46 4F 52
3AA0   20 34 38 4B 20 53 59 53 54 45 4D 00 DE 3A 02 00
3AB0   96 31 33 33 2C 48 4D 3A 96 31 33 32 2C 4C 4D 3A
3AC0   96 31 32 39 2C 48 4D 3A 96 31 32 38 2C 4C 4D 3A
3AD0   8E 20 50 52 4F 54 45 43 54 20 4D 45 4D 00 0D 3B
3AE0   04 00 94 21 22 43 41 20 42 46 30 30 3D 34 39 2C
3AF0   31 22 3A 96 35 37 34 2C 31 32 30 3A 96 35 37 35
3B00   2C 31 39 31 3A 8E 20 24 42 46 37 38 00 40 3B 06
3B10   00 58 AB B0 28 58 29 3A 84 22 44 4F 20 59 4F 55
3B20   20 57 41 4E 54 20 54 4F 20 55 53 45 20 54 41 42
3B30   4C 45 20 46 4F 52 20 47 4F 54 4F 22 3B 41 24 00
3B40   70 3B 08 00 8A 20 C1 28 41 24 2C 31 29 AB 22 59
3B50   22 20 A0 20 96 20 34 39 31 34 35 2C 30 3A 88 31
3B60   32 3A 8E 20 24 42 46 46 39 3D 34 39 31 34 35 00
3B70   7E 3B 0A 00 96 20 34 39 31 34 35 2C 31 00 91 3B
3B80   0C 00 81 49 AB 31 9D 33 30 30 30 3A 88 31 35 30
3B90   00 9B 3B 0E 00 82 49 00 9E 3B 10 00 80 00 A6 3B
3BA0   12 00 88 31 34 00 AC 3B 14 00 8E 00 B2 3B 16 00
3BB0   8E 00 B8 3B 18 00 8E 00 BE 3B 1A 00 8E 00 C4 3B
3BC0   1C 00 8E 00 CA 3B 1E 00 8E 00 D0 3B 20 00 8E 00
3BD0   D6 3B 22 00 8E 00 DC 3B 24 00 8E 00 E2 3B 26 00
3BE0   8E 00 E8 3B 28 00 8E 00 EE 3B 2A 00 8E 00 F4 3B
3BF0   2C 00 8E 00 FA 3B 2E 00 8E 00 00 3C 30 00 8E 00
3C00   06 3C 32 00 8E 00 0C 3C 34 00 8E 00 12 3C 36 00
3C10   8E 00 18 3C 38 00 8E 00 1E 3C 3A 00 8E 00 24 3C
3C20   3C 00 8E 00 2A 3C 3E 00 8E 00 30 3C 40 00 8E 00
3C30   36 3C 42 00 8E 00 3C 3C 44 00 8E 00 42 3C 46 00
3C40   8E 00 48 3C 48 00 8E 00 4E 3C 4A 00 8E 00 54 3C
3C50   4C 00 8E 00 5A 3C 4E 00 8E 00 60 3C 50 00 8E 00
3C60   66 3C 52 00 8E 00 6C 3C 54 00 8E 00 72 3C 56 00
3C70   8E 00 78 3C 58 00 8E 00 7E 3C 5A 00 8E 00 84 3C
3C80   5C 00 8E 00 8A 3C 5E 00 8E 00 90 3C 60 00 8E 00
3C90   96 3C 62 00 8E 00 9C 3C 64 00 8E 00 A2 3C 66 00
3CA0   8E 00 A8 3C 68 00 8E 00 AE 3C 6A 00 8E 00 B4 3C
3CB0   6C 00 8E 00 BA 3C 6E 00 8E 00 C0 3C 70 00 8E 00
3CC0   C6 3C 72 00 8E 00 CC 3C 74 00 8E 00 D2 3C 76 00
3CD0   8E 00 D8 3C 78 00 8E 00 DE 3C 7A 00 8E 00 E4 3C
3CE0   7C 00 8E 00 EA 3C 7E 00 8E 00 F0 3C 80 00 8E 00
3CF0   F6 3C 82 00 8E 00 FC 3C 84 00 8E 00 02 3D 86 00
3D00   8E 00 08 3D 88 00 8E 00 0E 3D 8A 00 8E 00 14 3D
3D10   8C 00 8E 00 1A 3D 8E 00 8E 00 20 3D 90 00 8E 00
3D20   26 3D 92 00 8E 00 2C 3D 94 00 8E 00 34 3D 96 00
3D30   89 31 34 00 00 00 01 4C 4D 88 7F 00 00 00 48 4D
```

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
BA00   7E 3A AB 3A DD 3A 0C 3B 3F 3B 6F 3B 7D 3B 90 3B
BA10   97 3B 9D 3B A5 3B AB 3B B1 3B B7 3B BD 3B C3 3B
BA20   C9 3B CF 3B D5 3B DB 3B E1 3B E7 3B ED 3B F3 3B
BA30   F9 3B FF 3B 05 3C 0B 3C 11 3C 17 3C 1D 3C 23 3C
BA40   29 3C 2F 3C 35 3C 3B 3C 41 3C 47 3C 4D 3C 53 3C
BA50   59 3C 5F 3C 65 3C 6B 3C 71 3C 77 3C 7D 3C 83 3C
BA60   89 3C 8F 3C 95 3C 9B 3C A1 3C A7 3C AD 3C B3 3C
BA70   B9 3C BF 3C C5 3C CB 3C D1 3C D7 3C DD 3C E3 3C
BA80   E9 3C EF 3C F5 3C FB 3C 01 3D 07 3D 0D 3D 13 3D
BA90   19 3D 1F 3D 25 3D 2B 3D 00 00 00 00 00 00 00 00
```

DEMO

```
0 LM=255:HM=185:REM $B9FF:REM FOR 48K SYSTEM
2 POKE133,HM:POKE132,LM:POKE129,HM:POKE128,LM:REM PROTECT MEM
4 DISK!"CA BF00=49,1":POKE574,120:POKE575,191:REM $BF78
6 X=USR(X):INPUT"DO YOU WANT TO USE TABLE FOR GOTO";A$
8 IF LEFT$(A$,1)="Y" THEN POKE 49145,0:GOTO12:REM $BFF9=49145
10 POKE 49145,1
12 FORI=1TO3000:GOTO150
14 NEXTI
16 END
18 GOTO14
20 REM
   .
   .
   .
   .  LINES 20-146 ARE ALL THE SAME
   .
   .
   .
   .
   .

146 REM
148 REM
150 GOTO14
```

ASSEMBLY LISTING

```
0000          ; S E A R C H   F O R   L I N E #   P R O G R A M
0000          ; FOR OS-65D
BF60             *=$BF60
BF60             POKER=$19          :TEMP PNTR FOR LINE# TO BE FOUND
BF60             BPTR=$C7           :BASIC PNTR CURRENT PROGRAM ADDRES
BF60             SOB=$78            :START OF BASIC PROGRAM
BF60             UPA=$E3            :UTILITY POINTER A
```

Continued

```
  80 BF60             UPB=$F5          :UTILITY POINTER B
  90 BF60             UPC=$F7          :UTILITY POINTER C
 100 BF60             BLIM=$BFFE       :TEMP LINE# COUNTER TO BUILD TABLE
 110 BF60             TBL=$BA          :HIGH BYTE FOR TABLE ADDRESS
 120 BF60             SYNERR=$0E1E     :LOCATION FOR SYNTAX ERR ROUTINE
 130 BF60            ;
 140 BF60            ;**************************************************
 150 BF60            ;********SEARCH BASIC FOR A GIVEN LINE#***********
 160 BF60            ;**************************************************
 170 BF60            ;
 180 BF60 A5 19       GOTO   LDA   POKER        :*****************
 190 BF62 85 F7              STA   UPC
 200 BF64 18                 CLC                :SET UPC TO POINT TO THE
 210 BF65 A5 1A              LDA   POKER+1      :ADDRESS OF THE LINE#
 220 BF67 69 BA              ADC   #TBL
 230 BF69 85 F8              STA   UPC+1        :*****************
 240 BF6B A0 00              LDY   #$00         :*****************
 250 BF6D B1 F7              LDA   (UPC),Y
 260 BF6F 85 C7              STA   BPTR         :FETCH ADDRESS TO BPTR
 270 BF71 C8                 INY
 280 BF72 B1 F7              LDA   (UPC),Y
 290 BF74 85 C8              STA   BPTR+1       :*****************
 300 BF76 38                 SEC                :INDICATES ADDRESS FOUND
 310 BF77 60                 RTS                :RETURN
 320 BF78            ;
 330 BF78            ;
 340 BF78            ;**************************************************
 350 BF78            ;******BUILD A TABLE OF BASIC LINE ADDRESSES******
 360 BF78            ;******LINE NUMBERS MUST BE 0, 2, 4, 6, ETC.******
 370 BF78            ;**************************************************
 380 BF78            ;
 390 BF78 A9 00       BUILD  LDA   #$00         :****************
 400 BF7A 8D FE BF           STA   BLIM         :SET 1ST LINE# = 0
 410 BF7D 8D FF BF           STA   BLIM+1       :****************
 420 BF80 85 F5              STA   UPB          :****************
 430 BF82 A9 BA              LDA   #TBL         :SET ADDR TABLE PTR = TBL
 440 BF84 85 F6              STA   UPB+1        :****************
 450 BF86 A5 78              LDA   SOB          :****************
 460 BF88 85 E3              STA   UPA          :SET POINTER UPA TO
 470 BF8A A5 79              LDA   SOB+1        :THE START OF BASIC
 480 BF8C 85 E4              STA   UPA+1        :****************
 490 BF8E            ;
 500 BF8E            ;***PLACE JMP INSTRUCTION IN BASIC***
 510 BF8E            ;***JMP TO TEST FOR USE TABLE FLAG***
 520 BF8E            ;
 530 BF8E A9 4C              LDA   #$4C
 540 BF90 8D A9 08           STA   $08A9        :STORE JMP INSTRUCTION
 550 BF93 A9 EB              LDA   #$EB
 560 BF95 8D AA 08           STA   $08AA        :STORE LOW ADDRESS
 570 BF98 A9 BF              LDA   #$BF
 580 BF9A 8D AB 08           STA   $08AB        :STORE HIGH ADDRESS
 590 BF9D            ;
 600 BF9D            ;
 610 BF9D            ;***REPEAT THIS LOOP FOR EVERY BASIC LINE#***
 620 BF9D            ;
 630 BF9D            ;
 640 BF9D A0 01       BILD1  LDY   #$01         :PTR TO HI LINK BYTE
 650 BF9F B1 E3              LDA   (UPA),Y      :FETCH THE HIGH LINK BYTE
 660 BFA1 F0 44              BEQ   GDONE        :DONE IF ZERO
 670 BFA3 C8                 INY                :****************
 680 BFA4 B1 E3              LDA   (UPA),Y
 690 BFA6 CD FE BF           CMP   BLIM
 700 BFA9 D0 08              BNE   ERR          :CHECK WHETHER THE LINE #
 710 BFAB C8                 INY                :IS IN CORRECT SEQUENCE
 720 BFAC B1 E3              LDA   (UPA),Y
 730 BFAE CD FF BF           CMP   BLIM+1
 740 BFB1 F0 03              BEQ   OK           :****************
 750 BFB3 4C 1E 0E    ERR    JMP   SYNERR       :SYNTAX ERROR ON WRONG #SEQ
 760 BFB6 A0 00       OK     LDY   #$00         :****************
 770 BFB8 A6 E4              LDX   UPA+1
 780 BFBA 38                 SEC
 790 BFBB A5 E3              LDA   UPA          :STORE THE ADDRESS OF
 800 BFBD E9 01              SBC   #$01         :THE BASIC LINE # AFTER
 810 BFBF 91 F5              STA   (UPB),Y      :SUBTRACTING ONE FOR LATER
 820 BFC1 B0 01              BCS   AA           :USE WITH "GETCHR"
 830 BFC3 CA                 DEX
 840 BFC4 C8          AA     INY
 850 BFC5 8A                 TXA
 860 BFC6 91 F5              STA   (UPB),Y      :*****************
 870 BFC8 88                 DEY                :SET Y =0
 880 BFC9 18                 CLC                :*****************
 890 BFCA A5 F5              LDA   UPB
 900 BFCC 69 02              ADC   #$02
 910 BFCE 85 F5              STA   UPB          :ADD #2 TO THE TABLE POINTER
 920 BFD0 8D FE BF           STA   BLIM         :UPB AND TO THE REF LINE#
 930 BFD3 D0 05              BNE   BILDA
 940 BFD5 E6 F6              INC   UPB+1
 950 BFD7 EE FF BF           INC   BLIM+1       :*****************
 960 BFDA B1 E3       BILDA  LDA   (UPA),Y      :*****************
 970 BFDC AA                 TAX
 980 BFDD C8                 INY                :FETCH THE LINK (PTR TO NEXT
 990 BFDE B1 E3              LDA   (UPA),Y      :BASIC LIN), STORE IN UPA
1000 BFE0 85 E4              STA   UPA+1
1010 BFE2 86 E3              STX   UPA          :*****************
1020 BFE4 4C 9D BF           JMP   BILD1        :NEXT LINE#
1030 BFE7 EE FE BF    GDONE  INC   BLIM         :INCREMENT THE LIMIT BY ONE
1040 BFEA 60                 RTS                :DONE; RETURN TO BASIC
1050 BFEB            ;
1060 BFEB            ;
1070 BFEB            ;**************************************************
1080 BFEB            ;************** TEST FOR USE TABLE ****************
1090 BFEB            ;**************************************************
1100 BFEB            ;
1110 BFEB AD F9 BF    TEST   LDA   FLAG         :TEST FOR USE TABLE FLAG
1120 BFEE F0 06              BEQ   JGOTO
1130 BFF0 20 0A 09           JSR   $090A        :FLAG NOT SET RETURN BASIC
1140 BFF3 4C AC 08           JMP   $08AC
1150 BFF6 4C 60 BF    JGOTO  JMP   GOTO         :USE TABLE
1160 BFF9 EA          FLAG   NOP
```

the lines are correctly numbered and reports an error if the line numbers are not correct. After the table is built, the first two bytes will contain the starting address of line number 0 (actually the address minus one), the second two bytes will contain the address of line number two, etc...

4. A second machine language routine is needed to find the address of the line whenever either a GOSUB or GOTO is encountered. In order to call this subroutine, three bytes of code are changed in the BASIC interpreter. When the new code is called, the routine first checks a flag to determine if the table should be used. If the flag is set, the routine will return to normal BASIC. IF the flag is not set, the routine will fetch the address of the correct line from the table, and then return to BASIC to begin executing the correct line number.

The principle benefit of the above technique is that long programs can run much faster. The other benefit is that the program can be organized anyway the programmer desires without affecting the speed at which the program runs.

THE NEW SEARCH FOR LINE NUMBER PROGRAM

In order to understand how the new search for line number program works, we need to understand the structure of a line of BASIC. Each line of BASIC has the following structure:

```
    PP PP           nn nn     bb bb ...     00

  NEXT LINE         LINE #    BASIC CODE    NULL
(LO BYTE, HI BYTE)            (TOKENS
                              AND ASCII)
```

If we look at the memory dump of the "DEMO" program, we find in line No. 0, the first address of the next line number at $3A7F. This address is $3AAC. At $3AAC we find the second address which is $3ADE, etc.. In this way, there is a link from one line to the next.

The routine in the Assembly listing from lines 390 to 1040 is the routine that creates the table. This is the routine that must be called from the BASIC program. In addition to building the address table, this routine also changes the BASIC code at the GOTO function so that anytime a GOTO is found (GOSUB also uses the GOTO function), the BASIC interpreter will jump to the TEST FOR FLAG routine of this program at line 1110 (this will be discussed later). If you look at the memory dump of locations $BA00 to $BAA0, you will see the table of addresses that the build table routine creates. Remember that the addresses in the table are always one less than the actual address of the line (BASIC likes it that way).

After the table is built and the program is run, anytime a GOTO is found, the BASIC interpreter will cause a jump to the "TEST FOR USE TABLE" routine. Lines 1110 and 1120 check the flag byte to determine if the table should be used to perform the search for line number function. If the byte has a value of zero, the program jumps to the new routine at line 180 to find the address in the table. If the flag byte is not zero, then the program returns to BASIC to find the line number. Note that the code at lines 1130 and 1140 duplicate the code starting at $08A9 in the BASIC interpreter. This is necessary because we modified the code at this location to cause the jump to this new machine code program.

The code that finds the address in the table is actually very short. Starting at line 180, the low byte of the line to be searched for is loaded from $19 (where BASIC has placed it) into utility pointer C (UPC). Lines 210 thru 230 load the high byte and add it to the high byte of the table address. Now what does this do for us? For example, if we are looking for line #2, we find that line 180 loads a $02 and places it in UPC, then it will load $00 and add it to the high byte of the table address ($BA) and stores it in UPC+1. We now have $BA in UPC+1 and $02 in UPC, this you will notice is the location in the table for the address of line #2. We then load #00 into the Y register and then use UPC to fetch the low and the high byte and store them in the BASIC pointer for the current line address. We then return to BASIC and the execution of the program continues as before until a new GOTO is found.

The next listing is of a BASIC program called "DEMO". This program simulates a program where a GOTO is 68 lines away and the GOTO is executed 3000 times. Line 2 loads the machine language program and line 4 changes the upper limit of BASIC and sets up the address for the USR(X) function. The POKEs in lines 8 and 10 either set or clear the use table flag. Obviously, the POKE values used in lines 2, 4, 8, and 10 need to be adjusted for the location of your machine code program.

If you run this program and don't use the table, it will run in 27 seconds. If you use the look-up table, it will run in 9.4 seconds. If you change the GOTO 150 in line 12 to GOTO 18 to simulate a situation where the jump location is close, the program will run in 12.6 seconds. From this it can be seen that using this look-up table program can improve the execution times even of programs that are written with the subroutines at the beginning.

If you load and run another program, you must remember to POKE the load flag with a value other than zero, otherwise, the new program will not run. Also, if you make a change to the program, you will need to rebuild the table. If on the other hand, you don't change the program you can restart the program by entering at a location after the build table routine is called. For example, if you want to restart the "DEMO" program, you can restart at line 12.

This routine can be used to speed up long programs that have many subroutines and/or branches frequently. For any application where memory space is critical, you should set up the table space to minimize the usage of memory. Although I have not tried this with OS65U, there is no reason why this technique would not work. Due to a bad memory board, I was limited to 40K when I wrote it, and thus I placed the table at $BA00 and the machine language program at $BF60. This provides space in the table for the line addresses of a 688 line program. This can, of course, be changed for shorter or longer programs if desired.

★

THE NEW OSI PERSONAL COMPUTER (maybe)

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

With the advent of new generation processors capable of addressing over one Megabyte of RAM and using sixteen bits of data, it is now time for OSI users to investigate likely replacements for the aged 6502. One likely candidate is the Western Design Center's W65SC816, hereafter referred to as the 65816.

A full sixteen-bit microprocessor, the 65816 has 24-bit addressing, 16-bit registers, more op-codes than the 6502 and is fully capable of 6502 emulation. In fact, it is pin- and signal-compatible with the 6502. and therefore, could be plugged directly into the 6502 socket. With this much going for it, let's take a more detailed look at this candidate for a new OSI computer.

Figure 1 shows the registers in the 65816 programming model. Close inspection will show you that the 6502 registers are still there, only extended for 16-bit operation, one register not present on a 6502 (the Direct register), and two 8-bit Bank registers. Also, note the Address register is now 24-bit. The Status register has been revised to use all bits, with two of them used differently; Memory Select was added, bit 4 is Index Register Select in 16-bit mode and Break in 6502-mode, Carry now also monitors Emulation mode, but is conceptually separate. All registers retain their names except the Accumulator. Each half of each register is distinguished by the letter 'H' or 'L' for high- and low-order bytes. The 16-bit Accumulator is identified as 'C' and the high- and low-order bytes by 'B' and 'A'.

The Status register illus-

OSI/ISOTRON

MICRO COMPUTER SYSTEM SERVICE

- - - - - - - - - - - - - - - - - - -

*C2 AND C3 SERIES
*200 AND 300 SERIES
*FLOPPY DISK DRIVES
*HARD DISK DRIVES
CD 7/23/36/74
*TERMINALS, PRINTERS, MODEMS
*BOARD SWAPS
*CUSTOM CONFIGURATIONS
*CUSTOM CABLES
*SERVICE CONTRACTS

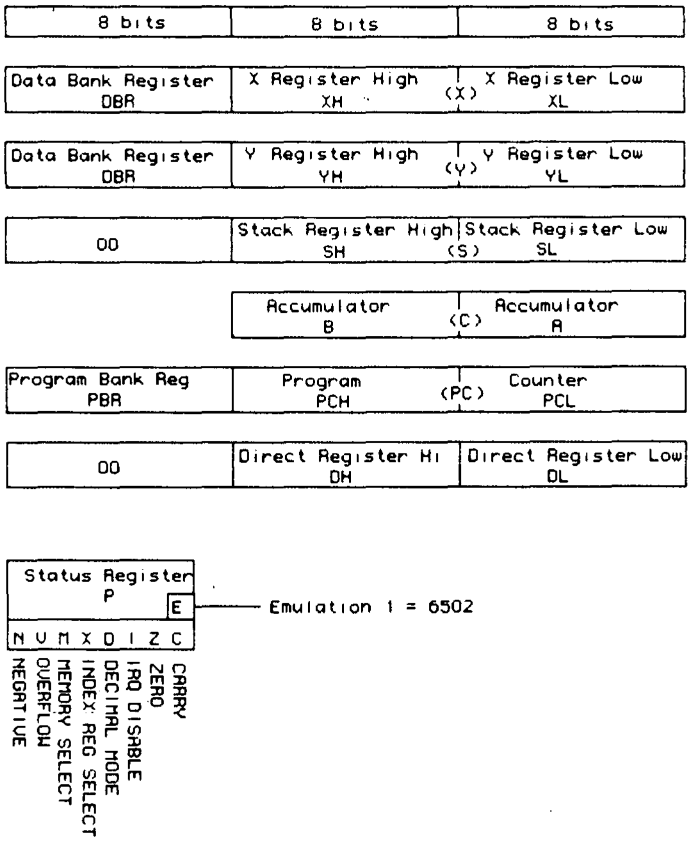
PHONE (616) 451-3778

COMPUTERLAB, INC.
307 MICHIGAN ST. N.E.
GRAND RAPIDS, MI. 49503

trates the major changes beyond the register extensions. The 'X' flag (Break), a software interrupt in 65Ø2 emulation, transfers control to the same routine as the hardware interrupt. However, in 65816 mode the vector is different. This gives programmers more latitude in interrupt handling since the vector now identifies what type of interrupt invoked it. Software won't have to determine whether hardware or software problems caused the interrupt which should speed up the handling of interrupts.

The 'M' flag (Memory Select) works in unison with 'X' to determine whether index and Accumulator registers are treated as 8- or 16-bit. When the flags are set to '1', the registers are treated as 8-bit; when set to 'Ø', registers are 16-bit. 'M' affects only the Accumulator, 'X' affects the indexes.

The 'E' flag is difficult to explain. It does not exist in the Status register; it exists behind the Status register 'C' flag and can be exchanged with 'C'. It cannot be directly tested, set or cleared; it must be exchanged with 'C'.



The added register, the Direct register, holds the address of the 'direct' or 'Zero' page of memory. Programmers are not limited to a 256-byte page at the beginning of memory, anymore; 'zero page' can now be anywhere and doesn't have to begin on a page boundary. However, an extra cycle of processor time is needed if the low order byte of the address isn't zero, to add the offset; if it is zero, special circuitry enables the procesor to skip the extra cycle.

New instructions include PHX, PHY, PLX and PLY for stack operations on index registers; PHB and PLB for data bank register; PHD and PLD for direct register; PHK for program bank register (no instruction for pulling the program bank register). There are new flag handlers, REP and SEP (reset and set status register flags) and TRB (test and reset bits). A new stack-relative addressing mode is available along with new stack instructions. One of these is PEA (push effective address) which pushes the third and second bytes of the instruction, in that order, onto the stack.

There are other instructions added to the 65Ø2 instruction set, too many to go into very much detail here. Most deal with 16-bit operation, some utilize the new addressing modes. The implementation of 16-bit operation in the 65Ø2 world appears to be very well done; new registers, extended registers, new addressing modes and new instructions as well as full 65Ø2 emulation without sacrifice of speed (extended addressing uses an unused half cycle of bus time).

Without going into any more detail on this processor, let's imagine a few things. High-density memory device prices have come down to fairly low levels in recent months. The board size of an OSI computer, using 4164 high-speed Dynamic RAM chips, could probably hold 512 Kilobytes of directly addressable memory. Disk drives, double-sided/double-density, are not too expensive anymore. High-resolution grahics boards and 8Ø-column video board designs are available. The OSI C4P backplane with four slots could be configured with 1 Megabyte of RAM, two double-sided/double-density disk drives and hi-res/8Ø-column video. The major problem facing OSI computers is a viable operating system written for the 65816 to take advantage of the 16-bit operation and speed. There are engineers out there capable of designing the hardware; are there any systems software people reading this?

★

Ky. ASM V1.1 ASSEMBLER A REVIEW

By: Damon F. Curry
SofTouch, Inc.
P.O. Box 31335
Dayton, OH 45431

As authors of some very large programs written entirely in 65Ø2 Assembly Language, we've been happy to see PEEK(65) publish a series of articles on Assembly Language programming. However, we feel that such articles would not be complete without a proper review of the MnM Assembler, produced by MnM Software Technologies.

We've written several large programs which required Assembly Language for speed of execution, or special computing functions not available in BASIC. Some examples are: virtually instantaneous searches through very large data structures, extremely rapid sorting, and numeric computation to higher accuracies than that permitted in BASIC (for example, double or even triple precision arithmetic). We were able to do some of these things, slowly and awkwardly, with OSI's 65D Assembler. We desperately needed a better assembler, though, primarily for two reasons: ease of use, and an ability to assemble very large programs. We've been using the MnM Assembler for over a year now, with superb results.

But why did we want a different assembler other than the OSI-supplied one we already had? Why was the OSI 65D Assembler unsuitable for us? Let us simply list features (desirable or not) of the 65D Assembler. Then, we'll compare the MnM Assembler's features.

OS65D Assembler:

1) Documentation: OSI published a short User's Guide years ago. (The current PEEK(65) articles are providing much of that information). Error documentation is poor, though, since Assembly errors are reported as error numbers, requiring look-up in a cross reference listing. More frustrating, though, is the occasional output of an error message with an undocumented error number!

2) Merging files: Strictly speaking, this is an operating system function, not as assembler function. Merging files is awkward under 65D. Merges require a fairly detailed knowledge of operating system calls and memory allocation.

3) Editing capabilities: Unless one purchases a separate word processor, there are very limited editing capabilities available. Only a very simple resequencer is normally available (although we wrote a much better one, if anyone is interested).

4) Symbols and labels: Assembler symbols (labels, constants, variable names, etc.) are limited to 6 characters in length.

5) Program size: All Assembly source text plus its resultant object code and the Assembler's symbol table must fit into memory at one time. Source text begins at $317E, so a 48k computer has 36,738 bytes available to hold source code, object code, and symbol table. Assuming an 8 to 1 ratio of source code to object code (a very conservative assumption), and a 1 to 5 ratio of symbol table to source code, then assemble programs with OSI's 65D Assembler are limited to less than 4k of object code.

6) Speed of assembly: The 65D Assembler is very slow. We've seen it take more than 5 minutes to assemble progams with source text less than 4k in length.

7) Assembler problems: Assembler "lock-up" is common. Many times we had our computer simply "freeze" during assembly. We were never able to determine the cause or see any pattern to "lock-up" occurrences, other than the probability of "lock-up" was directly proportional to the length of the source text.

8) Disk storage: Since OSI's Assembler works under 65D, one does not have access to hard disk or network storage.

9) Formatting of output: OSI's 65D Assembler formats outputs consistently the same. There are no provisions for partial printouts, or forced page breaks.

MnM Assembler:

1) Documentation: The MnM Assembler manual is a large book, with much useful information. It includes a 6502 Assembly Language tutorial section. Error documentation is in plain English.

2) Merging files: Since the MnM Assembler works under OS65U, file merges are far simpler than merges under 65D. Two files can be merged in one command line, and the procedure is always the same. 65D merges require the use of indirect files, memory pointers, etc., which will most likely be different for each merge operation.

3) Editing capabilities: By invoking the 65U editor, Assembly Language source text files can be easily loaded, edited, and saved. More extensive editing is possible with a word processor. The 65U resequencer is much more versatile than the one in 65D's Assembler. This extended resequencing power makes file merges even easier.

4) Symbols and labels: Assembler symbols can be of virtually any length. Practically, labels are limited only by the length of a single typed line. Notice how much easier source code is to understand with longer symbols.

```
OSI  LDY InOfst
     LDA InBuff,Y
     CMP #CR
     BNE ChkLF

MnM  LDY InputStringOffset
     LDA InputBuffer,Y
     CMP #CarriageReturn
     BNE
CheckIfLineFeedWasEntered
```

5) Program size: The single most useful feature of the MnM Assembler is its disk-to-disk assembly. This feature permits the assembly of programs whose source text is far larger than the available memory of the computer. Without the MnM Assembler, really large programs such as word-processors, compilers, and data-base managers written in Assembly Language are nearly impossible to write for an OSI machine. In our case, many of our programs could not have been developed without the MnM Assembler. Our 6502 Simulator/Debugger, for instance, is approximately 8k in length (object code), but it is assembled from source code greater than 80k in length, more than twice the available memory under OS65D, and over three times the available memory under OS65U!

6) Speed: MnM's Assembler runs circles around OSI's. On a 1 MHz computer, over 80k of source text (our 6502 Simulator/Debugger mentioned above) assembles in less than 5 minutes, less time than the OSI Assembler takes for files one-twentieth that size!

7) Assembler problems: None! The MnM Assembler has always worked without creating system problems, even handling disk read/write problems.

8) Disk storage: Under OS65U, one has access to all disks (floppies and hard disks), even network facilities for storage.

9) Formatting of output: MnM's Assembler is superior in output formatting to OSI's 65D Assembler. Comments are neatly lined up (all comments begin in column 40), 16-bit addresses are shown in human readable form rather than machine readable inverted lo-byte/hi-byte form, the symbol table is neatly printed at the end of the assembly, and titles and subtitles are available. Subtitles force a new page form feed, so subtitled program sections stand out in easily read form. Listing commands can be inserted into the source text to permit partially printed assemblies.

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.

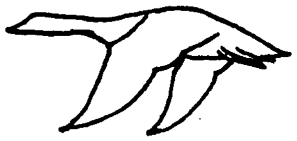
SIMPLY POWERFUL AND COMPLETE: This DBM has external simplicity, and a manual that is written in English; complete from Tutorial to Definition of Terms. Although Gander provides support by the author, it shouldn't be necessary.

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

After realizing the advantages of the MnM Assembler over the OSI 65D Assembler, we turned to MnM's exclusively. Without it, we would still be able to do 65Ø2 Assembly Language programming, but our productivity would be so heavily impacted that we would not be able to accomplish one quarter of our present work! We heartily recommend the MnM Assembler to anyone who is seriously contemplating 65Ø2 Assembly Language programming on OSI computers.

★

PRINTER FOR OSI MODEM PROGRAM

By: Stephen B. McGinnis
505 Smith Street
Ridgway, PA 15853

Sometime ago I submitted a revision of the simple Modem program originally supplied by OSI with my C1P series 2 computer. This revision automatically set the display for 48 characters so that the "video swap" routine was no longer needed. Well, the program works well (although I did make the modifications to make it run in 8K of memory) and I have been debating for sometime about how to get a printed copy of whatever happened to be coming over the telephone modem. With a $10.00/hour phone bill (weekends, no less) to access CompuServe's Pittsburgh number, my silent Heathkit H-14 printer was a constant reminder of wasted resources.

There is a simple (no, not elegant) and cheap solution to the problem: install a DPDT toggle switch. If your computer has a switch (J3) installed so that a printer, modem, or cassette recorder can be used with the serial interface, here is how to do it:

PART NEEDED
DPDT toggle switch (similar to Radio Shack P/N 275-1546)

TOOLS NEEDED
Soldering iron
Rosin core electrical solder
Solder wick (similar to Radio Shack P/N 64-2090)
Screwdriver
Needlenose pliers
Wirestripper (or sharp knife!)
OSI J3 switch diagram dated 10/21/80 (it's handy if available, but not really necessary)

PROCEDURE
1. Unplug your computer and any attached cables, etc.

2. Turn it upside down and remove the screws holding on the bottom of the case. Lift off the bottom and expose the computer board. Position the case with the power supply located away from you and to the left.

3. Identify the following:

a. The rotary 3-position switch mounted on the back of the computer case. This will be identified as J3. It is located immediately to the left of the opening reserved for the fan.

b. The 25-pin Printer jack which is mounted on the back of the case. This will be identified as J5. It is the jack located next to the line cord.

c. The 25-pin Modem jack mounted on the back of the case. This will be identified as J6. It is located between J5 and J3.

4. Identify pin 3 on the Printer jack (J5.3) and pin 2 on the Modem jack (J6.2). Notice that each is connected to J3 (the rotary switch) at the same point by brown (on mine, anyway) wires. This is J3 terminal 2 (J3.2). You have just identified the output line from the RS232 port.

5. Now, identify terminal 3 on your Modem jack (J6.3). Follow the white (?) wire to J3 and note the solder point. When J3 is switched to the Modem position, this point will be connected to the input of the RS232 port.

6. Study the DPDT switch closely. If it is from Radio Shack it should have the pins numbered from 1 to 6. If not, the 3 pins on the left of the switch are numbered 1 to 3 (top to bottom) and the 3 pins on the right are numbered from 4 to 6 (top to bottom). It doesn't matter how you hold the switch.

7. Note that at least one of the RCA jacks on the back of the case isn't being used. Later on, if you leave enough wire length (hint), you can remove one of these and install your toggle switch in its place. This will prevent you from drilling a new hole!

8. Now, de-solder the 2 wires where they attach to J3.2. Be sure you can identify these as coming from J5.3 and J6.2.

9. Solder the wire from J6.2 (Modem) to toggle switch pin 2. (Depending on where you are going to locate the switch, you may need to lengthen this wire and others later on, either by soldering and wrapping in tape or by butt connectors).

10. Solder the wire from J5.3 (Printer) to toggle switch pin 5.

11. Cut a length of new wire and strip each end. It'll help if you apply a small amount of solder to the bare ends to keep the wire from unravelling.

12. Solder one end of the wire to J3.2 (where you removed the other 2 wires).

13. Solder the other end of the wire to toggle switch pin 1.

14. Cut another length of wire and prepare the ends as before.

15. Solder one end to the point where the wire from J6.3 (Modem) connects to J3. A "tack" soldering job should suffice (note that a disk capacitor is already attached to this lug).

16. Solder the other end of the wire to toggle switch pin 6.

17. The remaining steps apply only to toggle switch connections:

a. Using a small length of wire, jumper pin 4 to pin 1.

b. As in "a", jumper pin 3 to pin 1.

DISK DRIVE RECONDITIONING
WINCHESTER DRIVES
FLAT RATE CLEAN ROOM SERVICE
(parts & labor included)

Shugart	SA4008	23meg	$550.00
Shugart	SA1004	10meg	$450.00
Seagate	ST412	10meg	$350.00

FLOPPY DRIVE FLAT RATES

8" Single Sided Shugart	$190.00
8" Double Sided Shugart	$250.00
8" Single Sided Siemens D&E Series	$150.00
8" Double Sided Siemens P Series	$170.00

Write or call for detailed brochure
90 Day warranty on Floppy & Large Winch.
1 Yr. Warranty on 5" & 8" Winchesters.
Phone: (417) 485-2501
FCS FESSENDEN COMPUTERS
116 N. 3RD STREET
OZARK, MO 65721

18. Except for locating the switch, you've finished.

You'll quickly find that one position of the switch will allow for the normal operation of the C1P (the switch will be pulled towards pins 3 and 6). After you've loaded your Modem program, flip the toggle switch, turn J3 to its Modem position, and all modem communications will be directed to your CRT and to your printer. Please note: the output will be at 300 baud, not the 1200 baud you've grown accustomed to at the Printer position of J3. Set your printer accordingly!

A brief diagram of the connections you made to the toggle switch follows:

RS232 out (J3.2)
X (1)

to modem (J6.2)
X (2)

RS232 out (J3.2)
X (3)

RS232 out (J3.2)
X (4)

to printer (J5.3)
X (5)

from modem (J6.3) and to RS232 in
X (6)

★

A $29.00 MODEM FOR OSI

By: Joseph Ennis
212 20 Street
Niceville, FL 32578

THIS MODEM IS SMALL, SIMPLE, CHEAP AND WORKS! It will work equally well on any personal computer that has a serial port, preferably a 6850. The cost of the hardware (cost of kit plus transformer) is $28.90, with another $14.95 you can buy the software. There are lots of reasons to add a modem to your OSI. To me the top three reasons are: 1) my own personal terminal for the times when I have to compete for the use of a terminal; 2) to swap software with non OSI personal computers; and 3) with the availability of computer nets, it might make more sense to buy an access number than a disk drive for bulk storage.

HARDWARE DESCRIPTION

The modem is a kit produced by Electronic Systems, Dept KB, P. O. Box 18220, San Jose, CA 95158, and has been around since at least 1977. The kit contains: 3 IC and about two dozen discretes on a double sided 2 1/2 inch by 2/1/2 inch glass board. The IC are socketed and the kit takes less than 1/2 hour to build. SUPER SIMPLE! The inputs to the modem can be either TTL or RS-232 from the computer side and Bell A-103 at any audio level from 3 mv. to 3 volts on the telephone side. This modem is configured as an originate-only type and although it can easily be set for answer-only tones, a single model cannot easily be switched back and forth between originate and answer. If one needs both functions, it would be advisable to buy two of these modems and configure one each way and switch them as needed. Also, there are no duplex nor baud rate controls on this modem.

These are handled by the software in conjunction with the ACIA (Asynchronous Communications Interface Adapter), the 6850 on the OSI board. The modem is designed to drive an audio coupler and the kit instructions tell how to connect one. To keep down size and cost, I chose direct connection. For those that want the acoustic coupler option, Rondure/Computer, Room 2522, Butler, Dallas, TEXAS 75235. sells a low cost one for $19.95. So far, the cheapest direct coupler that I have found is a 600 ohm to 8 ohm subminiature transformer from Radio Shack. Since the telephones in my area are touch tone with modular plugs, my direct coupler has an extension cable with a modular plug on the end. To keep the telephone company happy, the connection to the telephone line should be around 600 ohms inductive reactance, 200 ohm resistive and deal with audio levels between 3 mv. and 1.7 volt. The Radio Shack transformer has 77 ohm primary DC resistance so a 120 ohm 1/4 watt resistor was placed in series with the primary. The audio levels are around one volt on the telephone line so they do not need any changes. (Figure 2 shows the schematic for the two boards.)

The modem can be connected to either the OSI RS232 port or the TTL port. Since I want to keep the RS-232 port open for a printer and since the TTL port would never be used for anything, I chose to drive the modem from the TTL port. If your OSI does not have components in either of these blocks of circuitry (the usual case), then you will have to use Photograph 2 and 3 and the schematic in Figure 1 for parts, values and placement to populate these circuits. Unfortunately, a modification will have to be made to the circuit track on the 600 board to use this modem, for the TTL port, or even the RS-232 port. It is not a big mod, just cut the jumper at W-12 (See Figure 1 for what W-12 does.) W-12 is necessary because the cassette port is anti-social. There are three ports that the ACIA, U14 could connect to; the cassette port, the RS-232 port, and the TTL port. The transmit part of the ACIA connects to all three with no problems. The problem is with OSI's implementation of the Kansas City Standard for the cassette port. When there is no signal coming in on the cassette port, the output of the Kansas City Standard circuit (U63), cassette RxData is held low. A little better design would have resulted in the cassette RxData line being in the high-don't-care state. Then all three RxData lines could be tied together in an implied-OR circuit. Therefore, the easiest way to deal with this is to switch the cassette RxData line out when it is not needed. (Make the cut at W-12 as shown in Photograph 2.)

I originally added a jumper from W-12, pad C to J3-8 (an unused pin) and switched J3-4 between J3-5 and J3-8. (See schematic in Figure 1.) This proved to be an unnecessary jumper, the circuit works just

computer repair

Board level service on:
- OSI / Isotron
- TeleVideo
- IBM pc/xt

Floppy drive alignment:
- Siemens
- Shugart
- Teac

Terminal repair:
- TeleVideo
- Micro-Term

(1 week turnaround)

SEI
Sokol Electronics Inc.
474 N. Potomac St.
Hagerstown, Md. 21740
(301) 791-2562

as well if a much shorter jumper is placed between W-12 pads A and C, and only a single pole, single throw switch is used to jumper J3-4 and 5 when one wishes to load a cassette tape into the computer. Two additions that I added to the interface board are: 1) a LED and a resistor to act as a carrier detect indicator, and 2) a jack to be used by a touch tone dialer (See Figure 2).

SOFTWARE

Most Software Houses that write programs for OSI computers have some version of "Dumb Terminal" or a modem driver program. I have bought both the Progressive Computing and Aardvark Technical Services, (2352 South Commerce, Walled Lake, MI 48088) programs. I prefer the Aardvark, it is a little better documented. For purposes of checking this installation, I offer a Simple (like in low IQ) terminal program, given in List 1. This program will allow you to establish two-way communications.

HOW IT WORKS - HARDWARE

As long as one is the originator of the communication, the hardware requirements for a modem is very simple. The answering modem has to do all the work of checking and matching baud rate and word format. Therefore, if one eliminates the circuits that are used to do this checking and switch setting, one finds that all that is left is one voltage controlled oscillator (a single IC) and a single tone demodulator (one more IC). That is what this modem card is: two phase-locked loops, one CMOS hex inverter, a switching transistor, two frequency adjustment pots, 5 timing quality capacitors and not much else. It runs off a single +5 volt power supply. Referring to Figure 2, U1 (the 2211) is a straight forward tone discriminator, pin 7 is either high or low depending on whether the audio tone present at pin 2 is above or below the time constant formed by R5, R6 and C6. The audio received from the telephone line comes through the decoupling transformer T1 to U1 pin 2, the tones are either 2225 Hz (logic one, also called a Mark, the carrier rests at this frequency when no data is being sent), or 2025 Hz (logic zero, also called a Space). The whole job of U1 is to toggle its pin 7 between high and low as the signal on its

LISTING 1

```
40 REM List #1, Simple Terminal.
50 POKE235,32:POKE236,239:POKE237,255:POKE238,96:POKE530,1
60 POKE11,232:POKE12,0:POKE232,32:POKE233,235:POKE234,255
70 PRINT"SIMPLE TERMINAL":PRINT
80 INPUT"FULL OR HALF DUPLEX";A$:IFLEFT$(A$,1)="F"THENF=128
90 PRINT:PRINT"300 BAUD IS NORMAL, TO  GET 75 BAUD ADD 1 TO    CONT";
100 PRINT"ROL WORDS. TO GET":PRINT"4800 BAUD SUBTRACT 1."
110 PRINT:PRINTTAB(2)"300 BAUD CONTROL WORDS":PRINT
120 PRINT"WORD NO CHRS PARITY STOP"
130 PRINT" 01      7    EVEN   2"
140 PRINT" 05      7    ODD    2"
150 PRINT" 09      7    EVEN   1"
160 PRINT" 13      7    ODD    1"
170 PRINT" 17      8    EVEN   2"
180 PRINT" 21      8    ODD    1"
190 PRINT" 25      8    EVEN   1"
200 PRINT" 29      8    ODD    1"
210 PRINT:INPUT"ENTER ACIA CONTROL WORD";BAUD
220 STATUS=61440:BUFFER=61441:KEYBOARD=57088:NOOP=254:RESET=3
230 PARITYERR=64:FRAMERR=16
240 SAVE:POKESTATUS,RESET:POKESTATUS,BAUD:PRINTCHR$(26)
250 P=PEEK(KEY):IFP<>NOOPTHENPOKE100,F:X=USR(X):POKE100,0
260 P=PEEK(STATUS):IFNOT(PAND1)THEN250
280 IFPANDFRAMETHENX=PEEK(BUFFER):GOTO250
290 IFPANDPARITYTHENX=PEEK(BUFFER):GOTO250
300 X-PEEK(BUFFER):IFX>127THENX=X-128
310 POKE517,0:PRINTCHR$(X);:POKE517,1:GOTO250
ready
```

LISTING 2

```
SIMPLE TERMINAL

FULL OR HALF DUPLEX? FULL

300 BAUD IS NORMAL, TO GET  75
BAUD ADD 1 TO CONTROL WORDS.

TO GET 4800 BAUD SUBTRACT 1.

300 BAUD CONTROL WORDS

WORD NO CHRS PARITY STOP
 01     7    EVEN    2
 05     7    ODD     2
 09     7    EVEN    1
 13     7    ODD     1
 17     8    EVEN    2
 21     8    ODD     1
 25     8    EVEN    1
 29     8    ODD     1

ENTER ACIA CONTROL WORD? 17
```

input pin switches between 2225 and 2025 Hz. Since the output at pin 7 is low for a frequency higher than the RC time constant, it must be logically inverted before it is sent to the ACIA. This is achieved by one of the inverters in U3 (pin 12). This output connects to the 600 board (J2 pin 1) and becomes RxData3 going to W-12, and then on to the ACIA (600 board U14 pin 2). R4 on the modem card sets the band width or depth of null for the discriminator. In summary, U1 and U3 on the modem card are a frequency to TTL logic level converter and the rest of the receive function is done either by the ACIA or the software.

The modem transmit side is even simpler, U2 (a 567) is also intended for tone demodulator service, and it has a very accessible VCO (Voltage Controlled Oscillator), and is all that is used in the 567. The timing for the 567 is controlled by the resistances and capactances connected to pins 5 and 6 (see Figure 2). If Q1, the 2N222, switching transistor is biased off, then only C9 and C12 in conjunction with R8 and R9 set the frequency of the oscillator. When Q1 is biased on, C7 and C8 are added to the RC circuit causing the oscillator to oscillate at a lower frequency. For the modem transmit side, the frequencies are 1270 Hz (logic one, Mark, and is where the carrier rests when there is no data to transmit) and 1070 Hz (logic zero, Space). Again the logic is inverted, so one of the inverters is connected to the input of Q1. The rest of the inverters are used as audio drivers on the output audio to the telephone line. These output drivers are the only reason that the inverter has to be CMOS, anything else would load the oscillator pick-off point.

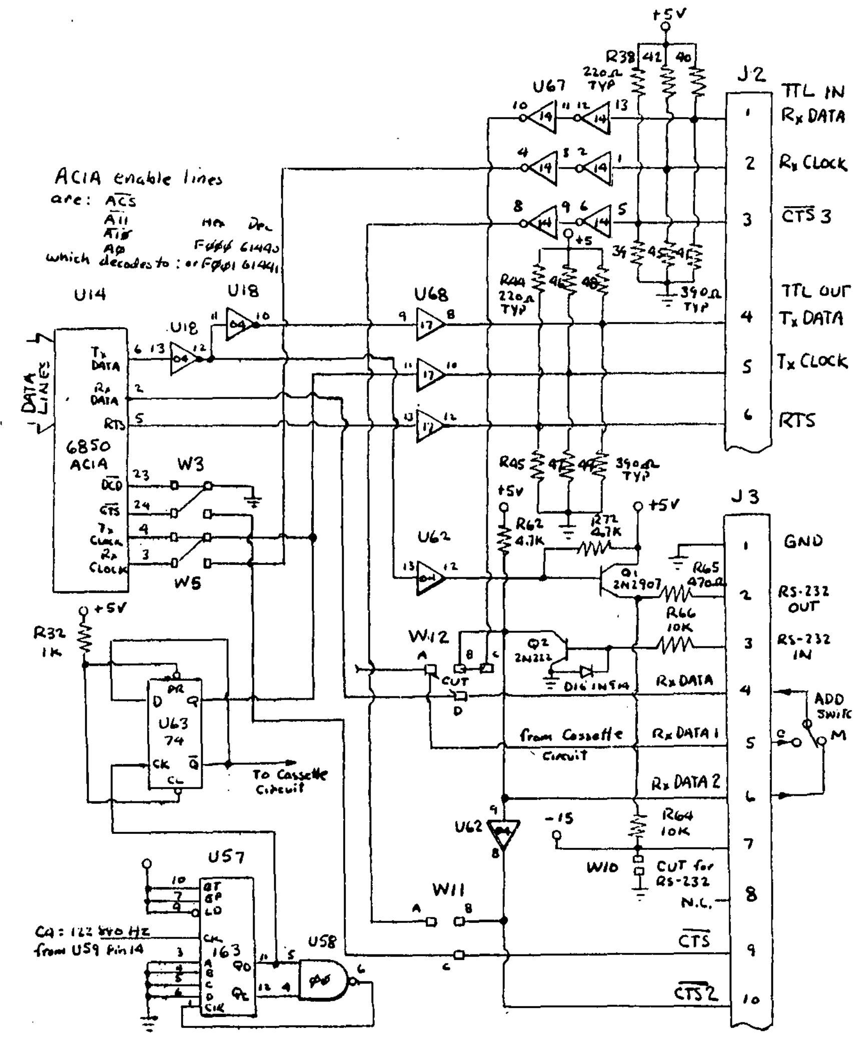
HOW IT WORKS -- SOFTWARE

If you already have a terminal /modem program, just LOAD and RUN and skip to the alignment section. If you just built this modem, and don't have any support software, but just can't wait to try it out, a modem demo program is given in List 1.

Before discussing how the software works, a review at the system level is in order. An Originate modem can only talk to an Answer modem and an Answer modem to an Originate modem. If both of you have Originate-only modem, no possible software mod will allow you to communicate; however, you could talk to a third party such as a computer bulletin board, which is always an Answer modem. With the bulletin board one could post a message or program for the other to copy. For the case where you will only be using your modem to talk to one other modem connected to an OSI computer (or most other personal computers using BASIC), no software is needed at all. One modem will still have to be tuned to the Answer frequencies. But for communicating and swapping programs, the SAVE and LOADS - which you already have in your computer's system monitor ROMS - is all the software that is required. One calls the other on the telephone, and both computers and their modems are energized and connected. Then when one person is ready to send something, he tells the other person to type LOAD CR on her end, then he types SAVE CR and LIST CR on his end, that's all. The data goes over the telephone line just as if it was going to or coming from the cassette port. Voice communication can occur at the same time. You ignore the tones the computers are putting on the telephone line and the computers ignore the voices.

FIGURE 1

Partial Schematic of OSI 6ØØ Board Showing ACIA, RS-232 Port, and TTL Port



Now, the ACIA (Asynchronous Communication Interface Adapter) needs to be discussed. In the OSI Computer, the ACIA is a 6850 chip. The ACIA has four internal 8 bit registers; however, two are write-only and two are read-only on the CPU side of the ACIA, so the four registers take up only two address spaces. The receive buffer and transmit buffer are one pair and the control register and status register are the other pair. When the ACIA is initialized, a control word is written to the control register by the System Monitor ROM, that selects one of several programs built into the ACIA and sets the divide-by value on the baud rate clock divider (sets baud rate). The ACIA uses its built-in program to deal with interrupts, to take in an 8 bit word from the CPU and shift it out, a bit at a time, on each (baud rate) clock pulse, receive a string of pulses and assemble them into an 8 bit word, and sets flags in the status register so the CPU can check on the ACIA's progress.

Now, finally, how the software in List 1 works. The program first presents a menu, and you select the communication mode you wish to use. This results in the ACIA's Control Register being reset and the desired control word being entered. Line 24Ø does the work. Now the program loops continuously between lines 25Ø and 27Ø. Line 25Ø PEEK's the keyboard for a key closure and if there is one, it uses a USR (X) routine to call part of the ROM's monitor routine (the vector was set in Line 6Ø) which will do all the work of outputting one ASCII character (equivalent to the key pressed) to the ACIA's transmit buffer address.

Now control passes to Line 26Ø which PEEK's the ACIA's status buffer and checks if the receive buffer flag is set. If it is, it jumps to the subroutine on Line 28Ø, and if not, it loops back to Line 25Ø. Subroutine at Line 28Ø checks the status word for indications of errors and if none, converts the contents of the receive buffer into an 8 bit ASCII word by chopping off the parity and stop bits, Line 3ØØ does the work. Line 31Ø prints the resulting character to the screen, then returns control to the calling line.

The received communication goes only to the screen in normal mode, and does not trigger the BASIC interpreter. If you want to send the received communication to memory by way of the BASIC interpreter, first make sure there are a lot of NULL's after each line and there are not any line numbers in the program to be received that correspond to the line numbers in the Modem Program, then hit the BREAK key, and W key. At this point you are in Local Mode. Now what you type on the keyboard goes only to your computer, and you can type anything except something that looks like a line number or something that returns an error message; however, the Modem is still sending the MARK tone so the line to the other computer is still being kept up. So, type LOAD (and of course, a carriage return). (If using HEXDOS type POKE 517,1 instead). Now until you hit either the Space Bar or BREAK Key all in-coming communication will go through the BASIC language interpreter in your computer, which will pack it in memory, if possible. The results will also scroll on the CRT screen. When the communication is complete and is observed not to contain anything that would be detrimental, hit the BREAK Key and the W Key again. Now you can either RUN the program by typing RUN 320 (and return) to see what you have. Or you could go back to two-way communication by typing RUN 240 (and return). THAT'S ALL THERE IS, TO HOW IT WORKS AND HOW TO USE IT.

CHECKOUT AND ALIGNMENT

If you have a frequency counter, an audio oscillator and an oscilloscope, you are in luck! The kit instructions give the alignment procedure and it takes less than five minutes (only two pots to set). Do it and skip the rest of this article. Otherwise, we will have to do it the hard way. The Modem kit instructions give three different techniques for those without test equipment. I will add a fourth technique for those that don't have test equipment but are computer timeshare users:

1) Borrow a terminal.

2) Using the terminal, sign-on the timeshare normally.

3) With the carrier up but no characters being sent either way, and Modem Pin 5 disconnected from the transformer, adjust R6, the pot closest to U1 (XR2211) for a High Level (or if the carrier detect LED option is connected, the LED ON) at the Modem Pin 4.

4) Load and start "Simple Terminal" or any other Modem program.

5) Go back to the borrowed terminal and command the Host

FIGURE 2

Schematic of Modem Board and Coupler Board

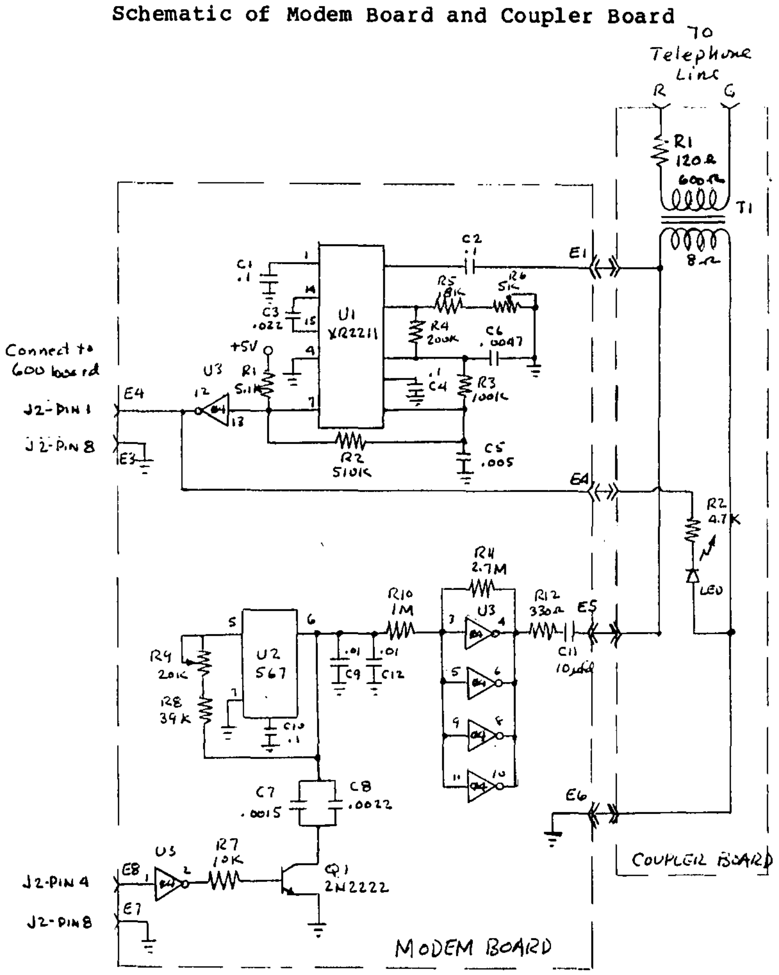


Photo #1

Modem Front & Back View

Direct Coupler Front/Back View

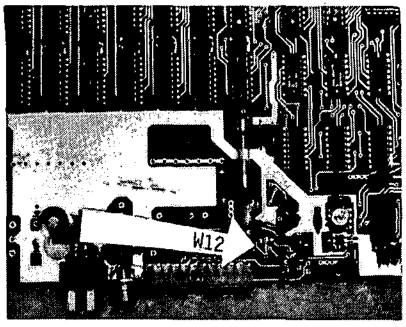


Photo #2

OSI Board Top View Showing Track Cut at W12.

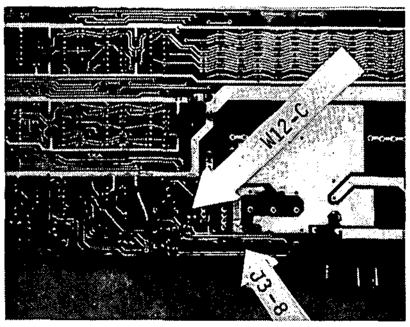


Photo #3

OSI Board Bottom View showing Jumper from W12-C to J3-8.

Computer to list a long file. R6 is already close to its proper adjustment point, and this file coming from the Host is the test pattern that will allow R6 to be fine tuned to its final position.

6) Go back to R6. A very small clockwise adjustment should result in the characters that are being transmitted by the Host to start being printed intelligently on your Computer's CRT. DOESN'T THAT LOOK GOOD! When R6 is close, characters will be printed but they will not be the right character and there will be a lot of skipped characters. When R6 is adjusted correctly, all the characters will be correct; however, there may still be some skipping as the Baud Rate Clock in the terminal is not necessarily the same frequency as the Baud Rate Clock in your computer (most likely won't be). This is because, when the Host is slaved to the borrowed terminal, your computer will be drifting through sync, this will cause occasional characters to be missed.

7) Now connect Modem Pin 5 to Modem Pin 1 permanently. This puts the Modem transmit carrier on the telephone line. Restart the Host, listing out that long file. Adjust R9 in small steps until the Modem transmitter carrier steals sync from the borrowed terminal, this will be indicated by the string of characters being printed to the CRT screen being printed without misses.

8) Now disconnect the borrowed terminal, and observe that the Host still stays on the line (if the LED option was added, the LED should stay lit) and recognizes your Modem as a Modem. Now, try pressing a few keys on your computer's keyboard, if the Host responds then the alignment is done.

9) If not, mark where R9 screwdriver slot is located and adjust R9 back and forth about one screwdriver slot width, hunting for the spot of best operation. When the alignment is complete, place a drop of fingernail polish on R9 and R6 and close the case.

Good luck and enjoy!

LETTERS

ED:

In my letter of July 30th concerning the WP6502 Word Processor and a letter from Carl King in the June issue, I stated that I was working on disassembling the WP6502. This was started in order to make the following modifications:

While printing; to be able to stop it and return to the Menu.

Replace the R/Tape and W/Tape in the Menu with Print selections.

Go with View and one tap of the space bar to the screen display.

The disassembling has progressed far enough so that the modifications could be made. Since my system does not use a cassette, the space for the R/Tape and W/Tape coding is used for the new coding.

An escape routine was added so that the "ESC" key, with the Shift Lock up or down, will stop the printing or screen viewing and return to the Menu.

The R/Tape in the Menu is changed to P/Std and coding is added that sets up the parameters for printing at the default values. The values are: Device 01, Margin 10, Form length 66, Text width 60, Page No. 00, Number of copies 01, AP Yes, Ho No. A "P" reply to the Menu, will start printing.

The W/Tape in the Menu is changed to OptPrt and the coding is changed so that an "O" reply will display the printer Menu so that the parameters may be changed by entry before the printing starts, just as originally written.

Does anyone know of a new address for Dwo Quong Fok Lok Sow or if they are still in business?

J. Edward Loeffler, Jr.
Huntsville, TX 77340

Edward:

We also have tried communicating with Dwo Quong, and like you have been unsuccessful. We fear the news it not good.

Peek Staff

* * * * *

ED:

I would like to point out a problem with the device driver for device #1 in OS-65D v3.2 and v3.3. On my C1P mod 1, with a printer on the output side of device #1 and nothing

TAXTIME FUN???

No, but maybe easier! Introducing TAX AIDS II for OSI 48K Video or Serial terminal systems. Zip through your Form 1040, Schedule A, B, C, Income Averaging and Alternate Minimum Tax calculations. Use the Tax Table and Tax Form files provided or create your own, to project future tax situations, using the friendly utilities included. Display all results on screen or in annotated printout, or with optional Print on Forms program, print directly on Tax Forms. Lots of traps and warnings to help reduce errors. A Depreciation Analysis program and full instructions are also included.

PRICE: 65DV3.3 Version $30. 65U version $40 (Serial only)
Add $10 for Print on Forms option.
California residents add applicable sales tax
Prepaid orders shipped postage paid, others UPS, COD.

Please specify Video or Serial System, Version and disk size.

Dealer Inquires Welcome.

RSB ENTERPRISES
4045 Ashbrook Circle San Jose, CA 95124

on the input side, the system will hang or issue a break while printing to this device. Referring to the dump in figure 1 shows the reason. After successfully transmitting a character OSI has decided to implement the <ctrl><s>,<ctrl><q> protocol. However, input for this is taken, not from the current input device according to the I/O distributor, but from device #1. Since there is nothing connected to the input side of device #1, it is reading garbage! A random <ctrl><s> will hang the system and a <ctrl><c> will cause a break to be issued. The problem can be corrected by including a POKE9433,96 at the beginning of BEXEC*. This will cause the device driver routine to return immediately after a character has been successfully transmitted.

Figure 1
Output Device Driver #1
Replace $24D9 PHA with RTS

```
24CD 48     PHA
24CE AD00F0 LDA $F000
24D1 4A     LSR A
24D2 4A     LSR A
24D3 90F9   BCC $24CE
24D5 68     PLA
24D6 8D01F0 STA $F001
24D9 48     PHA
24DA AD00F0 LDA $F000
24DD 4A     LSR A
24DE 9011   BCC $24F1
24E0 20F624 JSR $24F6
24E3 8D2523 STA $2325
24E6 C913   CMP #$13
24E8 D007   BNE $24F1
24EA 20F624 JSR $24F6
24ED C911   CMP #$11
24EF D0F9   BNE $24EA
24F1 68     PLA
24F2 8D6323 STA $2363
24F5 60     RTS
24F6 AD00F0 LDA $F000
24F9 EE2423 INC $2324

24FC 4A     LSR A
24FD 90F7   BCC $24F6
24FF AD01F0 LDA $F001
2502 297F   AND #$7F
2504 8D6323 STA $2363
2507 60     RTS
```

Frank Glandorf
Cincinnati, OH 45220

* * * * *

ED:

You have at least one devoted follower of Rick Trethewey's Assembly Programming class, but I would like to offer a couple of comments.

Your sample programs are impossible on a serial system - could you please include a couple of beginners samples for those of us who don't own video based systems?

I realize that I am jumping a number of lessons ahead but I hope you include some information on interfacing to a BASIC program. As an example, I have input a string under BASIC and now want to manipulate the string in machine language. I know the machine program can be loaded at $6000 in front of BASIC, but how do I call the required string into the work area and return it (probably a different length) to the BASIC area?

Another that comes to mind are printer and VDT commands. These are currently held in a dimensioned string and must be loaded from disk and created under CHR$ for each program run. It is a 22 dimension and up to 5 characters in each string - thus time consuming. Obviously, a machine routine at $6000 would speed this up tremendously, but how do I put the dimensioned strings where BASIC can use them?

The other area I hope you will include is input - output to a serial VDT.

Now to give you a laugh - I have almost completed my string manipulation routine. It is currently 161 bytes and has taken only three days of programming mistakes, hang ups, and "Hell, why did it do that?!"

It is my first attempt, and no doubt looks it, but hopefully practice will improve speed and method.

Thanks for the lessons, keep them coming.

Ian Mutch
Queensland, Australia

Ian:

Rick Trethewey reports that he spent most of a day working on the string interface to BASIC before success. That done, he has written lesson #9 of the Assembly Language tutorial.

Hang in there, it's coming.

Peek Staff

* * * * *

ED:

Month after month I see your request for articles by business people about their computer applications at work. I'd like to suggest a possible reason why you haven't received many such articles in the past - SECURITY.

It sounds a little like paranoia, but in our cut-throat economy, many business people are reluctant to reveal anything about their operations lest a competitor should profit from such a revelation. This would obviously apply where a firm in a given industry has developed some possibly unique software. The developer would naturally feel that his/her software is what makes him/her more successful than those without it. To have an article published about that software (or its application) could reduce his/her company's edge over competitors in the market.

As a counter-argument, I suggest that such reluctant business people submit articles merely describing the hardware. The software could be described with such generic terms as "payroll, inventory, general ledger", etc. No details of the origin of the software or its special features need be given.

So, c'mon you lawyers, doctors & accountants! Tell us that you're using an OSI computer - not some other brand. And please tell us how much time it has saved you and how much more productive you are because of it. Tell us how you came to choose OSI instead of BRAND X. After all, we want to hear about the computer, not your trade secrets.

Bruce Showalter
Abilene, TX 79601

Bruce:

Hear! Hear!

Eddie

* * * * *

FILE TRANSFER

8" OS65U FLOPPY TO 9 TRACK IBM COMPATIBLE TAPE. 1600 OR 6250 BPI.

MG Bookkeeping Svc.
(213) 598-8526

ED:

I recently purchased all of the available back issues of PEEK(65) and read through them. I recommend this -- a fascinating narrative on OSI!

I called Hazeltine at their (516) 549-8800 number recently and got Police Emergency there in Greenlawn, NY-- no forwarding number. Tried Information and was given (516) 261-7000 Hazeltine. They gave me 293-5600, Technical Support. Tech Support said that Hazeltine doesn't support the 1400/1500 line any more -- try TRW in Jersey, (201) 575-7110. I did, and eventually reached the Tech Rep, Jack Riggs at ext. 5289. He's a real help - cooperative and knowledgeable. They sell parts, too.

Does anyone out there have a 9-track tape drive, a hard disk, and possibly FIBACK? I have 4 or 5 files on tape (500K bytes or so each) and need them on OS-65U floppy so I can use them. $, trade, etc..

Are OSI's OS-65U utility routines copyrighted? I'm writing enhanced versions of CREATE, DELETE, and possibly PACKER and COMKIL, which I might want to sell for a small fee.

Whatever happened to the promised annotated OS-65U disassemblies? Since PEEK(65) told me that they don't know of any on the market, I'd like to do one -- I need it. If anyone out there has partials that they'd like to share, I'll provide a copy of the finished product in exchange for a significant (i.e., several routines) annotated disassembly (1.43 or 1.44).

Has anyone designed a mod or a kit to a 470 board to read IBM 3740 format? (J.B. Boardman's letter in the Oct '81 PEEK(65) is a start).

Does anyone have an OS-65U based routine which will read either OSI CPM or IBM 3740 format floppies?

What's happened to Rockwell's R65C02 pin-compatible 6502 chip with an enhanced instruction set and up to a 4 MHz clock? (I saw it mentioned in PEEK(65) about 18 months ago.)

Steve Hendrix in the October '83 PEEK(65) discussed the C1P's CPU board trying to drive the memory boards at 2 MHz and not always being successful. I've had the same sort of problem with a 510 and 3 520s -- is it the same problem with a similar fix?

Tom McGourin
1170 Nestor Ave., Apt. A16
Akron, OH 44314
(216) 745-3516

Tom:

Let's see. One thing at a time!

1) 9 track to floppy service is available at a number of places, two come to mind: MG Bookkeeping Service advertised in November, page 21, and although not advertised, Mike Sokol, also in November, page 19, is equipped to do it too!

2) Copyrighting is always ticklish. Yes, OSI does copyright. The question is, how much modifying have you done? It is a legal point. Best consult OSI. In the past they have been cooperative. It is to their advantage to have their machines work better.

3) We have a disassembly of OS-U in hand, just awaiting go ahead instructions from the authors. We would be happy to cooperate with you in anyway to get this information to those who need it.

4) If there is something new

MAKE YOUR OWN WORKSYSTEM

Isotron's new Mix and Match Options allow its dealers to configure the system a customer needs at **savings** similar to the two popular WORKSYSTEMS previously offered.

Mix and Match WORKSYSTEM options include:
★ All Isotron's Ohio Scientific Computers
★ Popular Peripherals
★ Application Software

For more information about Ohio Scientific products and the name of your nearest Isotron dealer call or write:

ISOTRON

140 Sherman St. Fairfield, Ct. 06430 • (203)255-7443

on 3740 readability, our readers haven't told us yet. The closest we can come is D&N's Disk Transfer.

5) R65C02: Shy of substitution for the sake of substitution, there remain a number of problems to utilize its power. For starters we need 4 MHz RAM. See page 10, Thorsen.

6) Let's ask the master. What do you say, Steve?

Eddie

* * * * *

Tom:

The problem with the C1P trying to drive memory boards at 2 MHz stemmed from a shortcut by OSI, in that they did not include any buffering in the phase-2 clock signal, but made the processor output pin try to drive the whole system. Even with the power-hungry 2114s, it was up to the job as long as the clock stayed at 1 MHz. I got a real shock the first time I hooked an oscilloscope on the bus with it heavily loaded, and observed the "square wave" as something more like a logarithmic ramp. Since memories or peripherals require a certain voltage before they consider the signal to be at a "true" level, a sharper rise time results in the signal effectively coming on sooner, which gives the device more time to respond before the processor latches the data. The effect is not linear: if the signal takes 250 nsec to rise to a "true", it leaves an additional 250 nsec for the memory to respond at 1 MHz (500 nsec low, 500 nsec high), but no time at all at 2 MHz (250 nsec low, 250 nsec high). If the rise time is longer or the clock is faster, the result is that the memory must respond sometime before it gets the signal, and I don't think there are any chips around yet that are quite that fast.

Though I am not familiar with the 510 and 520 boards specifically, it sounds like there would be more than enough devices driven by phase-2 to cause a similar problem. Even with a TTL buffer on my system, the problem does not go away completely. The processor phase-2 output is rated to drive one TTL load, but even standard "LS" series TTL is only rated to drive 10 additional inputs. A conservative design would require a separate driver for each row of memory chips, such that there would only be a total of ten inputs being driven from the bus. Of course, each additional level of buffering adds its own delays. In addition, the fact that there are several different boards being driven by a common bus with some significant length calls for some transmission-line theory (and I know just enough of that to be dangerous). In any case, the processor board should probably drive the system phase-2 signal through a buffer with some muscle, perhaps an 8T26 or 8T28 (the 8T26 inverts the signal; the 8T28 does not). The cleaner rise on the clock signal with a better driver results in more time for the memories to respond.

Steve Hendrix
4089 Alexander Rd.
Atwater, OH 44201

* * * * *

ED:

As promised, here is the program to calculate look-angles to the satellites in synchronous orbit. As you may know, there has been a great increase in the popularity of the home receiver or "Earth Station" recently. This program will help in aiming the dish, providing the azimuth and elevation angles necessary to point the thing in the right direction.

As you get into the program, you will notice that the coordinates of the earth station are in degrees. I find that it is not necessary to get to minutes and seconds, because measurements that precise are not always known, and because the dish will be peaked on the satellite by observing the picture. In line 35, the user should insert his own city, and his own coordinates in line 50. This helps speed the data entry when running the program if the user is interested mainly in his own installation. If not, the program allows input of other

```
3 REM
4 REM              PROGRAM NAME IS 'SAT'
5 REM
6 REM       WRITTEN  BY  DWIGHT FINGER, JULY,1984
7 REM
9 GOSUB 5000
10 PRINT TAB(16)"THIS IS A PROGRAM TO CALCULATE LOOK ANGLES TO "
20 PRINT TAB(24)" SATELLITES IN SYNCHRONOUS ORBIT"
25 FOR I=1TO7:PRINT:NEXT I
30 DO=0:K=6.626:C=.0174533
35 PRINT TAB(16)"IS EARTH STATION LOCATED IN ANCHORAGE? Y or N";
40 INPUT A$
50 IF LEFT$(A$,1)="Y" THEN L1=61.19:L2=149.81:GOTO 200
53 IF A$<>"N" THEN 30
55 PRINT
60 PRINT TAB(13)"ENTER THE LATITUDE AND LONGITUDE OF THE EARTH STATION"
65 PRINT TAB(17)"IN DEGREES, SEPARATED BY A COMMA";
70 PRINT;:INPUT L1,L2
80 IF L1>90 OR L2>180 THEN 60
200 PRINT:PRINT TAB(21)"ENTER THE LONGITUDE OF THE SATELLITE";
202 INPUT SA
203 LA=L1*C:LO=L2*C:S=SA*C
210 D=S-LO
215 IF LA=0 AND D=0 THEN  DO=1 : REM  CHECK FOR  DIRECTLY OVERHEAD
217 IF LA=0 AND D=0 THEN DO$="THE SATELLITE IS DIRECTLY OVERHEAD"
219 IF DO=1 THEN GOSUB 5000:PRINT DO$:GOTO360
220 IF LA=0 AND S>LO THEN AZ=270.00:GOTO270
222 IF LA=0 AND LO>S THEN AZ=90.00:GOTO270
228 REM
230 REM CALCULATE AZ
240 REM
250 AZ=ATN((TAN(D))/(SIN(LA)))
260 AZ=(AZ*57.29)+180
265 AZ=AZ*100:AZ=INT(AZ):AZ=AZ/100 : REM ROUNDS TO 2 DEC PL
270 GOSUB 5000:PRINT:PRINT"THE AZIMUTH IS";AZ;"DEGREES"
300 REM
310 REM CALCULATE ELEVATION
320 REM
325 M=COS(LA):N=COS(D)
330 X=((K*M*N)-1)/(SQR(((K*K)+1)-((2*K)*M*N)))
340 EL=ATN(X/SQR(-X*X+1))
345 EL=EL*5729:EL=INT(EL):EL=EL/100 : REM ROUNDS TO 2 DEC PL
350 PRINT:PRINT"THE ELEVATION IS";EL;"DEGREES"
360 PRINT:INPUT"HARD COPY? (Y)es or <CR>=No";QA$
370 IF LEFT$(QA$,1)="Y" THEN GOSUB 2000
500 PRINT:INPUT"DO YOU WANT TO TRY ANOTHER? Y or N";Q$
520 IF LEFT$(Q$,1)="Y" THEN GOSUB 5000:CLEAR:GOTO 30
530 IF LEFT$(Q$,1)<>"Y" THEN PRINT"END OF PROGRAM"
550 RUN"BEXEC*","PASS"
1000 END
2000 REM
2010 REM        **** SUBROUTINES
2020 REM
2030 PRINT#5,"FOR A SATELLITE LOCATED AT";SA;"DEGREES,"
2035 PRINT#5,"FROM AN EARTH STATION AT";L1;",";L2
2036 IF DO=1 THEN PRINT#5,DO$:GOTO 2060
2040 PRINT#5,"THE ELEVATION ANGLE IS";EL;"DEGREES AND "
2050 PRINT#5,"THE AZIMUTH ANGLE IS";AZ;"DEGREES"
2060 PRINT#5:PRINT#5:PRINT#5:RETURN : REM TO LINE 500
5000 REM
5010 REM
5030 PRINT CHR$(126);CHR$(28) : RETURN : REM  CLS  AND  HOME
```

coordinates for other locations.

The progam is written with North America in mind, and the entries of Long. and Lat. are positive numbers, degrees west and degrees north. The satellites are also positive numbers, in degrees west. To be honest, I don't know if the program will work under any other conditions.

The rest of the program should be self-explanatory by means of the REMs. As previously mentioned, I'm a bit new at this and there may be better ways of doing some operations, but this is what I have come up with and it does seem to work.

Dwight Finger
Anchorage, AK 99504

ERRATA

ED:

After years of cursing faulty program listings, I am ashamed to admit to creating one of my own. Please publish these corrections to the PIACOM program in the December issue:

Line 545: the variable BY should be BR

Line 585 should read:

```
EB=BU+BY:TN=INT(RT*BR/BY):
FZ=INT((RT+1)*BY-TN*BR)
```

BY is the number of bytes per track in the buffer. BR is the number of bytes per record in a random file.

While I'm at it, I also noticed that I twice used the expression "a non-ASCII character" in the documentation when what I meant, of course, is a non-alphanumeric character--a graphics or control character that would not occur in a text file.

My apologies for any confusion this may have caused.

Steve Donachie
Miami, FL 33143

Introducing
SCRIBE
WORD PROCESSOR
OS-65U 1.42 < Floppy/Hard Disk, Level 1 or Level 3
and DENVER BOARDS
- INTERFACED TO OS-DMS FILES
- AUTOMATIC WRAP AROUND
- COMPLETE EDITING CAPABILITIES
  - FULL CURSOR CONTROL
  - INSERT & DELETE TEXT
  - SEARCH/SEARCH & REPLACE
- USER FRIENDLY MANUAL
- AND MUCH MORE

IHS COMPUTER SERVICES
Route 1 Box 201B Port Republic, VA 24471
(703) 249-4833

$195.00

WAZZAT CORNER!

Continued from page 4.

```
1   REM PRINT USING BY LZ JANKOWSKI
5   :
10  PRINT !(28):INPUT "How many numbers ";H:DIM N$(H):PRINT
30  FOR C=1 TO H
40  :  INPUT "Number "; N: GOSUB 190
50  NEXT
60  END
70  :
190 Z$(1)="00": Z$(2)="0": T=40
200 M$=STR$(INT((N+.005)*100)):S$=LEFT$(M$,1):L=LEN(M$)-1
210 M$=RIGHT$(M$,L): IF L<3 THEN M$=Z$(L)+M$: L=3
230 PRINT TAB( T-L-2) S$ LEFT$(M$,L-2) "." RIGHT$(M$,2)
240 L=FRE(L): RETURN

1 REM PRINT USING, VERSION 2 —— BY LZ JANKOWSKI
2 :
5 FOR C=1 TO 9: Z$(C)=Z$(C-1)+"0": NEXT: P=40
6 PRINT!(28):INPUT"How many decimal places ";D: I=10^D
10 PRINT: INPUT "How many numbers ";H: DIM N$(H): PRINT
30 FOR C=1 TO H
40 :  INPUT "Number "; N
50 :   IF N>=1 000 000 000/I THEN PRINT"# too large!":GOTO 40
60 :    GOSUB 200
70 NEXT
80 END
90 :
200 M$=STR$(INT((N+.5/I)*I)): S$=LEFT$(M$,1):L=LEN(M$)-1
210 M$=RIGHT$(M$,L): IF L<D+1 THEN M$=Z$(D)+M$: L=D+1
230 PRINT TAB( P-L-2) S$ LEFT$(M$,L-D) "." RIGHT$(M$,D)
240 L=FRE(L):RETURN
```

How useful is the program for OS65U users?

AD$

Good news to the owners of C1P-MF, C2P-MF or C4P-MF OSI Computers with Single Disk Drives. 'Fantastic Copy II' has just been released by the author, Mike Putnam, with the copy guard removed allowing you to copy the copier ... it also works with drives that have head unload and motor shut-off modifications ... best of all, the price is only $15.00 for all versions and we will even update your present disk for $7.50. Order today from Thomas Technical Service, RD #1, Box 135, Linden, PA 17744. Mastercharge, Visa and COD's, phone 717-398-1893. Also ask for our bargain filled catalog of OSI used computers, boards and software.

* * * * *

WANTED: One 'BASE 2' printer, for parts. Henry Jata, 1424 Newbridge Rd. Bellmore, NY 11710.

* * * * *

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

* * * * *

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

* * * * *

FOR SALE: C4P, cassette based, 8K. Has OSI manuals, Sams Photo Facts Manual, about 1 dozen cassette programs (some Aardvark) a few years accumulation of PEEK(65)s and Aardvark journals. $75 (negotiable). Steven Defoster, 23 Cleveland Ave., Binghamton, NY 13905, (607) 797-4625 after 6 on weekdays. Anytime on weekends.

* * * * *

Ohio Scientific Business User Needs: Working Letter Express with WP3 or WP2 or other Mail-Merge Program. Serviceable terminal (No ACT-V). Will sell MIME314 w/defunct CPU for parts $100 or best offer. Call Haber & Associates (202) 328-0888.

* * * * *

FOR SALE: OSI C3-OEM (1-2 MHz) with dual double sided 8 inch Shugart floppy drives. 48K plus 8K for CP/M. Televideo 912C terminal. IDS 440 printer. Sola power conditioner. 4 new head cleaner disks. Disk storage tray with 73 disks. Software: OS65D V3.2, OS65U V1.42 and V1.44. OSDMS nucleus and sort. C3 utilities. WP6502 word processing. Misc other programs. Manuals. All in excellent condition. <u>BEST OFFER!!</u> Call John O'Malley, Miami, FL (305) 759-6468 evenings or weekends.

* * * * *

OSI C3A for sale. 56K RAM, 2 DISKS, OSI Software plus, CPM Wordmaster and Spellguard, plus Epson MX-80 printer. $650, C3A alone: $400. Brian Armstrong, 671 Live Oak, Menlo Park, CA 94025, (415) 325-7886.

* * * * *

C3-OEM, ADM 3A terminal, 65U-V1.43, 65D-V3.3, fully integrated accounting software - G/L, A/R, A/P, P/R, ready for complete business accounting, also W/P and data base mgmt software, all with complete documentation and free phone consultation. $1900 or best offer. Mac Claxton CPA, 11303 Kettering Place, Largo, MD 20772. (301) 336-6677.

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 OS-65U. Complete disk sort and merge, documentation shows you how to call from any BASIC program on any disk and return it or any other BASIC program on any disk, floppy or hard. Most versatile disk sort yet. Will run under LEVEL I, II, or III. It should cost more but Sanders says, "...sell it for just..."	$89.00	$
( )	**KYUTIL** - The ultimate OS-DMS keyfile utility package. This implementation of Sander's SORT/MERGE creates, loads and sorts multiple-field, conditionally loaded keyfiles. KYUTIL will load and sort a keyfile of over 15000 ZIP codes in under three hours. Never sort another Master File.	$100.00	$
	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 $1.90	$
Postage & Handling	$ 3.70
TOTAL DUE	$

POSTAGE MAY VARY FOR OVERSEAS