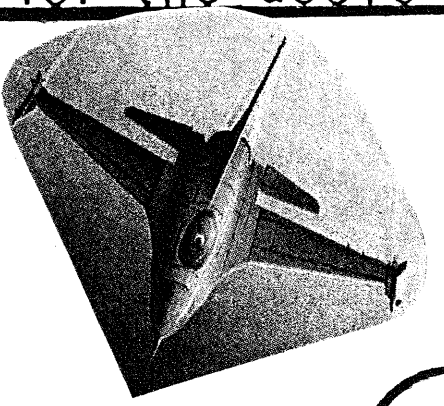
January 1988

# TS·2068 UP·DATE

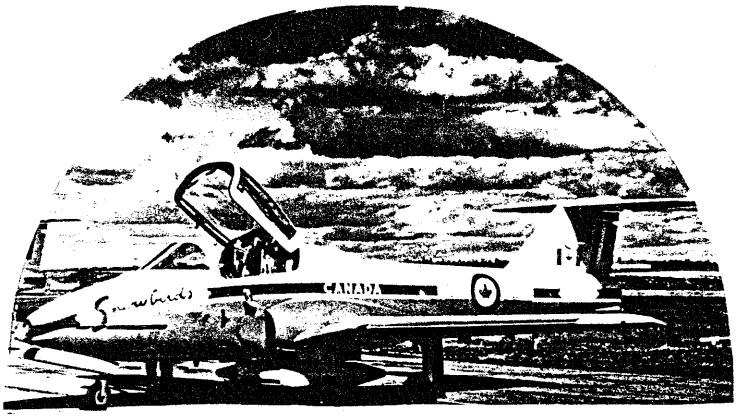
## the user's NEWS

*Take it to the limit*

A quarterly Magazine for the users



# EXTRA MEMORY



TS-2068 UP-DATE
1317 Stratford Ave, Panama City, FL 32404
(904 871 4513)

## The Extra Memory Section

### Do we Need Extra Memory?

Good question, especially now that disk drives are on the scene for the TS-2068. A Disk Drive is actually a "box of extra memory". The problem is, one has to place the data from disk INTO the TS-2068 memory chips for it to be used. So, that means that we are limited by the FREE memory of the TS-2068 as to the LENGTH of the program that it can handle. FREE MEMORY is 38652 bytes. Now if we fill up that memory with a program, there is no space left for DATA to be stored, so we have to reserve an amount of FREE to handle data that will be generated by the program. Usually the trade off is about 50-50. Then a program will take up about 19K and there will be an equal amount of space reserved to store and manage data.

19K of memory is OK for games and trivia but lacking to store extensive software programs. Also, 19K is a bothersome limit upon the amount of data that can be stored. There should be a way of "stretching" FREE memory so that longer and more useful programs can be operated and still have enough FREE memory for data. THERE IS! We can use the DOCK BANK for programs and the HOME BANK for DATA. We can put a 64K program in the Dock Bank, and keep almost all of the 38K of FREE memory for data storage and management. This capability of the TS-2068 has been neglected and few realize the increase in power that the use of EXTRA MEMORY BANKS offer. Let me put it this way. If you give me twice as much memory storage to store PROGRAMMING, I can give you TEN times as much software power.

A software program is built with "kernel routines", packages of programming that are used over and over by different elements of the program. As program length increases there is less need to create new kernel routines, and the programming becomes quite cryptic, using mostly "GO TO and GO SUB" within the program lines. Also, it is not necessary to create new variables. Then one more "K" of programming can produce functions that would require as much a 5 or 6K of programming to produce without the kernel cells. So then, picture the increased capability that results from using 64K of memory for a program in the dock bank, as versas the ordinary limit of 19K! Then, the FREE memory of 38652 bytes is available for DATA management and storage.

THE LONESOME DOCK: In the beginning Timex started producing Dock Cartridges with programs. These were mostly games like "Flight Simulator", good games but soon tiring. Since then few industries have produced software in cartridges. As a result, the Dock Door is hardly ever lifted, and the capability has languished. One reason for that is the cost of producing both the cartridge and the software in EPROM. A reasonable price for a good Dock Cartridge software would be around $60.00, too expensive for our tastes, and the reason for our ignoring the Extra Memory functions. One other reason is the trickiness of programming required for using extra memory. But, IS it all that tricky? And, CAN WE LEARN HOW?

Ordinarily, publications must reserve their page space to cover many subjects, and the detailed continuous coverage of one subject cannot be attempted. UP-DATE will undertake to bring you detailed and continuous coverage of EXTRA MEMORY in this section. We start in this issue with The Concepts Of EXTRA MEMORY PROGRAMMING. We will progress through the details of programming in extra memory banks, Non Volatile Ram, the use of EPROM Cartridges, Programming EPROMS, the use of RAM DISK, and the use of Disk Drive to support extra memory programs. This issue also brings an article in the Technical Section about converting a TS-1000 64K RAM to a DOCK Memory board. Programming will be given in both BASIC and in Machine Code, starting in easy to understand basics.

There are people who are intensely interested in this subject and have done some fantastic things. Mr. Craig Davis is engaged in transferring the Smart Text software to run in Dock Bank EPROM, coordinating disk drive for both screens and memory storage. Hopefully he will tell us about it in a future article. I will be discussing the use of RAM in Extra memory. Mr. Larry Kenny (Larken Electronics) can tell us things about RAM DISK. And, Mr. Eric Johnson has promised a construction article on Extra Memory RAM. If you have done any work in this area, UP-DATE will publish your articles. New user interest will bring new product development. Lets take extra memory "to the limit"!

## Jan. 1988 Editorial

Hi, you TS-2068 Die Hards! Die? Heck, I'll be using my '68 till they back up the hearse to my widow's door. This January 1988 issue of UP-DATE brings in some new writers, new to UP-DATE, but not new to the TS-2068 user groups. I'll give you a short introduction to each of these outstanding writers at the end of their columns. They are successful business people whose time is valuable, but still they find time to contribute their knowledge FREE. We can show appreciation by a card of thanks. Also, this issue begins new building block sections about Extra Memory, Larken Disk, Aerco Disk, and Telecomputing. These sections will build as more issues are published. Hope you like the direction that UP-DATE is 'a goin.

UP-DATE will let our writers "hang it all out", praising the systems that they like. Enthusiasm unbounded! All of our disk systems are good, each having its own better features. We tie these systems together with review articles in this issue that should form a departure point for continuing coverage. For those who dont yet have a disk system, this should be a mouth watering bit of reading. Even those who have disk systems should find some new information in the reviews. Want TWO DOS systems with your present equipment? That too!

UP-DATE DISK SUPPORT: The capability now exists to provide disk support to users who have the Amdek Amdisk 3" disk drive. However, due to the cost of the 3" diskette, all who order the Issue Disk on 3" disk should send a disk to have the programs recorded thereon. Issue diskettes are also available in all formats of 5 1/4", for Oliger Safe, Larken Disk, and Aerco FD 68. If ordering for 5 1/4" drive please specify for SS or DD, and 40 or 80 track. The October Issue Diskette is a success, with about 1/3 of the subscribers ordering. That Mail Merge program really does its thing, for almost every type of Listing needs, from Mail Lists, to inventries, to invoices, to the daily unit milk producing records for a dairy herd. ~~Print products~~

It being C O L D up Noarth, this issue will try to warm you up by filling space with some Sunny Florida file photos. They are direct from the Chamber of Commerce. For those who dont know, Panama City is up in the "Pan Handle" of Florida, on the North Coast of the Gulf of Mexico. There are rolling hills, large navigatable rivers, limestone springs that gush sparkling water, dense hardwood forrests, and beautiful clear lakes. We have four mild seasons of the year. Our tourist seasons are the Summer for the Yanks and the Winter for our "Snow bird" Canadian friends. But our best seasons are the Spring and Fall, together about five months of delightful temperatures ranging from 65 to 85 degrees. It is over 600 miles from my house to Miami and another hundred down to the Keys.

WINTER FEST IN ORLANDO, FLORIDA: Now iff'n you want to warm up the bod and attend a TIMEX COMPUTER FEST at the same time, then comoan down to the Winter Timex Fest in Orlando in early MARCH 88. Thats the time when you think that Winter should be about over, but it 'aint. The details are: Place: Mariott Hotel, Orlando Florida. Time: March 4 through March 8, 1988. Reservations are necessary ONLY if you want to stay at the Mariott Hotel. Otherwise it is no sweat for motel space elsewhere around town. You are encouraged to pre-register for the FEST itself, but you dont have to to attend. Registration at the door is $8.00 Single or $12.00 Family. Pre-registration by mail is $5.00 Single or $9.00 Family. Mail to: Winter Timex Fest, 249 N. Harden Ave., Orange City, FL 32763. Contact is Eric Johnson at that address. The Mariott Hotel is at the South edge of Orlando, very convenient to Disney World, Sea World, Circus World, Cyprus Gardens, and other World Family attractions. It is about 45 miles to the East Coast beaches and the Space Center. What a great opportunity for a combined Winter Fest and Family vacation! Your hosts are the combined TSUGs of Florida.

FILING UP-DATE PAGES: The pages are numbered by the color section that they belong to. Please remove the staples and file the pages in your UP-DATE booklet. New page separators are the front and back covers of this issue titled: "EXTRA MEMORY" and "Telecomputing". UP-DATE needs more contributing writers to cover these special subjects: Oliger Safe System, Larken Disk System, Aerco FD-68 Disk, TOS (Zebra) Disk, CP/M and RP/M, Telecomputing, and general articles on programming. The sharing of your knowledge will be appreciated by all.

NEW SOFTWARE: This issue brings some new and interesting software in the yellow pages. Herb Bowers, an eminent authority on INCOME TAX, is just in time for our dreaded TAX REPORTS. Herb promises to save you money with his software, and GET THIS! He climbs way out on a limb and sez that if you use his tax program and are penalized by IRS, HE PAYS THE PENALTY! Now that is what I call CONFIDENCE. Also, Herb brings us "Vanna Brite" (Hmmm).

Then JOHN McMICHAEL offers a interesting combination of hardware and sofware to use the inexpensive Commodore Plotter to draw airplanes in flight, pretty girls, and fantastic graphics. One needs to buy a $49.95 Commodore plotter to do alla that. I'm gonna do it because the plotter is a nice peice of equipment and the new capability will be both fun and useful.

Next, a new edition of Smart Text for Oliger V2-35 EPROM is offerred. This one has FULL EDIT during input typing, a expanded Mail Merge annex, Automatic Column printing, and a special printer set-up for Daisy Wheel printers. These columns are samples of the Auto Column Printing function. No Cut and Patch, the columns print automatically.

UP-DATE HEADQUARTERS has added quite a group of things. Added for your support are, the Amdisk 3" drives, Larken SKDOS Cartridge, Avantex 300-1200 baud modem, Specterm SI/O interface, Specterm 64 Software, another TS-2068 with a 4 Drive system, and Aerco FD-68 Disk system with 256K Memory. Already on hand was the Oliger Safe System. You should notice an improved print in this issue resulting from the addition of a Daisy Wheel printer. I hope to find some BBS boards somewhere to down load some CP/M software for the Aerco FD-68 users. Anybody got a good source? CP/M is a subject that needs exploring in depth. Up-Date needs a writer who is willing to take us by the hand.

THE UP-DATE BBS: As the learning curve flattens out, a UP-DATE BBS will be set up using the Aerco 256K RAM as RAM DISK, with the Larken SKDOS Cartride, augumented by about 3 Mbts of disk storage. What I'm planning is a informal BBS that is exclusively for the subscribers and has about three sub boards. One will be for program downloads, another for friendly message interchange, and one for shopper service. Suggestions and helpful guidance will be appreciated.

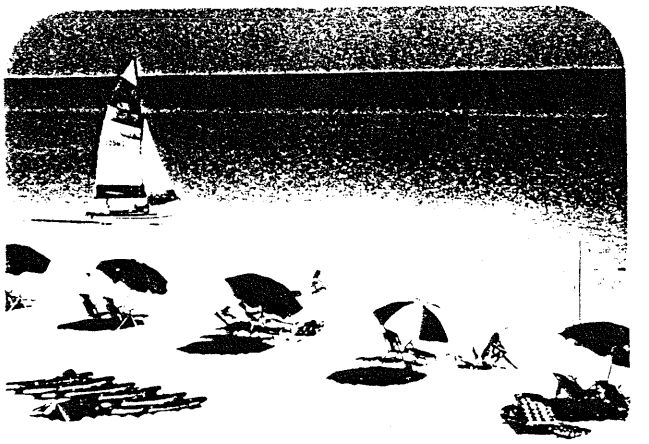
**A Cheap and Excellent Twin Disk Drive**
**For $49.95**

Yep, a Twin Disk Drive, in a cabinet, with built in power supply, ready to plug in and use, FOR $49.95 plus shipping!! That is the Amdek Amdisk III, a neat little 3 inch drive. There are a couple of catches to the deal. The diskettes are not "Floppies" but "Hardies". The diskettes are more like a thin cassette, a sandwich wafer of hard plastic that protects the enclosed disk. The disk itself has a center sprocket that the drive mechanism engages to spin the disk. It appears to be designed to enhance long life of the diskette. The catch is, the price of the diskettes ($4.95 each). That would be bad if one had to replace them often, but I believe the diskette should last for at least a year or more. Now ordinarily, a twin Disk Drive unit, will cost about $240.00, the price broken down to $160.00 for two drives, plus $80.00 for the drive enclosure and power supply. So, you can get the Amdisk III for $49.95, plus 10 diskettes ($49.95), and still be ahead about $140.00. One more catch is the fact that the disk drive reads and writes to one side of the disk at a time. This may not be a biggie as each side formats to 195K of disk space for program and data storage. When one side is filled with data you just flip the disk cartridge over for the next 195K of storage. The total disk capacity (both drives) is 780K. A nice affordable disk drive system.

Later I hooked up the Amdisk drives in series with two 5 1/4" drives to operate as "drives 2 and 3" of a four drive system. This allows easy copying of programs and data between standard 5 1/4" disk and the 3" drives or direct LOAD and SAVE to either.

The Disk Drive comes with a data cable that wont fit your disk controller. That is only a minor problem, as the edge card connectors of the drives inside are identical to the edge card connector of your controller card. I had a data cable fabricated at a local computer repair shop (cost $21.00). Then it took only a few minutes to remove the case top and replace the existing cable with the new one that fits the TS-2068 Disk Drive Controller. Problem solved, but you can add $21.00 to the system cost, making it $120.80 for the Amdisk twin drive plus the 10 diskette cartridges. The drives operated perfectly upon first hook-up. Also, it has a cooling fan to keep down heat problems. Many of the more expensive drive enclosures lack good heat dissipation. This is a fine disk drive system, but dont expect to be able to buy software on these expensive 3 inch diskette cartridges. I am glad that I purchased mine because they make excellent additional disk capacity to my other drives, and I believe that the diskette cartridges will last forever. Source: Peripherals Direct Ltd., PO BOX 4301, Northbrook, IL 60062, Tel: 1 800 332 9988. They take plastic money but charge a service charge for it.

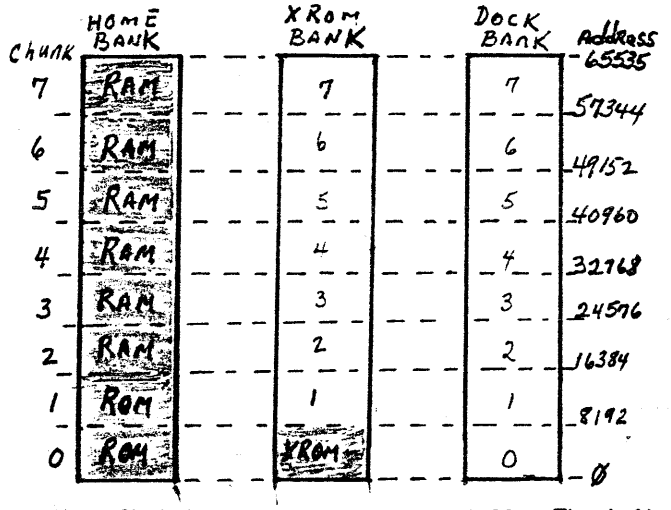
I am fascinated by a ad in Computer Shopper for a Zerox 16/8 bit CPU Computer with 10 mbt hard drive that sold about two years ago for over $4000.00 and is now offered at $349.00. That might make a better BBS system if I can find a BBS software for it. Subscriptions to UP-DATE are now at about 150 and still building. I have received a lot of compliments and no gripes, which makes me want to try harder. Wishing everyone a successful 1988 year. Ed.



## Extra Memory Programming

Most of us can visualize the computer's "regular" memory. We know that the ROM part of memory has "fixed" machine programming put there by Timex when they built the computer, and that there is a section called RAM that we store our own programs in. To carry this visualization further, we will do some diagrams that splits up the memory into CHUNK SECTIONS. The chunks will be identified by the ADDRESS that the computer's CPU uses to send and receive data to and from the chunks of memory. Please refer to Figure 1 for the visualization.

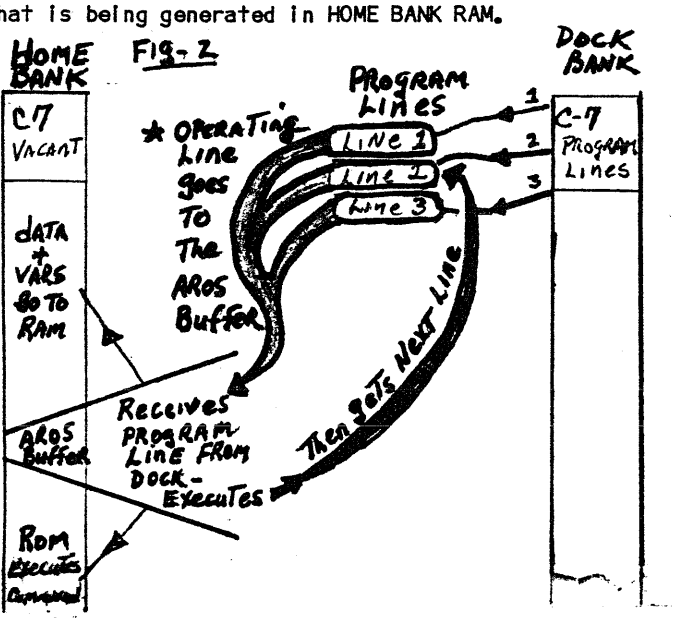
Figure 1




Now that is a busy diagram isn't it. The left block is what we normally visualize, having two sections of memory, "ROM and RAM". The center block depicts another 64K "BANK of MEMORY" called "The XROM BANK". Timex put only 8K of memory there to handle such things a Cassette Load and Save. When we type a LOAD or SAVE command, the computer switches OUT Chunk 0 from home bank and switches IN Chunk 0 of the XROM Bank, to do the LOAD or SAVE. Notice that the other "Chunks" of the XROM Bank are vacant, having no Memory Chips installed. Timex didn't even put sockets under the hood for more memory chips. Also, notice that the block on the right is named, "The Dock Bank", and all three BANKS have the same CHUNKS identified in the same manner.

A CHUNK is 8K (8192 bytes). As we get into Extra Memory we will transfer CHUNKS of Memory Chips IN and OUT of the range of addresses that the CPU controls. As we visualize the computer, the CPU "Sees" the memory chips that are in the left block, "the HOME BANK". Now suppose that we add some memory chips and fix them so that they are in "CHUNK 7" of the DOCK BANK (the right block of figure 1). Would the CPU "see" that group of memory chips? NO, because the CPU can see (address) only 64K at one time, and the TS-2068 is fixed so that the HOME bank is what the CPU normally sees. Here is where PROGRAMMING comes into the act. We can do some POKES, and wipe out CHUNK 7 of the HOME BANK and make the CPU see CHUNK 7 of the DOCK BANK instead.

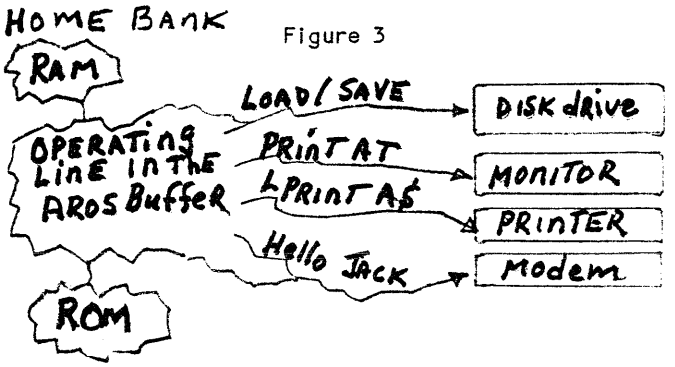
When we do that, Chunk 7 of the HOME bank will still be available for storage of data. Now that is confusing, so we will transfer attention now to figure 2 to see how that can be possible. In figure 2 we delve into the way a program line works from a Extra Memory bank. The senario is: We have a BASIC program in Chunk 7 of the DOCK Bank, and that program is running. The TS-2068 always runs programs lines in the HOME BANK, no matter what bank the program lines are stored in. The ingenuous way that this happens is kinda confusing and we will delve into that later. The key point here is: Since each program line runs in the HOME BANK, the CPU "sees all 64K of the home bank", and Stores DATA that is being generated in HOME BANK RAM.




Still referring to Figure 2, the lines of programming that are in the Dock Bank are picked up ONE LINE AT THE TIME, placed in the "AROS BUFFER" of the HOME BANK and operated. Say that the line has programming that sez <LET A$="I go to Church on Sunday">. Then the contents of A$ is stored in the HOME BANK RAM. Then when the last part of the line of programming is finished, it is replaced with the NEXT line of programming that is in the DOCK BANK, and that line is executed. The Dock Bank program continues to run, one line at a time, until the last line of the program has finished. So, in effect, the CPU can manipulate TWO 64K banks of memory at the same time.

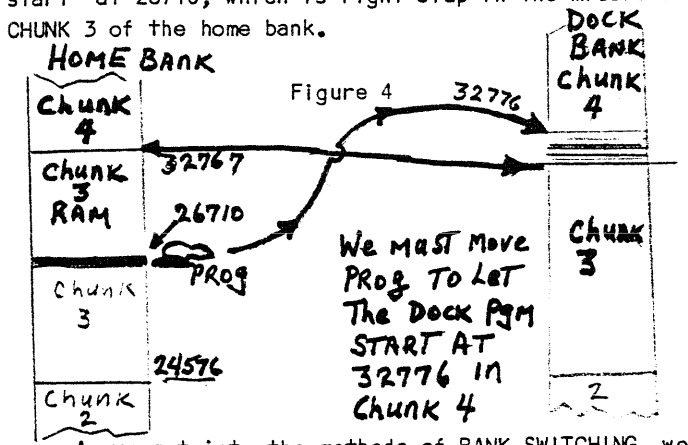
Now to complete the "visualization", lets transfer our attention to figure 3, where we have added a disk drive, a printer, a monitor, a modem, and a cassette. We are still operating the program that is in the DOCK BANK, one line at a time. The LINE of programming that is operating is in the AROS BUFFER of the HOME BANK. That line of programming can contain LPRINT COMMANDS, or PRINT commands, or DISK LOAD/SAVE commands, or Cassette commands, or just anything that the TS-2068 can do. If data variables are created, as when receiving data from a modem, the data is stored in the HOME BANK RAM. OR, data that is in the home bank Ram may be printed by a LPRINT command in the program line.

Figure 3



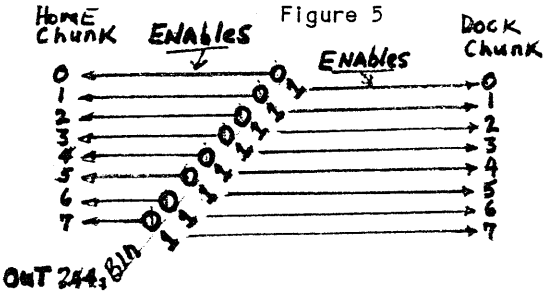

Now that we have the concepts of Extra Memory Operation in focus, we will go on to the "how to do it" phase. One could survive by just plugging in a cartridge and typing RUN, but since Dock Cartridge Software is about as scarce as hen's teeth, we dont have that easy option. Besides, we want to learn how to do it anyway. Now we will look at figure 4 and see a fly in the soup. The soup is HOME BANK Memory, and the pesky fly is "PROG". PROG is the memory address where the CPU finds the address of the first byte of a BASIC program. Normally PROG is at address 26710, and the first line of a basic program will start at 26710, which is right slap in the middle of CHUNK 3 of the home bank.

Figure 4




As we get into the methods of BANK SWITCHING, we must tell the CPU "where to find PROG". We will do that later. Right now we must learn how to switch banks. We will use the term "Enable" to mean the chunks of memory that the CPU sees for program lines. For example if we enable Chunks 4 through 7 of the DOCK bank, then the CPU will not see chunks 4-7 of the HOME bank, and WILL see chunks 4-7 of the DOCK bank. Bank switching is accomplished by using the OUT command. "OUT port 244" is used to designate the chunks of memory to enable. It is easier to visualize if we use BINARY numbers with the OUT command. OUT 244, BIN 00000000 would enable all 8 chunks of the HOME BANK, while OUT 244,BIN 11111111 would enable all 8 chunks of the DOCK bank. Please refer to figure 5 for a visualization of the effect that each "bit" of the binary number has upon the chunks of memory enabled.

Figure 5




The easiest and most simple way of using extra memory banks is to enable chunks 4 through 7 at one time. OUT 244,BIN 11110000 will enable chunks 4 through 7 of the Dock bank. Then if you want to stop using program lines in the dock bank, OUT 244,BIN 00000000 would enable all 8 chunks of the HOME bank. Remember, each of the bits affect a single chunk enable. Zero enables a Home bank chunk, and "1" enables a dock bank chunk. The "OUT 244,BIN number" can be done in the direct mode or from within a line of programming to switch bank chunks.

Now suppose that we want to use a program that is in the Dock Bank chunks 4 through 7. OUT 244,BIN 11110000 would do it. BUT, the CPU looks for the first line of a program at the address of PROG, and PROG is sitting at address 26710 in chunk 3 of the home bank. (re-visit figure 4). Now we must change PROG to the address of the first line of our program that is in the DOCK bank. The first address of CHUNK 4 of the Dock bank is 32768. But we cannot start there, as the first 8 bytes of a dock program is reserved for "control codes". See the table of control codes in figure 6. Please study the table a bit and come on back.

Figure 6

| Adr. | Meaning | Value Significance |
|---|---|---|
| 23768 | Language type | 1=Basic and MC. 2=MC only |
| 23769 | Cartridge Type | 1=LROS or 2=AROS |
| 23770 | Program Start | A two Byte Address |
| 23771 | | Second Byte |
| 23772 | Chunk Select | BIN 0=HOME. 1=Dock |
| 23773 | Auto Start? | 0=No and 1=Yes |
| 23774 | No. of Bytes | 2 Byt Nbr for lngth of |
| 23775 | reserved | MC pgm starting at 26688. |

## EPROM OR RAM?

O.K. You can now see that we want to move PROG to address 32776 which will be the first byte of the program lines in the dock bank. We can do that, but now we are getting to a point where procedures vary between using RAM or EPROM in the Extra Memory Bank.

The discussions so far have been applicable to both EPROM and RAM in the dock bank. We will now concentrate upon handling RAM Memory, and leave EPROM for another article. One canot just type and ENTER program lines into the Dock Bank as we are accustomed to doing. It is necessary to first put the program lines in HOME BANK Memory, and then move the program lines into the Dock Bank Memory for operation. The objectives and steps of operation are: 1. Move PROG to Adr 32776. 2. LOAD a Basic program from either Cassette of Disk. 3. Move the Basic program that is in Home Bank Memory into Dock Bank Memory. 4. Make the necessary POKES to the first 8 bytes of Dock Bank Memory. 5. RUN the Basic Program that is in the Dock Bank. Most of these functional steps may be accomplished by adding some short utilities to the Basic Program after it is first loaded from Cassette or Disk.

## THE UTILITIES

### Sources and Credits

### The TS-2068 Technical Manual,

### Mr. Thomas B. Woods' Non Volatile Ram Manual

PLANNING: We will create a group of utilities to ADD to ANY BASIC program. After the utilities are added, the program can be SAVED with a starting line number of 9800. When the program is RE-LOADED, a menu will appear. The menu choices will allow us to elect whether to MOVE the program into DOCK RAM, or to operate the program in the HOME BANK. If DOCK OPERATION is elected, the added utilities will be exercised, but if HOME Bank is elected, the utilities will be ignored. We may add this programming to all of our BASIC Softwares and utilities to have the choice of HOME or DOCK Bank operation. It is assumed that there is a set of RAM memory chips installed and set up to operate in the Dock Bank. To begin, we LOAD a Basic program from Cassette or disk and add the following program lines.

THE MENU: 9800 INK 7: BORDER 0: PAPER 0: CLS: PRINT AT 10,4;"YOUR CHOICES ARE:"''TAB 4;"1. Operate in HOME BANK"''TAB 4;"2. Operate in DOCK BANK": INPUT A: IF A<1 OR A>2 THEN GO TO 9800

9802 IF A=1 THEN RUN: REM** You may replace RUN with a GO TO the program starting line.

9804 CLEAR

9999 RESTORE 9999: FOR X=23296 TO 23340: READ Y: POKE X,Y: NEXT X: RANDOMIZE USR 23296: DATA 175, 6,2,33,83,92,94,35,86,213,33,89,92,16,247,225, 209,237,82,68,77,235,175,211,244,126,245,62,240, 211,244,241,119,11,35,229,33,0,0,237,66,225,32, 234,201

When DOCK is elected, the above routine MOVES the BASIC Program into the DOCK RAM. Next, the AROS Overhead bytes will be poked.

9999 RESTORE 9999: FOR X=32768: READ Y: POKE X,Y: NEXT X: OUT 244, BIN 00000000: DATA 1,2,8,128,15,0,0,0

Now SAVE your Basic Program, with the above additions to Cassette or disk. You may add the same lines to your other Basic programs that you may want to use in the dock bank. IMPORTANT: The basic program, when re-loaded WILL NOT operate in Dock Bank without the following being accomplished FIRST.

MOVE PROG FIRST

MOVE PROG: Before each LOAD of a Program to operate in the dock bank, PROG must be moved to the starting address of the Dock Program lines, which is 32776. Otherwise, the CPU will look for the normal starting address of 26710, which address will be in the HOME Bank chunk 3. Therefore, the operating procedure must be to FIRST, MOVE PROG, THEN LOAD the Basic Program that has the utilities installed. Then the program is moved into dock bank with a proper starting address. The following program to MOVE PROG must be the first step.

9999 RESTORE 9999: For X=23296 TO 23304: READ Y: POKE X,Y: NEXT X: RANDOMIZE USR 23296: DATA 33,85, 104,1,178,23,195,187,18

The above utility should be entered just after the computer has been turned ON. ENTER and RUN the program, then Delete the line. DONT type NEW, as that would RESET PROG to 26710. Imediately after running the above utility, and Deleting the line, go ahead and LOAD your Basic program that has the new menu and MOVE programming. Elect option 2 in order to move the basic program into the dock bank. When the utility has moved the program into dock, you should have a blank screen.

### The Situation

If you have been following through so far, you now have your Basic program in the Dock Bank. It may or it may not operate, depending upon several factors. First, there are no variables anywhere. Variables are not stored in the dock bank, and we didn't store any in Home bank. Also, we didn't delete the program from home bank, so it will still be there. And, because at line 9999 we did an <OUT 244, BIN 00000000> we are now operating in the Home bank. And, because at 9999, we POKED 32773,0, the program that is in Dock does not Auto Run. So, right now, the CPU "sees home bank" and the program that is

still there. So, to continue, we should type <DELETE 1, ENTER> to get rid of the home bank program.

Next, to get into operation in the dock bank, we need to <OUT 244, BIN 11110000> and <POKE 23750,128>. Then <GO TO the start line of your program>. Try it if you have moved a program. If it doesn't operate right, then join the 100% club, a group of extra memory users who ALL had their first Moved program to be not fully functional. Then we will proceed with some more learning. But maybe it worked fine and you dont need to read further? If so, please write an article for UP-DATE!

Continuing for the new member of the 100% club: Dont give up! Be persistant! Cuss maybe, but dont quit! The rewards will come! That program wasn't important anyway. And it is likely to be hiding in the dock bank right now. You cannot see it, because you cannot LIST a Dock program. Thats the reason why we must always carefully edit and fix a program before putting it in a extra memory bank. Some needed to know facts are: You can LOAD a variable file in two ways. 1. A program line in the dock bank can be: <LOAD/"name"VAL> or other disk syntax to load a vars file. 2. You can POKE 23750,0 to get into the home bank. Then vars files may be created with program lines, or LOADED from disk.

A third way of creating vars is to do it in program lines in the dock bank, like LET A$= "I really dig this". Then the vars so created will be stored in the home bank. You see, when a program line is executing, it is IN THE HOME BANK, the CPU "SEES HOME BANK", and variables created are stored in home bank RAM. Next we will look at some more key memory addresses that have names. ARSBUF is adresses 23748 and 23749, and is the pointer to the AROS BUFFER. ARSFLG is Address 23750, and is the pointer to the current start of a AROS Data Line. Also 23750 acts as a switch to switch operations between the HOME BANK and the Dock bank. POKE 23750,0 enables the HOME bank, and POKE 23750,128 enables the Dock Bank.

Other Key addresses are: ADATLN is Address 23751 and 23752 and is a pointer to the current Start line of a AROS Data Line. ADATLENG is address 23753 and 23754 which stores the length of the current AROS DATA LINE. Of that group you will get the most use of Address 23750, the switch for changing banks. After POKE 23750,0 you can enter program lines in the home bank, or LOAD another program from Disk or Cassette. While experimenting with changing between Home and Dock, try combinations of <POKE 23750,0: OUT 244,BIN 00000000> and <POKE 23750,128: OUT 244, BIN 11110000>. Keeping notes of results will be helpful.

PROGRAMMING LIMITATIONS: There are some restrictions on commands that can be used in EXTRA memory program lines. ON ERR GO TO cannot be used. But to skirt around this a ON ERR line can be put in the home bank, and "switched to" for use of that powerful routine. Example: your program line is <100 ON ERR GO TO 500> and <500 ON ERR RESET: do something: GO TO 102>. Now if those lines were in a program to go into DOCK Bank, they could be re-arranged as: <100 POKE 23570,0: OUT 244, BIN 00000000: GO TO 10>. Now that would enable the home bank where a line of programming could be: <10 ON ERR GO TO 20> and <20 ON ERR RESET: do the same thing: POKE 23570,120: OUT 244, BIN 11110000: GO TO 102>. Study that out for another way to skin the cat.

The Oliger SAFE "Fast FOR Counter" is not acceptable programming in Dock Bank. But, you can do a similiar switch to Home Bank, use the Fast FOR counter, and when finished, switch back to Dock Bank and continue. Disk Drive LOADS and SAVES of Character arrays, Vars files, and Screens work fine in Dock program lines, but Not BASIC Programs that use starting line numbers or machine code tables. When you are using Extra memory, PROG is set UP to Chunk 4, leaving about 6K of address spaces vacant in chunk 3 of Home bank. Since this space is un-used, it could be used for Machine Language programs to be executed by RANDOMIZE. Also, that 6K is subtracted from FREE for Home Bank programs.

Finally, an awful lot of territory has been covered in this article to get from ground zero to "somewhere". We have just scratched the surface of Extra Memory. Many facts have been left out,espically such things as how to use Machine Code utilities and programs. And, we have covered only the Dock Bank of extra memory. UP-Date will continue to provide as much useful information as can be dug up about Extra Memory programming and devices. Both UP-DATE and the subscribers will appreciate writer contributions to help in the understanding of this broad and relatively unexplored subject. B. Jones



## Dock Bank Memory for Twenty Cents
### Convert a TS-1000 Ram Memory For the TS-2068

Many TS-2068 users graduated from the TS-1000 and still have a 64K Ram for that classic that is collecting dust somewhere. I ask Mr. John Oliger to give us information about converting his 64K Ram for the TS-1000 to use as DOCK BANK MEMORY. He complied and these instructions are simple and easy. Other TS-1000 memory boards may be converted in similiar manner. Why dont one of you who have done it to a Memotech Memory send in the schematic changes? Given here is the Oliger Schematic with insets showing the modification details. When the changes are made, the 64K memory is split into two 32K banks addressed 32768 through 65535, chunks 4 through 7 of the TS-2068 memory map. The "Bank A/B" switch selects the Bank that will be active. The data that is placed in either bank will be kept refreshed by the computer when the computer is ON.

The "48-64K" switch is used to TURN ON or OFF the UPPER 16K of memory of the bank selected by the A/B switch. If the 48-64K switch is ON, then the entire 32K of the bank selected is active in the dock bank. If the 48-64K switch is OFF, then only the LOWER 16K of the bank selected (chunks 3 and 4) is active in the dock bank. But the upper 16K region is still kept refreshed by the computer. This allows one to elect programming in the dock bank in six combinations of 16 and 32K groups.

John Oliger warns that no one should consider building this memory board from scratch because of the high cost of the 4164 "pin 1 refresh" chips. But if one is already on hand, the cost is about 20 cents to add 64K of Dock Memory to your TS-2068 (two 1K resistors). I could afford twenty cents, so I dude it and it works! When finished, I just stuck the memory board in the empty slot of the Oliger Expansion Port and started programming in the dock bank. Maybe you know someone who has one of these outstanding 64K boards and can trade them out of it? Now, programming dont just jump into that new Dock Bank Memory. Thats a little tricky and one must learn how. See the articles in the UP-Date section titled "Extra Memory". A check list of the modifications is given below.

Check List of changes. Refer to scehematic on next page.

1. Cut the trace to pin 11 of U-4 (LS-86), OR Remove U-4 and Bend Pin 11 Outward.

2. Remove 12-16K Switch.

3. Remove 8-12K Switch.

4. Remove R-2 27K resistor and replace it with a 1K resistor.

5. Remove Diodes D-3 and D-4, OR remove U-5 (LS-138) and bend pins 12 and 13 outward.

6. Remove Diodes D-9 and D10 and Resistor R-4.

7. Remove Diode D-11.

8. Cut trace to pin 3 of U-4, OR Remove U-4 and bend pin 3 outward.

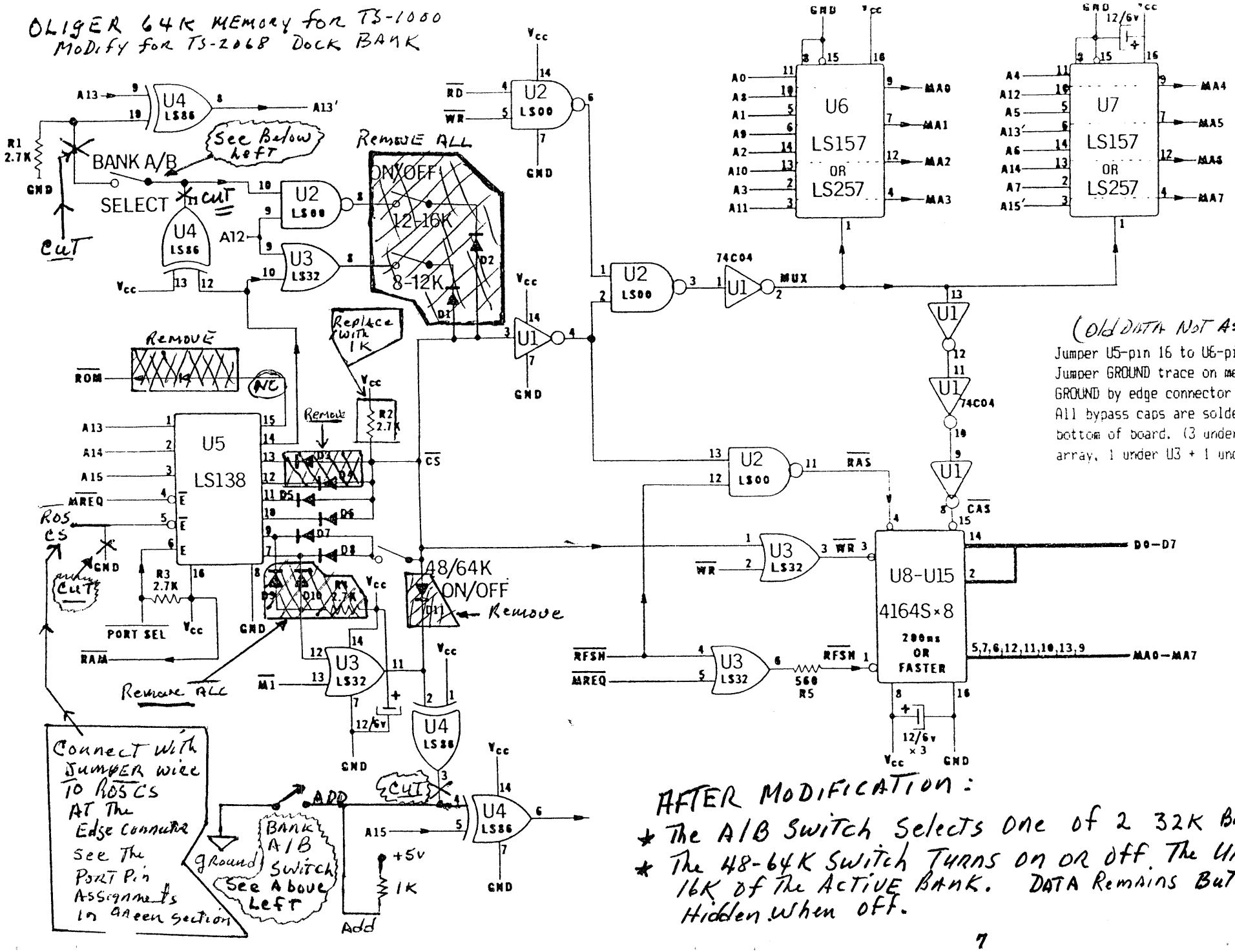
9. Remove U-5 (Ls-138) and bend pin 15 outward. Re-install.

10. Cut the trace from Bank A/B switch that connects to U-4 pin 10. Connect that side of the A/B switch to Ground. This leaves pin 10 of U-4 connected to +5 Volts.

11. Cut the trace from ground to pin 5 of U-5. Jumper the Pin 5 trace to ROSCS.

12. Connect a new 1K resistor between the side of the A/B Switch that is connected to U-2 (LS-00) pin 10, to +5 Volts.

# OLIGER 64K MEMORY for TS-1000
## MODIFY for TS-2068 DOCK BANK




(OLD DATA NOT Associated)

Jumper U5-pin 16 to U6-pin 16.
Jumper GROUND trace on memory to GROUND by edge connector 2 times
All bypass caps are soldered to bottom of board. (3 under memory array, 1 under U3 + 1 under U7)

## AFTER MODIFICATION:

* The A/B Switch Selects one of 2 32K BANKS.
* The 48-64K Switch TURNS ON OR Off The UPPER 16K OF The ACTIVE BANK. DATA Remains But IS Hidden When Off.

# HANDY REFERENCES AND A SAFE-DOS FILE INDEXER

Bob Hartung, 2416 N. Co. Line, Huntertown, IN 46748

When Bill Jones asked me to contribute some articles for SDU my first impulse was to reply that I would rather just sit on the sidelines and pick the brains of his other writers for awhile. Because the things I have written since the days of SYNC and SYNTAX have been simply the sharing of my learning experiences with T/S computers, I felt that most SDU readers probably would have advanced far beyond anything I might be able to offer. Like many T/S users, my first hands-on experience with a computer of any kind started with an unexpanded ZX80, to which I later added the 8K ROM and the notorious RAM-pack, moving up to the 2068 as it became available, and most recently a QL when they dropped in price below $100.

(As an aside, although I am very impressed by the QL SuperBasic as a quite powerful programming language, especially when it is extended by Super Toolkit II, I much prefer the 2068 Basic. When a T/S is extended by Beta Basic 3.0 which adds over 100 new or enhanced commands and functions, including procedures, to a Spectrum-emulated 2068, almost all of the more significant programming features of SuperBasic other than multi-tasking are made available but with much easier key-in, syntax-checking, editing, and often faster processing. Most of these new BB commands and functions have 2-10 optional formats to further extend their capabilities. For anyone interested in what Beta Basic does, by permission of BetaSoft I have written a 20K-byte demo program that includes an uneditable version of Beta Basic to actually run typical listings of about 85 of these BB commands. It is available for $5.00 to cover the cost of tape, postage and handling.)

After carefully reading through the first issue of SDU and the useful tables Bill gave us, it occurred to me that another one which might be included is a list of some of the more frequently used system variables. I have copies of appendices B and D from the 2068 manual on the wall back of my work desk, but I find it handy to have these excerpts (LISTING 1) taped to the right-hand side of the computer for quick reference. If the list is covered with clear packaging tape and attached only at the top edge it does not interfere with using the cartridge port.

## DOSDEX--AN AUTOMATIC FILE INDEXER

The DOSDEX file indexer is an adapation of the excellent menu loader written by Roelof Mulder and John Oliger and included in the SAFE DOS documentation. On my CPI 80-track drives, one DOSDEX disk can hold up to 165 disk catalogs, each of which can contain up to 165 individual file titles, or 27225 in all. These are displayed in the same format as the M/O menuloader and are sequentially accessed by keying (CAPS SHIFT L) or just lower-case (l). If the appropriate disk is inserted, keying (ENTER) as in the menu loader will load the title selected by the cursor.

I added item numbering to the display of titles and file-types, without using the fast FOR/NEXT loop, so the additional processing time causes the display of menu pages to be noticably slower. (At my age, more is passing me by than I'm catching up with anyway!) If the eighth byte (14) is changed to an 8 and the listing is entered exactly as given here, the machine code routine from your menu-loader listing may be poked into a 39-byte first-line REM instead of a VARS location as the program was originally written. No other variable definitions may be placed between CLEAR and DIM c$ if the code is stored in a line 1 REM. With deletion of some screen effects and the line 10 q$ definition this allows DOSDEX to be placed in file 0 with SAFE v2.32 or later.

Or if you choose, you may simply adapt your original menu-loader listing with the required changes for DOSDEX and save it as the very first file. It will not fit into file 0 but you can use it along with the menu-loader routine which is in file 0. Note that token-words must be used as much as possible and line 200 of the original menu-loader listing must be moved up to line 5. In condensed form DOSDEX will just fit into one cylinder (5120 bytes) when expanded with catalog data, or if you prefer to retain the original screen format each index will use two cylinders, cutting in half the maximum possible number of titles stored.

To use DOSDEX, first save it (or the menu-loader) to track 0 on your index file disk with (CLEAR) (SAVE /0). Then make a save of DOSDEX as the first file on the disk with (SAVE / "DOSDEX" LINE 2). (This is necessary in order to initiate the routine if the shorter form of DOSDEX is in file 0.) For convenience in identifying your file disks, number them consecutively with a felt marker, numbered labels, or white-out as 0,1,2,3 . . . beginning with your DOSDEX disk. Since the storing of the respective indices requires alternately obtaining the CATalog data from each file disk, then storing it on the DOSDEX disk, it's a good idea to have write-protect tabs on all the file disks. (You could also adapt DOSDEX to a 2-drive system so file catalogs would be read from one drive and saved to the DOSDEX disk on the other.) The sequence for indexing on a single-drive system is this:

Insert the DOSDEX disk (0) and key (LOAD). (Key (ENTER) to load DOSDEX if it is stored as file 1.) Remove DOSDEX disk and insert file disk 1. Key (C) to CATalog the file disk into DOSDEX. (If you have keyed in (LET / P = 0) before loading DOSDEX and have your printer on, a hard copy of the file titles will also be printed out here for disk

labeling or a loose-leaf binder.) Remove file disk 1 and insert DOSDEX disk. Key (S) to SAVE as index "1". Repeat with file disk 2, and so on.

To read the DOSdex files, key (LOAD) with disk 0 in place. When DOSDEX is loaded, either from file 0 or from file 1, then key lower-case (l) or (CAPS SHIFT L) to LOAD first set of titles. Use any key except L to run the cursor through the page(s) as with the menu-loader, or key either lower-case or upper-case (L) each time you want to sequentially load each of the file indices in turn. If you want to update only one particular index, reset NEXT # (disk number) by keying (N) and insert the corresponding disk. Key (C) to CAT the updated info, then insert the DOSDEX disk and key (S) to SAVE it.

HANDY REFERENCES

```
DFILE1: 16384,6912
KEY REPEAT STOP: POKE 23561,0
"    "   DELAY: POKE 23561,n
"    "    RATE: POKE 23562,n
CHARS TABLE [+256]: 23606-7
  (Normal: POKE 23607,60)
KEYBEEP LEN: POKE 23609,n
VARS START: PEEK 23627+PEEK 23628*256
  (+6 for first CHR$ byte)
BASIC start: PEEK 23635+PEEK 23636*256
  (+1 for LSB of line no. or +5 for
  first character byte of line 1 REM)
LINE 0: POKE 26711,0
CALC STACK: PEEK 23651+PEEK 23652*256
CAPS ON: POKE 23658,8
  " OFF: POKE 23658,0
TV FRAMES: 23672-3-4
TOP-LINE SCROLL: INPUT "" or POKE 23692,-1
LAST-LINE SCROLL: PRINT AT 21,0 after above
UNUSED SYSTEM BYTES: 23728-9
RAMTOP: PEEK 23730+PEEK 23731*256
PRNT DRIVER: POKE 26703,LSB: POKE 26704,MSB
UDG start: 65368 (168 bytes)
JLO SAFE printer commands:
  LET /p=0
  OUT 127,27: LPRINT "...."
  Col/line: POKE 23323,n (default=255)
  LF after CR: POKE 23324,10 (0=no LF)
SCREEN LPRINT TO PRINTER: OPEN #2,"P"
  (Normal: CLOSE #2)
WARM RESET:  RANDOMIZE USR 0
```

DOSDEX LISTING

```
  1 REM 123456789 MC code here 4567890123456789
  2 CLEAR : DIM c$(VAL "178",VAL "20"): LET t=VAL "10": L
ET s=SGN PI: LET o=NOT PI: LET d=s
  5 LET a=VAL "PEEK 23635+PEEK 23636*256+5": LET c=INT (a
/VAL "256"): POKE VAL "23549",VAL "195": POKE VAL "23550",
a-(c*VAL "256"): POKE VAL "23551",c: LET fi=USR VAL "23549
": LET row=s+s: LET col=VAL "9": LET n$=c$(VAL "178", TO V
AL "16")
  20 LET a$="": DIM f$(VAL "6",s+s): LET f$(s+s)=" DATA n"
: LET f$(INT PI)=" DATA $": LET f$(VAL "4")="CODE ": LET f
$(VAL "5")="ABS ": LET f$(VAL "6")="VAL "
 300 CLS : IF n$(LEN n$)=" " THEN LET n$=n$( TO LEN n$-s):
 GO TO VAL "300"
 320 PRINT AT o,o;" Disk#";d-s;": ";n$;#o;"Key: NEXT CAT S
AVE LOAD (1)#";d
 405 LET f=s: LET c=INT (fi/VAL "18"): LET dif=INT ((fi/VA
L "18"-c)*VAL "18"+VAL ".4"): LET loop=VAL "17"
 410 LET q$="p": LET it=s: IF loop)=fi THEN LET loop=fi: G
O TO VAL "425"
 415 FOR i=s TO c: FOR m=o TO loop: PRINT AT row+m,t;a$;AT
 row+m,t-LEN STR$ it;it;" ";c$(it, TO t);" ";f$(CODE c$(it
,VAL "11")+1): LET it=it+s: NEXT m: GO SUB VAL "500": NEXT
 i: FOR i=s+s TO VAL "19": PRINT AT i,t;a$: NEXT i: IF NOT
 dif THEN GO TO VAL "410"
 425 IF f THEN FOR m=o TO dif-s: PRINT AT row+m,t;a$;AT ro
w+m,t-LEN STR$ it;it;" ";c$(it, TO t);" ";f$(CODE c$(it,VA
L "11")+1): LET it=it+s: NEXT m: IF loop)=fi THEN LET f=o
 427 IF NOT f THEN LET it=fi+s
 430 GO SUB VAL "500": GO TO VAL "410"
 500 FOR L=o TO m-s: PRINT AT row+L,col-s-s;"  "; INVERSE
s;")": IF q$()"" THEN FOR a=s TO PI+PI: NEXT a: LET q$=""
 510 LET a$=INKEY$: IF a$="" THEN LET q$="p": GO TO VAL "5
10"
 520 IF CODE a$=VAL "13" THEN GO TO VAL "600"
 522 IF a$="C" THEN OPEN #2,"P": CAT : CLOSE #2: GO TO INT
 PI
 524 IF a$="L" OR a$="l" THEN LOAD /STR$ d
 526 IF a$="S" THEN LET d=d+s: SAVE /STR$ (d-s) LINE t*t:
GO TO INT PI
 528 IF a$="N" THEN INPUT " INPUT NEXT #";d: GO TO INT PI
 530 PRINT AT row+L,col-s-s;"  ": NEXT L: LET a$="
          ": RETURN
 600 CLS : LET pos=it-m+L: LET d$=c$(pos, TO t): LET a=COD
E c$(pos,t+s): IF NOT a THEN LOAD /d$
 610 IF a=s THEN LOAD /d$ DATA n()
 620 IF a=s+s THEN LOAD /d$ DATA n$()
 630 IF a=INT PI THEN LOAD /d$CODE
 640 IF a=VAL "4" THEN LOAD /d$ABS
 650 IF a=VAL "5" THEN LOAD /d$VAL
```

AUTHOR PROFILE

Mr. Robert D. Hartung, "Bob" to us, holds down two positions for Dekalb County, Illinois. Bob is Chaplain for the Sheriff Department, and is a member of the County Child Protection Team. Bob also serves as Pastor of Church of God Ministries in Fort Wayne, IN., and is the Founder of "Turning Point Services", a Ministerial outreach program for troubled youth. Bob previouly served with distinction in Pastorial work in England and Canada. His hobbies are; helping Youth development, fishing, antenna design, photography, and TS Computing. Many of us are familiar with his writings in SYNTAX, SYNC, TS Horizons, Syncware News, Time Designs, and CTM magazines. Bob Hartung is hereby appointed Chaplain and Spiritual advisor for UP-DATE and the subscribers. We need Bob in more ways than one.

# MASTERING THE MANAGEMENT OF CHARACTER ARRAYS

## THE MANAGEMENT PACKAGE

Wouldn't it be nice if we could CAT the disk directory, and with X-RAY eyes see the data that is in each program! But all we can see is the titles and the data length of each recording. Too bad that data can't be more visable! Probably the biggest problem of using computers is the need to plan ahead and organize indexes of files so that later we will be able to find the data that we need to use. Last issue, we started with "SDOS MAIL MERGE", a software that creates all kinds of Listings of Data, from "mailing lists, inventory lists, books and articles lists, to the daily milk producing records of a dairy herd". These lists are Saved for future use in "Character Arrays".

The lists produced by SDOS MAIL MERGE are just that, "lists". SDOS Mail Merge gives you a "Sorted, pure List", and provides options for print outs in several formats, LABELS being one of the most useful. Now if you work with many lists of things or people, you know that each list that you produce is likely to repeat some of the data that is contained in another list. If you print a lot of mailing labels using several mailing lists, you are likely sending out duplicate letters to the same persons. It takes a lot of time to cross reference several mailing lists, then more time to eliminate the duplications. So, we have two problems to solve, one- eliminate duplications, and two- "X-RAY the disk files to see their contents".

The Data Management Package consists of SDOS MAIL MERGE and the two Companion Programs, "PURITY" and "LIST LOOKER", both given in this issue. These two companion programs are short enough that the key-in project should be easy. But, as usual, the two programs will be included in the January Issue Diskette. "LIST LOOKER" is your "X-RAY Eyes", and "PURITY" is your program to PURIFY all of your various listings against each other to eliminate duplications within all of the lists. Each of these programs are independant, and designed to be used with the files produced by SDOS MAIL MERGE. Actually, they can be used to purify and process ANY Dimensioned Character Array files, whatever the source.

PURITY also has another purpose. Long Mail or Inventory Files cannot be LOADED to a Printing software because of the limited amount of FREE memory of the host program. MSCRIPT and SMART TEXT both gives us somewhere between 12K and 17K of FREE memory to be used for both TEXT and MAILING LISTS. A 96 Name File, having 7 lines of 31 Characters each, takes up 20732 bytes, and cannot be loaded to either program. BUT, if we break down such large mailing lists to, say, 12 names per list, and "process the groups automatically", then there is sufficient FREE available with almost any word processing software. Such a scheme would result in the handling of mail files of 2604 bytes each, well within the capacity of any Word Processing Software.

I wont speak for MSCRIPT, but Smart Text can take 40 such files of 12 name groups, and print them all without hesitating, as Form letters, or Labels, or as just print outs of the listings. So, "PURITY" is designed to enhance the capabilities of all Word Processing Softwares, by splitting up Large Listings so that they can be loaded to the operating programs. Now for the details.

## P U R I T Y

At right is a screen copy of the Purity Function Menu. This menu is line number 152 which is the starting line number of the program. As functions are accomplished, this menu is re-presented. The PURITY program may be SAVED and re-loaded with data aboard. Or, a selection of <6> will bring prompts to install a Data Diskette, then CAT the directory and prompt for input of "file name". When a file is loaded, the menu re-appears and processing can begin. Each loaded file is "purified" with itself, eliminating duplicates, and eliminating skips, by a selection of <7>. When finished, the file may be RE-SAVED by a selection of <1> or <3>.

YOUR CHOICES ARE:

1. RE-SAVE This FILE (SAVE OVER)
2. Break into 12 Name Files, SAVE to PURITY DISK
3. SAVE this Large File To PURITY DISK
4. COMPARE with all PURITY FILES for DUPLICATES
5. FLIP through File Names on Screen
6. LOAD another Large File
7. PURIFY this File

The selection of <2> will result in the large file being broken up into groups of 12 name files and each file SAVED with IN-SEQUENCE numeric file names, "mx1 through mx100". This begins the creation of your "Purity Disk". For example, if the large file being processed is a file of 96 name groups, your "purity disk" will have 8 group SAVES, and their titles will be "mx1 through mx8". The "purity files" will be used to "purify your other large group files". To process more large files, select <6> to load in the files one at a time, then <7>, then <4>. As that sequence finishes, the large file being processed has been compared with the 8 purity files, and all duplicates eliminated. The large file is now PURE, and may be SAVED as a large file.

Now that we have the second large file purified, it may also be saved to the purity disk by selecting <2>. Then, the "mx" files are again saved as "12 name groups", mx9 through mx16. There will be fewer saves of mx files if the elimination of duplicate names resulted in fewer name files in the large list. As more files are loaded in and processed, the purity file grows, and as it grows the number of comparisons with the next large file increases. For example, if we purify 10 files of 96 names, there could be as many as 80 mx files saved to the purity disk (no duplicates found), or 40 mx files if half of the names were duplicated within the files being processed.

## PROGRAMMING

This is a program of Nested Counters within Nested counters. The number of individual counts is astronomical. To purify just one 96 name file against itself requires 4608 comparison counts, plus 7 counts for each duplicate found, plus 4608 counts to eliminate interior skips, plus 7 counts for each skip eliminated. I haven't even attempted to sum up the number of counts required to compare all of the fields of a large file with the mx files in the purity disk (be my guest). One or two files are processed fairly fast, taking only about two minutes. As the purity files build, the time of processing increases. This is an ideal BASIC program to assemble with TIMECHINE to greatly increase its speed of operation.

PURITY will not recognize such repeats as, "John L. Sullivan, Mr. John L. Sullivan, and J. L. Sullivan". While using MAIL MERGE to create the files, you should be consistant with the format for entering names. Comparisons are made with the FIRST LINE of each name group, the "name line". If two Name Lines are the same, then the whole group of 7 lines "of the large file being processed" are deleted. The files in the Purity Disk are never changed. The menus are arranged so that prompts give the break needed to change diskettes. You can use PURITY in several different ways to process your listings. The pure files are then ready to print out with either SDOS MAIL Merge, LIST LOOKER, Smart Text, or any software that can handle character array listings.

## THE PURITY LLIST

```
  10 FOR n=1 TO 8: LOAD /"mx"+STR$ n DATA k$(): FOR
y=1 TO 12: PRINT n;" ";y: FOR g=1 TO 4: PRINT k$(y,g
): NEXT g: NEXT y: NEXT n
  20 STOP
  25 FOR n=1 TO 96: PRINT n'o$(n,1): NEXT n
  30 STOP
  55 CLS : PRINT AT 10,0;"Install your Mail File Dis
kette"'''"Touch ENTER": PAUSE 4e4: CLS
  60 LET i=0: LET b=1: LET x=0
  65 LET a=0: BORDER 0: PAPER 0: INK 5: CLS : CAT :
INPUT "Input Name of Array to LOAD";a$: LOAD /a$ DAT
A  o$()
  66 GO TO 152
  70 CLS : PRINT AT 10,0;"Check all 96 Names and DEL
ETE   DUPLICATES "; FLASH 1;"Line 80"
  80 FOR n=1 TO 96: IF n>96 THEN GO TO 135: REM **Fi
eld counter
  82 IF o$(n,1,1)=" " THEN NEXT n: GO TO 135
  90 FOR y=n+1 TO 96: REM **    comparison counter
 100 IF o$(n,1)=o$(y,1) THEN FOR m=1 TO 7: LET o$(y,
m)="": NEXT m: REM ** Delete dup-licate
 110 NEXT y
 120 NEXT n
 135 CLS : PRINT AT 10,0;"Eliminate skips in 96 Name
 array"; FLASH 1;"Line 140"
 140 FOR n=1 TO 96: IF n>96 THEN GO TO 152
 142 IF o$(n,1,1)<>" " THEN NEXT n: GO TO 152
 144 FOR y=n+1 TO 96: IF y>96 THEN NEXT n: GO TO 152
 146 IF o$(y,1,1)=" " THEN NEXT y: NEXT n: GO TO 152
 148 FOR g=1 TO 7: LET o$(n,g)=o$(y,g): NEXT g: FOR
g=1 TO 7: LET o$(y,g)="": NEXT g
 150 NEXT n
 151 CLS : PRINT AT 0,0;"The file "; FLASH 1;a$; FLA
SH 0;" is now Purified"  -Line 152 START-
→152 CLS : PRINT AT 2,0;"Your Choices are:"'''"<1> Re
-SAVE this file"'"    (SAVE OVER old file"'''"<2> Bre
ak into groups of 12          name files and SAVE to
the      Purity disk"'''"<3> SAVE this whole file to
the     Purity disk"'''"<4> Compare with ALL Purity
       files for Repeats."'''"<5> Flip Through File N
ames"'''"<6> LOAD another File"'''"<7> Purify this File"
'''"<8> Remove Skips Only": INPUT "Input choice ";z: G
O TO (z=8)*VAL "140"+(z=7)*70+(z=1)*160+(z=2)*178+(z
=3)*160+(z=4)*370+(z=5)*600+(z=6)*50+(z<1 OR z>8)*15
2
```

NOTE - Cont.

```
 160 CLS : PRINT AT 10,0;"Split the 96 name array in
to 8  12 name arrays and save each to disk "; FLASH
1;"Line 170"
 170 CLS : PRINT AT 10,0;"This is a PURE FILE SAVE.
 To  SAVE OVER EXISTING, keep same  disk."'"To SAV
E on PURITY DISK, Install PURITY DISK"''"ENTER When
READY": PAUSE 4e4: CLS
 172 INPUT "INPUT FILE NAME ";z$: PRINT '' FLASH 1;"
SAVING ";z$: SAVE /z$ DATA o$(): GO TO 152
 174 FOR n=1 TO 100: ON ERR GO TO 176: LOAD /"mx"+ST
R$ n DATA k$(): NEXT n
 176 ON ERR RESET : LET b=(n-1 AND n>1)+(1 AND n<2)
 178>LET a=0: REM * "a"counts the large fields in in
c- rements of 1.  "b" numbers the  disk files to sav
e.
 180 FOR c=1 TO 96 STEP 12: REM *get groups of 12 na
me fields
 190 IF c>96 THEN STOP : GO TO 340
 200 DIM k$(12,7,31)
 210 FOR y=1 TO 12: REM ** count to fill a 12 name f
ile array
 220 FOR g=1 TO 7: REM **count  to fill 7 lines
 230 IF o$(c+a,1,1)=" " THEN LET c=96: LET y=12: LET
 g=7: GO TO 340: REM ** If first chr is aspace then
quit, no more names  present
 240 LET k$(y,g)=o$(c+a,g): IF a=12 THEN LET a=0: GO
 TO 300
 250 NEXT g
 255 LET a=a+1: IF a>12 THEN NEXT c
 260 NEXT y
 300 CLS : LET z$=STR$ b: PRINT ;AT 15,0;"Saving ""m
x"; FLASH 1; INK 6;z$; INK 5; FLASH 0;""""";: SAVE /"
mx"+STR$ b DATA k$(): LET b=b+1: LET a=0: NEXT c
 310 NEXT c
 330 CLS : BEEP 1,20: PRINT AT 10,0;"File name ";a$;
" has been split"'"into ";b-1;" groups named ""mx";i
+1'"through mx";b-1;"""."'"ENTER to Continue.": PAU
SE 4e4: CLS : GO TO 350
 340 IF k$(1,1,1)<>" " THEN GO TO 300
 342 LET b=b-1: GO TO 330
 350 CLS : PRINT AT 10,10;"Your Choices:"'TAB 4;"1.
 Quit"'TAB 4;"2. Load Another File": INPUT z: IF z=
1 THEN PRINT ''TAB 4;"Install BASIC Disk, ENTER": PA
USE 4e4: CAT : INPUT "Input Title to LOAD";a$: LOAD
/a$
 360 CLS : PRINT AT 10,0;"Install Data file Diskette
"'"ENTER When READY": PAUSE 4e4: CLS : GO TO 65
 370 CLS : PRINT AT 10,0;"Now purifying against the
""mx""  files in Purity Disk."'' FLASH 1;"Install Pu
rity Disk and ENTER": PAUSE 4e4: FOR r=1 TO 200: ON
ERR GO TO 372: LOAD /"mx"+STR$ r DATA k$(): PRINT AT
 15,0;"Checking number of Files "; FLASH 1;"mx";r: N
EXT r
 372 CLS : ON ERR RESET : LET i=r-1: FOR r=1 TO i: P
RINT AT 10,0;"Processing file "; FLASH 1;"mx";r; FLA
SH 0;" to Delete  Repeats in "; FLASH 1;a$: LOAD /"
mx"+STR$ r DATA k$()
 374 FOR n=1 TO 12: IF n>12 THEN GO TO 388: REM **Gr
oup counter
 376 FOR y=1 TO 96: IF y>96 THEN GO TO 386
 378 IF o$(y,1,1)=" " THEN GO TO 384
 382 IF k$(n,1)=o$(y,1) THEN FOR m=1 TO 7: LET o$(y,
m)="": NEXT m: REM ** Delete dup-licate
 384 NEXT y
 386 NEXT n
 388 NEXT r
 396 LET b=i+1: GO TO 135
 500 FOR w=7 TO 12: LOAD /"mx"+STR$ w DATA k$(): FOR
 z=1 TO 12: LPRINT z;" ";k$(z,1): NEXT z: NEXT w
 550 STOP
 600 FOR w=1 TO 100: ON ERR GO TO 602: PRINT w;" ";o
$(w,1): NEXT w: STOP
 602 ON ERR RESET : PRINT FLASH 1: PRINT '"ENTER WHE
N READY": PAUSE 4e4: CLS : GO TO 152
```

## LIST LOOKER

SDOS MAIL LIST COMPANION No. 2

**Mail files, telephone listings, stock listings,** etc, all have one thing in common. After the second listing is tucked away in diskette, things degenerate into confusion. "Whos on first, what is in centerfield, and why is the pitcher". Remember that skit? "LIST LOOKER" is deceiving by its shortness. This utility will look at any LIST FILE that is saved as a Character Array of three dimensions. Example, you have a 100 name array containing "full name, Apartment number, Street address, City and State, plus other data, making each a 7 line field. The "dimension" of the array would be A$(100,7,31), "31" being the number of characters per line. Its kinda hard to see what is in a file just looking at the Disk Directory, isn't it?

**LIST LOOKER will X-RAY that file and show you** its bones. When List Looker loads in from disk, a prompt is on screen giving three choices: 1. LOAD an Array, 2. LPRINT Listing, 3. Screen Print Listing. When <1> is elected, you are prompted to "Install Data Disk", then a touch of ENTER gets the Disk Directory WITH A PROMPT to "ENTER ARRAY TITLE". If you are like me, your diskettes ae kinda disorganized. So if that directory doesn't contain the data that you are looking for, just BREAK, type GO TO 5, and you can install another diskette to get its directory and loading prompt.

**The data array automatically LOADS after input** of its title. Then <3> will get a screen print out of each of the groups, numbered from 1 to the end number. Example, if there are 50 names and

addresses, each will screen print under its field number so that you can identify the names that you may be interested in. The "3. LPRINT LISTING" lets you get a complete print out of all of the listing. This time however, you can select the "number of lines to print", "TAB", and "SPACES BETWEEN". So, what does this do for you? THINK! —— You got it! You can print print LABELS, by input of "4 Fields", 2 spaces, and 5 for TAB. Or, you can input 7 lines and get whole groups of data fields.

**Maybe you dont know how many names and how many** lines per name the array has. It really doesn't matter, as the program "sniffs for total number of fields", then tells you that, plus the number of lines, at line 130. The INPUTS then governs the print format. This is a excellent "companion utility" to use with mailing list data files generated by "SDOS MAIL MERGE". Also, it works with "Purity Files". The three programs go hand in glove to give you complete processing of all kinds of Array Data Files.

## THE PROGRAMMING

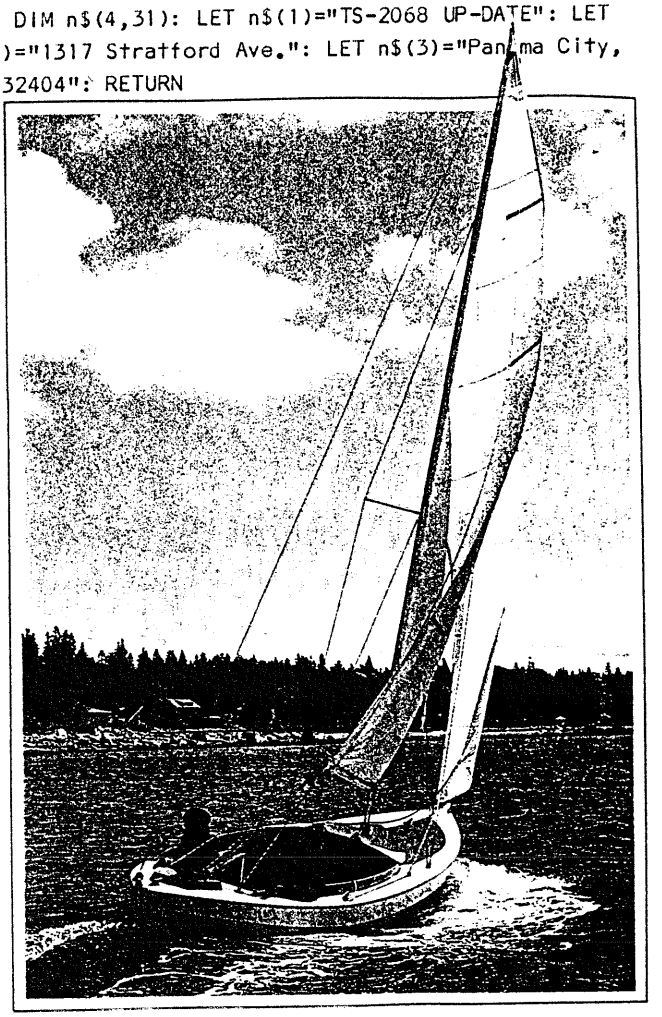
**There are several significant tips of** programming in LIST LOOKER. Lets take them line by line. Line 5 (the starting line), gives the <LET/P=0> to initialize the Oliger SAFE printer driver. Other Disk Operating systems will need to have a different program statement here to initialize a printer driver. Then the POKE 23324,10 is to tell the system to give a LINE FEED with each Carriage Return. If your printer does not need a LINE FEED, then Omit this. Then LINE 10 gives the way to CAT the Disk Directory, and put a prompt on the screen directory. After the prompt, the Array Title is LOADED by LOAD/A$DATA Z$().

LINE 90 is executed to LOOK AT the ARRAY and find its DIMENSIONs. The "Z Counter" uses a ON ERR to trip when the Number of Name fields have been counted. The "G Counter" does the same for the Number of Lines". Then Vars Z and D are used for counter limits in the LPRING routine at line 20 and the Screen Print routine at line 60. This program is short enough to really bite into and discover all of its tricks. Have fun doing it!

---

## LIST LOOKER, the LLIST

```
  5 LET /P=0: POKE 23324,10: BORDER 0: PAPER 0: INK
 6: CLS : GO SUB 138: PRINT AT 10,10;"OPTIONS"''TAB
 4;"1. LOAD ARRAY"''TAB 4;"2. LPRINT Listing"''TAB 4;
 "3. Screen Print Listing": INPUT "Input Your Choice?
 ";a: IF a<1 OR a>3 THEN GO TO 5
  7 GO TO (a=1)*10+(a>1)*90
 10 CLS : PRINT AT 10,4;"Install Data Disk, ENTER":
 PAUSE 4E4: CAT : INPUT "Input CHR ARRY TITLE";A$: L
OAD /A$ DATA Z$(): GO TO 5
 20 CLS : FOR N=Z1 TO Z2: ON ERR GO TO 40: IF Z$(N,
1)(1)=" " THEN STOP
 25 IF C>4 THEN LPRINT TAB TB;n
 30 FOR Y=1 TO C: LPRINT TAB 10;Z$(N,Y): NEXT Y: FO
R M=1 TO SX: LPRINT : NEXT m
 35 IF tw=1 THEN FOR Y=1 TO C: LPRINT TAB 10;N$(Y):
 NEXT Y: FOR M=1 TO SX: LPRINT : NEXT m
 38 NEXT n
 40 ON ERR RESET : GO TO 5
 60 CLS : FOR N=1 TO Z: ON ERR GO TO 80: IF Z$(N,1)
=" " THEN STOP
 70 PRINT n: FOR Y=1 TO D: PRINT Z$(N,Y): NEXT Y: P
RINT : NEXT N
 80 ON ERR RESET : PRINT '''ENTER WHEN READY": PAUS
E 4E4: GO TO 5
 90 FOR Z=1 TO 200: ON ERR GO TO 100: LET C=LEN Z$(
Z,1): IF Z$(Z,1)(1)=" " THEN STOP
 95 NEXT Z
 100 ON ERR RESET : LET Z=Z-1: FOR G=1 TO 100: ON ER
R GO TO 110: LET C=LEN Z$(1,G): NEXT g
 110 ON ERR RESET : LET D=G-1
 120 IF a=3 THEN GO TO 60
 130 CLS : PRINT AT 10,0;"The Array has ";D;" Fields
."''AT 12,0;"Input Nbr of Fields to Print. "''"( 4 f
or Labels )": INPUT C: CLS : INPUT "Input Start Nbr.
 ";Z1: INPUT "Input END Nbr. ";Z2: INPUT "Input TAB:
 ";TB: INPUT "Input Spaces Between: ";SX
 132 CLS : IF c<5 THEN INPUT "Twin Labels? <1> Yes-
<2> No ";TW
 136 CLS : GO TO 20
 138 DIM n$(4,31): LET n$(1)="TS-2068 UP-DATE": LET
 n$(2)="1317 Stratford Ave.": LET n$(3)="Panama City,
 FL 32404": RETURN
```



## PRINTER PROGRAMMING

- OR -

## Adapting Software to Printers

**The article about printer control in the** October issue brought forth enough response to indicate that a more comprehensive reference should be attempted. Apparantly many users are having problems in this area. One nice Lady wrote her thanks, and several Gents wanted more information. But John Oliger wrote and pointed out a couple of mistakes. These will be corrected in the text of the following.

Why doesn't software writers include enough programming to make the software print "right out of the box" with ALL TYPES of printers? Well, that would be nice, but would likely require about 38K of programming, and our TS-2068 has only 38K of FREE memory to start with. So, usually a software is designed to print with a type of printer that is "compatable with" several brands, and instructions given to make program line changes to adapt to other printers. There are several DOT MATRIX printer brands that use the "EPSON STANDARD", which usually means that the printer maker copied the Epson "Control Codes". Two "Standards" widely used with Daisy Wheel printers are "QUEME Compatable" and "DIABLO Compatable".

In order for a computer to communicate with a printer, both devices must understand a common language. That common language is called ASCII (American Standard Code for Informaton Interchange). Page number 239 of the TS-2068 User manual gives the ASCII Codes and calls them "The Character Set". Actually there are more codes in ASCII than the TS-2068 uses, and a few of the codes in the TS-2068 CHR SET are not standard ASCII, but for printer control, the codes in the User Manual will suffice.

So, we have the "common language", which is ASCII. The computer understands it and so does the printer. Now there are two jobs that the printer must do. One is to PRINT CHARACTERS, and the other is to SWITCH its own modes of printing. To PRINT characters the printer must be able to receive and to respond rapidly to "streams of characters" sent by the computer. The charcters are processed by a "printer driver", a machine language software that is supplied by the manufacturer of the "PRINTER INTERFACE". Since the TS-2068 contains only a printer driver for the little TS-2040 printer, ALL interfaces for large printers are supplied by our Cottage Industries. Some of these are "Serial Interfaces", but most are CENTRONICS PARALLEL interfaces.

We will delay the discussion of Serial (RS-232) interfaces until the next Issue of UP-DATE. For this discussion we will deal only with CENTRONICS PARALLEL interfaces, and only with the OLIGER and AERCO Interfaces. These two "CPI" devices have become the dominant ones for the TS-2068, the most simple to use, and they use a minimal amount of computer memory for their printer driver code. In fact, the Disk Drive controller hardware contains the printer driver code in its EPROM, thus using no computer memory at all. These interfaces provide the electronics circuitry to process character streams and coded directions to a printer and to RECEIVE the "interrupt signals" from the printer.

The software "driver code" is an extension of the TS-2068 ROM, which lacks the "built in instructions" necessary to send data and commands to the printer. While the Oliger and Aerco CPI interfaces are different in circuitry, the driver codes supplied with each can be used with the other interface. Now lets get to the two functions of these CPI interfaces. One function, and the most complicated, is the processing of Character streams to the printer. Actually this is the most simple to use. <LPRINT> does it! You dont see the many complicated functions that take place, and you dont have to worry about it.

The other CPI function is to "Process Control Codes to the Printer", to make it do such things as Change from Elite Style to Pica Style, Roll up a Page, or the other mechanical functions that the printer does. The Interface uses "OUT PORT 127" as the communications path to the printer, and the path back from the printer for "interrupts". So, the "ASCII COMMAND" that a printer needs to do a desired function is sent "OUT through PORT 127". A typical direct command to click up a line space is <OUT 127,10>. "10 decimal" is the industry standard ASCII CODE for printers to perform a LINE FEED. There are 32 "single character codes", 0 through 31, in the TS-2068 Character set that can be "sent out" in this manner.

Actually there are 255 character codes that can be sent out to the printer, but only 32 can be sent out "without a character being printed". All of the other 223 codes will cause something to be printed. Example, <OUT 127,65> will result in the character "A" being printed. This is because ASCII CODE 65 is assigned to the character "A", and the interface processes data characters to be printed. Some printers use as many as 80 ASCII CODES to perform internal changes. For example, a Diablo Daisy Wheel printer uses ASCII CODE 79 to "SET BOLD PRINT". BUT <OUT 127,79> PRINTS a "O"! The solution to "sending printable ASCII CODES" to the printer is to first send the "ESC COMMAND".

The "ESC" code is "27". So, using the above example, <OUT 127,27> then <OUT 127,65> will command

the Diablo printer to SET BOLD PRINT. The ESC code (27) tells the printer to "Expect a CONTROL CODE NEXT". Printer manuals vary as to how their CONTROL CODES are given. Most manuals have a table of control codes with numbers given in both HEXADECIMAL and DECIMAL. A typical such expression would be given in brackets as (1B,4D)H (27,77)D. The first group is given in Hex and the second group in Decimal. In this case, our TS-2068 command would be <OUT 127,27> <OUT 127,77>. Another way that the SAME command group could be presented is <ESC M>. "ESC=27 and the Character Code of "M" is 77".

Still another way given in some printer manuals is <LPRINT CHR$(27)+M>, which isn't the correct way of sending such codes with the TS-2068, but can be intrepeted as OUT 127,27:OUT 127,77. So, with all of these different ways of saying the same thing, it's no wonder that printer manuals are confusing! Incidentally, that ONE command group is used by Epson printers to SET ELITE PRINT MODE. Command codes to perform a single function may be as many as six codes chained together. Example: (27,120,1)D (155,120,1)D. The "D" for Decimal may or may not be present. That command group SETS HI QUALITY MODE for Epson printers. The command for the Aerco and Oliger CPI would be <OUT 127,27: OUT 127,120: OUT 127,1: OUT 127,155: OUT 127,120: OUT 127,1>, quite a long group of OUTs to do just one switching function!

Now its gonna get longer, because that group of six OUT commands execute in about 100 milliseconds, and the printer requires much more time than that to respond to SWITCHING commands. For Software programming lines to Command the printer, we must have a "CHECK OF THE PRINTER STATUS" routine TO SEE IF THE PRINTER IS BUSY before sending a control code. When the printer is busy it places a interrupt signal on IN PORT 127. If the printer is busy then the software must wait until the printer is READY before sending the control code. The correct way to do this with the Oliger CPI is to use the loop given in the interface manual, which is <100 IF INKEY$ #3="B" THEN GO TO 100> <102 RETURN>. Then a control code group such as (27,45) would be programmed in a line as: <50 GO SUB 100: OUT 127,27: GO SUB 100: OUT 127,45>. The line 100 will loop itself until the printer is READY, then the line 100 IF condition will be FALSE and the RETURN will allow the next OUT command to execute.

The above "INKEY$ #3" polls the IN PORT 127 for the bits used by printers to signal its status to the computer. But, this procedure requires a interface driver that processes the INKEY #3 syntax. If you use another type of interface you should check its manual for a PRINTER STATUS CHECK routine. The use of INKEY$ #3 returns a "Improper I/O device" report when used with some other interfaces. If you get such a report code you can turn OFF the printer and type <PRINT IN 127>. Then use that number in the status check routine. My system produces 253 when the printer is BUSY. Mr. Oliger tells me that all bits of IN PORT 127 are not not controlled the same with all models of the TS-2068. His words are quoted: "A program loop such as the example given (100 IF IN 127=253 THEN GO TO 100) (102 RETURN) should not be used and this is not how the printer interface manual instructs this to be done. This is the kind of thing that works with one computer but not on another, because all of the bits on IN PORT 127d are not used, and thus are floating. The function INKEY$ #3 should be used for this purpose as detailed on page 6 of the Oliger Interface manual."

Pardon me for digressing into the complicated. This is supposed to be a SIMPLE treatsie about how to clear the fog in printer manuals and CONTROL your printer with program lines. We will get back to that. Other interfaces such as TASMAN and A&J use LPRINT CHR$ instead of OUT 127. Many printer manuals express their EXAMPLE commands in this manner, LPRINT CHR$ 27, as the ESC command, instead of OUT 127,27. Thats alright if you know how to intrepet such red herrings. The Oliger and Aerco interfaces just wont work with LPRINT CHR$, and must have OUT 127,number.

To sum up: When ESC is given, it means OUT 127,27. When a letter character is given as a command code, look up the ASCII code for the letter on page 239 of the TS-2068 User Manual and use the CODE number as the command. When constructing program lines to Command the Printer, each OUT 127 should be preceeded with <GO SUB> to the "STATUS CHECK" routine, which for the Oliger and Aerco CPI is <100 IF INKEY$ #3="B" THEN GO TO 100> then a following line <102 RETURN>. The line numbers can be of your choosing. There is no limit as to the number of "chained commands" that can be in one program line. Many printers require as many as six chained commands to perform one function change of the printer.

You cannot use HEXIDECIMAL numbers in your OUT 127 command. HEX numbers must be converted to decimal, if not given in the printer manual. Page 239 of the TS-2068 User manual gives the codes in both Hex and Decimal. Printer control sequences given in a printer manual, such as: (ESC * 0 2) must be intrepeted by looking up the codes for "*" (42), for "0" (79), for "2", (50). We know that ESC is 27. So, the chained commands would be entered in a program line as follows: <500 GO SUB 100: OUT 127,42: GO SUB 100: OUT 127,79: GO SUB 100: OUT 127,50: RETURN >. GO SUB 100 would be to the INKEY$ #3 routine to check the printer status.

Your printer and the software can both be OK,

but all you get is partial lines of print that lap over onto the next line, because of improper printer switch settings. Most softwares completely control the printer and require that no MARGINS be set at the printer, the auto PAGE ADVANCE be turned OFF, the printer LINE JUSTIFICATION be turned OFF, the PORPORTIONAL PRINT PRINT SPACING be turned OFF, and for letter size pages, the LINES PER PAGE setting at the printer should be 60 lines per page. Then the software and the printer wont be fighting each other to control margins, character spacing, and page length. Let the software do the controlling, and let the printer be DUMB, just responding to "Simon Sez" commands given by the software.

The printer switch that sets a LINE FEED with each CARRIAGE RETURN should be ON. Then a line feed will occur each time a line is printed. A Carriage Return command is sent by the sotware each time a line has printed, or the "partial last line" of a paragraph. Now this just about sums up "everything that anyone will ever need to know about Commanding printers with the Oliger and Aerco CP Interfaces". Next issue we will discuss a RS-232 Serial Interface. In the meantime perhaps someone would like to send in a treatsie about "printer commanding with the TASMAN CPI. Permission is hereby given for TSUG Clubs to make re-prints of this article or excerpts thereof.

## PRACTICAL USES OF SCREEN FILES

- AND -

## DISK DEPENDANT PROGRAMMING

**If I had to complain about something I would** gripe about the 6912 bytes of disk space that is required to SAVE a Screen File. A whole screen, when full, contains 22 X 32= 704 characters. Of course there is much more to a displayed screen than just the places reserved for characters to be printed, but 6912 bytes? A program line that constructs a screen takes up quite a lot of our precious FREE Memory. The more a software can do, the more MENUS and screen displays are needed. Usually a well designed screen will cost about 600-1000 bytes of FREE MEMORY. A software that has a lot of functions may need as many as a dozen such Menu and Information Screens. The FREE Memory used for a dozen such screens would be between 6 and 12K bytes. Look at FREE memory in another way. A full page of printed data is usually about 3000 Characters. So, 12 screens within the software programming costs us the memory capacity of about 2 to 4 full pages of data.

Is it economical to spend 6912 bytes, times 12=82944 bytes of disk space to save 12 screens and "boot the screens in when needed in a program"? You bet it is!! And, the SCREEN$ LOAD and SAVE is one of the most useful functions of a Disk Drive. The programming (SAFE) <1000 LOAD /"1"SCREEN$> (AERCO) <CAT "1.SCR",> is all that is needed to bring aboard a screen menu that would take up 1000 bytes of FREE Memory. 12 such lines of programming costs only 132 bytes of FREE memory. We have a million K of disk space to waste, but every byte of FREE memory is like the last lick of a ice cream cone. How sweet it is! And who cares about a 6912 byte chunk of DISK Space? I rememember once testing the practicality of booting screens from Cassette. While the screen was loading in the grass on the lawn grew enough that it needed mowing! And I watched as a Cardinal built a nest, layed her eggs and the young birds learned to fly! All while that screen was loading from cassette. Well, maybe thats stretching it a bit. Anyway, <LOAD /"1"SCREEN$> pops a menu on screen in 2 and one half seconds flat! Thats FAST and efficient!

The "doing it" is too easy to talk about. You just program your screen in the normal way with PRINT AT statements in a line of programming. But now you can let it all hang out and create a real beautiful screen with color changes, full words instead of abbreviations, and put nice borders around it. When finished and the screen is really beautiful, just BREAK, and type <SAVE/"1"SCREEN$>. Next DELETE the program line, and re-enter the same line with 1000 LOAD /"1"SCREEN$. Do this as many times as you have Menus and Display screens in the software. When finished, your program will have more meaningful menus and you'll have saved enough FREE memory to add many more functions to the program.

Now what we have just done is to create a "Disk Dependant" software. It wont work without the disk drive. You are sacrificing a good bit of disk space to conserve Computer FREE Memory. Later we will use disk space to store "Segments of the Software itself", and MERGE these segments to the main program in memory as the program routines are needed. For example, suppose that you have a software that has a Data Base Manager, a Word Processor, a Mail Merge. You have used a lot of programming for these three comprehensive functions, and have thus sacrificed FREE memory for DATA STORAGE. While you are using the Data Base Manager, you really do not need the Word Processor nor the Mail Merge to be on board.

So, when the SAFE Merge function is implemented, we can have our cake and eat it too. Software programs can be segmented so that only the major function that we need will be in memory. The other major functions will be on stand-by in disk.

## TIMEX CLUBS OF NORTH AMERICA

Following is a list of Timex Clubs of North America. A few may be missing and will be published as information is received. Some of these clubs publish excellent newsletters that are worthy of subscribing to. For those clubs that may be interested, Up-Date is willing to publish a "Club Edition" that consolodates inputs from participating clubs, if sufficient interest is forthcoming. Suggestions will be welcomed.

S.E. Michigan Group
Box 614
Warren, MI 48090

G.U.T.S. Group
6625 Clifford Drive
Cupertino, CA 95014

Kansas Area Users
4557 Cherry
Wichita, KS 67217

TAS BAM User Group
PO Box 48961
St Petersburg, FL 33743
Newsletter

TSUG of Philadelphia
PO Box 53490
Philadelphia, PA 19105

Westmoreland TSUG
PO Box 3051
Greensburg, PA 15601

Vashon Isl User Group
PO Box 199
Vashon, WA 98070

Mile High
914 S. Victor Way
Aurora, CO 80012

Hampton Roads
112 Kohler Crescent
Newport News, VA 23603
Newsletter

Sincus News
1229 Rhodes Road
Johnson City, NY 13790
Newsletter

L.I.S.T.
10 Idle Way Drive
Center Point, NY 11721
Newsletter

Toronto Canada Group
PO Box 7274, Stn A
Toronto, Canada M5W 1X9
Newsletter

San Francisco Area
PO Box 1312
Pacifica, CA 94044
Newsletter

North Carolina Group
206 James St.
Carrboro, NC 27510

Milwaukee User Group
5052 N. 91st St.
Milwaukee, WI 53225
Newsletter

Sun Lake Group
1200 Lake Drive
Grand Island, FL 32735

Vancouver Group
2006 High View Place
Port Moody, BC V3H 1N5
Canada N.L.

CCATS
1419 1/2 7th St.
Oregon City, OR 97045
Newsletter

NE Florida TSUG
6634 Oriole Ave.
Jacksonville, FL 33216
Newsletter

TSUG of Orlando
808 E. Anderson St.
Orlando, FL 32801

S.T.U.N.
Rt. 1, Box 21
Glade Hill, VA 24092



## TS-2068 SUPPLIERS

Following is a listing of Suppliers of Software and Hardware for the TS-2068. There must be more, and Up-Date will publish additions to this listing as information is obtained.

A+ Computer Response
220 Centre St.
Sullivan, NH 03445

APR Software
1606 Pennsylvania Ave. #204
Miami Beach, FL 33139

Aerco
Box 18093
Austin, TX 78760-8093

BG Micro
PO Box 280298
Dallas, TX 75228

Basically Programming
2528 W. Olive
Fullerton, CA 92633

Beaver Computer Products
756 Fleming Ave.
Winnipeg, Manitoba R1K 1V5
Canada

Bob Crecco
1627 Dewey St.
New Albany, IN 47130

Byte Back
Rt. 4, Box 54
Leesville, SC 29078

Byte Power
1748 Meadowview Ave.
Pickering, Ontario L1V 3G8
Canada

Chai-Chi Chao
73 Sullivan Drive
Morago,CA 94556-1209

Curry Computers
PO Box 5607
Glendale, AZ 85312-5607

Quantum Computing
PO Box 1280
Dover, NJ 07801

RMG Enterprises
1419 1/2 7th St.
Oregon City, FL 97045

S&K Enterprises
2107 SE 155th
Portland, OR 97233

Sharp's Inc.
RT 10, Box 459
Mechanicsville, VA 23111

Sinclink Co.
6675 Clifford Drive
Cupertino, CA 95014

Cuyahoga Valley Software
615 School Ave.
Cuyahoga Falls, OH 44221

Dan Elliott
RR-1, Box 117
Cabool, MO 65689

Foote Software
PO Box 14655
Gainesville, FL 32604

G. Russell Electronics
RD-1 Box 539
Centre Hall, PA 16828

Grey & Clifford Computer
PO Box 2186
Inglewood, CA 90305

Gulf Micro
1317 Stratford Ave.
Panama City, FL 32404

Heath Computer Services
950 E. 52 South
Greentown, IN 46936

Herb Bowers,Sr.
2588 Woodshire Cir
Chesapeake, WA 23323

JRC Software
PO Box 448
Scottsburg, IN 47170

Jack Dohany
390 Rutheford Ave
Redwood City, CA 94061

Jameco Electronics
1355 Shoreway Rd.
Belmont, CA 94002

Sincpac Software
5206-1 Cedarbend Dr.
Fort Myers, FL 33907-7514

Sirus Ware
6 Turning Mill Rd.
Lexington, MA 02173

Sunset Electronics
2254 Taraval St.
San Francisco, CA 94116

The Olger Co.
11601 Whidbey Drive
Cumberland, IN 46229

Thomas B. Woods
PO Box 64
Jefferson, NH 03583

John Mathewson
1852 Appleford St.
Gloucester, Ontario K1J 6T4

John McMichael
1710 Palmer Drive
Laramie, WY 82070

Knighted Computers
707 Highland St.
Fulton, NY 13069

Larken Electronics
RR-2
Naven, Ontario K4B 1H9
Canada

Mark L. Fendrick
PO Box 2392
Secaucus, NJ 07004-0992

Markel Enterprises
PO Box 2392
Secaucus, NJ 07094-0992

Meta Media
726 W. 17th
Vancouver, BC V5Z 1T9
Canada

Mountaineer Software
749 Hill St. #6
Parkersburg, WV 26104

Novelsoft
35 Candle Liteway
Willowdale, Ontario M2R 3J5
Canada

Paul Bingham
Box 2034
Mesa, AZ 85204

Peripherals Direct LTD
PO Box 4301
Northbrook, IL 60065

Variety Sales
325 W. Jersey St. #2D
Elizabeth, NJ 07202

WMJ Data Systems
4 Butterfly Drive
Hauppauge, NY 11788

Wymil Corp.
Box 5904
Bellingham, WA 98227-5904

Zebra Systems
78-08 Jamaica Ave.
Woodhaven, NY 11421

## The TS-2068 DISK DRIVE SYSTEMS

### Combinations to Achieve Increased Capabilities

| OWNER'S Basic Disk System | Plus | Additional Device | EQUALS Additional Capabilities |
|---|---|---|---|
| Oliger SAFE Disk System | Plus | SPDOS Disk Software | Oliger SAFE DOS + RAMEX DOS Formats<br>Two Simultaneous Disk Operating Systems |
| Oliger SAFE Disk System | Plus | LARKEN SKDOS Cartridge | Oliger SAFE DOS + Larken DOS Formats<br>Two separate Disk Operating Systems |
| Larken Disk System | Plus | Oliger SAFE Board A | Larken DOS + Oliger DOS Formats<br>Two Separate Disk Operating Systems |
| Larken Disk System | Plus | Aerco FD-68 Disk System | Larken DOS + Aerco DOS<br>Two Separate Disk Operating Systems |
| Aerco FD-68 Disk System | Plus | Aerco on-board Extra Memory | Two Disk Operating Systems (RP/M + Aerco DOS)<br>Extra Memory banks for programming |
| Aerco FD-68 Disk System | Plus | Larken SKDOS Cartridge | Three Disk Operating Systems<br>(above, plus Separate Larken DOS)<br>Plus RAM DISK if Extra Memory present on FD-68 |
| RAMEX Disk System | Not aware of possible Combinations | | RAMEX SPDOS Operating System<br>RAMEX Disk System is no longer supported by a manufacturer. |
| TOS (Zebra) Disk System | Not aware of Possible Combinations | | TOS Disk Operating System<br>TOS Disk is no longer supported by a manufacturer. |

All of the above Disk Systems are compatable with Cassette LOAD and SAVE operations. Cassette is the "Transfer Medium" to SAVE programs that are formatted by any disk system; then re-load to the TS-2068 that has a different disk controller installed; then EDIT the program lines and MODIFY the DOS SYNTAX; then SAVE to Disk with the DOS that is in use. The Computer must be turned OFF to transfer from one Disk System to another. There are two exceptions to this. The Oliger SAFE and SPDOS software combination provides two DOS systems Co-resident. The Aerco FD-68 Disk System has Aerco DOS and RP/M Co-resident.

# SPDOS for North America

Ian Robertson

SPDOS was originally marketed in the UK by WATFORD ELECTRONICS as a disk interface and operating system. KEMPSTON (the joystick interface people) later marketed a version which used a smaller amount of RAM, approx. 700 bytes (Watford used about 8k). A varient of the SPDOS interface was marketed in North America by RAMEX INTERNATIONAL LTD. and known as the MILLENNIA K. This is the system that in 1985 introduced me to the wonders of a disk system (later to be followed by the OLIGER, LARKEN, BETA PLUS, KEMPSTON and finally the CUMANA). The SPDOS presently being marketed by the CUYAHOGA VALLEY SOFTWARE WORKS is a modification of SPDOS for the 2068 running on Oliger hardware. It is sold under license from ABBEYDALE DESIGNERS LTD., who wrote SPDOS. As a matter of fact, the amount paid (by CVSW) to Abbeydale is a large portion of the selling price. This group has not written this software to make money, but rather to help extend the life of our computers. The DOS operating system is available on two 5.25 inch disks, for either 40 track DSDD or 80 track DSQD drives.

The first disk is an Oliger disk with a "boot" that replaces some of the hardware of the Millennia K interface. The second disk is the "SPDOS System" disk.

**HOW IT WORKS:**

SPDOS is a RAM based DOS. Because of that it uses several K of RAM and can be run concurrently with the Oliger DOS (which uses no RAM) in operation. Certain SPDOS commands are read from disk and the system disk should be in Drive #1 (Oliger drive #0). Since SPDOS is also compatible with the Millennia K system, it works only in the 2068 mode. Disks which were written on the Ramex system can be read and written to. However these disks should be considered as Data disks (disks without the operating system on them).

The DOS occupies memory from 58500 to roughly 63500. It supports sequential files and program overlays. The overlays are program segments with line numbers that are loaded in from disk replacing existing ones, making memory use more efficient and adding structure to the program.

The DOS is as fast as the Oliger in Loading and Saving. SPDOS allows 144 directory entries per disk and uses a minimum of 1K for a file. A formatted 80 track drive will have aproximately 785K of space free to write to. Each SPDOS track is organized into ten 512 byte sectors. SPDOS supports commands like MOVE, ERASE, and will autorun programs named "AUTO" when the system is Booted up. It has the ability to perfom a selective CAT. All SPDOS commands must be prefixed by the command "PRINT # 4:". Other commands are used in the normal way. Those of you using the latest version of the LARKEN disk interface will recognize this syntax.

Several utility programs are included: SYSCOPY which will back up SPDOS, COPY which will assist in selectively copying disks, ERASE which does the same for erasing files, MINIDOS a truncated DOS located at 49644 to 53600, and MAIL/BAS a demonstration data base.

The SPDOS system should be considered an enhancement to the Oliger Safe Disk Operating System rather than a replacement. It will be of the most use to people who write a lot of their own programs, and those who have a Millennia K system or know users who do as it adds a certain amount of compatibility to the 2068. For example, Ramex marketed 2068 disk versions of MASTERFILE, TASWORD II and OMNICALC. This software should allow these programs to be run on the OLIGER disk system, but I am not sure about the ability to print, using an 80 column printer. The RAMEX system is compatible with the TASMAN C centronics interface, while the OLIGER is compatible with the OLIGER/AERCO centronics interface.

**WHERE TO BUY IT:**

The cost is $24.95 + $1.50 for postage and is available only from the **CUYAHOGA VALLEY SOFTWARE WORKS, 615 School Ave., Cuyahoga Falls, Ohio 44221.**

2068: Larry Kenny of LARKEN ELECTRONICS has done it again! He has produced a disk interface cartridge, that plugs into the cartridge port, which turns your RAMEX interface into a LARKEN interface. And it works! It comes with the DOS on a 2764 Eprom and the FORMAT software on tape. The FORMAT programme is loaded into the computer and after configuring it to suit your (up to 4) drive system, it loads itself to disk. I have tried it on both SSDD 3" and 5 1/4" DSQD drives, without a problem. The LARKEN extended basic commands also work on my RAMEX hybrid. Now for the interesting part - the cartridge does not have to be removed from the cartridge port when the RAMEX DOS is used AND by switching off the RAMEX DOS eprom, the RAMEX does not have to be modified in any way.

Profile of the Author

Mr. Ian Robertson is the Liaison Officer for the Toronto Timex Sinclair User Group and a regular writer for their fine publication "Sinc Link". Ian has every computer that bears the Sinclair name and is current on all. Despite pressures of business and family Ian finds time to contribute his knowledge. We look forward to his future article about SPDOS, The QL, and other items of interest.

LARKEN DSK400 DISK OPERATING SYSTEM
A review
by G.F. Chambers

Larry Kenny has come out with what in effect is his third generation disk interface system. Possibly one should say his 2nd generation, since it could be said that the first generation had two phases.

Lets review the background:

The first system was based on an EPROM DOS (which was bank-switched into the 63488 area of memory), plus several disk DOS's. The second phase of this first system provided a cartridge DOS which was bank-switched into the ROM area of the computer (when disk functions were required), and eliminated the need for the EPROM- and disk-based DOS's. The cartridge also provided several supplementary features not available on the intial system. This system was confined to the use of two DSDD drives, and formatted disks to a 160K capacity and maximum of 50 files.

::::::::::::::::::::

The new system, the DSK400, subject of this review, consists of a cartridge board very similar to the original, plus a new interface board. The new interface board measures about 6 inches by 2 inches, and mounts horizontally behind the computer. A ribbon cable for the drive plugs into a connector at the right-hand end of the board. At the other end of the board are an NMI button and a 9-pin joystick connector. Possibly the unit could fit into an empty Memotech or Gladstone ZX81 64K RAMpack case. The board is equipped with a gold-plated female connector with a tinned male through-connector, for adding other peripherals. Four of the nine chips on the interface board are socketed. The quality of the board is very high.

The system can handle a variety of drives, up to a maximum of four. It is said to be capable of handling 3", 3½", and 5½" (not 8")single or double sided drives in single-, double-, or quad density. I have only tried it on a SA455-type 5¼" drive in the DSDD mode. In this mode disks can be formatted to hold just over 400K.

The sytem uses all the familiar Sinclair commands, including CAT, ERASE, LOAD, SAVE, MERGE, OPEN #, CLOSE #, FORMAT ,also GOTO and PRINT. All disk commands are prefaced with PRINT USR 100: or PRINT #4:

The system is compatible with the 2068 and the Spectrum mode. A Spectrum EPROM can be mounted on the cartridge and switched in with a short BASIC command from the 2068 mode. It has an AUTOSTART mode, by which you can boot a program into memory by holding the ENTER key operated while powering up the computer. Only one program per disk can be handled this way. I usually make this a MENU program to select the program that I want from the disk. This AUTOSTART program can be used to automatically boot the computer into the SPECTRUM mode from a cold start, and produce the menu.

With an appropriate EPROM DOS the cartridge is also suitable for use with several other disk systems, such as the AERCO, RAMEX, OLIGER; improving the performance of these sytems significantly. It can also be used with the OS-64 and LROS cartridge chips, by mounting them on the cartridge board.

A program to format disks is supplied on disk, along with a program to facilitate copying disks using two drives. The FORMAT program will format disks as either double or single sided, and to 40 or 80 track density.

Programs can be saved (and loaded) with all the command forms used with tape, as for example:: SAVE "progrm.B1", SAVE "progrm.B1" LINE 100, SAVE "prog.C1" CODE start, length, SAVE "progrm.C1" SCREEN$, SAVE "progr.A1" DATA (), SAVE "progr.A$" DATA $().

Programs can be removed from the disk using the ERASE command. Tracks so erased become available for subsequently saved programs.

The NMI (snapshot) button means that you can capture to disk those unbreakeable programs on tape. Simply load your program, press the NMI button and the program will be transferred to disk. It captures the complete memory from address 22490 upwards, and uses 9 tracks of a disk. Loading one of these programs from disk takes just under 7 seconds. A disk can hold 8 of these programs, and have 7 tracks spare for menu, etc.

Programs such as Tasword, Timachine, Pro/file, MtermII are readily modified to run on the DSK400 system. Mscript with Jack Dohony's mods can be similarly adapted.

The joystick port accepts the 9-pin connector standard on joysticks, and works with those programs which are provided with the Kempston joystick option.

The DUMP.B1 and COPY2D.B1 programs supplied on disk with the system use a modified disk DOS which can be used from BASIC programs to control the drive cperation. This means that disk utilities can be written in BASIC to perform a variety of tasks, similar to what has been done with the earlier LARKEN systems. Typical of this are three programs which I have modified for the DSK400. They are a RENAME program ,which will rename programs on disk; DOCTOR with which one can inspect and repair errors on disk tracks; and INDEXER which maintains a file of programs on a disk collection.

The DSK400 includes a number of Extended Basic commands. These permit the placement onscreen of up to three "windows" of any size, colour, and position; and writing to them in proportionally spaced characters. Also there is a set of 7 graphics patterns which can be called up as desired to fill areas of screen. A CIRCLE command will fill in selected areas of the

screen with a selected pattern. INK and PAPER commands will instantly change paper/ink colours. POKE will poke numbers higher than 256 into two addresses automatically.

There is buint-in printer driver software by which a large printer may be used, making use of the LLIST and LPRINT commands. This feature is accessed by the command: PRINT #4: OPEN #3,"lp". A drawback to this feature is that I found it impossible to control line length; if there is a control, I could not find it. As a consequence I found my printer putting out lines with 108 characters in them.

Final Conclusions: An easy-to-use system which will greatly enhance the use of the TS2068. Highly recommended.

Available from: Larken Electronics, R.R.#1, Navan, Ontario, Canada K4B 1H9 Price for interface board, cartridge board and drive connector cable: Can $145. (You will also need to purchase the disk drives and a power supply for them)

## The LKDOS Cartridge

In addition to the Larken Disk system that is reviewed above, there is another Larken development that deserves mention. It is the LKDOS CARTRIDGE, which contains the Larken DOS and operates with two other Disk Controllers for the TS-2068. The LKDOS Cartridge plugs into the Cartridge Compartment of the TS-2068. When used with OLIGER SAFE Disk system, Board B of the Oliger system is removed. Then the LKDOS cartridge utilizes the Oliger disk controller board as the controller. The result is a LARKEN DOS System that formats and reads disks in the Larken format, and has all of the Extensions to Basic as discribed in the above review.

The LKDOS Cartridge can also be used with the AERCO FD-68 Disk Controller. The LKDOS uses a Separate EPROM for use with Oliger or Aerco disk controllers. The results are the same, a Larken DOS. When used with Aerco FD-68, the built in extra memory of Aerco is used as "RAM DISK". This unique feature "formats" 40K banks of RAM as if the RAM were anther disk drive in the system. A 256K Aerco Ram is used as if it were four more disk drives, each having 40K of storage. The Ram Disk is somewhat faster operating than a disk drive, but of course, the data storage is gone when you turn off the computer.

Before using the LKDOS cartridge with the Aerco FD-68, a small modification to the FD-68 is required to add a switch. Details are given in the Larken documentation. No modification is needed with the Oliger Safe Board A. Please refer to the article in this issue titled "The TS-2068 Disk Drive Systems" for more information about the use of the LKDOS cartridge. For further information write to Larken Electronics, address given in the above review.

The Larken disk system for the Timex 2068 is fully Spectrum compatible and also can be used with Aros or Lros cartridge roms It supports all token keywords - CAT, ERASE,LOAD,SAVE,MERGE, OPEN#,CLOSE# and also GOTO and PRINT that were intended to be used with a external mass storage device. FORMAT and MOVE are supplied as programs that run in ram.
The way these commands are impliamented by the Larken system is to preceed them with a Rand usr 100:
eg: RAND USR 100: CAT "",
For easier typing and a shorter command the PRINT #4: command can be used instead of Rand Usr 100:. To use PRINT #4: you must first Open Channel 4 to the diskdrive with -
RAND USR 100: OPEN #4,"dd"
You can now preceed all LKdos and Ex-basic commands with PRINT #4: eg: PRINT #4: LOAD "filename.ex"
If the Print #4: command is used before it has been initialised, error 0 - Invalid Stream will result.

LKDOS File Names
The only other difference between a Lkdos command and a standard cassette command, is the file name.
Lkdos uses a file name that contains a program name up to 6 characters followed by a two character 'extension'. A period separates the program name from the extension. The first letter of the extension tells the dos what type of file it is . It must be an A,B or C
'A' for Array 'B' for Basic 'C' for Code
The second letter of the extension can be any character exept, if you are saving a string array, it must be a '$'.
The extension must be two characters long.
The only character that can't be used in a file name is a '?'. This is used as a 'wildcard' character for use in CAT searches.
Some examples of file names:

| | |
|---|---|
| "Progrm.B1" | A Basic program |
| "zeus.Cx" | A code file |
| "Names.A$" | A String Array |
| "Numbrs.A1" | A Numeric Array |

SAVE COMMANDS
Before you can save your programs and data to the disk, your disk must be formatted. (See section on setup and formatting) Any formula or expression in a command can be used.
eg: PRINT #4: SAVE a$ (TO 6)+".CT" CODE Start,End-Start
All variations of cassette commands are supported.

| | |
|---|---|
| PRINT #4: SAVE "Prog.B1" | - Basic program |
| : SAVE "Prog.B1" LINE 100 | - Basic Auto run |
| : SAVE "Prog.C1" CODE start,length | - Bytes save |
| : SAVE "Prog.C1" SCREEN$ | - Screen save |
| : SAVE "Prog.A1" DATA () | - Numeric Array |
| : SAVE "Prog.A$" DATA $() | - String Array |

Before saving make sure that the write protect notch on the disk is not covered by a protect sticker and that the disk is in the drive properly with the door closed.

EXTENDED BASIC COMMANDS :
These commands consist of extended graphic, additional Print and I/O device Channels and up to 3 scrolling windows on the screen in proportional spaced characters. (42 to over 80 charac ters per line)
GRAPHIC COMMANDS:
- PRINT #4: DRAW width,heigth,pattern (This is a versatile box fill or clear command that starts at the last plot command that fills in the selected pattern (0 to 9 (or 10= user defined pattern that consists of 8 bytes starting at address 23540)
- PRINT #4: CIRCLE x,y,pattern (This is a region filling command that fills from right to left, selected blank areas of the screen in the patterns described above. It will fill a line until it finds a set pixel) x,y sets the starting pos.
- PRINT #4: INK x (immediatly changes ink color on screen)
- PRINT #4: PAPER x (immediatly changes paper color on screen)
- PRINT #4: POKE a,b (this does a double poke for 16 bit numbers . b can be 0 to 65535
CHANNEL AND WINDOW COMMANDS :
- PRINT #4: OPEN #n ,"device" This attaches a channel to a display device. (n can be channel 2 to 15 ) device can be "w0", "w1", "w2" (window 0 to 2) or "lp" (larg e printer 'aerco type IF' only)
- PRINT #4: CLOSE #n (n can be 2 to 15 . This is used for closeing channels opened with the above command. Note- never use the Sinclar Close command (unpreceeded by Lkdos switch) to close a channal opened by Lkdos)
- PRINT #4: INPUT#(window),top,left pos,right pos,bottom pos
This command defines a windows size, position and color. "window" is 0-2 , other parameters are standard print positions in absolute numbers (0 to 20or31) EG: PRINT#4:INPUT#1,0,10,20,8
This would define window #1 to have its upper left corner at print pos 0,10 and the window would be 10 units wide (20-10) The bottom of the window would be at the 8th line.
The screen colors that are being used when the window is defined will stay with that window.
- PRINT #4: CLEAR w This will clear the window (0-2) and restore the print pos to the top of the window . It will also recolor the window.

## Profile of the Author

Mr. George Chambers is one of them "rotating officials", always involved in the Toronto TSUG, and now the club Secretary. George contributes a regular column in "Sync Link" about the Larken Disk system and other subjects. If it can be done with Larken, George knows how to do it. We look forward to other articles and programs by George.

# THE OLIGER TS-2068 SAFE DISK SYSTEM

A Review By: Bill Jones

Mr. John Oliger has now finished "phase 4" of the JLO SAFE DISK Operating System, which I have dubbed "SDOS". This review will attempt to give an overview of this versatile hardware and DOS. I believe that even the users of SAFE may glean some new information. First, the hardware is designed to the "SHUGHERT STANDARD" which means that it conforms to an existing standard for FORMAT and CONTROL of Double Density Soft Sector Disk Drives. Several types of drives are usable with this system including; 5 1/4 inch Half or Full height, Single or Double Side format; 3" Disk Drives by Amdek, and some newer 3 1/2" disk drives. The key as to compatability is whether the drive is "SHUGHERT STANDARD".

The Oliger SAFE hardware consists of two circuit boards that fit up-right in the Oliger Expansion port. This expansion port has four slots and a extruding edge card connector to accommodate other add-on devices. It also contains a "RGB INTERFACE" to connect to a RGB Monitor. "BOARD A" of the system contains the heart of the disk interface, the Western Digital Controller chip WS-1770, and its interface circuitry. "BOARD B" of the system contains 16K of extra memory. 8K is a EPROM that contains the assembly code of the operating system, some new BASIC Language functions, and a Centronics Printer Driver Software. The other 8K is RAM that is used for instruction processing of the new Basic Language and the Disk Control Basic Commands.

One should think of the Oliger Safe Disk System as the centerpiece of a larger system that includes the Printer Interface, the Disk Controller, Extra Memory, Additions to the TS-2068 ROM, RGB Monitor interface, and a "Control System" that includes control of the printer, Extra Memory, and the Disk Drive. The Extra Memory Board has not yet been released. "SDOS", the controlling software has been a on-going development since the SAFE DISK hardware was released during the winter months of 1986. Each stage of development was made available to the users in a Up-Dated EPROM. The latest EPROM carries the identication of "V2-40". For the user to up-date his system involves only the replacement of the EPROM that is in use.

As of now, the 8K EPROM is almost completely filled with assembled code, and unless Mr. Oliger decides to piggy back another EPROM, it appears that SDOS is in a finished state. A table of the SDOS Language Syntax is listed below. Note the similiarity to the regular TS-2068 Cassette instructions for easy learning. Also note the brevity of the command syntax (fewer key strokes). All of these commands can be performed by Keyboard entry or from within program lines. Disk file "titles" can be concantated as, LET A$="Title", and then use A$ as a title within the disk command. Or, <LET A$=A$+STR$ 5> and the program title to SAVE or LOAD would be "Title5".

## THE SDOS COMMAND SYNTAX

FORMAT/"name" A diskette is formatted to established parameters with title of "name".

CAT Loads the disk directory to screen. CAT can be used with a input statement to place a prompt on screen to input a title to LOAD.

MOVE/ Copies a diskette from current drive to the next drive in the system.

MOVE/"title" and type TO n Moves a file named "title" to disk n. The type of file is VAL, SCREEN$, DATA, ABS, or if left out, a BASIC file is assumed.

SAVE/0 Saves a auto loading menu program. Then LOAD ENTER will load the file 0 program.

LOAD ENTER Loads a file 0 auto running menu program.

SAVE/"name" Saves a BASIC program to disk.

SAVE/"name"LINE n Saves a BASIC program which, when re-loaded will auto run at line n.

LOAD/"name" Loads a BASIC program named "name". The program auto runs if it was saved with a starting line number.

SAVE/"name"CODE ,adr,byts Saves a machine code table from memory address and byte length specified.

LOAD/"name"CODE adr Loads a code program from disk to starting memory address specified.

SAVE/"name"VAL Saves the TS-2068 Variable file to disk.

LOAD/"name"VAL Loads a Variable file from disk to memory.

SAVE/"name"SCREEN$ Saves the existing monitor screen to disk.

LOAD/"name"SCREEN$ Loads a screen from disk and replaces the existing monitor screen.

SAVE/"name"DATA x() Saves a dimensioned Numeric array to disk.

LOAD/"name"DATA x() Loads a dimensioned Numeric array fromdisk.

SAVE/"name"DATA x$() Saves a dimensioned

Character array to disk.

LOAD/"name"DATA x$() Loads a dimensioned character array from disk and assigns the data to "x$".

SAVE/"name"ABS Saves the entire state of the computer as a BASIC program, including all vars and all code.

LOAD/"name"ABS" Loads the above State of the Computer BASIC program.

COPY/ Copies the screen to printer.

MERGE/"name" Merges a BASIC program and appends it to the existing BASIC program in memory. Variables of the MERGED program take priority.

All of the above SAVE commands will check for a identical named program in disk and TOOT a warning if one exists, then pause 5 seconds to allow an abort.

SAVE//"name" This variation by-passes the same name check and SAVES the program whether a same name exists or not.

RESTORE/"old name" type TO "new name" Changes the name of a program in disk.

RESTORE/s Re-initializes the SDOS system to TURN ON parameters.

VERIFY/"name type" Checks file in disk and verifies accuracy of the file data.

DISK DRIVE CONFIGURING COMMANDS

LET/s=1 or2 Sets up to FORMAT a diskette for either single or double side read-write.

LET/d=0-3 Changes active disk to disk number specified.

LET/t=10-250 Sets up to FORMAT a diskette to the number of tracks specified. Range is 10 to 250, default value is 40.

LET/h=0-3 Resets disk head step rate to accommodate a slower drive. Default value is 0=6ms. 1=12ms. 2=20ms. 3=30ms.

LET/p=o Selects the printer driver code software that is stashed in the EPROM to print with a Centronics printer.

LET/p=t Deselects the Centronics printer and selects the TS-2040 printer.

The NMI PUSH BUTTON FUNCTIONS

ABS STATE SAVE via numbered keys. The title of the SAVE will be the number pressed.

SCREEN$ SAVE via keys Q through T on same row of keyboard. The title will be the key pressed.

COPY SCREEN to Centronics Printer via press of key Z.

BREAK to BASIC via press of key C.

RETURN TO INTERRUPTED PROGRAM via the ENTER key.

Holding the key "N" (for NEW) while pressing NMI button causes a system RESET to copyright display.

Holding keys 1 through 3 upon power-up selects the Disk Drive number to be active (instead of Drive number 0).

JLO SAFE puts more data into a diskette than does most other disk control systems, 405K as versus 360K for a IBM system for example. A 80 track drive formats for 815K maximum. In addition SDOS is likely the fastest disk system in existance for data SAVE and LOAD to and from Disk, faster even than the IBM XT. Which brings up a point to discuss. Often the question arises; "Does it have Sequential File SAVE and Append?". The answer to that is, "No, such a group of functions are not needed, and would not be used if provided with this fast disk system".

Opening and Appending a existing file in disk is a function created to mitigate the deficiencies of some computers. For example, the C-128 and some other computers limit string and character array content to 255 characters, too little to contain paragraphs of textual data. The only way to SAVE textual data to disk with such computers is to use counters to save the data in small "line length" groups to disk, and keep on OPENing the disk files to APPEND the text files as additional data is being generated. Each OPEN and APPEND action takes tens of seconds of time. With SDOS, we SAVE a Character array of data as large as 25K in three seconds flat. If we want to APPEND it, we bring it back into memory (three seconds) and append it at will

A review can never do justice in such limited space to a comprehensive system such as the SAFE DISK Controller and SDOS. The literature supplied with the system is about 80 pages of type. How can that be boiled down to a couple of review pages? It can't, so one should realize that this review is just a "overview" of the system. The manual supplied is written for the kit builder and it goes into much more detail than one normally would expect of a user manual, giving theory of operation, testing instructions, and operating instructions. The manual is an outstanding reference for both the builder and the the user who buys the system assembled and ready to use.

COMPATABILITY WITH OTHER TS-2068 HARDWARE - SOFTWARE: Since the Safe Disk system is fully decoded and does not use the TS-2068 RAM, there is →



no hardware or software that this system conflicts with. Safe Disk operates with TS-2068, TS-2068 with Spectrum Switch, either in the TS-2068 or the Spectrum Mode, and with the Zebra OS-64 Cartridge. No software conflicts exist.

OLIGER SAFE USED WITH THE LARKEN SKDOS CARTRIDGE: Board A of the SAFE system is the disk controller. When Board B is removed, and the Larken SKDOS Cartridge is installed, the system becomes a LARKEN DOS. Then the diskettes are formatted to Larken format and diskettes recorded with Larken disk systems can be loaded and operated. See the review of the Larken Disk system in this issue.

OLIGER SAFE USED WITH THE SOFTWARE "SPDOS". This software on disk is supplied by Cuyahoga Valley Software Works, 615 School Ave., Cuyahoga Falls, OH 44221. ($24.95 + $1.50 postage). The SPDOS Disk provides the RAMEX DISK OPERATING SYSTEM "in addition to the Oliger SDOS system". The two disk operating systems are combined to have almost all conceivable disk command functions. Moreover, when the SPDOS software is loaded, diskettes prepared by a RAMEX DISK system will load and operate. The SPDOS Disk is highly recommended for all Oliger Safe users.

FOR THOSE WHO WANT EVERYTHING, the Oliger Safe system, with Larken SKDOS Cartridge, and with the SPDOS Disk Software, gives the user three disk operating systems and the capability to use software formatted with Oliger Safe, Larken Disk, and RAMEX Disk.

HARDWARE OPTIONS AND PRICING: The manual is written for those who build the two circuit board projects. This is a very educational project and one can save a considerable amount of money by assembling the two boards. Also, one may purchase both boards fully assembled and ready to use. The building project manual is a very detailed source of reference even for the non-builder. THE PRICING OPTIONS: Bare Circuit Boards with the SAFE EPROM=$43.95. Kit of both circuit boards with all parts=$97.95. Complete system Fully assembled and ready to use=$119.95 ($129.95 with the NMI Option). The Four Slot Expansion Buss: Bare board=$11.95. Board with parts=$43.95. Fully assembled=$54.95.

If you are ever going to build a electronics project, then a Oliger Kit is what you should start with. His instructions are clear and easy to understand. The only equipment needed to build these projects is a small soldering pencil. However it will help to have a desk magnifying glass to better see the tiny circuit traces, and an ounce of Acetone with a soft toothbrush are needed to clean off the solder flux as the job progresses. The building time for both boards will be about 5-10 hours. Mr. Oliger wont let you fail. For a nominal charge of $10.00 you can send him a completed board for check out and repair. So, if it doesn't work after you build it, that is your insurance.

SOURCE: The Oliger Company, 11601 Whidbey Drive, Cumberland, IN 46229. Send SASE for product list.

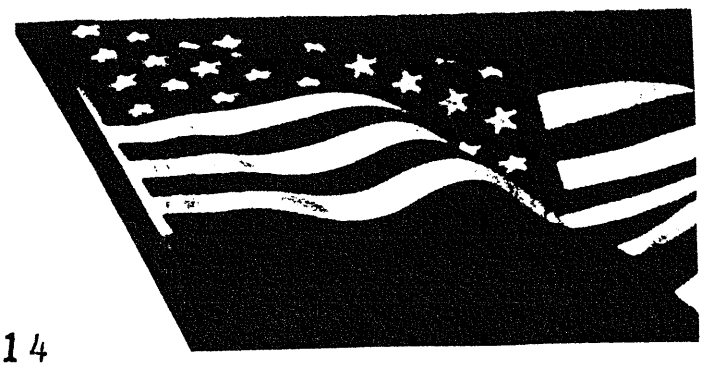
---

## TS-2068 SUPPORTING PUBLICATIONS

Changes at Syncware News: Tom Woods departs the scene, and Jeff Moore steps up from Editor to Publisher. Basil Wentworth, well known to most of us "timers" becomes Editor of Syncware News. Fred Nachbaur, the original founder and publisher, continues as Technical Advisor. Sounds like an enduring combo to me. Of interest is that Syncware News is beginning to support the IBM Clone Users, a move that should enhance their subscriber base and advertising potential. Up-Date recommends that all TS-2068 users continue their subscriptions to Syncware News and encourage their efforts. Syncware News, 602 S. Mill St., Louisville, OH 44641. Subscriptions, $16.95 U.S. $19.95 Canada and Mexico. (1 year 6 issues).

TIME DESIGNS and CTM MAGAZINE continues to be leader publications in support of Timex Sinclair Computers. Both are excellent publications, highly recommended by UP-DATE. Time Designs, 29722 Hult Road, Colton, OR 97017. Subscription $15.00 per year, six issues. CTM Magazine, 1704 Sam Drive, Birmingham, AL 35235. Subscription $18.00 per year (U.S) $32.00 per year (Canada, Mexico). 12 issues.

COMPUTER SHOPPER, a huge magazine, having a regular article series about Timex Sinclair, and a mammoth source of computer products advertisements. Computer Shopper, 5211 S. Washington Ave, PO Box F, Titusville, FL 32780. Subscription $21.00 per year (12 issues), U.S. Subscriptions only.



# The Aerco FD-68 Disk System

A Review by: Syd Wyncoop

The FD-68 is a disk system that has subscribed to the 'Theory of Evolution'. That is to say that the FD-68 is not a finished system and undergoes constant revision. However, Aerco has taken care to make sure that each update remains compatible with previous versions.

Some features of the FD-68 that I like are: ** Accepts any mixture of 4 disk drives in 3", 3 1/2", 5 1/4", or 8" sizes, which can be SSSD, SSDD, DSDD, or DSQD. ** Additional 64K of RAM memory addressed in the dock bank which is expandable to 256K. *** Built-in RGB interface. ** Entire system is contained on one board of very good quality. The board is uncased but has a backing on the solder side. *** The Operating system is in a ROM and uses none of the 'normal' system RAM. ** Uses standard Sinclair Keyword commands. *** Uses standard disk file names and extensions. ** Does not use a channel number, so there are no conflicts with other device drivers. **** Runs all CP/M 2.2 programs. ** Can be purchased as a complete system or you can economize by puting together your own system.

While the unfinished system has many unique enhancements, the process of evolution has created a few drawbacks. I feel that the biggest drawbacks are: ** The entire Basic variables area must be saved, as individual arrays cannot be saved. ** The system is not compatible with Spectrum programs. ** There is limited directory space. The array problem can be worked around rather easily, however, the Spectrum compatibility has been longed for by the users. The FD-68 will only recognize 32 program titles, at this time. I understand the directory space has been increased however, I have not received my latest EPROM up-date.

The real problem with the deficiencies is the long delay between updates (Spectrum capatability was to be completed two weeks after the 5/87 Sinclair Mini-fest). The updates are sent from Aerco in a new ROM. There is a small fee ($5.00 for the last update I received) for this and the old ROM is returned to Aerco to be recycled for the next update. Also, your ROM can be reprogrammed at any time, if you make changes in your system configuration. The Basic FD-68 system allows saving of Basic programs, binary memory images (code), the entire variables area (with or without a Basic program) and SCREEN$'s. There is also the capability of running an AROS program, from disk, in the dock bank! Provision has been made for the future addition of numeric arrays (.DAT), character arrays (.CHR) and LROS program (.LRO) extensions.

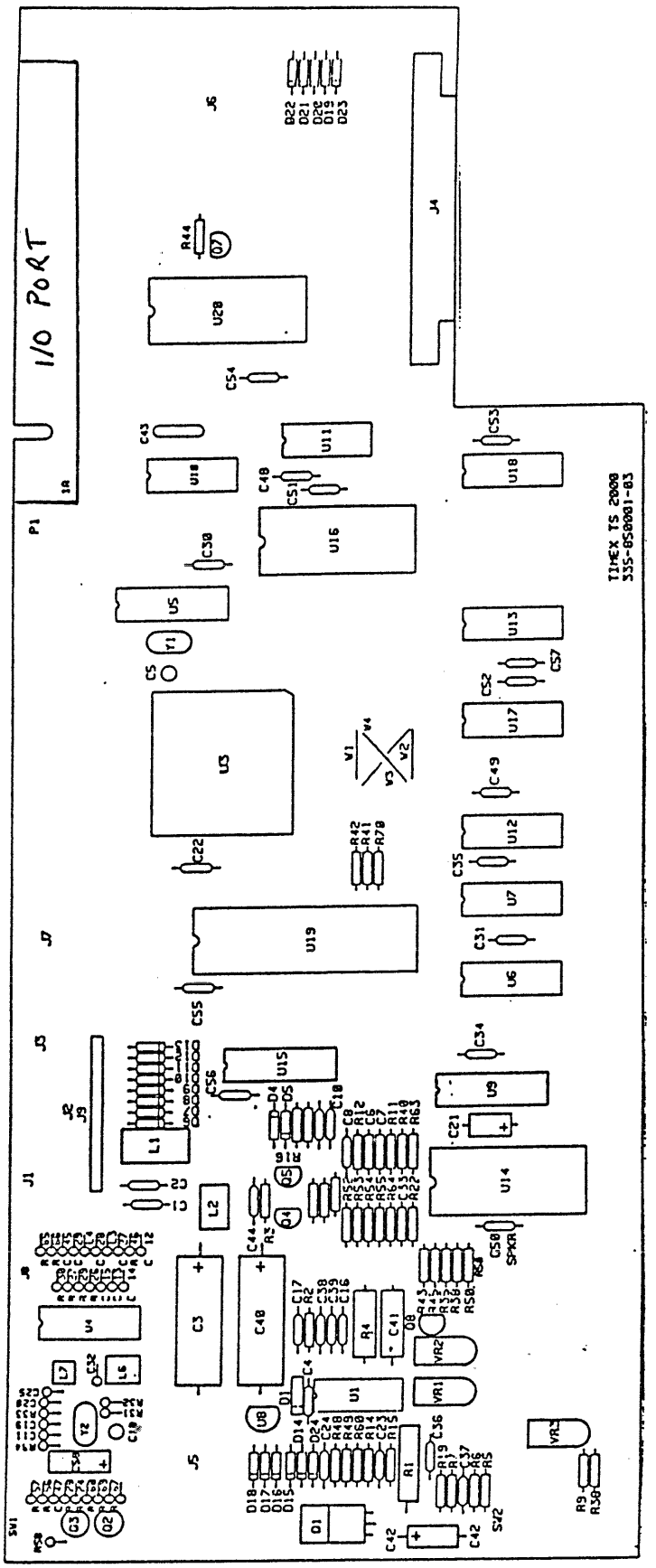
There is a special extension (.BUT) that allows entry into a very nice implementation of CP/M, called RP/M. CP/M provides an entirely new, to Sinclair users, operating system. There is a wide range of public domain and commercial applications packages available to us under CP/M. More on this in future issues, if there is a request for it. Each disk can store one autostart program by naming it 'Boot.bas' which runs upon turning on the computer, with the disk in drive A. I usually place a program such as M-Script or Zeus on a disk as the boot program. I then use the rest of that disk for files that the Boot program uses. All of my general purpose disks have a special Boot program I wrote that sets-up the system variables to my taste, catalogs the disk and loads the files I indicate. This has proven to be a very efficient procedure.

I cannot say enough good things about the FD-68. I looked at the Oliger and Zebra systems when I was shopping for a disk system and I chose the Aerco for the above reasons. I feel that it provides the widest range of options (Though I must admit that I do like the Zebra tree structure). It is a good value and a must have. I since looked at the Larken system. Larken has some unique features but, I am not convinced to change. In short, the Aerco FD-68 provides many unique features.

Source: AERCO, PO-Box 18093, Austin, TX 78760. Send SASE for literature and prices.

## Author Profile

Mr. Syd Wyncoop works with the J. I. Case Co. of Portland, Oregon, manufacturing Farm and Construction Equipment (How unique! An American Manufacturer!). A Boy Scout leader, and member of the Executive Committee of St. Therese School, Syd is involved but finds time to teach Machine Code to the Portland SIG group and hold office as Chairman of CCATS, one of the most active Timex User Groups. Syd bought his first computer, a TS-1000, to acquaint his two boys with computers. Then he had to teach himself in order to be the teacher. Now he is the Guru of the Portland area user groups. We UP-DATE'rs encourage Syd to continue covering the Aerco FD-68 and CP/M, as well as sharing his programs and Utilities. SALUTE!




**T82068: PC BOARD COMPONENT LAYOUT**

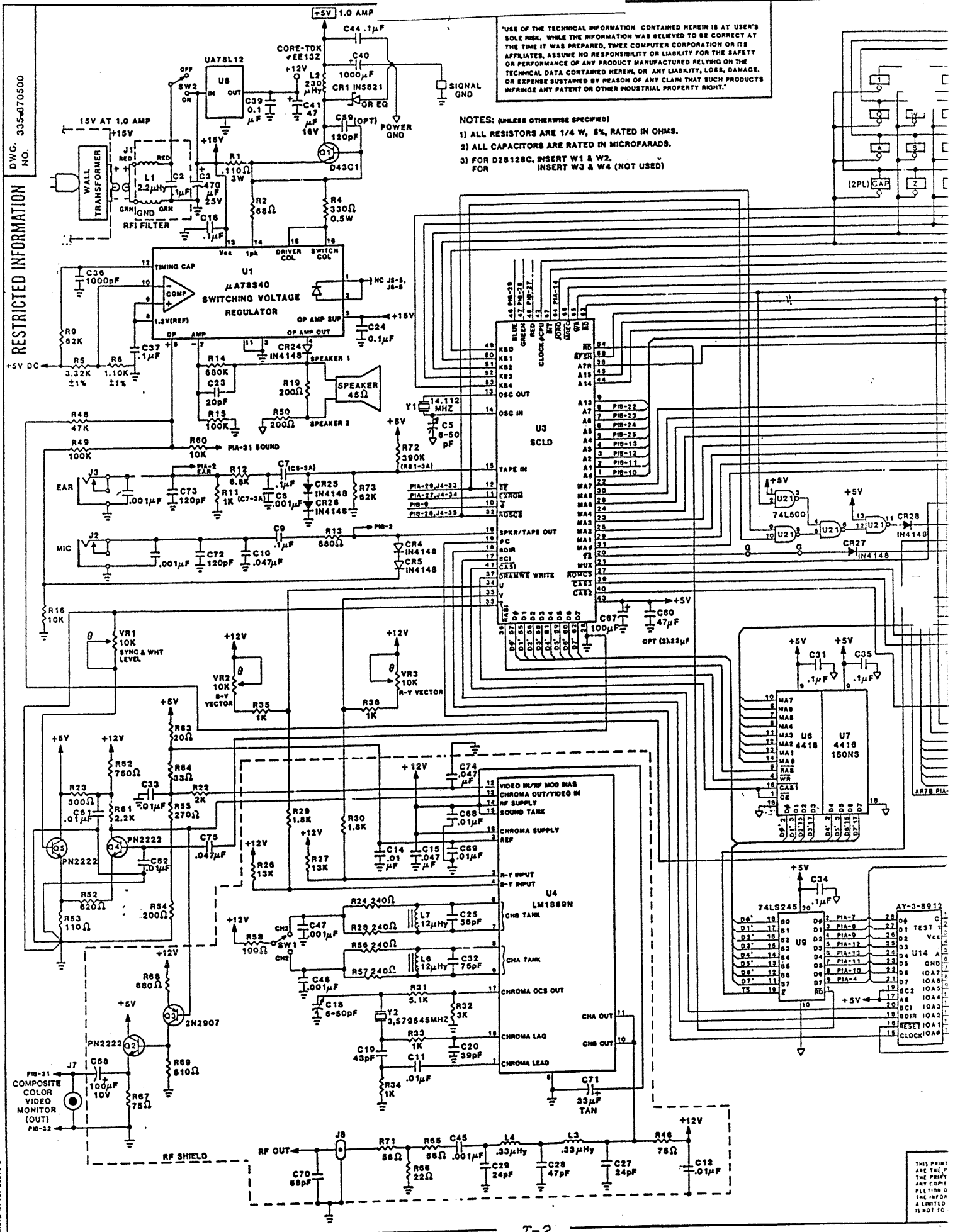
RESTRICTED INFORMATION

DWG. NO. 335-870500

"USE OF THE TECHNICAL INFORMATION CONTAINED HEREIN IS AT USER'S SOLE RISK. WHILE THE INFORMATION WAS BELIEVED TO BE CORRECT AT THE TIME IT WAS PREPARED, TIMEX COMPUTER CORPORATION OR ITS AFFILIATES, ASSUME NO RESPONSIBILITY OR LIABILITY FOR THE SAFETY OR PERFORMANCE OF ANY PRODUCT MANUFACTURED RELYING ON THE TECHNICAL DATA CONTAINED HEREIN, OR ANY LIABILITY, LOSS, DAMAGE, OR EXPENSE SUSTAINED BY REASON OF ANY CLAIM THAT SUCH PRODUCTS INFRINGE ANY PATENT OR OTHER INDUSTRIAL PROPERTY RIGHT."

NOTES: (UNLESS OTHERWISE SPECIFIED)

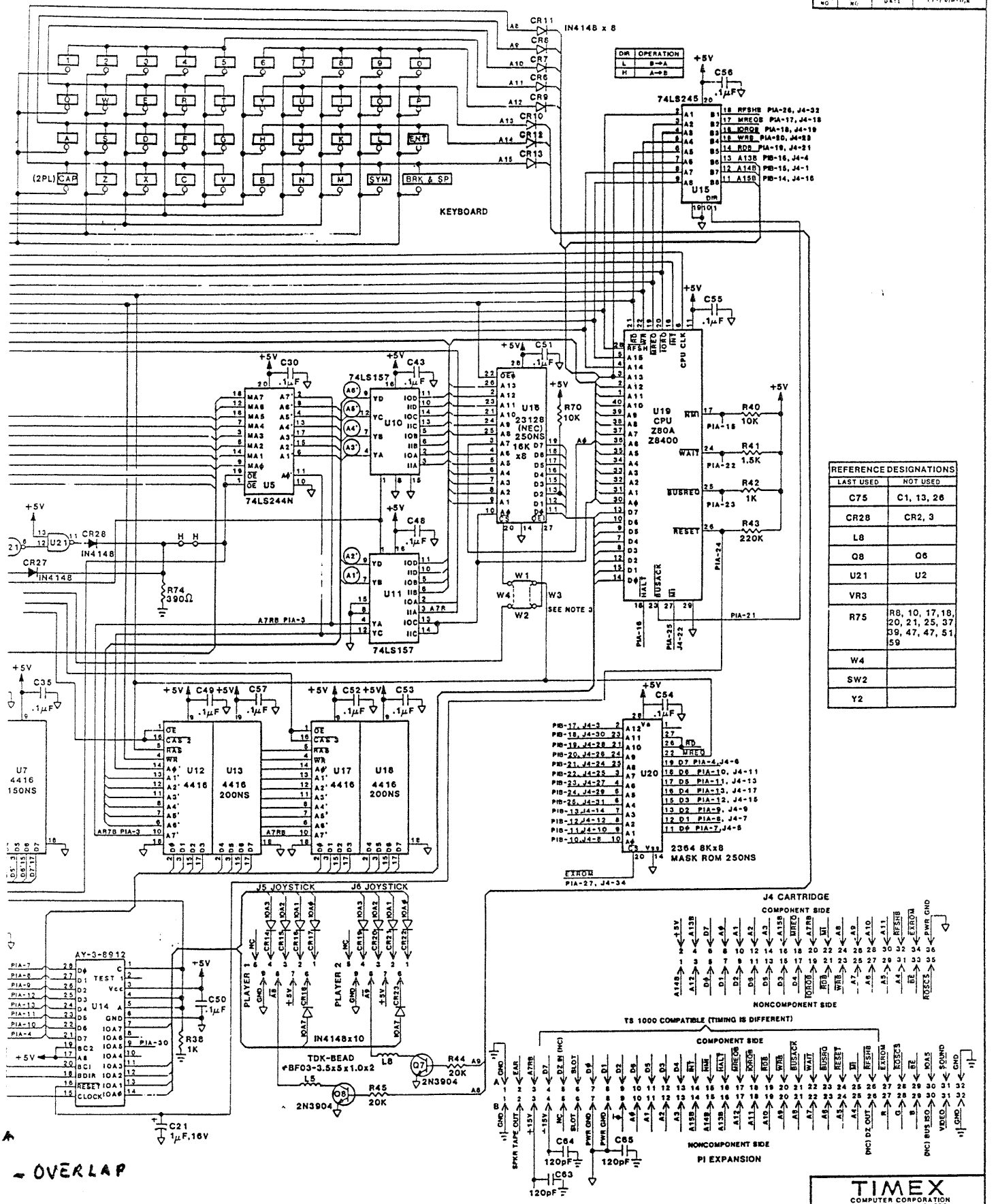
1) ALL RESISTORS ARE 1/4 W, 5%, RATED IN OHMS.
2) ALL CAPACITORS ARE RATED IN MICROFARADS.
3) FOR D28128C, INSERT W1 & W2.
   FOR INSERT W3 & W4 (NOT USED)






BRUNING 44-131 22870-3

| CHANGES | | | |
|---|---|---|---|
| ISSUE NO | C.O. NO | DATE | |




| REFERENCE DESIGNATIONS | |
|---|---|
| LAST USED | NOT USED |
| C75 | C1, 13, 26 |
| CR28 | CR2, 3 |
| L8 | |
| Q8 | Q6 |
| U21 | U2 |
| VR3 | |
| R75 | R8, 10, 17, 18, 20, 21, 25, 37 39, 47, 47, 51 59 |
| W4 | |
| SW2 | |
| Y2 | |

- OVERLAP

TIMEX
COMPUTER CORPORATION
WATERBURY, CONN. 06720

| | | | |
|---|---|---|---|
| THIS PRINT AND THE INFORMATION IT GIVES ARE THE PROPERTY OF TIMEX CORPORATION THE PRINT IS NOT TRANSFERABLE AND IT AND ANY COPIES ARE TO BE RETURNED UPON COMPLETION OF THE CORRESPONDENCE OR ORDER THE INFORMATION [illegible] A LIMITED PURP[illegible] IS NOT TO [illegible] | MATERIAL | | DO NOT SCALE DRAWING |
| | SPEC | | |
| | SIZE & TOL | | |
| | TEMPER | | |
| | PHY SPEC | | |
| | T-3 | DWN N PARSONS | |
| | | DATE FEB 84 | |
| | | SCALE NONE | SUPERSEDES / SUPERSEDED BY |

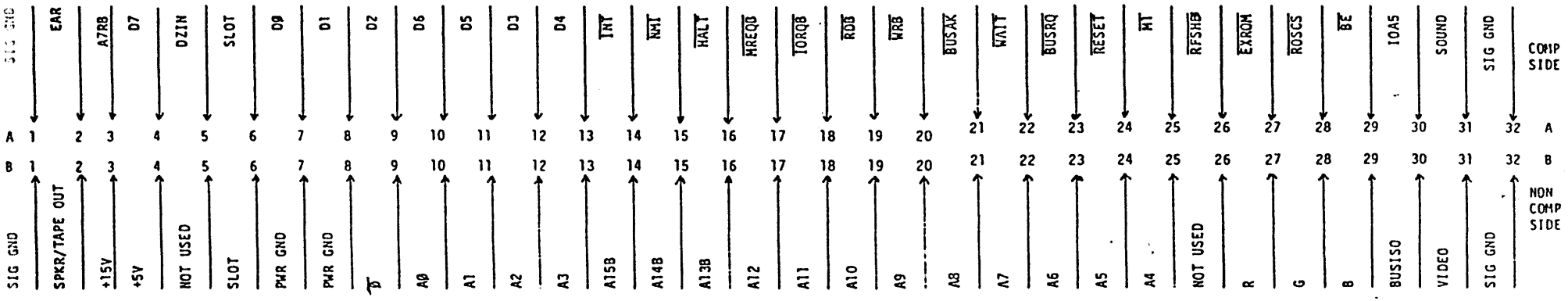
NAME: TS2000 SYSTEM SCHEMATIC DIAGRAM

DWG NO 335-870500

# P-1 CONNECTOR SIGNAL LAYOUT

## COMPONENT SIDE

TS1000 COMPATIBLE




## NON-COMPONENT SIDE

(VIEW FROM FRONT OF COMPUTER)

# NOW! AFFORDABLE
# 2068/1520
# HI-RES PLOTTING

DISCOVER A NEW DIMENSION IN T/S 2068 COMPUTING!

Just look at all of the "plot-a-bilities":

> 1520 is both a hi-res plotter AND a printer.
> 1520 is compact-slightly larger than a 2040.
> High quality ALPS made plotter mechanism.
> 4 color plotting: black, blue, green, & red.
> 4 character sizes: 10, 20, 40, and 80 c.p.i.
> 16 selectable dashed line sizes.
> 90 degree character rotation possible.
> 479 "x" step / +-998 "y" step plotting area.
> Extra paper & pens avail. at RADIO SHACK.
> Quality PC board with plate-through holes.
> "LPRINT" commands control plotter functions.
> Expanding variety of software for the I/F.
> 1520 available @ TOYS-R-US stores for $29.95.
> Write for more infomation and sample plot.

NAME: _____

ADDRESS: _____

CITY/STATE/ZIP: _____

| QUANTITY | DESCRIPTION | PRICE EACH | TOTAL |
|---|---|---|---|
| | Bare Board w/cassette | $14.95 ppd | |
| | Complete Kit w/cassette | $20.95 ppd | |
| | Assembled/tested w/cass. | $30.95 ppd | |
| | Keyed Edge Connector-If purchased with hardware | $4.95 ppd | |
| | SCREEN-TO-PLOTTER util. | $8.95 ppd | |
| | PIC-PLOT utility | $8.95 ppd | |
| | BANNERIFIC utility | $8.95 ppd | |
| | CMS/1520 patch kit | $8.95 ppd | |
| 1 set/order | COLORED REPLACEMENT PENS | $2.00 ppd | |
| | | TOTAL ENCLOSED>> | |

SEND ALL ORDERS TO:
John McMichael, 1710 Palmer Dr., Laramie, WY 82070

---

## ‡‡HERB BOWERS PROUDLY PRESENTS‡‡



# MONEY MACHINE II

**Based on a Popular TV Game Show**

STARRING LOVELY MISS BANNA BRITE

The Ultimate Word Game

The all new MONEY MACHINE II FEATURING:

New theme!!
Fantastic Family Fun!!
"BANNA" turning the letters
Hundreds of NEW Puzzles
Solo or 2 or 3 Players

All this for only
T/S 2068 Cassette **$12.00** (US)
Includes Postage-via 1st Class


HERB BOWERS, Sr.
2588 Woodshire Circle
Chesapeake, VA 23323
Phone: 804 487-5924


A·B·B·A | Accurate Business, Bookkeeping and Accounting Software

---

NEW FROM A·B·B·A SOFT

# Don't File Your 1987 Tax Return
UNTIL YOU HAVE
# QWIK-1040 ©

**********
INCLUDES: Form 1040, 1040A & EZ, Sch. A, B & R * Forms 2106, 2441 Automatic Calculation Income Tax Liability, Child Care Credit Taxable Social Security, Earned Income Credit *Multiple W-2 & 1099 Entry * 2040 Compatible * More

* Easy Manual Transfer to all Official Internal Revenue Forms *

Only **$29.50** Postpaid
Jan. 88 Delivery
For Timex/Sinclair 2068 Computer

SPECIAL! If we receive your order before by January 5, 1988 you will receive FREE of charge "THE NEW TAX LAW and YOU". This program has been review in TIME DESIGNS, COMPUTER SHOPPER, U.A.U.G., TIMELINES and other User Publications and is being used all over the United States. IMMEDIATE SHIPMENT OR if you have already purchased "THE NEW TAX LAW" from us you may deduct $7.00 from the purchase price which will give you a full credit for the original purchase.

GUARANTEE: If you must pay any fine or penalty as a DIRECT RESULT of using "QWIK-1040", I will pay such fine or penalty for you. Herb Bowers, Sr.

Schedules "C", "D", "E", "SE", & Forms 2119, 2210, 3903, 4137, 4255, 4562, 4972, 8582, 8598, 8615 & other forms will be available soon.

CHECK OR MONEY ORDER. (US Funds)


HERB BOWERS, Sr.
2588 Woodshire Circle
Chesapeake, VA 23323
Phone: 804 487-5924


A·B·B·A = Accurate Business, Bookkeeping and Accounting Software

## UP-DATE ISSUE DISKETTES

The software and utilities presented in each issue of UP-DATE are consolodated and offered on disk. These are all NEW programs having a combined value of at least $50.00. The authors of the titles share in the sales proceeds of the Issue Diskette. As you add these useful programs to your library you are supporting your Up-Date magazine and the authors who bring you these useful programs.

Disk System Support: By the time you read this UP-DATE should have support capability for LARKEN, AERCO FD-68, RAMEX, and OLIGER SAFE, disk systems. Disk Drive support includes: 3" Amdisk, 5 1/4" DS/DD 40 or 80 track, and 5 1/4" SS/DD 40 track. Due to high cost and scarcity, orders for 3" Amdisk programs must be accompanied by a 3" diskette for dubbing.

### JANUARY 88 DISKETTE

The January disk contains: MTERM DATA CAPTURE, by George Chambers: DOSDEX File Indexer, by Bob Hartung: PURITY, by Up-DATE: LIST LOOKER, by Up-Date: EXTRA MEMORY UTILITIES, by Up-Date. PRICE: $19.00 postpaid.

### OCTOBER 87 DISKETTE

The October 87 disk contains: MAIL MERGE, by Up-Date: J-UTILITIES, by Up-DATE: A Menagerie of 20 Sound and Visual utilities: A Group of Programming Tips. Mail Merge is a major software. PRICE $16.00 postpaid.

Special offer: Get both January and October Issue Diskettes for $32.00 postpaid. Order through UP-DATE, address on cover.

### Smart Text Disk

The Smart Text programs are being prepaired in LARKEN, AERCO FD-68, and RAMEX, disk formats. Disk drive formats are: 3" Amdisk, 5 1/4" DS/DD 40 or 80 Track, or 5 1/4" SS/DD 40 Track. Due to the cost and scarcity of 3" diskettes, orders for 3" disk must be accompanied by two 3" diskettes for dubbing. Each diskette has an additional software for use with the Zebra OS-64 cartridge for 64 column display. Standard price for all is $26.00 (postpaid US and Canada). Order through UP-DATE. Satisfaction Guaranteed.

### Smart Text Cassette

The Cassette version of Smart Text has many of the automatic data processing and printing functions of the disk drive software, and it includes the program for use with the Zebra OS-64 Cartridge. This software is available only for use with Aerco or Oliger Centronics Parallel Interface. As programmed, ST works with EPSON compatable printers. Program line changes are required for others. 80 page manual includes instructions. The TS-2040 printer is also supported. Price $26.00 (Postpaid). Order through UP-DATE. Satisfaction Guaranteed.

Continued from preceeding page.

vswr=Voltage standing wave ratio
vtvm=Vacuum tube volt meter
VU=Volume unit
wb=Wide band
wdg=Winding
whdm=Watt hour meter
wm=Wattmeter
wv=Working voltage
X=Reactance
Xl=Inductive Reactance
za=Zero adjusted
vt=Vacuum tube
VOX=Voice operated transmitter keyer
w=Wide
wd=Watt demand meter
wg=Waveguide; wire gage
wl=Wavelength
wrg=Wiring
ww=Wire wound
Xc=Capacitive reactance
y=Admittance
Z=Impedance; Zone

## SEMICONDUCTOR ABBREVIATIONS

The following abbreviations have been adopted for use with semiconductor devices.

$\alpha$—Alpha. Common-base short-circuit current gain
B, b—Base electrode for units employing a single base
$b_1$, $b_2$, etc.—Base electrodes for more than one base
B—Beta. Common-emitter short-circuit current gain
$BV_R$—Breakdown voltage, reverse
C, c—Collector electrode
$C_{cb}$—Interterminal capacitance, collector-to-base
$C_{ce}$—Interterminal capacitance, collector-to-emitter
$C_{ds}$—Drain-source capacitance, with gate connected to the guard terminal of a three-terminal bridge
$C_{dso}$—Open-circuit drain-source capacitance
$C_{du}$—Drain-substrate capacitance, with gate and source connected to the guard terminal of a three-terminal bridge
$C_{eb}$—Interterminal capacitance, emitter-to-base
$C_{rdo}$—Open-circuit gate-drain capacitance
$C_{rso}$—Open-circuit gate-source capacitance
$C_{ibo}$—Open-circuit input capacitance (common base)
$C_{ibs}$—Short-circuit input capacitance (common base)
$C_{ieo}$—Open-circuit input capacitance (common emitter)
$C_{ies}$—Short-circuit input capacitance (common emitter)
$C_{iss}$—Gate-source capacitance, with drain short-circuited to source
$C_{obo}$—Open-circuit output capacitance (common base)
$C_{obs}$—Short-circuit output capacitance (common base)
$C_{ods}$—Short-circuit output capacitance (gate-drain short-circuited to ac)
$C_{oeo}$—Open-circuit output capacitance (common emitter)
$C_{oes}$—Short-circuit output capacitance (common emitter)
$C_{oss}$—Drain-source capacitance, with gate short-circuited to source
$C_{rbs}$—Short-circuit reverse transfer capacitance (common base)
$C_{res}$—Short-circuit reverse transfer capacitance (common collector)
$C_{rss}$—Short-circuit reverse transfer capacitance (common emitter)
$C_{rss}$—Drain-gate capacitance, with the source connected to the guard terminal of a three-terminal bridge
D—Duty cycle
d—Damping coefficient
E, e—Emitter electrode
$f_{hfb}$—Small-signal, short-circuit, forward-current, transfer-ratio cutoff frequency (common base)
$f_{hfc}$—Small-signal, short-circuit, forward-current, transfer-ratio cutoff frequency (common collector)
$f_{hfe}$—Small-signal, short-circuit, forward-current, transfer-ratio cutoff frequency (common emitter)
$f_{max}$—Maximum frequency of oscillation
$f_T$—Transition frequency
$g_{MB}$—Static transconductance (common base)
$g_{mb}$—Small-signal transconductance (common base)
$g_{MC}$—Static transconductance (common collector)
$g_{mc}$—Small-signal transconductance (common collector)
$g_{ME}$—Static transconductance (common emitter)
$g_{me}$—Small-signal transconductance (common emitter)
Ge—Germanium
$G_{PB}$—Large-signal average power gain (common base)
$G_{pb}$—Small-signal average power gain (common base)
$G_{PC}$—Large-signal average power gain (common collector)
$G_{pc}$—Small-signal average power gain (common collector)
$G_{PE}$—Large-signal average power gain (common emitter)
$G_{pe}$—Small-signal average power gain (common emitter)
$G_{pg}$—Small-signal insertion power gain, common gate
$G_{ps}$—Small-signal insertion power gain, common source
$G_{TB}$—Large-signal transducer power gain (common base)
$G_{tb}$—Small-signal transducer power gain (common base)
$G_{TC}$—Large-signal transducer power gain (common collector)
$G_{tc}$—Small-signal transducer power gain (common collector)
$G_{TE}$—Large-signal transducer power gain (common emitter)
$G_{te}$—Small-signal transducer power gain (common emitter)
$G_{tg}$—Small-signal transducer power gain, common gate
$G_{ts}$—Small-signal transducer power gain, common source
$h_{FB}$—Static value of the forward-current transfer ratio (common base)
$h_{fb}$—Small-signal, short-circuit, forward-current transfer ratio (common base)
$H_{FC}$—Static value of the forward-current transfer ratio (common collector)
$h_{fc}$—Small-signal, short-circuit, forward-current transfer ratio (common collector)
$h_{FE}$—Static value of the forward-current transfer ratio (common emitter)
$h_{fe}$—Small-signal, short-circuit, forward-current transfer ratio (common emitter)
$h_{FEL}$—Inherent large-signal, forward-current transfer ratio
$h_{IB}$—Static value of the input resistance (common base)

Continued on next page.

Continued from preceeding page

$h_{ib}$—Small-signal value of short-circuit input impedance (common base)
$h_{IC}$—Static value of the input resistance (common collector)
$h_{ic}$—Small-signal value of short-circuit input impedance (common collector)
$h_{IE}$—Static value of the input resistance (common emitter)
$h_{ie}$—Small-signal value of short-circuit input impedance (common emitter)
$h_{ie}$ (real)—Real part of small-signal value of short-circuit input impedance (common emitter)
$h_{OB}$—Static value of open-circuit output conductance (common base)
$h_{ob}$—Small-signal value of open-circuit output admittance (common base)
$h_{OC}$—Static value of open-circuit output conductance (common collector)
$h_{oc}$—Small-signal value of open-circuit output admittance (common collector)
$h_{OE}$—Static value of open-circuit output conductance (common emitter)
$h_{oe}$—Small-signal value of open-circuit output admittance (common emitter)
$h_{rb}$—Small-signal value of open-circuit, reverse-voltage transfer ratio (common base)
$h_{rc}$—Small-signal value of open-circuit, reverse-voltage transfer ratio (common collector)
$h_{re}$—Small-signal value of open-circuit, reverse-voltage transfer ratio (common emitter)
I, i—Intrinsic region of a device (where neither holes nor electrons predominate)
$I_B$—Base current (dc)
$I_b$—Base current (rms)
$i_B$—Base current (instantaneous)
$I_{BO}$—Breakover current, direct
$I_C$—Collector current (dc)
$I_c$—Collector current (rms)
$i_C$—Collector current (instantaneous)
$I_{CBO}$—Current cutoff current (dc), emitter open
$I_{CEO}$—Collector cutoff current (dc), base open
$I_{CER}$—Collector cutoff current (dc), with specified resistance between base and emitter
$I_{CES}$—Collector cutoff current (dc), with base short-circuited to emitter
$I_{CEV}$—Collector cutoff current with specified voltage between base and emitter
$I_{CEX}$—Collector current (dc), with specified circuit between base and emitter
$I_{CO}$—Collector leakage current (cutoff current).
$I_D$—Drain current (dc)
$I_{D(off)}$—Drain cutoff current
$I_{DSR}$—Drain current, (external) gate-source resistance specified
$I_{DSS}$—Drain current, zero gate voltage
$I_{DSX}$—Drain current, gate-source condition specified
$I_E$—Emitter current (dc)
$I_e$—Emitter current (rms)
$i_E$—Emitter current (instantaneous)
$I_{EBO}$—Emitter cutoff current (dc), collector open
$I_{EC(off)}$—Emitter-collector offset current
$I_{ECS}$—Emitter cutoff current (dc), base short-circuited to collector
$I_{E1E2}$—Emitter cutoff current (double-emitter transistors)
$I_F$—Forward current (dc)
$I_f$—Forward current, alternating component
$i_F$—Forward current (instantaneous)
$I_{F(AV)}$—Forward current, dc value with alternating component
$I_{FG}$—Forward gate current (direct)
$I_{FGM}$—Peak forward gate current
$I_{FM}$—Forward current, peak total value
$I_{F(OV)}$—Forward current, overload
$I_{FRM}$—Forward current, peak repetitive
$I_{FSM}$—Forward current, peak surge
$I_G$—Gate current, dc
$I_{GF}$—Forward gate current
$I_{GR}$—Reverse gate current
$I_H$—Holding current, direct
$I_I$—Infection-point current
$I_O$—Average output rectified current
$I_{OV}$—Overload on-state current
$I_P$—Peak-point current (double-base transistor)
$I_R$—Reverse current (dc)
$I_r$—Alternating component of reverse current (rms value)
$i_R$—Reverse current (instantaneous)
$i_{R(REC)}$—Reverse recovery current
$I_{RRM}$—Peak reverse current, repetitive
$I_{R(RMS)}$—Reverse current, total rms value
$I_S$—Source current
$I_{SDS}$—Source current, zero gate voltage
$I_{SDX}$—Source current, gate-drain condition specified
$I_{TRM}$—Peak on-state current, repetitive
$I_{TSM}$—On-state current surge (nonrepetitive)
$I_U$—Substate current
$I_V$—Valley-point current (double-base transistor)
$I_Z$—Regulator current, reference current (dc)
$I_{ZK}$—Regulator current, reference current (dc near breakdown knee)
$I_{ZM}$—Regulator current, reference current (dc maximum rated current)
$K_O$—Thermal derating factor
$L_c$—Conversion loss
M—Figure of merit
N, n—Region of a device where electrons are the majority carriers
NF—Noise figure
$NF_o$—Overall noise figure
$NR_o$—Output noise ratio
P, p—Region of a device where holes are the majority carriers
$P_{BE}$—Total power input (dc or average) to the base electrode with respect to the emitter electrode
$p_{BE}$—Total power input (instantaneous) to the base electrode with respect to the emitter electrode
$P_{CB}$—Total power input (dc or averge) to the collector electrode with respect to the base electrode
$p_{CB}$—Total power input (instantaneous) to the collector electrode with respect to the base electrode
$P_{CE}$—Total power input (dc or average) to the collector electrode with respect to the emitter electrode
$p_{CE}$—Total power input (instantaneous) to the collector electrode with respect to the emitter electrode
$P_{DS}$—Drain-source power dissipation
$P_{EB}$—Total power input (dc or average) to the emitter electrode with respect to the base electrode
$p_{EB}$—Total power input (instantaneous) to the emitter electrode with respect to the base electrode
$P_F$—Forward power loss (dc)
$p_F$—Forward power loss (instantaneous)
$P_{FM}$—Forward power loss, total peak value
$P_{IB}$—Large-signal input power (common base)
$P_{ib}$—Small-signal input power (common base)
$P_{IC}$—Large-signal input power (common collector)
$P_{ic}$—Small-signal input power (common collector)
$P_{IE}$—Large-signal input power (common emitter)
$P_{ie}$—Small-signal input power (common emitter)
$P_{OB}$—Large-signal output power (common base)
$P_{ob}$—Small-signal output power (common base)
$P_{OC}$—Large-signal output power (common collector)
$P_{oc}$—Small-signal output power (common collector)
$P_{OE}$—Large-signal output power (common emitter)
$P_{oe}$—Small-signal output power (common emitter)
$P_R$—Reverse power loss
$p_R$—Reverse power loss (instantaneous)
$P_{SM}$—Surge nonrepetitive power
$P_T$—Total power input (dc or average) to all electrodes
$p_T$—Total power input (instantaneous) to all electrodes
$Q_s$—Recovered charge (stored charge)
$R_B$—External base resistance
$r_{BB}$—Resistance between two bases, emitter zero (double-base transistor)
$r_b'C_c$—Collector-base time constant

Continued on next page.

# MTERM II
## Modifications for the LARKEN DD System

by G. Chambers
14 Richome Court
Scarborough, Ont M1K 2Y1 CANADA

When I started to put MTERM II onto the LARKEN system I had some ideas of what I wanted to do. Primarily I wanted to be able to make disk saves of the buffer at any time while on-line. I soon found that as MTERM was constituted, this was impractical. MTERM as presently constituted cleared out any BASIC program and variables that were initial- ly present. Thus my idea of having a BASIC program ready to perform the disk-save immediately upon an exit from MTERM was not possible. I have overcome this by a few judicious POKEs into MTERM, plus some unique ideas in a companion BASIC program. The following notes describe the changes and their purpose.

I found that one of the first things that MTERM does upon initialisation is to look at the system variables VARS and PROG. It uses these values to clear out the BASIC program and variables, also to help in determining the ongoing buffer status.

Bytes which do this are located at 54035/37, 54043/45, and 61165/66. I modified the program by POKEing new values into several MTERM addresses, as follows.

54034,33: 54035,12: 54036,123
54043,33: 54044,12: 54045,123
54523 to 54537, POKE all zero's in these addresses.

The first six POKEs cause MTERM to set the start of the buffer space. This figure could be set anywhere; it is simply a trade-off between buffer size and BASIC program size. Initially, I considered having MTERM look at the system variable E-LINE, however I found that this became corrupted whenever I entered a direct command. I fastened on an arbitrary value of 31500 since I felt that with a disk-save routine in place, maximizing buffer space was less important than having sufficient space for the BASIC program. This sizing will allow a LARKEN-modified LOADER V to be used.

I found that with this, even though the BASIC remained untouched, the program variables were being wiped out. Looking around, I found a block of MTERM code (54523/54537) that appeared to do this. I simply wiped it out with no apparent ill-effect.

Addresses 61164/65 store the used/free buffer status. They are loaded with the values found in addresses 54044/45 (i.e. the starting address of the buffer) at initialisation, and are incremented as the buffer fills. I have made use of this in LINE 22 of the BASIC program to determine the size of the block of data to be saved.

These addresses 61164/65 can be useful in another way. If, for example, you inadvertently erase the buffer while in the Buffer Menu mode it is possible to restore it. Option 4 in the menu allows you to insert an arbitrary buffer length. This will change the "buffer used/free" pointer in MTERM so that one can view and/or save the buffer.

These addresses 61164/65 can be useful in another way. If, for example, you inadvertently erase the buffer while in the Buffer Menu mode it is possible to restore it by POKEing address 61165 with a value between 124 and 211. The buffer itself has not been erased, simply the pointer, and this POKE creates an arbitrarily-sized buffer.

There a number of features about the program which are worthy of explanation. LINE 100 contains POKEs to addresses 23662/63. It's purpose is this. Usually, if you execute a RAND USR from a BASIC program and return, the BASIC program will continue from where you left it. However with MTERM this is not the case. You will come out to an OK report at the bottom of the screen. I was unable to find the MTERM code which caused this. Failing that, I solved the dilemma by POKEing a number into the system variable OLDPPC (address 23662/63) corresponding to the LINE number where I wish to continue in the BASIC program. Then, when I exit from MTERM it is simply necessary to press the C key (CONT), and the ENTER key, to find myself back into the running BASIC program at the spot where I want to be.

The BASIC has a couple of other things about it which are of interest. If one accesses MTERM by a RAND USR 54016 the program is reinitalized with the buffer set as cleared. However, if it is desired to re-enter MTERM without disturbing the state of the buffer then one can use RAND USR 54089. This option has been included in the BASIC at LINES 30 and 32.

There probably will be occasions when several successive buffer SAVES will be required during the same download. The BASIC contains an arrangement whereby the name of the saved buffer remains the same while it's suffix is incremented. Where the initial SAVE has a suffix of "Ca", the suffix in subsequent SAVEs will be incremented as "Cb", "Cc", etc.

Although we have shown an original BASIC program here for use with MTERM there is every reason to imbed these program ideas into LOADER IV or LOADER V. They will work there in the same way, and provide for the additional features of these programs.

Also, even without a disk system, these features may prove to have some merit, and are worth considering.

**In adapting this program you will have to watch carefully when you tamper with the makeup of LINES 20 21, 32, and 100. What you are trying to do is direct the return of the computer to the appropriate line and statement number, and this can be tricky.**

Note that the program is designed to be used in the LARKEN AUTOSTART mode. That is, you should save it to disk with a GOTO 3.

```
Ø>REM Modified MTERM II for
     the LARKEN disk system
        by G. Chambers

  2 GO TO VAL "9"
  4 CLEAR VAL "31499": RANDOMIZ
E USR VAL "1Ø2"
  6 CLEAR VAL "54Ø15": RANDOMIZ
E USR VAL "1ØØ": OPEN #4,"dd"
  7 PRINT #4: LOAD "mterm2.C1"C
ODE
  9 LET g$="buffer"
 1Ø LET g=VAL "15Ø": LET q=VAL
"97": LET f=VAL "2Ø": LET o=NOT
 PI: LET i=PI/PI: BORDER o: PAP
ER o: INK VAL "7": CLS : PRINT
TAB VAL "1Ø";"MTERM II"'''TAB
PI*PI;"MAIN MENU:"'''TAB PI+PI;
"1) RUN MTERM"''TAB PI+PI;"2) L
OAD BUFFER"''TAB PI+PI;"3) NAME
 A BUFFER FILE"''TAB PI+PI;"4)
RESTORE BUFFER"''TAB PI+PI;"5)
SCAN DISK FILE":: GO SUB VAL "1
3": IF LEN z$<>i OR z$<"1" OR z
$>"5" THEN  RUN
 12 GO TO VAL z$*f
 13 INPUT "Your Choice? "; LINE
z$: RETURN
 2Ø GO SUB g: RANDOMIZE USR VAL
"54Ø16"
 21 CLS : CLS : LET q$=CHR$ q:
PRINT AT VAL "1Ø",NOT PI;"Save
buffer to disk?(Y/N)"''"File Na
me is "; FLASH PI/PI;g$+".C"+q$
: FLASH o: PAUSE o: IF INKEY$="
n" OR INKEY$="N" THEN  GO TO VA
L "24"
 22 LET n=PEEK VAL "61164"+VAL
"256"*PEEK VAL "61165"-VAL "315
ØØ"
 23 LET q$=CHR$ q: LET q=q+PI/P
I: PRINT #4: SAVE g$+".C"+q$COD
E VAL "315ØØ",n
 24 IF INKEY$<>"" THEN  GO TO V
AL "24"
 25 CLS : PRINT AT VAL "1Ø",NOT
 PI;"Press ENTER if you wish to
     clear the buffer"''"Otherw
ise press any other key": PAUSE
 o
 3Ø IF CODE INKEY$=VAL "13" THE
N  GO TO VAL "2Ø"
 32 GO SUB g: RANDOMIZE USR VAL
 "54Ø89"
 4Ø CLS : PRINT TAB 9;"LARKEN/M
TERM II"''''''''Insert disk con
taining program  to be loaded &
 press a key": PAUSE o: CLS : P
RINT #4: CAT "",: PRINT '''Full
 name of File (9 char$): ";: IN
PUT "Filename? "; LINE n$
 44 PRINT #4: LOAD n$CODE VAL "
315ØØ"
 48 INPUT "Enter the file lengt
h"'" (approx. if not known)";l:
 LET l=l+VAL "315ØØ": IF l>VAL
"53966" THEN  LET l=VAL "53966"

 5Ø POKE VAL "61164",l-256*(INT
 (l/256)): POKE VAL "61165",INT
 (l/256)
 55 CLS : GO TO VAL "32"
 6Ø CLS : INPUT "Buffer Name (6
 Chars) ";g$: GO TO VAL "1Ø"
 8Ø IF Z$="4" THEN  GO TO 48
 1ØØ CLS : PRINT AT VAL "17",NOT
 PI;"Enter full Disk File Name
": INPUT g$
 11Ø PRINT #4: PRINT g$
 115 GO TO 1Ø
 15Ø POKE VAL "23662",VAL "21":
POKE VAL "23663",NOT PI: RETURN
```

COMMENTS: An old "ZX'er can readily recognize another by his programming style which minimizes byte waste. <LET i=PI/PI> saves four bytes of memory as opposed to <LET i=1>. George Chambers started computing with a ZX-80 back in 1981, graduated to a ZX-81, and then to his TS-2068. He chartered the Toronto Timex Club in 1982 and has served in an executive role since. He is retired from the engineering dept of Bell Canada, a Cub Scout Leader, and a volenteer worker for the National Institute for the Blind. About the program: I have tried for a year to cut into MTERM II and make it dump data to disk. Now George has dude it where I failed. SALUTE! and Thanks George! Now you Oliger or Aerco users: The line numbers to edit and change to your disk syntax are: 6, 7, 27, 40, 44, and 100. Also, <RANDOMIZE USR 100> and <PRINT #4> are LARKEN SYNTAX and each use must be deleted for other disk systems or cassette. (Ed.)

# TS·2068 UP·DATE
# the user's NEWS

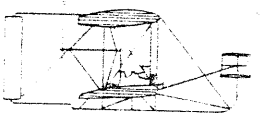
*Take it to the Limit*

A quarterly Magazine for the users




# TELECOMPUTING

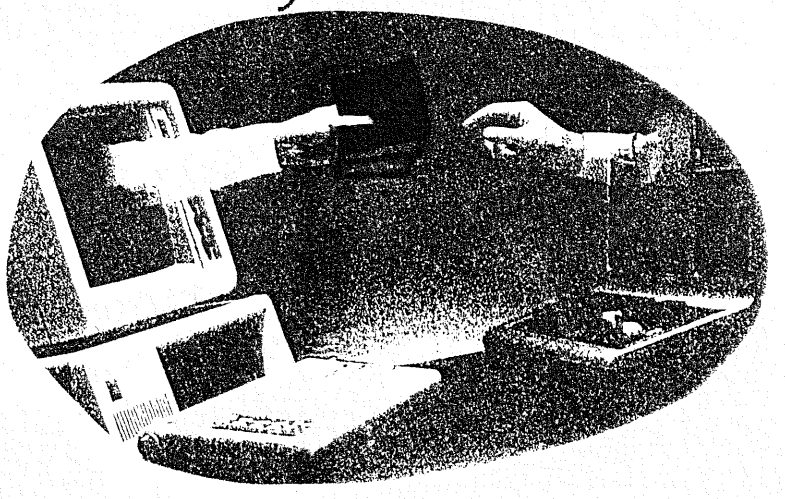
## The Ragged Edge



Wright Flyer

ERA Of Telecom 1988



TS-2068 UP-DATE
1317 Stratford Ave, Panama City, FL 32404
(904 871 4513)

COMMUNICATING WITH COMPUTERS

Progress will be the break through during this decade.

## *EDUCATION & REFERENCE (GO REFERENCE)*

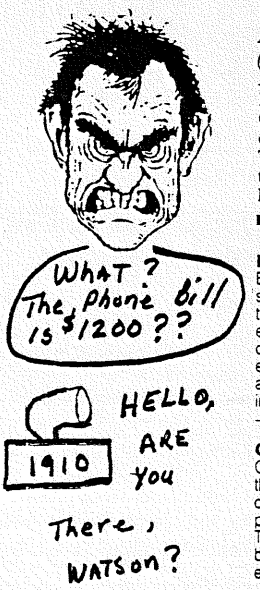
CompuServe provides educational resources for students of all ages and educators at all levels.

**For Professionals**
Access a variety of curriculum planning resources. Consult profiles of publishers of specialized educational materials. Join other educators in specialized Forums.

**Reference and Research Tools**
Enjoy instant access to an electronic version of Grolier's Academic American Encyclopedia. Plus, CompuServe's IQuest lets you access nearly 700 other databases from business, commerce, industry and academia. College Press Service keeps you current with Education News Services.

**Career Planning and Guidance**
The College Board and Peterson's College Databases help choose a college, prepare for the SAT, plan an adult student's return to academia, and evaluate strategies for securing financial aid.

**Support Services For Special Education**
A resource for those with handicaps, as well as those who assist, train, educate or employ the learning disabled and the vision, hearing or mobility impaired.




## *FINANCIAL TRANSACTION SERVICES (GO BANKING)*

Forward-thinking full service banks and brokers use CompuServe as an electronic gateway for out-of-town or out-of-state customers, local depositors, and investors. These services cater to consumers who prefer to serve themselves, electronically. No lines. No crowds. No traffic. No distractions.

**Banking**
Banks accross the country offer online financial and information systems where consumers and businesses can review transactions, transfer funds, pay bills, compare current interest rates and even exchange electronic mail with bank officers. Millions of people will do their banking this way in the future. You can start now. Take more effective control over personal finances. Save time and money. Take advantage of important new technology. Be one of the first to bank in an all new way.

**Quick & Reilly**
Quick Way brokerage services from Quick & Reilly Inc., the nation's third largest discount brokerage, include: online purchase and sale of securities twenty-four hours a day, current stock and option prices, portfolio management and automatic tax record keeping. Transactions are executed immediately when entered during business hours and at the start of the next business day when entered evenings and on weekends.

**Max Ule**
Tickerscreen is a financial information service of the discount brokerage Max Ule & Co. It offers closing NYSE prices, commission comparisons, and direct order entry of buy and sell orders for clients of Max Ule.

**Unified Management Corporation**
Liquid Green is a family of no-load mutual funds provided by Unified Management Corporation. Online access allows you to manage your account activities at your convenience, shifting cash reserves in and out of funds to take advantage of attractive money market interest rates.

# Computer, Telecom Gear Makers See Revenues Doubling By 1996

WASHINGTON — U.S. computer, business and telecommunications equipment industries expect revenues to nearly double to three quarters of a trillion dollars by 1996.

The projections amount to an average annual compound growth rate of 9.2 percent for computer and business equipment and 5.9 percent for telecommunications, said a recent Computer and Business Equipment Manufacturers Association (CBEMA) marketing analysis. Annual growth rates for the two industry segments were 13.9 percent and 9.3 percent, respectively, during the past 10 years.

In the shorter term, combined industry revenues are projected to be $384.2 billion this year and grow by 8.9 percent to $418.4 billion in 1988. Revenues are expected to grow by about 7.1 percent this year.

Computer and business equipment industry revenues are expected to advance 9.7 percent, from $198.2 billion in 1987 to $217.4 billion in 1988. An estimated increase in telecommunications industry revenues from $186 billion this year to $201 billion in 1988 amounts to an 8 percent gain.

"The future of the computer and business equipment industry continues to look bright. The accelerating shortage, due to a shift in population demographics, of entry level office workers and data processing personnel, will favor increased automation in factories and offices," the report said.

"Continuing advances in technology, declining costs and consumer education will promote automation and maintain the high rates of real growth experienced in the past," it added.

The predictions are based on trends that existed before the October stock market crash, but Oliver Smoot, acting CBEMA president, said there is no evidence that unstable stock prices will significantly affect the industry in 1988.

Revenues derived from sales of computers and related equipment are projected to grow from an estimated $116.6 billion this year to $128.7 billion in 1987. Revenues from software and services are expected to increase to $53.3 billion.

**PROJECTED REVENUES U.S. COMPUTER, BUSINESS AND TELECOMMUNICATIONS EQUIPMENT INDUSTRY**

($ Billions)

| | Computer & Business Equipment | Telecommunications | Total |
|---|---|---|---|
| 1986 | 183.65 | 174.92 | 358.57 |
| 1987 | 198.20 | 186.00 | 384.20 |
| 1988 | 217.44 | 200.98 | 418.42 |
| 1989 | 240.40 | 213.41 | 453.81 |
| 1990 | 256.69 | 223.83 | 480.52 |
| 1991 | 281.82 | 237.22 | 519.04 |
| 1996 | 443.37 | 311.48 | 754.85 |

Source: Computer & Business Equipment Manufacturers Association (CBEMA)

# Cable Will Hook Up 3 Countries

LONDON — Mercury Communications Ltd., the British independent telecommunications operator, has reached an agreement with France Telecom International and Companhia Portuguesa Radio Marconi (CPRM), under which an optical fiber undersea cable system will be laid to link Britain, France and Portugal.

France Telecom International is a marketing venture by the Directorate General of Telecommunications, the state-owned operating organization.

**Where It'll Go**

The cables will terminate on the south coast of England, in the north of France and near Lisbon, Portugal. The direct England-France cable will be a multiple fiber unrepeatered link over 90 miles long. It will be one of the longest such links in the world.

The link connecting Portugal will be a repeatered system operating at a capacity of 565 Mbit/s per fiber pair, one of the largest-capacity submarine links in service.

Those first telephone words "Hello, are you there Watson?" were spoken shortly after the turn of this century. About the same time Wilbur and Orville Wright were assembling bicycle parts to make a flying machine. That is about the stage of Telecomputing as we enter year 1988. At least 500 millionaires will make their fortunes in telecomuting within the next 10 years as progress overcomes existing communications monopolies.