UPDATE

JULY 1989

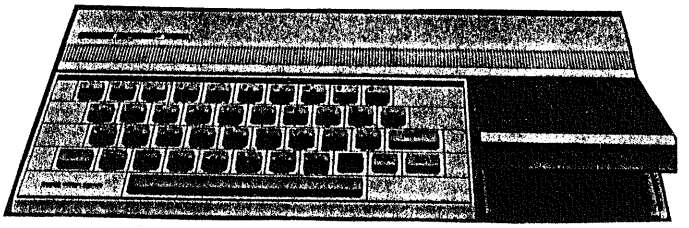
1944

ComputerSystems

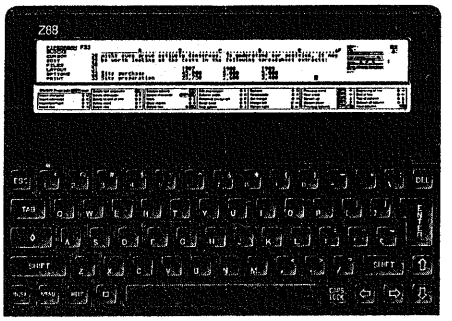
Take it to the Limit

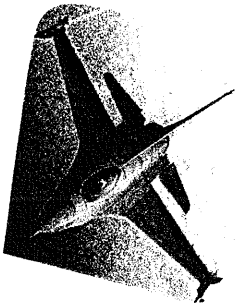
A quarterly Magazine for the users

The TS-2068 Workhorse
-best 8 bit Computer ever made!

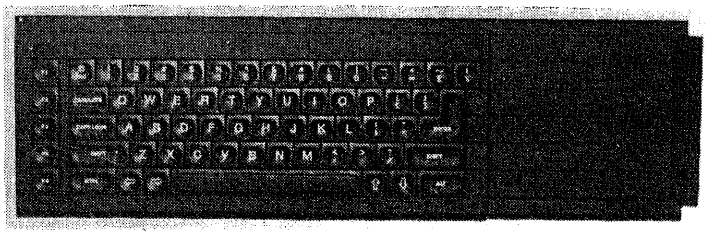


The Sinclair Z88
-revolutionary Lap-Top Computer!





The Sinclair QL
-a 32 bit CPU Powerhouse!



UPDATE Magazine
1317 Stratford Ave.
Panama City, FL 32404

NEWS AND NEW PRODUCTS

This is the first "USER GROUP DIGEST" page. UP-DATE invites each Timex-Sinclair User Group to send in a one page narrative about their organization. CATS is the first and it should serve as a guide. So come on you Club Secretaries, tell us about yourself.

TIMEX-SINCLAIR USER GROUP DIGEST

C.A.T.S. The Capital Area Timex Sinclair User Group

We are the Capital Area Timex Sinclair (CATS) Users Group - located in the nation's capital, Washington, DC. We are a not-for-profit organization and have been around since there were ads in Popular Science for ZX-81 kits. We have about 90 members - about a third of them are active users of TS-2068's and more than half have TS-2068's. Most of our members are from the the local area but we have dedicated members who come from as far away as Wilmington, DE and Harrisburg, PA to attend our meetings. We meet on the second Saturday of every month in the New Carrolton Public Library, New Carrollton, MD just inside the Capital Beltway (I-95). We usually have hardware sessions covering repairs and modifications starting at 11 a.m. and then the general meeting at 2 p.m. Each meeting usually has a specific topic or theme.

We provide many services to our members for annual dues of $18. Some of these are:

- Monthly 12-page newsletter (available also to non-members for a $12 annual subscription)
- Group library of software (ZX's and TS-2068's) available at a nominal fee (i.e., the cost of the tape)
- Group buys of hardware and software to get volume discounts (e.g., TS-2068 SCLD's, non-volatile RAM boards, disk controllers, disk drives and cases, printer interfaces)
- Hardware sessions with expert technical guidance where we repair or modify hardware, and build items that come in kit format
- Software sessions as part of our general meetings where people present software that the have developed or used.

In addition, we have many members who still actively use their ZX's and TS-1000's In addition, we have a growing number of QL users and even a few Z-88 users. We actively try to represent the entire spectrum (pardon the pun) of Timex and Sinclair users.

Our Executive Board is comprised of:

President	Tom Bent
1st Vice President	Hank Dickson
2nd Vice President	George Rey
Secretary	Mike Warmick
Treasurer	Ruth Fegley
Members-at-Large	
	Bill Barnhart
	Bob Curnutt
	Mark Fisher
Newsletter editor	Vernon Smith

For more information or a sample of our newsletter, please feel free to contact Vernon Smith at (703) 978-1835, or c/o CATS, P.O. Box 467, Fairfax Station, VA 22039. If reading this has made you want to join up without waiting another minute, send your name and address along with a check for $18 ($12 for newsletter subscription only) made payable to "CATS" to the above address.

The Last Issue of your Subscription

Do you want another Year?

UP-DATE began last October with 40 subscribers and it has been a tough climb to the present 220. A few subscriptions are still trickling in. The number of TS-2068 users is very small, smaller than most realize. Analysis by UP-DATE leads to the conclusion that there are fewer than 800 active users of the TS-2068. Further, it seems that there are fewer than 500 "serious" TS-2068 users who have already, or who are interested in developing their TS-2068s beyond its original configuration. In order to assure publication support for such a small group there must be a feeling of "family" and the family must be mutually supportive. I believe that UP-DATE has been successful in promoting the TS-2068 family concept. Several outstanding writers and programmers have joined in to share their knowledge out of their desires to help.

The TS-2068 family of users need continuing publication support. UP-DATE will continue to provide the support IF there are sufficient numbers who want to subscribe to a Quarterly magazine of this size. There is one business factor which is missing in a publication about ORPHAN computers. The missing factor is ADVERTISING, which all profit motovated publications depend upon. Ordinarily a magazine that has the article content of UP-DATE would have about 100 pages and about 60 pages would be advertising. The publisher would be making enough to send the kids to Harvard or Oxford. Without such advertising income, which really isn't possible to achieve by a TS publication, the subscriptions must pay the costs. This is true for all of our TS-2068 and QL publications.

The real reason for the decision to "hang on" is the attitude of the subscribers to UP-DATE as expressed in mail received and numerous phone calls. All have been full of encouragement and want UP-DATE to continue. Most who are not owners of the Sinclair QL are tolerant of the idea of sharing UP-DATE with the QL user group, just so long as the coverage of the TS-2068 is not diminished. The subscribers who are QL owners are delighted. In a way, the lack of advertising is a PLUS for the subscribers, because UP-DATE can be (and is) completely dedicated to the interests of the subscribers and not influenced by a syndrome of catering to the paying advertisers. Never the less, UP-DATE and the other ZX-TS publications are hanging on by their teeth and their survival is tenuous, completely dependant upon a high renewal of subscriptions.

So, here is the plan and the promise. UPDATE will hang in there and begin another year of issues while asking for subscription renewals. The article content of UP-DATE will be at least 30 pages of TS-2068 specific coverage. The Sinclair QL and the Z88 will be covered in extra pages and the magazine will grow with the QL and Z88 coverage. The October issue will start the next year. The future of UP-DATE depends upon most of the present subscribers renewing subscriptions. If there is an insufficient number of renewals, then you will get your October issue along with a 3/4 refund, and your editor will begin to devote more time to other hobbies. UP-DATE cannot start over at the half way point to try to build to the present level. Judging from the attitudes of the subscribers, such a back step will not occur.

The subscription goal of UP-DATE has not quite met expectation. But with the present FULL SUPPORT of the three current Disk Systems, more Larken DOS and Aerco FD-68 DOS users should be joining the UP-DATE family when the word gets around that these disk systems ARE supported. This July issue should go a long way toward proving that point. A lot of money has been spent to gain a fully capable TS-2068 Lab, including all current DOS systems and all disk formats. This full capability allows full service to the subscribers. A similiar well equipped Sinclair QL Lab is in operation. The programs and utilities given in UP-DATE are all fully tested before publishing, right here at the UP-DATE Lab. From now on the programs and utilities will be given in three DOS formats, Oliger DOS, Larken DOS, and Aerco FD-68 DOS (when pratical).

While the sales of new disk drive controllers have peaked, there are still a number of users who want to upgrade to disk capability. We will continue to seek more TS-2068 user subscribers who want to "take it to the limit". As UPDATE phases into supporting the Sinclair QL, we will see a repeat of the past, where most programs presented will have "dual application". Although the TS-2068 and the QL are different computers, most programs that operate in one can be changed in syntax to operate in the other. This will provide some fresh new conversion programming for both computer users. We TS-2068 users need such an infusion of new programming, and most of our good programs can be carried into the QL, maybe expanded. So, let us join forces with the QL users and keep on truckin, but faster.

A few personal notes: As I get deeper into the operation of the three active Disk systems for the TS-2068 I am struck with the enoromous capability. I guess that for the money spent on the TS-2068 I could be operating a fully exanded IBM AT system with two floppies and a 60 Mega byte hard drive. I have operated the excellent IBM computers at the College. The concept of that system is different. Everything is planned for you. Software is rigidly constructed for a specific range of functions, copy protected, and everything is costly. There is not the flexibility that one has with a fully developed TS-2068. I am finding that this comparison is also true when comparing the IBM and the Sinclair QL. One has more flexibility with our computers and much less expense of operation. And the overall capability is not much different.

This JULY UP-DATE Issue- New Programs and New Utilities

Putting this issue together involved taking the BUDGET program (by Bob Mitchell) and doing a re-write of the supporting BUDGET.B1 loader and constructing the Disk Menus for Oliger DOS and FD-68 DOS. Then each DOS version had to be extensively tested. Then the same process was required for the other programs given. During this process some new utilities were developed and will be given. Deficiencies were discovered in the Aerco printer driver code, which many use with TS-2068 printing software. I will give you the procedures that I used to work around these built in glitches. Then next issue we will go further into PRINTER CODES and how to do a better job with them. Jack Dohany's early "RALOADER" program was used to work in a relocatable code table. I suspect that Jack has some up-dates to the Aerco Printer handler code and will see if we can bring them to you.

INIT: This is a utility program to quickly get a printer port code loaded, initialized and ready to print with a Centronics printer. Given are two INIT utilities for using the Aerco Printer Port code with the Aerco FD-68, and with the Larken LKDOS. Another INIT is given for setting up the OLIGER printer driver that is contained in the SAFE EPROM. Many Larken LKDOS users use the Aerco prcode and its set up is kinda tricky. This prcode is the one on cassette as supplied with the Aerco Centronics interface. The INIT utility for the AERCO FD-68 is reloactable (to a starting address of your choice).

Perhaps in a future issue a similiar utility for the Oliger prcode that accompanies the Oliger Centronics interface can be presented. Mr. Oliger's driver code requires much less RAM and is error free, while the Aerco prcode has a "first LPRINT" glitch that is difficult to work around. Also the Aerco Prcode lacks support of TAB and printer status check. Incidentally, the Aerco and Oliger codes work equally as well with either centronics interface. I will ask Mr. Oliger to let us use his prcode, as it is much better. The Oliger Disk users do not have to worry with loading a prcode as the SAFE EPROM contains the code and is initialized with <LET /p=o>.

Help with the Aerco FD-68

A problem exists in using the Aerco FD-68 DOS. Since there is no Character Array SAVE LOAD, the data files must be either CODE files or a complete Variable file save load. When a vars file is loaded the existing basic program vars are replaced. This means that I could not do what I'm now doing; ie, I could not SAVE several data files as VARS files, and re-load them sequentially to print a continuous document. This is because the printing variables (line length, print style, etc) would be wiped out with the LOAD of a vars file. The INIT utility for the Aerco FD-68 takes care of this by providing 67 extra bytes of code ABOVE the 800 byte prcode, for storage of variables. So, the solution is to "hide the essential vars" above the prcode before loading in a new vars data file for printing.

A small counter from 1 to 67, using a READ DATA within the counter can be used to stash these vars before a vars file is booted in. The DATA line can contain the letter designations of the essential vars to be preserved. Then AFTER the new Vars file (a data file) has been loaded, the essential vars (numbers) can be PEEKed out and re-assigned to the essential vars. Then the basic program can resume with identical printing and set-up variables. This is the way the Aerco FD-68 version of Smart Text conducts automatic printing of multiple data files booted in from disk. This requires the variables to be poked to the (appended) prcode just before the new data file (a vars file) is loaded.

A Code File Printer

MS-TAS: I donate this new utility to Public Domain. MS-TAS is a utility to peek and print out Mscript and Tasword CODE files "without need to have either of the softwares" loaded. Some of us do not use either Mscript or Tasword, but receive data files saved with these two popular word processors. I got tired of wondering "what the heck's in thar" and MS-TAS is the result. It CATs the disk directory, prompts to input a code file name, then LOADS the code and begins printing with a Centronics printer. All you need to input is printing line length, the STARTING code address and the BYTE LENGTH of the code. For Aerco and Larken DOS, the printer code is also booted in and initialized.

Using STR$ to Modify Disk Program Titles

Here's a little quib for Disk Users: The senario is, several data files have been saved with a name followed by a number, as SAVE /"page 1"DATA h$() through "page 30". Now Smart Text can print all 30 of these file in one continuous document without pausing for breath. BUT, the titles are not accepted. ST wants just numbers as titles for auto printing from disk. This little quib will fix things. << FOR n=1 TO 30: RESTORE /"page "+ STR$ nDATA $ TO STR$ n: NEXT n >>. CAT the directory and the 30 "page num" files have been changed to just numbers 1 - 30 for printing. Hope I didn't lose you with that. All we did was to re-name 30 files in disk within about 30 seconds. A numeric variable, such as n, is added to a literal "string of characters" by using STR$ n.

Similiar concats of disk titles for SAVE LOAD RESTORE ERASE CAT MOVE are practical. Say a variable "n" needs to be appended to a title "page" to SAVE with LARKEN DOS. << LET a$="page "+ STR$ n+".a$": PRINT #4: SAVE a$ >> ort to do it. Here's an easier one for Aerco FD-68 to save PROGRAMS under a "program number" instead of a SAVE OVER. The title is "mscript" and the auto RUN line is 10. Your save line would be: << OUT 244,1: MOVE "mscript.bas",10 >>. We start with a variable like <<LET ss=1>> then change the SAVE line to: << OUT 244,1: LET a$="""mscript "+ STR$ ss+".bas""": MOVE "a$",10: LET ss=ss+1 >>. The first save is "mscript 1.bas". The next save will be "mscript 2.bas". To get to be friends with STR$, do this: <LET a=1> <PRINT STR$ a>. <LET b$="test"> <LET c$=b$+" "+ STR$ a> <PRINT c$>. See? STR$ is pretty friendly after all!

A Print Wheel Problem

The print character style of the previous issue looked pretty good to me, until a subscriber called about having trouble deciphering listings. The problem is one of discerning whether a character is a "i", a "l" (el), or a "1" (one). This has always been a problem with type and some typewriters use the literal "l" (el) for the number "1" (one). I looked around town and found a print wheel that seems to give better identification of the "i" (eye), but alas, the letter "l" (el) and the number "1" (one) looks the same. Can you tell the difference- llllll? I'll try to solve this ident problem by using upper case "L" in listings, while looking for a print wheel that has better characters.

A TOS (Zebra) Disk Newsletter

TOS DISK (Zebra FDD): UPDATE receives a nice newsletter "FDD EXPRESS" done by Mr. Ron Havlen, 4307 Chambers Road, Horseheads, NY 14845. FDD Express represents a resumption of the (defunct) TOPS Newsletter. Ron is a subscriber to UP-DATE and I invited him to contribute regular articles in support of the TOS DISK. Instead, Ron gave permission to re-print anything found in his newsletter. I will print excerpts of FDD EXPRESS from time to time. This issue givesone sample two page isue for TOS disk users to evaluate. Ron promises to get heavy into CP/M and some of that should be interesting to the Aerco FD-68 users. In the meantime, those who have the Zebra FDD (TOS) disk really should have this newsletter. (Look in

the Orphan Disk section)

A Save Nothing File

"Cclear"CODE: Did you know that once a CODE program has been loaded, the durn thing cannot be gotten rid of? CLEAR only clears the vars. NEW and the code is still stuck in thar. Set RAM TOP again, and the code is still there. You may not want to lose your Basic program by OFF or RANDOMIZE USR 0, just to clear the code out. There is a easy way to do this, so easy that its one o them "Why dint I thunk of it befores". Turn ON the computer and SAVE "Cclear"CODE 30000,65535. Thats a SAVE of 35535 bytes of "zeroes". Later when you want to clear out a code program, just LOAD "Cclear"CODE. You can make several Cclear saves to clear out smaller segments of memory. SAVE "Cclear2"CODE 65368,167 will clear just the UDG area (when reloaded).

Check the VARS Area

"VARS_DIS" is a utility to MERGE to any Basic program to give you a disassembly of the entire VARS area. Two additional bits of often needed information are presented, system variable VARS (the starting address of the vars area) and ELINE (top byte of vars+1). Then to get P-RAMT (the last byte of the BASIC PROGRAM), just substract 1 from the given "VARS" address. But both VARS and P-RAMT will be moved UPWARD by the exact byte length of the utility VARS_DIS, when the utility is merged. Admitedly, this one dont set the barn on fire, but I needed it once er it wont a been hatched. And, it mought jest turn you on to doin more peekin and pokin, which is the way to learn it durn it.

A BREAK THROUGH DOS for the TS-2068

Take DOS to The LIMIT - with MAX DOS

"MAX DOS": Now I've saved these last two to the last. A BREAK THROUGH in DISK CAPABILITY, and indeed it is, is the STACKING of the two most versatile disk operating systems, Oliger DOS and Larken DOS. I'm gonna coin a name for it because UP-DATE made the discovery. How bout "MAX DOS", which sez mor'n IBM's "MS DOS". Theirs sounds like a "mess" whil'st our'n sounds like the mostest. Gotta get a leg up, right? Well, I'll declare, swear, affirm, bet, and guarantee that this "MAX DOS" is the best disk operating system that has come down the pike for ANY personal computer! "TWO WELL DESIGNED DISK SYSTEMS THAT OPERATE TOGETHER AS ONE"!! AND NEITHER TAKES UP ANY OF THE TS-2068 RAM!! I dont want to take the thunder out of my own article, so will just call your attention to it.

EXTRA MEMORY

EXTRA MEMORY: Gobs of it! A TS-2068 that has twenty times the maximum capacity of a IBM XT! Do pictures and proof schematics lie? An interesting article by Mr. Charles Bothner sets the technical basis for our cottage industry to provide more memory expansion hardware.

Did you modify the TS-1000 as per the January UP-DATE (Dock Bank Memory for 20 cents)? Well, I did and the durn thing didn't work right. So, I sent the board to John Oliger with a limit placed on repair cost. It came back and now it works ($15.00 repair cost). Seems that one little detail was left out of the modification instructions. Gotta remove a feed through on the board. Now the "twin 32K dock bank Ram" sits in my expansion port most of the time, ready to take basic programs and utilities for instant operation. There is a "kind of" multitasking capability here, where a dock bank utility can generate data in the vars area to be used with a basic program in the home bank. I think that I'll try adding a battery to it to make it non-volatile memory so that when the computer is turned OFF, the data will remain. If successful I'll tell you about it.

The New Oliger V2-52 EPROM

I recently received the latest Oliger DOS EPROM, which includes some real fine additions. Included now are the functions ERASE / (file name), RESTORE / (old file name to new file name), RESTORE /"new disk name", RESTORE /(used alone to re-initialize to default parameters), VERIFY /(file name) to check the file for errors, COPY/ copies screen to printer, MERGE /(program name), SAVE //(Save without warning that file name may be present). Of particular note is that the ERASE/ function "completely re-organizes the disk contents and recovers the disk space from an ERASED file".

The new SAFE CATALOG has new data that I find to be very valuable. The starting line of each Basic program is given, and for CODE files the starting address and byte length are given. Also, the finished SAFE manual is now available. This (I believe) is the final coup de ma tre of the Oliger SAFE DOS. Any further EPROM versions will be for corrections (if un-forseen problems are found). A CAT of the directory of the April Issue disk is given with the new V2-52 format. It serves two purposes, to show the nice catalog, and to show the contents of the April issue disk. (The April Issue Disk has the same programs for Larken DOS and Aerco FD-68 DOS).

JLO SAFE V2.52 1988, J. Oliger

DISK NAME: APRIL UPDATE
FORMATTED @40 TRACKS, 0 SIDE(S)
CAPACITY: 79 CYLS/395K BYTES
FREE: 11 CYLS/55K BYTES

FILENAME	TYPE	CYLS	SIZE	START
code.Cl	BYTES	1	1826	63417
DOSDEX MGR	BASIC	2	6007	2
LOAD	BASIC	1	87	10
BUDGET.Bl	BASIC	1	986	60
Menu.Bl	BASIC	1	702	10
INIT.Bl	BASIC	1	583	9980
budget.CJ	BYTES	6	25923	32000
spread.CJ	BYTES	1	3904	58400
budget.BJ	BASIC	5	24554	10
spread.BJ	BASIC	1	2855	8000
BRIEF.Bl	BASIC	2	5445	5
MS-TAS.Bl	BASIC	1	924	8000
Ll MCLOAD	BASIC	1	487	1
PEEK VARS	BASIC	1	196	
budget.CY	BYTES	6	25923	32000
SEARCH	BASIC	1	4978	1
FILE_DEX	BASIC	1	1163	1
MARK-MOVE	BASIC	1	2092	1
VERI_DISK	BASIC	1	855	1
Chopin	BASIC	2	7584	220
Btls	BASIC	3	14780	9900
Mzrt	BASIC	2	7674	1
Spell	BASIC	2	7411	100
Ohm's Law	BASIC	1	1257	100
Multiply	BASIC	2	6038	10
Type Style	BASIC	1	2055	4608
Piano	BASIC	1	4747	10
budgetD	BYTES	1	652	64716
bujE.Cl	BYTES	4	17642	47047
bujJb.CT	BYTES	2	8320	33280
bujJs.CT	BYTES	1	2048	33280
help.CT	BYTES	1	3968	33280
bujJ3.CT	BYTES	3	11200	33280
bujJx.CT	BYTES	1	2944	33280
OLI-MS-TAS	BASIC	1	1512	8990
OLI CODE	BASIC	1	1841	
prod	BYTES	1	256	23296
Cclear	BYTES	2	5367	60000
Super MS-T	BASIC	1	3060	8972

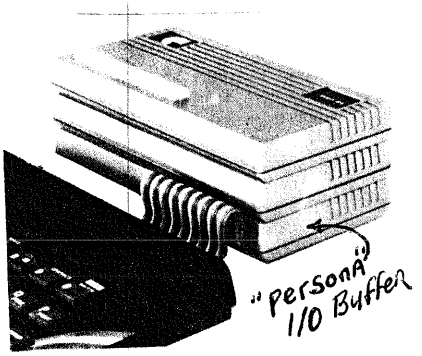
TOTAL FILES: 39

LETTERS

Dear Bill, Please send the Jan 88 disk. Check inclosed. How about asking the readers if anyone can help me out with schematics or other info concerning the TS-1000 "BASICARE" modules that I have.

The units are: a connecting module, a 64K RAM, two 16K RAMs, a DROM expandable to 8K, and a printer interface. These should be much less expensive to convert to TS-2068 than to buy new ones.

Ray Norton, PO Box 331, Bonner Springs, KS 66012-0331



Ray, this reply wont be in specific detail but may help. Someone out there may have schematics but I doubt that the maker released their tech data. YES! You can make modifications to any TS-1000 memory device to use with the TS-2068, and it is worth the effort to try. Generally, the memory chips of that era were 2 and 8K chips. It would not be cost effective to replace these old chips, but if they are still good, then the 64K RAM board would be worth salvaging.

The BASICARE system is the neatest expansion package that I have ever seen for any computer. It consists of several "stackable" modules. As each module stacked on top of the one below, the I/O buss of each mated. The first module which connects to the computer is named "Persona" and is a I/O Buffer for the Z-80 CPU buss lines. Since all of the TS-1000 I/O buss lines are "pin compatable" with the TS-2068, the Persona" unit should work almost AS IS with the 2068. One more TS-2068 buss trace (MREQ) should be brought into the Persona unit as is shown in the diagram given in this issue about the Oliger 64K Ram. This MREQ line should be furnished to the 64K Ram that you modify. The rationale given in the January UPDATE article about modifying the Oli RAM should be followed.

Generally, the ICs used and the method for coding for memory addressing during the TS-1000 era was fairly standard, and it is likely that the 64K Ram that you have will have almost identical coding chips to those in the Oliger RAM. If not, then you will need to obtain data sheets for the chips used (Jamesco Electronics is a good source). Or call me and I'll likely have the data sheets that you need. The memory of the TS-1000 modules must be re-coded to the TS-2068 memory map.

If I were you I would set aside the modules that contain the printer interface and the DROM. The task of making these useful for the 2068 involves re-design of the circuits for another purpose. Instead, the more productive approach would be to re-code the 64K Ram for two 32K dock banks as per the Oliger Ram conversion. Then later, as the project develops, you may want to re-design these boards for another purpose.

The Basicare system offered two branches of modules for using one 64K RAM or (up to) 16 64K modules. The "second" stacked module, which could be left out, is named "Minimap". This module provided addressing for 16 "pages" of 64K RAM Modules, for a total of one mega byte of RAM. If you have this unit, and more RAM modules, then a study of Mr. Bothner's article in this issue could provide guidance for modification of your "minimap" module. Notice that Mr. Bothner starts his project with a I/O Buss BUFFER board built upon a Zebra Proto board. You already have 99% of that unit in your "Persona" module. Believe me, a I/O buffer is one BIG project to build. About half way through it takes on the appearance of a "Afro head".

So, you really have two options. You can make your Basicare modules into "Switchable Dock Bank RAM" as we do with the Oliger Ram modification. Or, you can follow the lead given by Mr. Bothner to make a more elaborate system. UPDATE will present a follow on article that gives Mr. Bothner's corrected TS-2068 PROM. But as shown in his article, he already has several 64K banks in operation. Also note in his article that a auxillary power supply is a part of his system. Basicare also had a auxillary power supply module that was needed as more 64K Rams were stacked.

The advantage that you start with is having neatly packaged modules which can be modified to become a TS-2068 system. I can visualize them pretty silver modules stacked behind the TS-2068 providing a substantial increase in memory capacity. Go fer it!

QUIB ? (SPACE FILL)

Non-exclusive Statements

You may not always want parenthetical statements in a variable re-valuation to be mutually exclusive.

Consider a game where a player receives 10 points for hitting at least ten targets, and a bonus of 20 points for hitting exactly ten targets.

```
LET SCORE=SCORE+
  (10 AND H>9)+
  (20 AND H=10)
```

When H is ten, both expressions are true and SCORE is incremented by 30.

*** MAX DOS ***

A Study of DUAL DOS in the TS-2068

For several weeks UPDATE has been testing TWO SIMULTANEOUS Disk operating systems in the TS-2068. This is the Oliger SAFE DOS and the Larken DOS. The hardware hook-up is the FULL "two board" Oliger system and the Larken LKDOS cartridge in the Dock Port. Extensive operation with the combination has revealed an almost complete compatability between the two systems. The only incompatabilities are minor and will be discussed. For the the test two disk drives were used, but a single drive could have been used as well. Drive zero had a disk formatted as a Larken Disk, and drive 1 had a disk formatted as a Oliger disk.

In the beginning the Oliger V2-40 EPROM was being used. Actually the instructions provided with the Larken LKDOS cartridge calls for the removal of the Oliger "board A" which contains the SAFE EPROM. But I'm adventurous and I decided to keep both Oliger boards in and see what would happen. The system seemed to work O.K. Both the Oliger DOS and the Larken DOS operated normally. After about a week of operation I kinda got used to having DOUBLE the amount of EXTRA BASIC functions. All of the Oliger functions worked and so did all of the Larken functions. AND, it did not make any difference which DISK was used for actual SAVE and LOAD. In fact, ALL of the SYNTAX structure for BOTH systems can be used in program lines, or can be used in the direct mode. We will explore this a bit.

Oliger SAFE commands use the slant bar "/" as a designator. Example: "SAVE/ "title". The Larken commands use <PRINT# 4: SAVE "Title.B1">. The disk in use is selected by <LET/d=0> or <PRINT# 4: GO TO 1>. Now suppose a program line were: <100 SAVE/"file 1"DATA a$(): PRINT #4: SAVE "file 1.a$"DATA $()>. The TS-2068 accepts the syntax and the line enters OK. When the line executes, the first command results in the disk "automatically switching" to the Oliger disk and the data file is SAVED to the Oliger disk. Next, the Larken syntax results in an automatic switch to the Larken disk, and a SAVE of the same data file to the Larken disk. Both systems "remember" its own disk as elected by <LET/d=> for Oliger, and <PRINT#4:GO TO> for Larken.

Likewise, all other disk commands and all Extra Basic commands operate exactly as they would normally. The nice part is that all of these functions are additive, and operational as ONE BIG DOS. The Oliger FAST FOR counter works with LARKEN programs. The Larken Window functions work with a Oliger program. In fact, a program loses its idenity as a Oliger program or a Larken program. It can be saved or loaded in either disk format. One can coose which simply by using one DOS SAVE or LOAD command, or the other. Now, if I receive a disk from someone, I can load it if it is in either format. One can operate entirely with one disk format if he chooses, AND use All of the extra basic functions of BOTH Larken and Oliger systems. But there are strong advantages of using two disk formats.

Each of the two systems commands the Disk Drive through the Western Digital controller chip. There are certain Files Management functions which are more versatile when using the Oliger format, and others that give more capability when using the Larken Format. So the use of two diskettes does provide more options of disk files management. In effect, "MAX DOS" is controlling double the amount of disk storage- two diskettes under automatic controll. Now "MAX DOS" is my own coinage. I wanted something that would denote a BIG Disk Operating System, as opposed to "a choice of one or the other". Besides, I always like to twit the IBM users. Their "MS DOS" sounds like a "mess", while "MAX DOS" sounds like the "mostest".

Compatability with Existing Hardware: Both Oliger DOS and Larken DOS are compatable with the Spectrum ROM Switch. Both are compatable with the Zebra OS-64, but the LKDOS Cartridge takes up the Dock Port. The Larken literature discribes an additional IC that can be piggy-backed to accept either the OS-64 EPROM or the Spectrum ROM IC. UPDATE has not tested this modification. The SPDOS software on-Disk is compatable with both Larken and Oliger systems. The loading of SPDOS would give a third FORMAT and enable the use of RAMEX disk programs.

COMPATABILITY of EPROMS: UP-DATE received the new Oliger V2-52 EPROM and the system crashed when the LK cartridge was installed. A call was made to John Oliger who seemed confident that the conflict was caused by "two different brands" of EPROM used by the two systems. Apparantly the EPROM that has the "penthouse" on top cannot be used with the Larken Cartridge. I sent an old "flat top" EPROM to John for re-programming. When it was returned, the compatability was restored. Since then, all of this magazine issue pages have been prepared and printed with BOTH DOS systems installed. One non-compatability exists. Both Larken and Oliger systems have NMI functions. These functions clash and the NMI button cannot be used. To date, this is the only conflict found.

Conclusions: MAX DOS is a reliable combination of two very capable disk operating systems, and two excellent Tool Kit extensions of the Basic language. This combination operates as if the two systems had been designed as one large system. The tool kit extensions are not just Basic Commands, but are groups of functional routines. MAX DOS provides as many disk management functions as any DOS used by any computer. The extra tool kit functions greatly extends the job tasks that the computer can perform. These new functions are many and are unique. No other computer has them in their repertoire of functions. Some of the notable attributes of MAX DOS are listed below. Then a sample bit of programming will be given to conclude the report.

1. Disks prepared by either Larken DOS or Oliger DOS can be read by MAX DOS.

2. Disks may be prepared for use by others who have single Larken DOS or Oliger DOS.

3. The plethora of Extra Basic Tool Kit functions of Oliger DOS and Larken DOS are combined.

4. The Larken and Oliger Command Syntax for both Disk operation and Tool Kit Basic may be mixed in programming.

5. The system exercises direct conrol of two disk drives simultaneously. Switching is automatic.

6. The system makes use of the excellent Oliger

printer driver in EPROM. No need to load another printer driver.

7. No RAM memory is used by either Oliger or Larken DOS, thus MAX DOS does not use any RAM.

8. The TS-2040 printer is directly supported, simultaneously with a Centronics printer, or separately.

9. Direct low res Screen Copy is available by simple command.

10. No conflict with Spectrum ROM, or any Extra Memory device.

```
9000 BORDER 1: PAPER 1: INK 7: C
LS : LET /p=o: RANDOMIZE USR 100
: OPEN #4,"dd"
9030 PRINT AT 10,0;"Install Form
atted disks in:"'''Oliger Format
 in Drive 0"'''Larken Format in
Drive 1"''''Enter When Ready": P
AUSE 0: CLS
9040 PRINT AT 10,0;"SELECT:"''"<
1> To Move Screens TO LARKEN DIS
K"''"<2> To Move Screens TO OLIG
ER DISK": INPUT b: IF b<1 OR b>2
 THEN GO TO 9040
9050 PRINT AT 10,0;"Which way to
 Move?"''"<1> Oliger to Larken"'
'"<2> Larken to Oliger": INPUT c
: IF c<1 OR c>2 THEN GO TO 9050
9060 CLS : PRINT AT 10,0;"Instal
l Disk in Drives"''"Then ENTER":
 PAUSE 0: CLS
9070 INPUT "Number of Screen fil
es to move? ";m
9080 FOR n=1 TO m
9085 IF b=1 THEN LET /d=0: CAT :
 INPUT "Screen file Name?";a$: L
OAD /a$SCREEN$ : LET a$=a$+".C1"
: PRINT #4: GO TO 1: PRINT #4: S
AVE a$SCREEN$ : NEXT n
9090 IF b=2 THEN PRINT #4: GO TO
 1: PRINT #4: CAT "",: INPUT "Sc
reen file Name?";a$: PRINT #4: L
OAD a$SCREEN$ : LET /d=0: SAVE /
a$SCREEN$ : NEXT n
9100 STOP
```

CTM MAGAZINE QUITS

Chet Lambert, owner publisher of CTM Magazine, announced in the June issue of CTM that he is quitting publishing CTM magazine. I regret that CTM is finished and wish Chet and Tina well in whatever they decide to do. CTM was oriented toward the support of Amateur Radio Operators and has contributed eminently to the furtherance of PACKET Telecomunications. Chet really tried to develop support for the Timex-Sinclair users, by fulfilling the remaining subscrition obligations of TS Horizons when it quit publishing. Many TSU subscribers did not realize that Chet did this FREE.

A Monthly magazine requires a steady volume of industry advertising to support costs. The lack of advertising support from Amateur Radio suppliers reveals a steadily declining market for Amateur equipment. Being a licensed Amateur Radio Operator, I can express my belief that the decline in the market is caused by our own ARRL, which pursues a diehard policy designed to limit the licensing of new General Class operators and thus avoid growth. The result is an aging "old fogy club" wanting to guard their ranks and unwilling to up-date with modern equipment.

Telecomunications with RADIO and COMPUTERS will be the break through in the 1990's. Chet had a vision of this with PACKET. Unfortunately the doddering old fogy ARRL wants to keep on keying away nonsense with Morse Code and twenty year old Comm gear. The result is a stagnated radio industry and a lack of opportunity for a fresh new magazine like CTM to survive. So, Computerists must take up the challange and develop Telecomunications. I kinda got carried away, excuse me. Good Luck Chet and Tina! Stop by when you're down this way and I'll take you sailing.

Did you know:

'that data is moved to / from cassette at a rate of about 150 chrs per second?

'that data is moved to / from disk at a rate of about 10200 chrs per second?

'that disk is about 68 times faster than cassette?

that the ordinary jogger jogs at a speed of 4 mph

'that if you jogged as much faster than the ordinary jogger, as

disk data transfer is faster than cassette data transfer, then you would jog at a speed of 172 mph?

'that a flea can jump more than 20 feet?

'and his departure accellertion is more than 60 G's?

'and if you could jump as far, in comparison to size, you could jump from Miami to Detroit?

-three hops Coast to Coast, then three to Alaska, one to Attu, another to Japan, then 6 hops across Asia to Siberia, and 4 more to the East Coast- around the World in 18 leaps!

Airplanes would be obsolete if only we humans had comparative hopping abilities as fleas!

"DOS-64" A Review By Pete Fischer

"Phoenix Pete" Fischer is THE GURU of TS-2068 Telecomputing, having written a book on the subject. Thanks "P.P" for this nice input. Your expertise is always welcome in the pages of UPDATE!

ABSTRACT: This is a review of MakeDOS-64, a machine code system which allows use of the OS-64 cartridge code on an AERCO Disk Drive without loss of Aerco DD commands, and with access to the full 256K ram via bank switching. (review by Pete Fischer)

Short Description of DOS-64

DOS-64, like the OS-64 cartridge itself, sits in the Dock Bank. However, it DOES take away a small portion of ram in the Home Bank. It features the full use of AERCO Disk commands, just like the AERCO version of OS-64, but that version gives you only 17K of BASIC area. This version keeps the normal 33K that OS-64 would leave you. It also includes a number of other features not found in either the Aerco DOS nor the OS-64. Read on!

Description of the OS-64 Cartridge

Before I go into MakeDOS-64 itself, though, let me describe the OS-64. It's not as cut and dried as it first seemed. True, you just plug it into the cartridge port & turn on the machine. But it's not QUITE that simple. For one thing, you've lost nearly 5K of free memory at the bottom of BASIC. This is never discussed in the Zebra docs. Thus, loading any program longer than 33+K will crash the 2068. Also, you've lost some of the commands in BASIC. FLASH is useless, since it required Display File 2 in order to work. Once you set either PAPER or INK, the other is set to it's compliment. (i.e. PAPER 4 [green] will set the INK to 2 [red]). This change effects all the ink on the page- so that it's impossible to have more than one color of INK on any page. Also, you can't change the BORDER to a different color than the PAPER. You also lose BRIGHT and INVERSE.

Jeff Street, the Zebra programmer who wrote OS-64, DID manage to include printer drivers for 4 centronics interfaces, namely, the Aerco/Oliger, Tasman-B, Tasman-C and A&J. The 2040 is NOT supported. These are meant to drive Epson- compatible printers. But you can also load your own custom driver, and the docs tell you where to load it and how to call it (can be either centronics or serial).

Since screen display uses display file 2, a normal SCREEN$ will not work to SAVE or LOAD. You must SAVE/LOAD two seperate CODE files in order to deal with a SCREEN in 64 col mode.

At first I thought a BASIC program had to be specially written for OS-64. But then I thought, why don't I just LOAD in a program written for 32 col.,and see what happens?

So I did, and to my surprise and delight, most things worked. There WERE problems; such as when a program used 2 or more INKs in the same menu. The screen would flash through all the different colors as it printed out the screen, finally ending up with green ink and red paper which was unreadable even on my RGB. So I had to fix all those (this can take some time on a long program).

Any text which was centered in 32 col. is now to the left, and depending on how the PRINT command was used, some lines of text may stop at 32 col, while others go on to 64 col. So these PRINT statements may need some editing. But once you've converted a program, it's done. You never have to do it again.

So what is the advantage of 64 col.? The first attraction to me was the capacity to run Zterm-64. But the small buffer and bugs made that program less appealing.

It lets you put twice as many columns on the screen, with the same number of rows. This doubles the information on a page. This is particularly useful when you're using software that stores and displays lots of data, such as word processing, spreadsheet, accounting, database or telecommunications. It can make much better menus, as I always seem to run out of screen before I run out of options. You will see less of the "scroll?" prompts on your screen. Although I can't help but wish we had more rows as well as more columns...

One last note on the OS-64, if you have the JLO disk drive, you need to have OS-64 ver 1.72 in order to use the NMI "snapshot" save.

The Difference Between OS-64 & DOS-64

The main difference is complete compatibility with the Aerco system without any loss of free ram. Also of critical interest is the access to the full 256K rambank via bank switching. These banks can be had by a simple OUT command. Oh, what wailing and gnashing of teeth could have been saved if Aerco had sold this software with their 256K upgrade! (NOTE: In his recent book on advanced M.Code Programming on the 2068, Dr.Lloyd Dreger discusses this scheme of access to the 256K. He gives credit to David Hill. Keith Watson wrote the code originally, and gave it to David to publish in his newsletter. This is just a misunderstanding on Dr. Dreger's part).

Another feature that Keith worked in is a program by Jack Dohaney called KEYWORDS. For those of you who are unfamiliar (as I was) this program allows you to enter the TIMEX BASIC keywords, one letter at a time. This is useful to those who have custom keyboards without legends. You can toggle this feature on and off with a PRINT USR statement.

There are two additional character sets, BOLD and IBM PC style. These improve readability on marginal monitors.

You can alter the total maximum number of directories on a disk. We always complained that 31 was too few. But now that the number is doubled for DD diska and tripled for QD disks, it seems to take a significantly longer time for the system to search a directory. This gives you the best of both worlds.

While it's true you can "customize" the

OS-64 cartridge for a large number of options, you do so using POKE statements which must be re-entered everytime you turn on the computer. Since Make-Dos comes off a disk, however, you can customize it once and for all (and easily re-configure it later). Then, you save the customized version and DOS-64 "Wakes-up" in any setup you want!

This "customizing" is done by a menu-driven program that's quite painless and easy to follow. This is MakeDOS. All four of the original printer drivers are still there. You can choose one of those, or your own custom driver as permanently enabled. You can also choose various printer configurations such as line length, and print density. There is also an option to increase the total of numbered lines in BASIC from 9999 to 16383.

THE GOOD POINTS

1. Permanent 64 Col. display with completely normal DOS operation.
2. Full access to the 256K ram upgrade.
3. All the original printer drivers of OS-64 are there and any one can be permanently enabled.
4. There are 2 new character sets.
5. Jack Dohaney's KEYWORDS program is available for use.
6. Most 32 column BASIC programs are compatible, or require minor modification.

THE BAD POINTS

1. I miss a hardcopy manual for this product. You don't need one much, but it's good to have.
2. Some of the explanations given on the help screens are a bit technical. For technical users, they will be welcome. But others may be thrown.
3. You lose about 5K of BASIC (which you also lose with OS-64)

"I was just writing something for my own uses." That's how Keith described it to me in his self-effacing manner. He ended up spending several hundred hours and many calls to AERCO to get the straight story on the 256K. How many really great software packages started out as someone writing something for their own use.

This software is available from RMG, who has gone to great lengths to support AERCO users. It costs $15 +$2 P&H. Or from Keith. Brilliant programmers are worth supporting. See pg 14 of January UPDATE for addresses of Suppliers.

The Mscript File Printer that printed the above Article

UPDATE Addendum: The Zebra OS-64 cartridge provides a nice 64 column display, yet when a menu or information display does not require such packing on screen, the regular 32 character display provides more comfortable reading. The OS-64 does not LOCK one into the small print of a 64 column line. One can switch AT WILL between the two display files. When the 32 col display file is switched IN, all of the color attributes may be used for glorious screen displays. Following are two "switch routines" to be used with <GO SUB zb - or GO SUB ts>. The ts routine: <9000 OUT 255,0: PAUSE 10: OUT 244,0: CLS: RETURN>. The zb routine: <9002 OUT 255,62: PAUSE 10: OUT 244,3: CLS: RETURN>.

Just remember that when you have a Zebra OS-64 cartridge installed, all LPRINT, LOAD, SAVE, commands exercise the routines in the OS-64 ROM, and NOT the normal TS-2068 EXROM. Therefore one must always be in the 64 col (OS-64 Display) before using these commands. Otherwise a crash will result. Preceed all LOAD, SAVE, LPRINT commands with <GO SUB zb>. If you want 32 column presentations on screen, <GO SUB ts> first. If you want 64 col screen, <GO SUB zb>. Neat, eh?

The review article by Phoenix Pete came in in a MSCRIPT file, which gave me a chance to try out the "MS-TAS" utility to print it without loading the MSCRIPT software. I always get tangled up with Mscript's and Tasword's elaborate Control Codes for trapsying around in the two programs. So, the following Utility was developed and tested to print mscript files without need to have mscript loaded (or to learn it durn it). Note that Pete's text has a ragged right margin, which gave the other "MS-TAS" utility a problem. This was solved by using a "variable stepping counter" to end the line print each time a ENTER (code 13) is detected. Note that the last input of "line width" in line 9000 results in POKING the MAXIMUM printer line to the Aerco prcode.

Then counter "n" is stepped to the "line width". However the n counter STEP is a pseudo value. It could be any figure "more than the longest line to be printed". This is because at line 9015 the "y" counter stops when a code 13 (ENTER) is detected. Then the "n" counter is set back, so that when it STEPS it will step to the memory address of the next character to be PEEKED and printed. Chomp on that a bit and you may get some more ideas about making STEP in a FOR counter a variable figure.

Now Pete just assumed that I'd load up Mscript and print his document, so he didn't tell me what address the code started, nor its byte length. So, the "FIND" routine at line 9965 was added. <GO TO 9965> will find START and END memory addresses of the Mscript code file and print the info on screen. Then <RUN> and use this information for the INPUT PROMPTS after the DATA disk catalog is on screen.

In line 9000, the prcode file name, Randomize #, and the line width POKE, are for the prcode version given in the Apr/July Issue Disk. Your customized Aerco prcode may need different figures. This utility will be included in the October (next) Issue Disk (no room in the current Apr/July disk).

```
8980 REM * MSCRIPT FILES PRINTER

8985
8990 CLOSE #2: BORDER 0: PAPER 0
: INK 7: CLS : PRINT '''TAB 8;"
TS-2068 UP-DATE"'''"* Utility t
o PEEK and Print out MSCRIPT Da
ta Files from Disk."'''"* Does
not require Word Proc to be loa
ded."'''"* DOES Require;"'''"1. A
erco or Oliger Printer intfc
to be installed."'''"2. Centroni
```

cont →

```
cs Printer -ON- READY"''"SET E
QUIPMENT, THEN ENTER": PAUSE 0:
 CLS
9000 OUT 244,1: CAT "Cclear.bin"
,: CAT "Cprcode.bin",: RANDOMIZ
E USR 64500: INPUT ;"Install Da
ta Disk, ENTER";y$: CAT "",: IN
PUT "Input MS-TAS Disk File Nam
e     (all between Quotes).";z$
: INPUT "Input File Starting Ad
dress";sa: INPUT "Input File By
te Length";bl: INPUT "Printer L
ine Width";ll: INPUT "Start lin
e count?";Lc: INPUT "Total Col
Lines?";tc: POKE 64504,ll-1: CA
T "z$",
9002 OPEN #2,"p": FOR n=sa TO sa
+bl STEP ll
9005 FOR y=n TO n+ll-1
9010 LET c=PEEK y
9015 IF c=13 THEN  PRINT : LET n
=y+1-ll: GO TO 9027
9020 PRINT CHR$ c;
9025 NEXT y
9027 LET lc=lc+1: IF lc>tc THEN
 PRINT : CLOSE #2: LET lc=1: OU
T 127,12: BEEP 1,20: CLS : PRIN
T "Adjust Page, THEN ENTER": PA
USE 0: OPEN #2,"P"
9030 NEXT n
9035 CLOSE #2
9040 PRINT ''"<1> For next file"
''"<2> To Quit": INPUT "Choice?
 ";a: IF a<1 OR a>2 THEN  GO TO
 9040
9045 IF a=2 THEN  STOP
9050 GO TO 9000
9060 REM ** Find Start and End
          of MSCRIPT file.
9965 LET a=1: FOR n=46000 TO 653
67
9970 IF PEEK n<>0 THEN  PRINT "S
tart Adr=";n: GO TO 9990
9975 NEXT n
9990 FOR n=n TO 65535: IF PEEK n
=0 THEN  PRINT "End of File=";n
: STOP
```

DENSE PACK BASIC, and the use of LOGICAL OPERATORS

There has been a considerable amount of questions and interest about Dense Pack programming techniques and the use of Boolean Logic. While browsing through antique ZX publications I came across an interesting article written in 1983 by Sharon Aker (Where are you now Sharon?). Since Sharon's article merges nicely with my feeble attempts to explain the use of Boolean Logic in Basic programming, her excellent article is re-printed. Also interesting is the setting: The time- Sept. 1983. The computer- Sinclair ZX-81 with 1K total Ram. Boolean Logic is useful in programming ANY computer and can provide increased programming efficiency.

The Logical Operators

Sharon Zardetto Aker

AND and OR have two distinct usages in Sinclair Basic. One mirrors their use in English and is easily understood; the second is less straightforward, but is an extremely versatile programming technique. NOT, the third logical operator, is unfortunately overlooked or ignored by many beginner programmers; it, too, can be a powerful programming tool.

AND and OR

```
IF A<10 AND B<10 THEN...
IF A<10 OR B<10 THEN...
```

In the first example, the command following THEN is executed if *both* conditions are true (the conditions being A < 10 and B < 10). In the second example, as long as *either* condition is true, the command will be executed.

AND has a higher priority than OR and is performed first regardless of its position in the conditional statement.

```
IF X>5 OR Y>5 AND Z>5 THEN...
```

This statement sets up two conditions:

1) X>5
2) Y>5 AND Z>5

Since they are linked by OR, either one being true will cause the command to be executed.

To circumvent the computer's automatic ordering of operations, parentheses should be used.

```
IF (X>5 OR Y>5) AND Z>5 THEN...
```

The two conditions in this statement are:

1) X>5 OR Y>5
2) Z>5

Since they are linked by AND, both conditions must be true for command execution.

You should note that, in this last example, there are actually two combinations that will satisfy the computer's truth-check:

1) X>5 AND Z>5
2) Y>5 AND Z>5

You must understand the computer's view of "truth": a true condition is "1"; a false condition is "0".

Instead of IF-THEN

AND and OR also have another, entirely different, usage in Sinclair Basic, one that allows you to combine several IF-THEN statements into one logical statement that does not even use IF-THEN.

```
IF A<10 THEN GOTO 125
IF A=10 THEN GOTO 300
IF A>10 THEN GOTO 480
```

can be rewritten:

```
GOTO (125 AND A<10)+
  (300 AND A=10)+
  (480 AND A>10)
```

To translate this into understandable English, read each AND as "if." The computer looks at each one of the parenthetical statements and checks the truth of the expression following AND. If that conditional statement is true, the value of the parenthetical statement is set at whatever precedes AND; if the expression is false, the statement is assigned a 0.

So in this last example, if A is 14, the line will work out as in Figure 1. The computer will GOTO 480.

Mutual Exclusivity

The parenthetical statements in the above example are mutually exclusive; that is, if one is true, the others have to be false. That is not always the case, and you should plan carefully in a situation like the following:

```
IF N<10 THEN GOTO 100
IF N=10 THEN GOTO 150
IF N<20 THEN GOTO 200
```

Obviously, N can be less than 10 and less than 20 at the same time; so can it be both equal to 10 and less than 20. As long as the IF-THEN statements are in the right order in your program ("right" depends on the effect you want), there is no problem because the computer will act upon the first true statement it encounters and never see the one(s) following.

However, if you rewrite these statements using the logical operators, you

Figure 1.

```
GOTO (125 AND A<10) + (300 AND A=10) + (480 AND A>10)
          false              false             true
GOTO      (0)       +        (0)       +       (480)
```

Figure 2.

```
GOTO (100 AND N<10) + (150 AND N=10) + (200 AND N<20)
          false              true              true
GOTO      (0)       +        (150)     +       (200)
```

would have a *big* problem. If N is 10, the computer would make the evaluation found in Figure 2 and execute GOTO 350.

Further definition of one or more of the conditions, until the parenthetical statements *are* mutually exclusive, is necessary. For instance:

```
IF N<10...
IF N=10...
IF N<20 AND N>10...
```

Written logically,

```
GOTO (100 AND N<10)+
  (150 AND N=10)+
  (200 AND (N<20 AND N>10))
```

In the last parenthetical statement, the first AND operates as "if." The second AND operates in its usual manner requiring that both conditions be true for the total expression to be true. The inner parentheses are not strictly necessary, since the first AND will always be used as the "if"; they were included here only to make the example clearer.

Other Commands

Since the entire logical expression boils down to a number, it can be used in place of a number with many different commands:

```
PRINT AT (5 AND Z>14)+
  (2 AND Z<5),10;''OKAY''
PLOT X, (17 AND B=12)+
  (27 AND B<>12)
GOSUB (500 AND L=10)+
  (800 AND M=10)
PRINT (0 AND N<>10)+
  (N AND N=10)
PAUSE (120 AND T<10)+
  (380 AND T>=10)
```

Strings

Strings can also be used with logical statements. If the conditional expression is true, the parenthetical expression is "equal to" the string before AND. If the conditional expression is false, the statement is considered to be an empty string. So:

```
IF A>B THEN PRINT ''TOO HIGH''
IF A<B THEN PRINT ''TOO LOW''
```

can be rewritten as:

```
PRINT (''TOO HIGH'' AND A>B)+
  (''TOO LOW'' AND A<B)
```

When A is larger than B, the statements will be evaluated:

```
PRINT (''TOO HIGH'')+('''')
```

You can also assign a string value with this syntax:

```
LET P$=(''TOO HIGH''
  AND A>B)+
  (''TOO LOW'' AND A<B)
```

A logical expression for strings or numbers does not have to consist of alternative choices; a parenthetical logical expression can be inserted into an otherwise straightforward command. Consider a program that would display a multiplication problem of two randomly generated numbers (A and B) and a player's answer (C). If the answer is wrong, you might want it marked with an asterisk.

```
PRINT A;''*'';B;
  TAB 10;(''*''
  AND C<>A*B);TAB 11;C
```

If the answer is correct, nothing will be printed at TAB 10.

Variable Re-valuation

Logical expressions that change the value of a variable are easy to use. Here is one example, and the ways it would be evaluated.

```
LET X=X+(5 AND B<A)+(7 AND B>A)
If B<A  X+ (5)  +  (0) =(X+5)
If B>A  X+ (0)  +  (7) =(X+7)
If B=A  X+ (0)  +  (0) =(X)
```

As you can see, there is no need to write a statement for the B=A possibility if you want X to remain the same in that situation.

If the variable is to be decremented, change the plus sign to a minus sign:

```
LET N=N+(5 AND P=0)-
  (5 AND P<>0)
```

This will be evaluated in one of two ways:

```
If P=0:   N+(5)-(0) = N+5
If P<>0:  N+(0)-(5) = N-5
```

Using OR

While AND is used for strings and numbers, and variable re-valuation involving addition and subtraction, OR is only practical for re-valuations involving multiplication or division. A study of the chart in Figure 3 might help you understand why this is so.

When OR is used in this construction, read it as "unless":

```
LET N=N*(10 OR A>B)
```

If B is less than A, the statement is true. It is evaluated as 1 (as noted in the chart), and reads:

```
LET N=N*(1)
```

and the value of N does not change.

If B is *not* less than A, the parenthetical statement is assigned the value of the number before OR:

```
LET N=N*(10)
```

So, N is multiplied by 10 *unless* B is less than A, in which case N remains the same.

If you were writing a program to calculate new prices for merchandise going on sale, where items less than $100 were to be reduced by 10% and all others by 20%, your statement would read (P is the current price):

```
LET P=P*(.8 OR P<100)*
  (.9 OR P=100)
```

The evaluations are:

```
If P<100:   P*(1)(.9) =P*.9
If P>=100:  P*(.8)(1) =P*.8
```

Applications

If you would like a command executed every second time it is encountered in a loop, use NOT to change the true/false value of a variable back and forth. With:

```
IF V THEN...
```

at the beginning of a loop, and:

```
LET V=NOT V
```

later in the loop, V will be true on every other loop.

By using

```
LET E=N/2-INT(N/2)
```

E is zero whenever N is an even number. To have a command executed only when N is even, use

```
IF NOT E THEN...
```

Similarly,

```
LET I=N-INT N
```

means that I is zero whenever N is an integer, and

```
IF NOT I...
```

will be a true condition when N is a whole number.

Priority

NOT is assumed to apply only to the number to its right, unlike the other logical operators, which automatically apply to an entire expression.

```
NOT B<C
```

is interpreted as:

```
(NOT B)>C
```

NOT has priority over both AND and OR, so the following conditional statement would be processed in the order of innermost brackets first:

```
IF [[[NOT A] AND B] OR C]
```

The use of NOT can, of course, be altered by the use of parentheses.

NOT

To make use of NOT, you must first understand the computer's view of "truth." Conditions in an IF-THEN statement are evaluated, and a true condition is assigned a one, while false one are assigned zero. (Note the "result" column in Figure 3.)

Furthermore, any mathematical expression whose result is zero is considered false, while a non-zero result (even a negative number) is true.

NOT changes the true/false value of an expression:

If A is false, NOT A is true.

If A is true, NOT A is false.

Figure 3.

Operator	Conditional Choice	Conditional Expression	Result
AND	A$	true	A$
AND	A$	false	"" (empty string)
AND	N	true	N
AND	N	false	0
OR	N	true	1
OR	N	false	N

QL Characteristics

About half of the subscribers to UP-DATE are owners of both the TS-2068 and the Sinclair QL, the latest member of the Orphan Computer family. Comments received about the QL range from pure disgust to enthusiastic praise. But there is one common denominator amongst the QL owners and that is a desire to learn how to use this strange computer. It is indeed a strange one (to me) and one that requires a considerable amount of study. Having only recently purchased a QL, I am not yet ready to become your QL Guru. So, let us begin learning the QL together while continuing to treat our TS-2068 as our primary system. Eventually the QL will become a work horse to endure for many years to come. But I am sure that my TS-2068 system will never be discarded no matter how useful the QL becomes.

We TS-2068 users enjoy several advantages when we begin to use the QL. First, with few exceptions, all of our programming knowledge can be transferred to the QL. The key-words and tokens mean the same. "QL BASIC" just gives us an expansion of the language that we have been using. So, that part of the QL learning curve should come fairly easy. Since we can use our learned TS-2068 language, this allows us to program the QL in "simplese" while picking up the more advanced QL BASIC and Super Basic. Then, as we upgrade, still more language extensions and capability are added, such as "super tool-kit". In fact, the QL can become the most powerful of all personal computers available in the market places of the World, more powerful than the IBM XT.

There is also a PLUS with equipment compatability. Our disk drives and monitors are compatable with the QL. I first started off sharing a pair of 80 track drives and a RGB monitor with my TS-2068. But that soon became too much of a switching job and I gave in and spent some money on a second pair of drives and a Magnevox RGB monitor. WHICH, leads us into the subject of EXPENSE to get going with the QL. One can go "whole hog" as I did, or progress in stages. A composite monitor can be picked up for as little as $65.00 via catalog shopping while letting the budget build up for a RGB monitor. And, a full page width serial printer can be purchased for as little as $39.95. I have one of these old IBM thermal printers to do listings and scratch printing.

There have been many comments to the effect that "By the time one buys disk drives, monitor, and Trump Card, one has enough invested to have purchased a IBM Clone system". So, let us look at that first. "SYSTEM" is the key word. If you carefully examine the adverisments for IBM systems (Clones), you will find that a COLOR system, with 2 disk drives, with 600K of memory, will cost about $1800.00. Then you'll need to add a printer. My QL system cost was: $75.00 for the QL, $300.00 for the Trump Card, $300.00 for a RGB Monitor, which comes to $675.00. This is about the price of a "starter" IBM XT Clone with only 256K of memory, no monitor, and one 360K drive. It would take another $900.00 to upgrade the IBM to "aproximate" the capability of this QL system.

Really we are comparing Zebras to horses. The QL and the IBM are totally different critters. In my way of thinking a IBM system locks one into a "closed system", while the QL provides versatility. The Trump Card gives 896K of RAM memory, where the IBM is not that expandable. Even more ROM memory can be tagged on in cartridges. Much is said about the need to be "IBM Compatable". For what reason? Why try to be compatable with an INFERIOR system, and when such compatability reduces one to a lesser level of capability? The QL has more capability than the IBM XT. We just need to learn to tap that capability. If we need to compare the QL with a computer of similiar capabilities, the computer would be the Northgate 386, a 32 bit CPU powerhouse, as is the QL.

The price tag of the QL reveals a fact that needs to be understood by all. The computer was not ready for marketing. Because of that, the reviews of the QL were all bad- very bad- CRUEL- and Killing! While designed to sell for around $600.00 for the basic machine, reviews killed it and its price plummeted to $75.00. Now the few remaining QLs in stock are priced higher, and soon none will be available. An estimated 8000 QL computers were sold in the USA and Canada. So, we QL owners are rather few in numbers on this side of the Atlantic. However, the QL is a popular computer in the UK and supporting hardware and software are plentiful. Because of the low price and the known end to QL production, many American buyers bought a spare QL. I would advise any who did not to do so quickly.

The QL, as it is purchased, has quality control deficiencies that need to be (must be) corrected. The easiest path to take is to send the QL to Tom Bent, 9015 Flicker Place, Columbia, MD 21045. Tom does a series of fixes that corrects the problems. I will go through these. 1. The QL Power regulator needs to be fixed. As purchased it creates too much heat which causes crashes after a hour or so of operation. 2. Some CPU chips overheat and need to be evaluated, replaced if necessary. 3. The existing two ROM chips are high power consumers and need to be replaced with one EPROM which runs cooler and also corrects some ROM bugs. The new EPROM also contains several very worth while additions to the QL operating system which are normally sold in (add on) ROM cartridges.

4. The IC that provides Television monitoring is removed (optional). This reduces both internal noise and relieves the load on the power regulator, but eliminates the capability of using a TV for monitoring. 5. The "Spider board" needs re-soldering. As purchased, the spider board has loose connections that cause crashes. 6. A lube job is needed! Tom uses WD-40 to free up the keyboard keys and make them operate smoothly. 7. The RGB output needs Buffering. All of these fixes are needed. You can either have them done or do them yourself. But for the QL to operate successfully, these fixes are necessary. Quantum Levels magazine has devoted quite a bit of space to describing these problems.

To accomplish all of the above fixes, Tom charges about $65.00. This is really a bargain, especially if you consider that the addtions to the ROM, if bought in a Rom cartridge costs that much. Tom Bent is the Editor of Quantum Levels and one of the best authorities on the QL. I highly recommend that every QL owner subscribe to Quantum Levels, which is the only North American publication exclusively devoted to the QL users. Quantum Levels started only about two years ago and I advise getting all back issues. A new QL magazine has been announced. "QL USA" has not yet published its first issue. If interested, write to: Bill Fischer, 419 Parkwood Dr. E., Orange Park, FL 32073 (editor

publisher).

So, now you know why one could buy a QL for as little as $75.00, which is far less than the total cost of the IC chips contained in the computer. The QL simply was not ready to be marketed and Sir Clive payed the price for not fixing all of the bugs before going into production. Then the deficiencies in production line quality control was the final thing that did him in. One can look at it this way. We buy the computer in two stages. The first stage is buying the QL for $75 - $100. The second stage is getting its bugs fixed for about $65.00. Then we can go about assembling a SYSTEM, including the monitor, the printer, memory expansion, and disk system. This is a new concept for buying things and is kinda hard to get accustomed to.

Presently there are two devices that provide both Memory upgrade to 896K RAM, additional ROM, and a Disk Drive controller. One is the TRUMP CARD and the other is the SANDY BOARD. Both contain extra ROM that provides a plethora of additional Super Basic functions. AND, both are quite expensive, around $300.00. One can delay such upgrades and still have a QL system that is quite capable, more capable and more powerful than a un-expanded IBM XT. The four Pison Softwares, which comes with the QL, are excellent and they represent a combined value of at least $300.00 when compared with any software group offered for the IBM. These three softwares work well with a un-expanded QL. Some users may not ever need more software capbility.

The two Micro Drives are the legacies of Sir Clive Sinclair who spent a decade perfecting the highest speed low cost tape deck in the World today. While I prefer using Disk Drives, the two micro drives do add to the QL's capabilities.

I have already found that the two micro drives provide excellent and reliable mass storage. Operation is faster than disk drive with the Commodore, but slower than disk drive with the TS-2068. Never the less, micro drive lets one wait until one's budget is ready for the expense of up-grading to Trump and a disk drive system. Even then there is a means available to upgrade in stages. The Trump card provides the huge memory expansion, plus a great increase in Super Basic functions, plus a disk drive controller, plus RAM DISK. The addition of disk drive can be delayed as the final stage of upgrade. RAM DISK with Trump Card lets one operate "as if there were a disk drive", then save the data or program to micro drive when finished.

Since most QL users have a spare QL (insurance), why let the spare lay around collecting dust? It can be "networked" with the main QL by simple means of connecting a single conductor wire between computers. The distance limit between computers has not yet been found, but would certainly be as far as from one corner of the house to another. Since "networking" is an exclusive capability of the QL you wont find much published information about the principle. This capability is usually the domain of very expensive "main frame" computers. Briefly, the "networked" computers use their own memory for operation, plus that of the HOST computer, and can operate any peripheal that is attached to either. For example, my son can be in his room doing a college report. When finished, he sends the data to be printed with the printer that is connected to the main QL computer. During the process, the micro drives and disk drives that are connected to either computer can be used for both programs and data saves and loads. This is kinda mind boggling to me, but I'm inquisitive enough to "try" to take it to the limit.

The Printer: There are some very good bargains in printers now a days, especially SERIAL printers. Serial printers connect directly to the serial I/O ports (2) of the QL. Most of us are not familiar with the computer syntax to print with serial printers. My first week of QL ownership was devoted to figuring out how to use a printer. This does it with a serial printer connected to Serial Port #2: <10 OPEN #4,ser2c> <20 BAUD 9600> <30 PRINT #4,"This is a Serial Printer"> <40 PRINT #4> <50 LIST #4> <60 CLOSE #4>. Missing in QL Basic are our familiar LPRINT and LLIST. Instead we must OPEN a channel to the printer and then use PRINT or LIST. To use a Parallel Centronics printer one needs to purchase a Parallel Interface.

A suggestion (as I am doing): To make myself learn the QL and QUILL, I am beginning to do all correspondance with the QL and QUILL, and my more serious data processing and publishing with the TS-2068. Later the priority may be reversed. How about "maintainability" of the QL? Well, we all have some experience with that subject, dont we? Already we have two sources for repairs, Tom Bent and Dan Elliott. And, if you think that IBM Clones are repaired everywhere, you are a good prospective customer for the salesman of snake oil, hair growing lotion, and astrology predictions. Even the giant Tandy Corp. does not have full repair capability for their IBM Clones. One does not buy maintainability when one buys a IBM Clone.

So, belay your fears! The QL is a good computer, better than any other that one can buy for less than about $2000.00. It can be fixed when it breaks. It is expandable in stages. It will take us at least a year of study and usage to learn what the QL can do for us, and it will serve us well for many years. Now the QL user Group becomes another select group of independant thinkers, as unique as we TS-2068 users. Both refuse to ride the main stream of mediocrity. And neither accepts the idea that a popular Name Brand necessarily means the best. Both groups look to learning as a challenge and help each other to that end. Both groups will survive will take it to the limit!

UP-DATE invites more qualified users to help out in coverage of the QL- to help us novices to learn this fine computer. The disparaging comments about the QL and its bugs in this article will stand as the only group of negativisms as UP-DATE launches into support of "Computing with the Sinclair QL". However "bug reports and the details of their fixes" will be covered, along with programming, software operation, the availability of products, and the whole range of user interest subjects. And, there are plenty of subjects to cover. After studying the QL manual, the Trump Card manual, and the Speed Screen manual, I have hundreds of questions that need answers. The QL is a BIG and highly diversified computer. I would compare the learning task with that of a stranger to San Francisco who has just been hired as a taxi driver. Fortuntely for us, we can learn a segment of functions at a time and continue to progress through the learning curve. It should be a lot of fun and also productive.

Thought I would mention a few more letters I have received.

Dear Ron,

This is a cry for help! I'm completing the FDD Technical Manual. So far I've expanded the manual from 33 to 70 pages. But I don't even own a system. (I have a borrowed one.) I have descriptions and schematics of the TS2068 Bus Adaptor, FDD Interface, FDD 3000 Controller, FDD Drive, and Amdisk drives. Also I've a good start on disassembling the TOS BASIC extension software. Where I need help is as follows: How many versions of the FDD system are there (my borrowed system is silver boxes)? Does anyone know the three PAL logic? Has anyone done a disassembly? What else do I need to know that I don't know? I will make all this information available to anyone doing repairs or writing machine code needing access to the internal code. It should be possible to get all parts except for the pc boards and cases. The TOS is in Eprom so this can be corrected and expanded. One more thing. I need a FDD Controller (silver box or 3000). I have enough other parts to build a system. The controller would not have to be operating if it is repairable. Can I get the back issues of T.O.P.S. - I guess they are all back issues. Thank you for the FDD Express. Sincerely, Bob Orrfelt, 3436 Bay Road, Redwood City, CA 94063

Bob, as far as I know, there are just very slight differences between the silver boxes and the 3000 system. I say slight, because in doing some work with Mike Finn's Track Reader program's, I found a slight difference. Nothing to get emotional about. I will be explaining more in a future issue. There was only one issue of T.O.P.S., and I don't have more than just my one copy, which has notes all over it. Maybe somebody else out there has a copy they can send you. Ron

Jack Dohany wrote me a nice, informative letter a (long) while ago.

Hi Ron,

Thanks very much for sending me your newsletter. I like it, and want to remain on your list, even though I don't own a Zebra system right now...To provide fullest possible support, I need to have ALL disc systems at my elbow. The only one I now lack is Zebra, so I'm looking for a very 'inexpensive Zebra system, working or not, old-type or (preferably) FDD 3000...In order to stay alive this year, I need to expand my mailing list, and get my software known to as many people as possible. So please ask your subscribers to send me their address so I can send them a catalog. Preferably they should enclose $1 + SASE to help out... Something you or other Zebra users may be able to help me with: I have never seen an FDD 3000 or had access to one. I need to know how it differs from the original "many-little-box" Zebra system. If someone could undertake to make me a copy of the FDD 3000 documentation and mail it to me, I'd be happy...But a short, complete and correct symopsis of differences between old and new might be just as useful, since I already have documentation for old systems...Does anyone out there have a TOS source listing?..An expert programmer, with access to this listing, would be able to answer just abou any question or solve just about any problem. For example: how to get Mscript to talk to a serial printer over Zebra RS232?..Jack Dohany, 390 Rutherford Ave., Redwood City, CA 94061

Everyone is on notice to help Jack out any way possible.

Dear Ron,

I bought one of Zebra's early single drive systems (with it's problems) and later added the second drive. I am rather happy with my system now. I would be interested in knowing if anyone has done much with programming for the Amdek DXY-100 Plotter, which is addressed by lprint commands, I would also like to know if anyone has produced a program to make use of the 2068 sound capability. Dennis Bryant, 16148 Foliage Ave., Rosemount, MN 55068

Anybody who can answer this would be appreciated.

Dear Ron,

I don't remember if I read it in your newsletter or when talking to Owen Christianson but I'd like to transfer my one 2068 program Profile 2068 to disk. Your series covered Tasword but I don't even have that wordprocessor program, so if that is coming up I'll be looking forward to that. Tom Dunlop

Look no farther, Tom, this is it. Actually, not even looking at the program from way back when I transfered it, I can tell you that Tom Woods made tranfering the program INCREDIBLY easy! Hopefully you have the full manual, because all the information you need is right there. First, let's just MERGE the loader program. Now change the lines that LOAD the code and program to LOAD *, making sure to keep the names slightly different. If I remember correctly, both the code and the BASIC had the same name, which on tape wouldn't matter. So just rename the code "p/f" and the program "profile.bas". Now SAVE * "profile" LINE 1. That takes care of the loader. Now reload the original loader from tape, and in fact go through the entire loading process. Once you get the prompt to Create or Load a file, press CAPS SHIFT-EDIT, which will get rid of the quotes, and then STOP. Now SAVE *"p/f" CODE 63488,2046. Now LIST the Profile program you have in your computer. If you see only a blank screen, then just change the INK color to white: INK 7. Change the necessary part of line 107 to SAVE * f$ LINE 1n. Be sure to use the trailing "n", so that the program will overwrite the existing file on disk (if you want to do that). Change the part of line 5510 to LOAD * f$. Now SAVE *"profile.bas" LINE 9996. And you should have Profile on disk. There are some improvements that can be done. For example, I included a small section after the "L" or "C" prompt to do a CAT * if I choose "L". That way, I can see just what files I have to load. I also set up my version to automatically add the ".fil" extension to a name I saved, and to add that extension to any filename I wanted to load, so I don t have to add that when I want to load something. You can do that kind of stuff your own way. All the other additions where just as discribed in Toms manual, full-size printer addition, auto search improvements, etc. I do have a small "start" program on the disk to auto-load the "profile", but you can just rename the "profile" loader program to "start".

On to other things. First off, I have recently gone through all the letters I have received from you, (I keep a complete archive, no "file 13") and find that there are many questions I have not answered, or perhaps answered in a somewhat disorganized manner. And in fact, I kind of got confused as I read through as to what I had and had not related to in the

newsletter. Also, I have been informed by some people that my explainations of different items left a lot to be desired, sometimes more confusing than helpful. So I want to ask you to do me a favor. Anyone who had a topic they wanted answered, or that wasn't answered completely and understandably, please rewrite me with the request, and I will try to get it in as soon as possible. Please forgive me for the unprofessional manner in which I have handled the newsletter, but I said at the beginning I had never done anything like this before.

Just to let you know something I am working on: I have developed the idea of making an "archive" program, and in fact I have started with it. What it (hopefully) will be, is a program that reads the directory track of a disk, (Thanks to Mike Finn's TRACK READER program) and then is able to load up all the programs, code, arrays, on disk and place them on a cassette tape. There are a couple of ways this can be accomplished. One, by using Mike's program to read each track of a disk, encoding it, and placing it directly to tape. This would be faster, but a major drawback would be that the programs on the tape would not be able to be used from tape. I choose to get each file one at a time, and place them on tape so that they can be used from tape if necessary. Imagine having your drives go down, and not being able to access an important program.

Yes, you could load and save to tape each program, code, etc. manually, but how often does anyone do that? I have had my drives for, golly has it been almost three years already? And I have NEVER saved any of the files to tape, even though most of the programs have been created since I had the drives. So if my drives go down, I am up the creek without a paddle. And I am too lazy to sit here for the hours it would take to individually load and save each and every program. So if I can write a program to automatically do that, even if it can only do one disk at a time, then so much the better.

Now, some problems with it. One, I am writing it in Basic, since my knowledge of assembly leaves a lot to be desired. Oh, I can read assembly alright, but writing is a different matter. So with this in Basic, that means that I am using variables, which may conflict with variables in the programs I am trying to save. So some rules I will have to follow are: Use variables that I NEVER(!) use in programs. And surprisingly, there are quite a few, even with the string variables. Ever notice how we tend to use just certain variables in all our programs? For example, I almost always start with A$, then B$, then C$, etc. Then I skip to Z$, Y$, and sometimes X$. I NEVER use U$, V$, and W$. See if you don't tend to follow something like that as well.

Since the program is written in Basic, that means that a MERGE * will merge the program on disk with this one. So I have to make sure that all the line numbers are such that they don't interfere with the merging programs, and are not intertwined with them. Also, since the entire program, along with this special program, will be saved to tape, it is important to make sure that this routine is not executed when the merged program is run.

Another major consideration is the amount of memory the program and variable take up. Though I am trying to make it as small as possible, still it will take up some room. So a program on disk that takes full memory will not be able to be merged. Profile is one of those that would not be able to be backed up in this manner.

Obviously, machine language is the answer to most of this. When I get it finished, I will publish it and let those assembly programmers of you out there translate it to machine code. Then, you may be able to locate it in some innocuous space in memory so everything can be loaded in and saved.

Also, I haven't figured out how to load and save to tape those sequential and random access files. They could be much larger than memory (which is why they are so valuable) and the routine to convert them to code would take up room I don t want to use. I am going on the assumption that those kind of files would need continual update anyway, and so a backup would be too old to bother with.

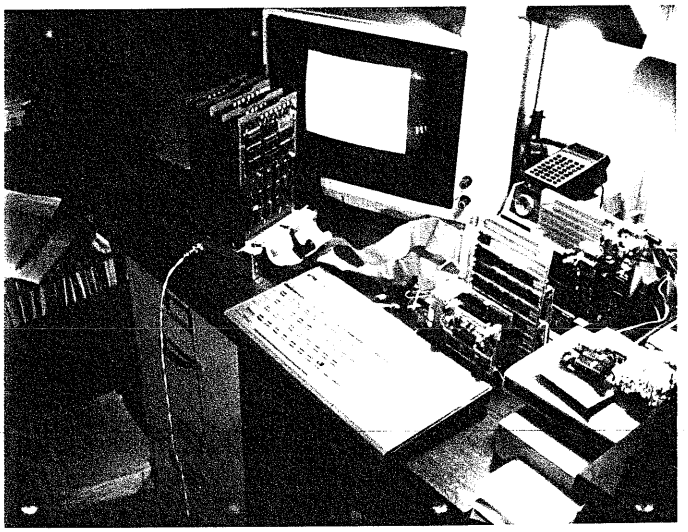
As you have probably noticed, I am using Tasword again to write this, even though you know I have Mscript. Well, the truth is, I have tried it a few times, used it for the last newsletter, but don't care for it all that well. My preferences are Tasword first, Wordstar (CP/M) a close second, and Mscript a distant third. Just no accounting for taste I suppose. I just prefer the "what you see is what you get" type of display, which both Tasword and Wordstar offer. Yes, I can set the screen of Mscript to the width I am printing, but I prefer on screen formatting, which Mscript can't do. Also, I don't like all that extra on screen jumble interfering with my reading. I will continue to use it and get used to it, but that doesn't mean I have to like it. I have to learn it inside out though, so I can try to figure out how to send the output through the serial port of the drives.

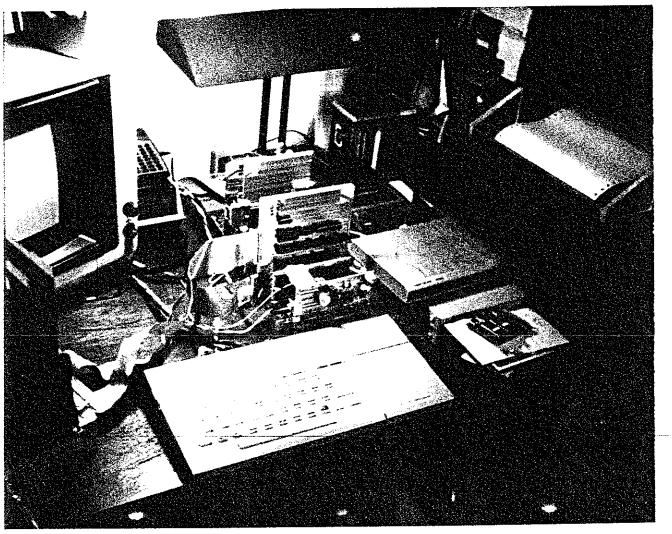
Now let me tell you the greatest(?) news. I have finished that Accounting software I was writing for my boss, and have been paid. So I have ordered an XT clone. Haven't received it yet, only ordered it two days ago. But now I will be able to be compatible with more business people in my area. A client of mine has an Amstrad, based on my recommendation, but I have been unable to get some of his software installed, since he has the computer in his home, and his schedule and mine conflict so much that I can't get there. So the answer is for me to have a compatible machine here, install the software properly on disk, and then give the disks to him.

So now the first thing that comes to your mind is, "Oh, no, is Ron going to abandon us too?". NEVER!!!! I feel too much loyalty to my fellow Zebra users, and in fact, will need to keep using my Timex system for some time to come. One of the first things I have to do is to try to transfer the PC telecom program PROCOMM from my CP/M disk to the XT. I downloaded it to my system so that I can hopefully do a direct transfer serial port to serial port, and thus get it to the clone, without having to buy a telecom program for it. (I'm incredibly cheap) If I accomplish this, you will be the first to know how. Why? Some of you do not have modems for your CP/M, but would like more CP/M programs. Everyone knows somebody who has a PC type computer. Let him download a program for you, and transfer it to your machine.

No, I did not get an Amstrad. I was (and still am) quite enthused about it, especially the 1640, but at the last minute decided I would need more expandability that it would allow. Just hope I haven't made a mistake.

Ron





TS-2068 with ** UNLIMITED RAM capability

The plug in cards to the left are expansion banks. The tall card behind the 2068 is the controller card. Interconnect is via 2 40 wire ribbon cables.

*** E X T R A M E M O R Y ***
(editor's note)

I wrote to Mr. Charles G. Bothner because I had heard of his work in the TS-2068 Extra Memory area. His response flabbergasted me. This is the most complete engineering analysis of 2068 Extra Memory that has been produced to date. It goes far beyond the expertise of the Timex engineers and corrects their mistakes. While interesting to read, most will be swamped by the technical detail. The real reason for publishing Mr. Bothner's extensive thesis is to provide the information to our own hardware developers who are still producing products for the 2068. It can save 2 years of duplicate work and cut the time needed to bring us products to use. Mr. Bothner, we appreciate your gigantic contribution and we applaud your expertise! You are really dedicated to our slogan, "take it to the limit!"

MAY 10, 1988

Bill Jones
TS-2068 UP-DATE
1317 Stratford Avenue
Panama City, FL 32404

Dear Bill,

You are right about me having operating hardware for TS-2068 expansion bank operation. I have spent the major portion of the past two years in debugging the Timex ROM software and constructing some hardware for expansion banks.

Let me fill you in on my present status of development. At the present time, I have completed debugging of the home PROM and verified operation of all modified routines with diagnostic software. The routines include the FORMAT, MOVE, CAT, and ERASE functions as well as the LOAD *, SAVE *, VERIFY *, and MERGE * functions. One of the Timex routines required for communication with the expansion banks is the PASSING routine which is located in the EXROM. The original Timex implementation of this routine (which saves parameters on the machine stack) required that memory chunk 0 be enabled in the EXROM bank simultaneously with chunk 1 being enabled in the HOME bank. Apparently the SCLD chip does not allow this to occur since I could never get this routine to work properly in the EXROM. After moving the PASSING routine into the HOME PROM, it worked perfectly.

I realize that this letter is somewhat rambling, but I wanted to give you some idea of the types of problems that I encountered during the development program. You may be wondering by now where I found room to cram any more functions into the HOME PROM. There was a great deal of redundant code in the original ROM, and several small routines which were never used. Also, I deleted the original PRINT and COPY routine area which dealt with the 2040 printer only. Another significant change to the the HOME PROM was the modification of the calls to the routines located in home RAM. Originally they only communicated with the RAM routines when they were in chunk 3. I included routines to determine whether the RAM routines are currently in chunk 3 or chunk 7. The calls to these routines were originally used by only the DOCK bank communication routines, but now they are also used by the expansion bank communication routines.

The present hardware status allows communications to up to 15 expansion banks via a controller interface card attached to the bus connector. Each expansion bank is on a separate plug-in card on a separate chassis assembly. I only have a total of 5 expansion banks on-line at the present time because I have not had time to wire up any more cards.

During the course of the development, I found it necessary to do major re-writes of about half of the O.S. RAM routines. I completely scrapped the original routines which did the actual setup communications with the expansion banks (B.S. READ and B.S. WRITE). The use of ports FC and FD as originally mentioned in the Timex Technical Manual seemed to make more sense so I went that route in the new hardware.

At the present time, I do not have the expansion bank initialization routines in the EXROM completely debugged. When I boot up the system, I load the SYSCON table and the MAXBNK values via a separate program which is loaded from tape.

You asked about some data on extra memory. Probably the simplest add-on is the full expansion of the EXROM bank. It is necessary to add a small circuit card containing two I.C.'s to the internal circuit of the computer to provide full address decoding to the EXROM PROM to allow chunks 1 through 7 in that bank to be used for either RAM or additional PROM routines. As you can see on the schematic, only one of the chips is used for the EXROM decoding. The other

chip allows de-selection or disabling of both the EXROM and DOCK banks when the line marked BUSISO is in a low state. This line is an unused function on the original computer, but it is the key element in the functioning of the bank expansion hardware.

Let me explain the bank expansion hardware operation a little further at this point. As you are aware, the activation of chunks of memory in either the EXROM or DOCK banks is accomplished by setting corresponding bits high in port F4. The identity of the bank using these chunks is established by the state of bit 7 of port FF (0=DOCK, 1=EXROM). The use of the BE signal by expansion banks would cause disabling of ALL HOME, DOCK, and EXROM memory so the expansion bank would not have direct access to the system variables or the RAM which is accessed by the SCLD for display refresh. In my implementation of the bank switching hardware, the bits of port F4 are still used to determine bank usage. A 0 in any bit position still signifies that the HOME bank has control of that chunk and a 1 signifies that either an expansion bank or EXROM/DOCK banks (depending on the state of bit 7 of port FF) have control of that chunk. If an expansion bank has been assigned the chunk, it will pull the BUSISO line low whenever addresses in that chunk are active, thereby disabling both the EXROM and the DOCK banks at that time. (The GET STATUS routine in RAM has been modified to properly identify which chunks are assigned to which banks by subtracting all expansion bank chunks from the port F4 mask and identifying all remaining non-HOME chunks as either DOCK or EXROM)

I am including the schematic for a full expansion of the EXROM bank via a circuit card which plugs in to the expansion bus connector. Note that the BUSISO line as well as the BE line can disable this memory when required.

I sent an article on port expansion hardware to TDM over a year ago, but I guess that they were not interested so I am including a copy in case you might want to do something with it. Since that article was written, I have added a number of other ports to the system including the Larry Kenney disk system, a 16 bit parallel I/O port, a Digitalker port, and a 20 milliamp loop interface port. The photograph shows that the outrigger hardware swamps the original computer in sheer volume.

I really don't know where to start in writing any articles on my memory expansion proect. Over 1000 bytes have been rewritten in the HOME PROM alone and you can see from the expansion bank schematics that they will take a great deal of explaining to make the overall operation clear to your readers. If you are interested, I can send you copies of the modified PROM's so you can try out any of the hardware you may care to build. I am not personally interested in marketing anything but will freely make information available if we can figure out a rational way to present it.

Please feel free to give me any advice or comments relative to generating articles. I am a hardware engineer, not a writer.

sincerely,

Charles G. (Chuck) Bothner

P.S. In regard to extra memory, there are over 1000 bytes of RAM going to waste on the normal 2068 computers. Most people are familiar with the 256 bytes available in the 2040 printer buffer area starting at address 23296 (5B00H), but I have never seen any mention of the 542 bytes available in the System Variables area starting at 23756 (5CCCH) or the 265 bytes available in the Syscon Table area starting at 24311 (5EF7H). My system boots up with a modified version of the AERCO printer handler and a COPY routine for an Olivetti model PR2300 printer being loaded into the spare RAM in the System Variables area. (The expansion banks are rapidly filling up the Syscon Table area.)

2068 PORT EXPANSION

Page 215 of the Timex/Sinclair 2068 User Manual contains a statement that up to 2 peripherals can be simultaneously connected to the expansion connector. If you are like me, you have acquired a number of different peripherals over the last few years, such as a full size printer interface, real time clock, telephone modem, graphics interface, RS-232 interfaces, etc. It becomes somewhat of a nuisance to keep reconfiguring the units connected to the rear expansion port, as well as producing unnecessary wear and tear on the connectors.

There are two primary reasons for the 2 expansion unit limitation. The first one is the 2068 power supply limitation of 1 ampere maximum on the +5 volt power bus. The second reason is the lack of buffering on most of the output signals on the expansion bus.

With this in mind, I designed and built a port expansion buffer unit. The schematic for this unit is shown on Figure 1. The power limitation was corrected by providing an external power supply which provides +5 volts @5 amps., and a + & -15 volt supply @500 milliamperes. This powers all port add-ons beyond the buffer card.

An examination of Table 2.4.1-2 in the 2068 Technical Manual will show the acceptable loading that can be applied to each of the output signals on the expansion connector. This table however, like the system ROM's, contains a number of errors. The following table contains some corrections which can be readily verified through an analysis of the 2068 schematic:

FUNCTION	TIMEX MAX. LOAD(I)	REALISTIC LOAD(I)
A15B	1.8 mA.	12 mA.
A14B	1.8 mA.	12 mA.
A13B	1.8 mA.	12 mA.
A7RB	0.35 mA.	6 mA.
$\overline{\text{EXROM}}$	12 mA.	1.8 mA.
$\overline{\text{ROSCS}}$	12 mA.	1.8 mA.

These corrections were noted during the design of the port expansion card and are listed here only for general information. The functions required for standard port I/O operations are $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$, data bits DØ through D7, and the lower 8 bits of the address bus, AØ through A7. The control bits $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ are among those that are already buffered inside the computer by U15, a 74LS245. The data bits and lower address bits are not buffered and have a limited drive current capability. Since the address lines are always driven by the Z80 chip (unless presently unavailable bank switching hardware is in use), a simple unidirectional buffer chip can be used for these signals as shown on Figure 1. U2 on the buffer card is a 74LS244 which has an output current drive capability of more than 20 milliamperes, so this will support at least 50 expansion ports if only one standard LS chip load is present on each line at each port.

The buffering of the data lines presents more of a problem since these lines are bidirectional. The buffering of the data lines is accomplished by using a 74LS245 chip (U1) which also has an output drive capability of more than 20 milliamperes. The data flow direction is controlled by the logic provided by U4. The 74LS245 chip is always operating in the data out mode unless a read on the port bus is requested. this could cause complications when the Z80 chip requests a read of the ports controlled through the SCLD chip, the sound chip, or the T/S 2040 printer. To prevent loading of the internal data bus by U1 when a read of the SCLD ports is requested, U5 is used to tri-state, or float, the 74LS245 when any of the port addresses FØH through FFH

are in use. All of the presently known 2068 internal functions are contained within this block of addresses. This leaves a total of 240 I/O addresses available for peripheral units. These ports are in the range from ØØH through EFH.

Some additional lines were buffered on this card to isolate the external ports from the internal 2068 hardware. the clock, M1, and RESET signals are buffered by a 74LS367 (U3) chip. This chip was used instead of another 74LS244 to save some real estate on the card to allow for the addition of a real time clock/calendar port on the same card (Figure 2). A 10uF capacitor was added to the unbuffered RESET line to compensate for the lowered pull up resistance presented by the 74LS367 chip (the 2068 internal pull up resistor is 220K ohms).

The port buffer was constructed on a Zebra Systems experimenters card using their expansion connector units. All of the functions not shown on the schematic of Figure 1 were connected straight through from the female connector to the male connector card. The pins from the female connector were not connected to the output male connector for all of the functions shown on Figure 1. See Figure 3 for connector mechanical details. Independent input and output connections were provided as shown on the schematic. The +5 volt and +15 volt pins on the female connector were cut off at the connector to isolate the computer power from the I/O ports. A -15 volt supply was added to allow use of operational amplifiers and other analog type chips on A/D and D/A ports. The -15 volts was tied to pin 5 on the bottom of the connectors. This is a previously unused pin on the expansion bus. Not shown on the schematic of the port buffer card are the usual 0.01 to 0.1 uF capacitors on the +5 volt supply pin for each chip on the board. The external power supply was connected to the buffer card through a 5 pin Molex connector, but these lines could be hard-wired, if desired.

As noted before, if you are using a T/S 2040 printer, it will be necessary to connect it between the computer and the buffer card since it uses port FBH for its communications.

At the present time I have a total of 8 I/O cards stacked behind the 2068 computer (see photos). They contain the following functions:

Card #1- I/O Port buffer and real time clock/calendar

Card #2- RGB Monitor interface and phoneme entry speech synthesizer

Card #3- Text-to-speech synthesizer with exception word EPROM added.

Card #4- 16 Channel A/D interface and paper tape reader interface.

Card #5- Zebra Systems graphics interface

Card #6- RS-232/ 20mA current loop interface(Zebra Systems modified 2050 Modem card)

Card #7- T/S 2050 MODEM unit

Card #8- AERCO printer interface

Cards 5 through 8 are commercial units while the first 4 are home brew assemblies. A total of 47 port addresses within the ØØH to EFH range are presently in use with more peripherals yet to come.

This stacking of I/O cards produces a lot of connectorization but I have never had any problems with system crashes or other problems due to connector wobble as reported for the T/S 1000 computers.

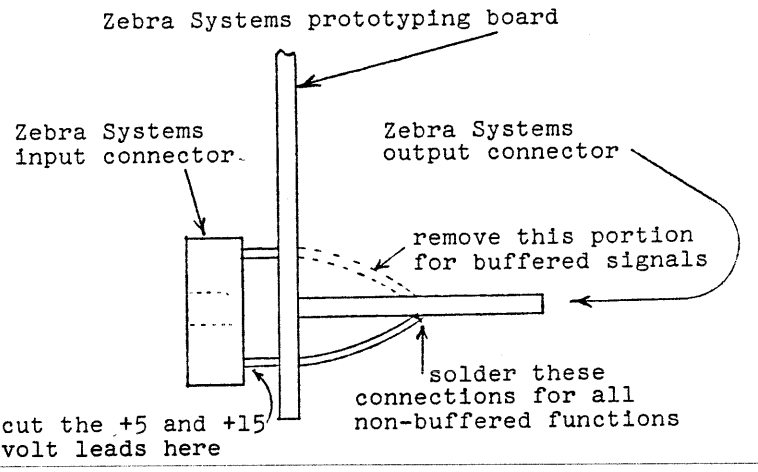


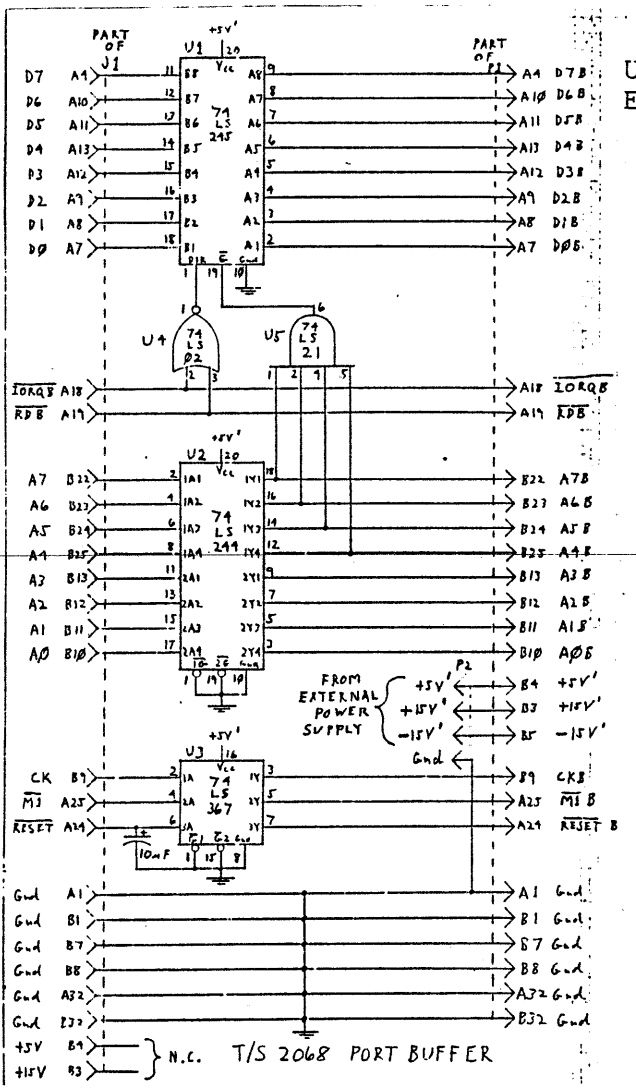
FIGURE 3

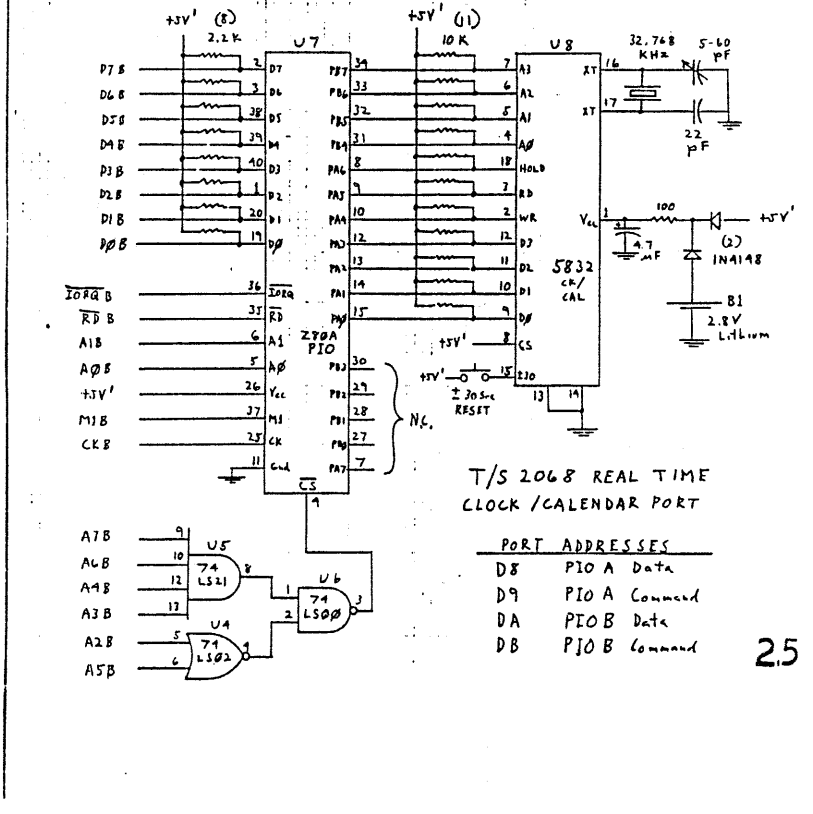
CODE IN LINE 1 REM. OF CLOCK SET PROGRAM

ADDR.	DATA	OPCODE	FUNCTION
685B ENTRY	E5	PUSH HL	Save registers
685C	D5	PUSH DE	
685D	C5	PUSH BC	
685E	3E0F	LD A,0F	PIO set for output
6860	D3D9	OUT (D9),A	PIO A port
6862	D3DB	OUT (DB),A	PIO B port
6864	3E40	LD A, 40	Clock HOLD command
6866	D3D8	OUT (D8),A	Start HOLD on clock
6868	0E22	LD C, 34	150 uSec. delay for
686A DELAY	0D	DEC C	HOLD setup
686B	20FD	JRNZ, DELAY	
686D	21F05F	LD HL, 5FF0	Addr. for setup data
6870	16D8	LD D, D8	PIO A port data addr.
6872	1EDA	LD E, DA	PIO B port data addr.
6874	06D0	LD B, D0	Starting clock chip addr.
6876	78	LD A, B	Clock addr. into A
6877 LOOP	D610	SUB A, 10	Decrement by 16
6879	47	LD B, A	Restore addr. to B
687A	7E	LD A, (HL)	Get ASCII data
687B	C610	ADD A, 10	Convert to BCD + HOLD
687D	4B	LD C, E	Get PIO B port data addr.
687E	ED41	OUT (C), B	Addr. to clock chip
6880	4A	LD C, D	Get PIO A port data addr.
6881	ED79	OUT (C), A	Output BCD char. + HOLD
6883	C610	ADD A, 10	Add write bit
6885	ED79	OUT (C), A	Char. + WRITE + HOLD
6887	D610	SUB A, 10	Turn off write bit
6889	ED79	OUT (C), A	Output BCD char. + HOLD
688B	23	INC HL	Next char. addr.
688C	78	LD A, B	Get clock chip addr.
688D	A7	AND A	Set flags
688E	20E7	JRNZ, LOOP	Continue until done
6890	ED79	OUT (C), A	Release HOLD on clock
6892	C1	POP BC	Restore registers
6893	D1	POP DE	
6894	E1	POP HL	
6895	C9	RET	Done

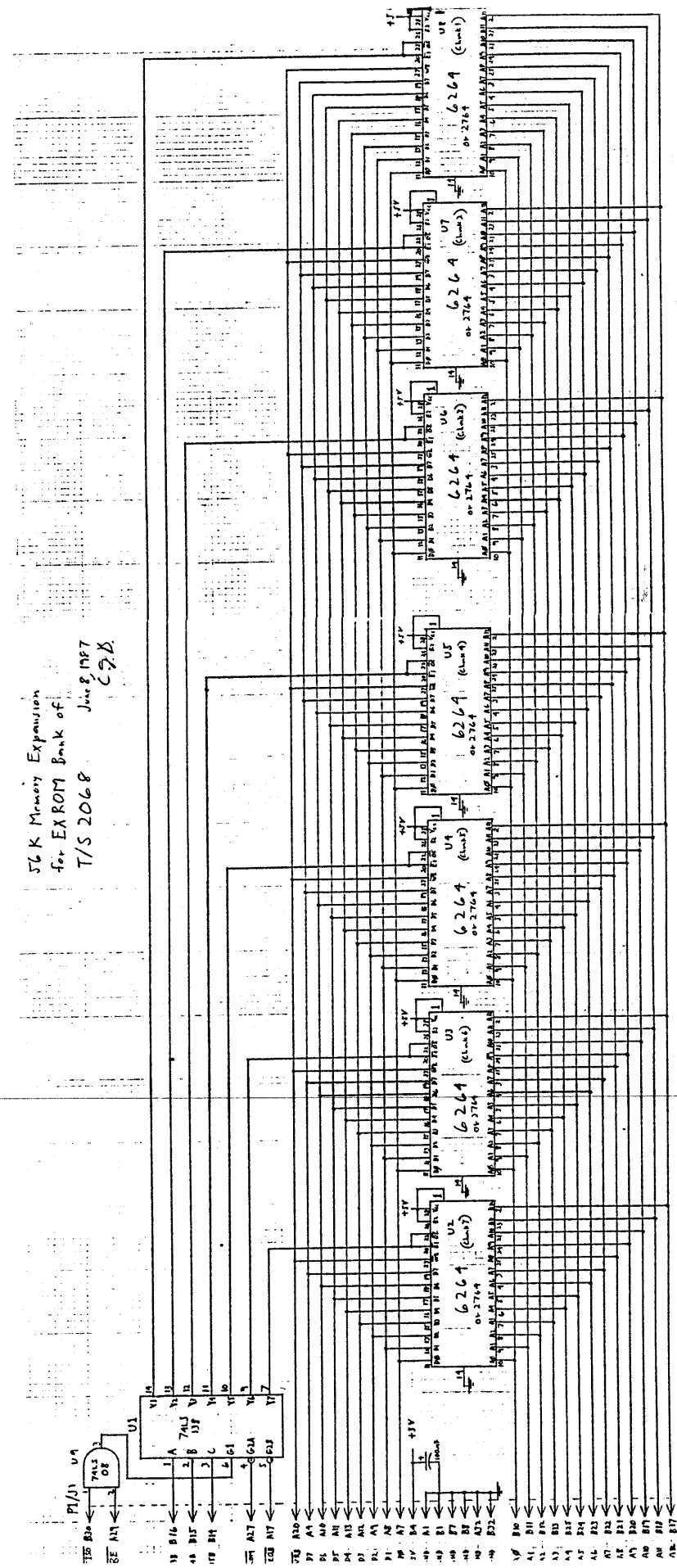
UPDATE July 1988
Extra Memory

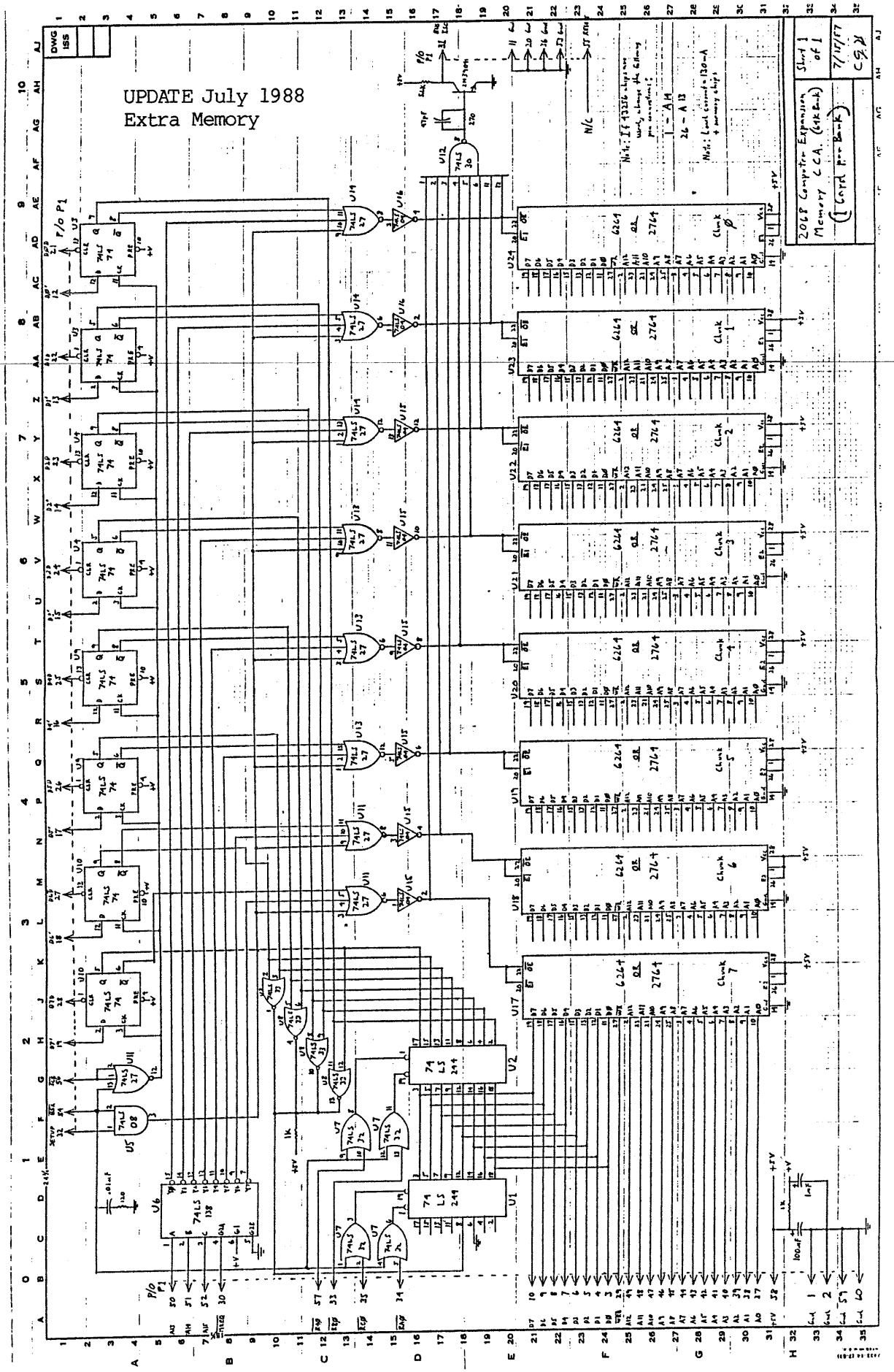
UPDATE July 1988
Extra Memory

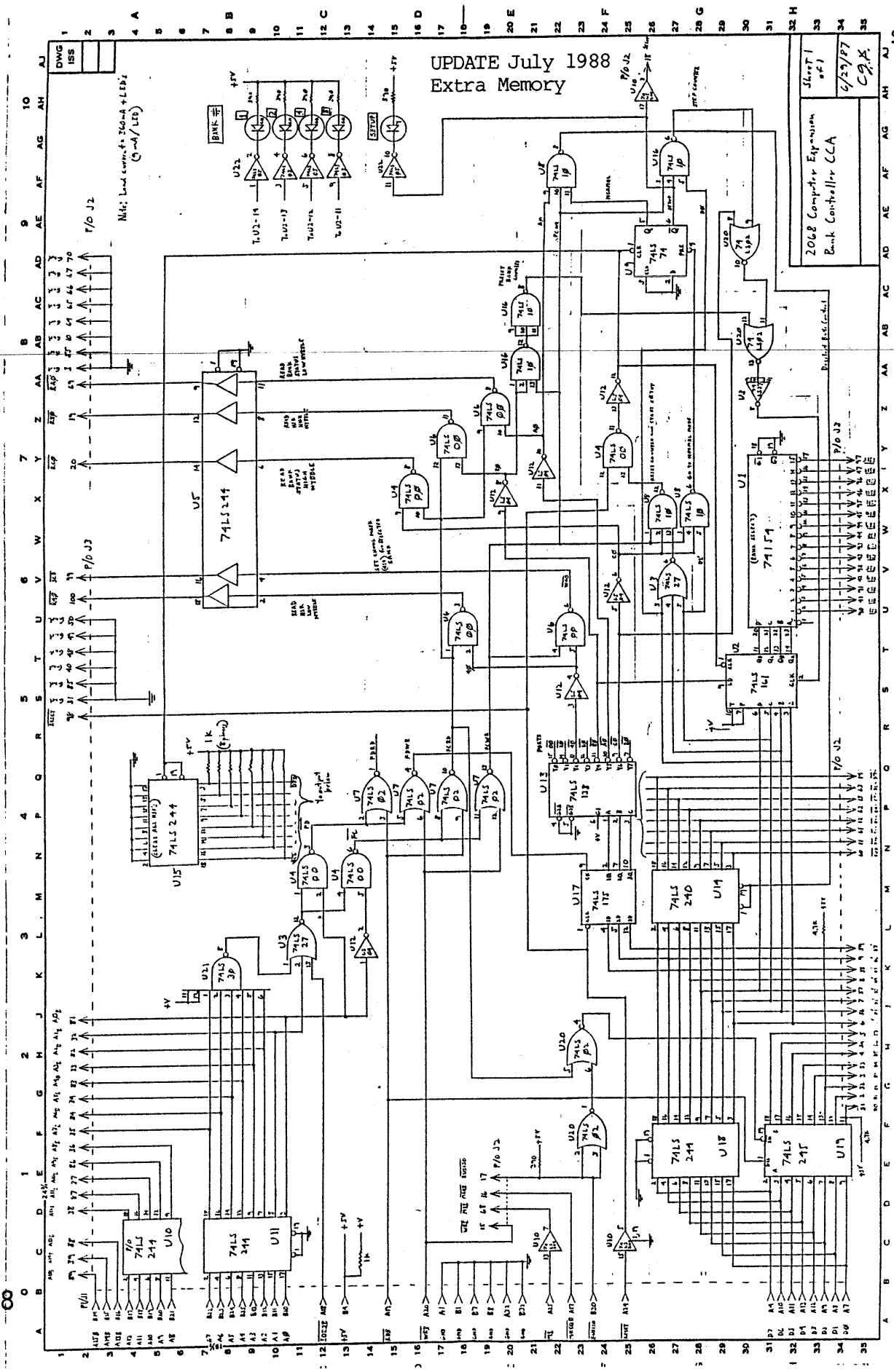




UPDATE July 1988
Extra Memory

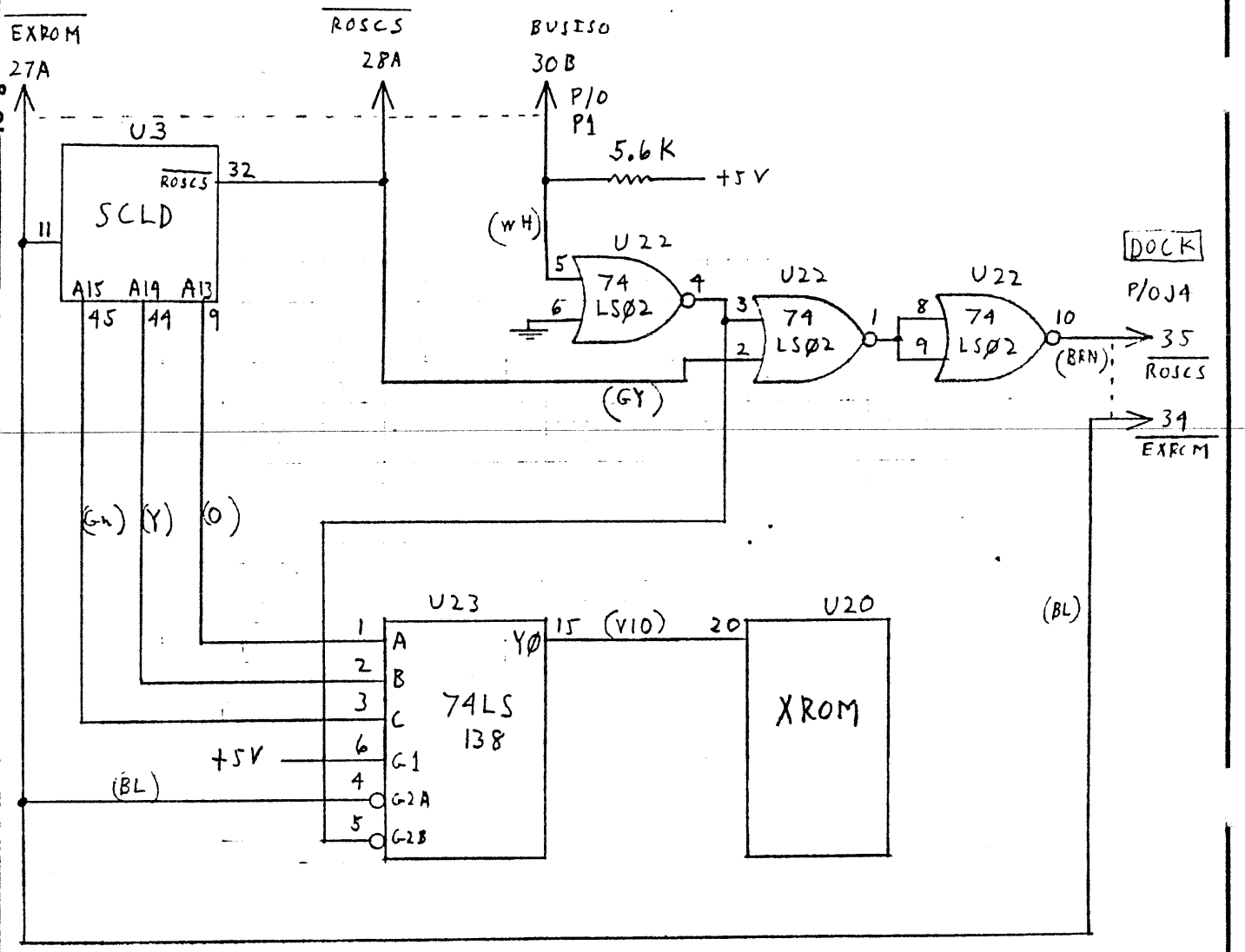



UPDATE July 1988
Extra Memory
2068 Computer Expansion
Memory CCA. (64k Bank)
(Card Two Bank)
Sht 1 of 1
7/15/87
CSD


UPDATE July 1988
Extra Memory
2068 Computer Expansion
Bank Controller CCA
Sheet 1 of 1
6/29/87
C.G.K.
DWG
ISS
74LS244
74LS245
74LS240
74LS241
74LS74
74LS00
74LS08
74LS10
74LS20
74LS27
74LS32
74LS132
74LS138
74LS139
74LS157
74LS161
74LS173
74LS174
74LS175
74LS193
74LS240
74LS241
74LS244
74LS245
74LS295
74S74
74S139
74S240
74S241
74S244
74S245
74S288
74S289
74S412
74S472
74S473
74S474
74S475
74S476
74S477
74S478
74S479
74S480
74S481
74S482
74S483
74S484
74S485
74S486
74S487
74S488
74S489
74S490
74S491
74S492
74S493
74S494
74S495
74S496
74S497
74S498
74S499
74S500
74S501
74S502
74S503
74S504
74S505
74S506
74S507
74S508
74S509
74S510
74S511
74S512
74S513
74S514
74S515
74S516
74S517
74S518
74S519
74S520
74S521
74S522
74S523
74S524
74S525
74S526
74S527
74S528
74S529
74S530
74S531
74S532
74S533
74S534
74S535
74S536
74S537
74S538
74S539
74S540
74S541
74S542
74S543
74S544
74S545
74S546
74S547
74S548
74S549
74S550
74S551
74S552
74S553
74S554
74S555
74S556
74S557
74S558
74S559
74S560
74S561
74S562
74S563
74S564
74S565
74S566
74S567
74S568
74S569
74S570
74S571
74S572
74S573
74S574
74S575
74S576
74S577
74S578
74S579
74S580
74S581
74S582
74S583
74S584
74S585
74S586
74S587
74S588
74S589
74S590
74S591
74S592
74S593
74S594
74S595
74S596
74S597
74S598
74S599
74S600
74S601
74S602
74S603
74S604
74S605
74S606
74S607
74S608
74S609
74S610
74S611
74S612
74S613
74S614
74S615
74S616
74S617
74S618
74S619
74S620
74S621
74S622
74S623
74S624
74S625
74S626
74S627
74S628
74S629
74S630
74S631
74S632
74S633
74S634
74S635
74S636
74S637
74S638
74S639
74S640
74S641
74S642
74S643
74S644
74S645
74S646
74S647
74S648
74S649
74S650
74S651
74S652
74S653
74S654
74S655
74S656
74S657
74S658
74S659
74S660
74S661
74S662
74S663
74S664
74S665
74S666
74S667
74S668
74S669
74S670
74S671
74S672
74S673
74S674
74S675
74S676
74S677
74S678
74S679
74S680
74S681
74S682
74S683
74S684
74S685
74S686
74S687
74S688
74S689
74S690
74S691
74S692
74S693
74S694
74S695
74S696
74S697
74S698
74S699
74S700
74S701
74S702
74S703
74S704
74S705
74S706
74S707
74S708
74S709
74S710
74S711
74S712
74S713
74S714
74S715
74S716
74S717
74S718
74S719
74S720
74S721
74S722
74S723
74S724
74S725
74S726
74S727
74S728
74S729
74S730
74S731
74S732
74S733
74S734
74S735
74S736
74S737
74S738
74S739
74S740
74S741
74S742
74S743
74S744
74S745
74S746
74S747
74S748
74S749
74S750
74S751
74S752
74S753
74S754
74S755
74S756
74S757
74S758
74S759
74S760
74S761
74S762
74S763
74S764
74S765
74S766
74S767
74S768
74S769
74S770
74S771
74S772
74S773
74S774
74S775
74S776
74S777
74S778
74S779
74S780
74S781
74S782
74S783
74S784
74S785
74S786
74S787
74S788
74S789
74S790
74S791
74S792
74S793
74S794
74S795
74S796
74S797
74S798
74S799
74S800
74S801
74S802
74S803
74S804
74S805
74S806
74S807
74S808
74S809
74S810
74S811
74S812
74S813
74S814
74S815
74S816
74S817
74S818
74S819
74S820
74S821
74S822
74S823
74S824
74S825
74S826
74S827
74S828
74S829
74S830
74S831
74S832
74S833
74S834
74S835
74S836
74S837
74S838
74S839
74S840
74S841
74S842
74S843
74S844
74S845
74S846
74S847
74S848
74S849
74S850
74S851
74S852
74S853
74S854
74S855
74S856
74S857
74S858
74S859
74S860
74S861
74S862
74S863
74S864
74S865
74S866
74S867
74S868
74S869
74S870
74S871
74S872
74S873
74S874
74S875
74S876
74S877
74S878
74S879
74S880
74S881
74S882
74S883
74S884
74S885
74S886
74S887
74S888
74S889
74S890
74S891
74S892
74S893
74S894
74S895
74S896
74S897
74S898
74S899
74S900
74S901
74S902
74S903
74S904
74S905
74S906
74S907
74S908
74S909
74S910
74S911
74S912
74S913
74S914
74S915
74S916
74S917
74S918
74S919
74S920
74S921
74S922
74S923
74S924
74S925
74S926
74S927
74S928
74S929
74S930
74S931
74S932
74S933
74S934
74S935
74S936
74S937
74S938
74S939
74S940
74S941
74S942
74S943
74S944
74S945
74S946
74S947
74S948
74S949
74S950
74S951
74S952
74S953
74S954
74S955
74S956
74S957
74S958
74S959
74S960
74S961
74S962
74S963
74S964
74S965
74S966
74S967
74S968
74S969
74S970
74S971
74S972
74S973
74S974
74S975
74S976
74S977
74S978
74S979
74S980
74S981
74S982
74S983
74S984
74S985
74S986
74S987
74S988
74S989
74S990
74S991
74S992
74S993
74S994
74S995
74S996
74S997
74S998
74S999
74S1000

CODE IN LINE 1 REM. OF CLOCK READ PROGRAM

ADDR		DATA	OPCODE	FUNCTION
685B	ENTRY	E5	PUSH HL	Save registers
685C		D5	PUSH DE	
685D		C5	PUSH BC	
685E		3E0F	LD A, 0F	PIO port B output only
6860		D3DB	OUT (DB), A	Set it up
6862		0ED9	LD C, D9	PIO port A command reg.
6864		21AA68	LD HL, PIO	Configuration table addr.
6867		0604	LD B, 04	4 byte transfer
6869		EDB3	OTIR	Set up port A
686B		3E40	LD A, 40	HOLD bit for clock chip
686D		D3D8	OUT (D8), A	Send it
686F		0E22	LD C, 22	150 uSec. delay routine
6871	DELAY	0D	DEC C	for HOLD setup
6872		20FD	JRNZ, DELAY	
6874		C620	ADD A, 20	Add READ bit
6876		D3D8	OUT (D8), A	Send READ + HOLD
6878		1EDA	LD E, DA	PIO port B data addr.
687A		16D8	LD D, D8	PIO port A data addr.
687C		21EF5E	LD HL, 5FEF	First data store addr.-1
687F		06C0	LD B, C0	First clock chip addr.
6881	LOOP	CDAE68	CALL STORE	Store data
6884		F5	PUSH AF	Store character
6885		78	LD A, B	Get current clock addr.
6886		D610	SUB A, 10	Decrement it
6888		47	LD B, A	Put back into B
6889		FE70	CP A, 70	Is it 1's day addr.?
688B		280B	JRZ, MASK	Jump if so
688D		FE40	CP A, 40	Is it 1's hours addr.?
688F		2807	JRZ, MASK	Jump if so
6891		FEF0	CP A, F0	Is it done yet ?
6893		2808	JRZ, END	If so, wrap it up
6895		F1	POP AF	Get ASCII data back
6896		18E9	JR, LOOP	Continue until done
6898	MASK	F1	POP AF	Get ASCII data back
6899		E6F3	AND A, F3	Mask off AM/PM,24 hr,29 day
689B		18E4	JR, LOOP	Continue until done
689D	END	F1	POP AF	Get ASCII data back
689E		0600	LD B, 00	Units seconds address
68A0		CDAE68	CALL STORE	Store last data char.
68A3		AF	XOR A	Clear A
68A4		D3D8	OUT (D8), A	Clear READ & HOLD
68A6		C1	POP BC	Restore registers
68A7		D1	POP DE	
68A8		E1	POP HL	
68A9		C9	RET	Done
68AA	PIO	CF		Select mode 3 on PIO chip
68AB		0F		Select I/O bits
68AC		37		Interrupt control byte
68AD		F0		Mask byte
68AE	STORE	77	LD (HL), A	Store ASCII character
68AF		23	INC HL	Next addr.
68B0		4B	LD C, E	Get PIO port B data addr.
68B1		ED41	OUT (C), B	Clock chip addr. in B
68B3		4A	LD C, D	Get PIO port A data addr.
68B4		000000	NOP's	Wait for clock addr. delay
68B7		ED78	IN A, (C)	Get clock chip data
68B9		E60F	AND A, 0F	Mask off bits 4-7
68BB		C630	ADD A, 30	Convert to ASCII
68BD		C9	RET	Back to main routine



Note; +5V wire = R
Gnd wire = BLK

Cut original circuit traces to:
pin 20 of XROM socket
pin 35 of J4 (Cartridge socket)

BUSISO line is used to disable XROM/DOCK banks when an expansion bank is active.

UPDATE July 1988
Extra Memory

Rev. Ø1

ISSUE

ISSUE	ENGR C9	TITLE T/S 2068 Modifications to Provide Proper Address Decoding for XROM to allow full use of XROM bank addresses		
	DRAWN 5/5/87		NO. OF SHEETS PER SET 1	SHEET 1

PRINTED IN U.S.A. E-1812-A-3 (3/84)

"LARKEN DISK DOCTOR"

By George Chambers

UPDATE July 1988
Programming

This utility will be found useful in the inspection, analysis, and correction of errors on disks used on the LARKEN DD system. This utility is designed for use with the Larken 360K per disk I/F board.

Entering the listing is straight-forward, however there are a couple of things which will not be evident from the listing.

Firstly, there are a number of buried color codes in the program which contribute to it's ease of use.

A buried color code is entered into a listing by going into the "extended" mode, then pressing one of the keys 0 to 7 (the color keys). This has the effect of immediately changing the listing color. This effect also appears in program operation, which is why we are doing it. This new PAPER color will continue through the listing and the program, so it is necessary to make a change back to the normal PAPER color immediately after the effect is no longer wanted. Do this by entering a new color using the same routine. Usually this will be the "extended mode" key 7, to give a PAPER WHITE effect. If this command has been inadvertently entered incorrectly it can be deleted using the DELETE key in the normal manner except that you will find that two operations of the DELETE key are required.

Lines which require this color switch are as follows:

LINE	1060 yellow:	"½"
"	1070 green:	"-"
"	2510 red:	"REMOVE PROTECT TABS"
"	" red:	"while using Rename"
"	2520 yellow:	"ready?"
"	4060 green:	"initial letter of each command"
"	4140 yellow:	"*"
"	4320 yellow:	initial letter of each command
"	4350 yellow:	letters Q and A
"	4640 yellow:	initial letters of "Same" & "New"
"	7601 Red:	"Remove WRITE PROTECT sticker"

You may also choose to highlight the REM statements. (Don't forget to insert the"extended" mode 7 after each of these entries)

Save the program by GO TO 110. Or to make it an AUTOSTART program by changing the line to read:

110 RANDOMIZE USR 102

The program makes use of the LARKEN "Hcode.C1" disk DOS routine, so it must be on the same program disk.

The program has been written to allow compilation by TIMEMACHINE. Lines 120 and 2535 would need to be removed however, due to the inability of TIMEMACHINE to handle uniquely LARKEN commands.

OPERATION:

Upon running the program you will be presented with a menu offering 8 choices.

Option 1. will give a screen display of the USED/IDLE status of all the tracks on the disk under scrutiny. As written it will handle DSDD disks correctly.

Option 2. is a Program Header Reader, which gives a track by track report of programs on the disk.

Option 3. provides track information about each program on the disk taken from the directory track.

Option 4 provides the capability of examining and modifying the data on a track. Most useful when used on track 0; the directory track.

Option 5. Offers a program rename capability.

Option 6. A Disk Formatting routine.

Option 7. To select a particular drive.

Option 8. To quit this program. Calls up a Menu.B1 program. Can be modified to suit needs.

The use of option 4 requires more explanation. It's biggest use will be on track 0 as mentioned earlier. When Option 4 is called up you will be asked for a track number. Most often wanted will be track 0, the track which holds the disk directory. Enter a 0. The program then proceeds to load the contents of that track into a buffer area in the computer starting at address 45000. It is at this point that we can access the buffer, inspect it, make changes to it, and resave it, with or without changes as desired.

First we will briefly cover the makeup of track 0. The first 168 bytes of this track are concerned with disk status. After that comes a series of "cells" containing a record of one of the programs that has been saved on the disk. Each cell contains the program name and the track numbers that the program occupies, plus 3 or 4 markers used by the DOS to mark the program location. Each cell occupies 34 bytes. There is room for as many as 100 such cells on this track (though rarely are that many used). Towards the end of the track is the disk name.

This is the briefest of outlines of the track makeup; sufficient only to permit an explaination of the use of the commands shown in option 4.

When Option 4 has been selected and track 0 chosen you will be presented with a display of the contents of that track starting with byte 0. This is not particularly interesting at the moment so press the C (continue) key. This will advance you to the first program name cell. The screen shows the first 17 bytes of this cell. Pressing C again will display the next 17 bytes of the cell. Pressing C key a third time will bring the first half of the second cell onto the screen.

To move back by one screen press the B (back) key. Or, if you wish to move onto the start of the next cell then press the E (etc) key. This will move you forward by a complete name cell, rather than a half a cell. If you wish to move forward even more quickly then press the J (jump) key and you will advance by 5 cells.

If you press the D (dire) key you will jump forward approximately 3600 bytes to where the Disk name is stored. Or by pressing the T (top) key you will return

to the very start of the buffer where you first entered. These commands then, control your movement through the buffer.

Now to cover those commands which allow you to modify the buffer.

Press the A (alter) key, and you will be presented with the option "Name or Track". Choose N if you wish to change a program name, or T if track numbers are to be modified. A cursor will appear on screen, and you can position it on the screen by using keys Q, A, and ENTER. This will determine where the new entry is to start. Then enter the desired changes. Terminate the entry and you will see the changes you have made reappear on the screen.

When you have made the desired changes you may then choose to save them to disk by pressing the S (save) key. You will be asked whether you wish to save the buffer to the same track (usually) or to a new track. The L (load) key permits you to load another track into the buffer, while the M (menu) key will bring back the main menu.

This has necessarily been a very abbreviated description of the operation of this program, and it may be adequate for some. If wish a more complete set of instructions you may obtain these by sending $3 to my address shown at the top of this listing.

```
  1 REM  George Chambers,
14 Richome Court                    Scarborou
gh, Ontario                CANADA   M1K 2Y1

 100 GO TO VAL "160"
 110 RANDOMIZE USR VAL "100": SAVE "doctor
.B1"
 120 CLEAR VAL "39999": RANDOMIZE USR VAL
"100": LOAD "Hcode.C1"CODE VAL "40000"
 160 DIM a$(640)
 170 LET oo=0: LET oa=1: LET ob=2: LET oc=
3
 180 LET drive=VAL "43301": LET dr=oo
 190 LET settrack=VAL "40003"
 200 LET nexttrack=VAL "40006"
 210 LET sprt=VAL "40009"
 220 LET load=VAL "40012"
 230 LET save=VAL "40015"
 240 LET track=VAL "43300"
 250 LET drive=VAL "43301"
 260 LET buffer=VAL "45000": LET bufflenth
=VAL "5120": LET lastcell=VAL "4500": LET
 diskname=4483
 270 POKE drive,ob
 280 LET treg=VAL "18"
 290 LET side=ob
 300 GO TO VAL "5000"
1000 REM Track Used/Free Routine
1005 CLS : LET number=oo: GO SUB VAL "7500
"
1010 LET c=oo: LET ot=c: LET ft=c
1041 FOR n=buffer+VAL "25" TO buffer+VAL "
103"
1050 LET t=PEEK n
1060 IF t=VAL "245" THEN  LET n$=">Used":
LET ft=ft+oa: GO TO VAL "1080"
1070 IF t<>VAL "245" THEN  LET n$="-Open":
 LET ot=ot+oa
1080 PRINT TAB (c);(n-(buffer+VAL "24"));T
AB (c+ob);n$;
1090 LET c=c+VAL "8"
1100 IF c=VAL "32" THEN  LET c=NOT c
1110 NEXT n
1120 PRINT AT VAL "20",oo;"Tracks Used: ";
ft;
1130 PRINT AT VAL "21",oo;"Tracks Open: ";
ot
1150 PRINT #0;"z to copy-Any other key for
 menu"
1170 PAUSE oo
1180 POKE VAL "23658",oo: LET i$=INKEY$
1190 IF i$="z"  THEN  COPY
1200 GO TO VAL "5000"
2000 REM  Program Header Reader
2010 LET m=oo
2020 CLS : PRINT '" This option displays t
he file  information saved on each block
 along with the data."
2030 PRINT '"TRACK  - The track number on
 disk"
2040 PRINT '"NAME - The file name is save
d  on each block for reference only"
2050 PRINT '"START - The start address of
 the file. Useful when saving code."
2060 PRINT : PRINT '"LENGTH - The total le
ngth of the file."
2080 PRINT #0;"Press any key when ready":
PAUSE oo
2090 CLS : PRINT "     DISK DATA-HEADER RE
ADER    ---------------------------------"

2100 PRINT "TRACK    NAME    START   LENGT
H"
2110 PRINT "-----------------------------
--"
2120 PRINT AT VAL "21",VAL "7";"(M returns
 to menu)": PRINT AT oc,oo
2130 POKE track,oa: RANDOMIZE USR settrack

2140 PRINT "0   Disk Directory Track"
2150 FOR t=oa TO VAL "79":
2160 IF PEEK VAL "23689"<VAL "7" THEN  GO
SUB VAL "2340"
2170 IF IN 16=oo THEN  RANDOMIZE USR settr
ack
2180 RANDOMIZE USR load
2190 PRINT PEEK (buffer+oa);TAB VAL "5";
2200 IF PEEK (buffer+oa)<>oo THEN : IF PEE
K (buffer+oa)<>t THEN  PRINT : PRINT "Tra
ck Number Error on Track ";t
2210 IF PEEK (buffer+oa)=oo THEN  PRINT :
GO TO VAL "2280"
2220 FOR a=(buffer+ob) TO (buffer+10)
2230 PRINT CHR$ PEEK a;
2240 NEXT a
2250 PRINT TAB VAL "17";PEEK (buffer+VAL "
12")+VAL "256"*PEEK (buffer+VAL "13");
2260 PRINT TAB VAL "26";PEEK (buffer+VAL "
22")+VAL "256"*PEEK (buffer+VAL "23")
2270 REM  PRINT TAB VAL "26";PEEK (buffer+
VAL "17")+VAL "256"*PEEK (buffer+VAL "18"
)+oa
2280 IF side=oa THEN  LET t=t+oa: POKE tra
ck,(PEEK track)+oa
2290 RANDOMIZE USR nexttrack
2300 IF INKEY$="m" OR INKEY$="M" THEN  GO
TO VAL "5000"
2310 NEXT t
2320 PRINT "Last track-Press key for menu"
: PAUSE oo
2330 GO TO VAL "5000"
2340 PRINT 'TAB oc;" Press z key to copy o
r          any key to clear screen"
2350 IF INKEY$="" THEN  GO TO VAL "2350"
2360 IF INKEY$="z" OR INKEY$="Z" THEN  COP
Y
2370 IF INKEY$="m" OR INKEY$="M" THEN  GO
TO VAL "5000"
2380 PRINT AT VAL "4",oo;a$( TO VAL "544")
: PRINT AT oc,oo
2390 RANDOMIZE USR load
2400 RETURN
2500 REM Rename Program Routine
2510 CLS : PRINT AT VAL "5",VAL "7";"Remov
e Protect TabS                while using Re
name."
2515 PRINT '''"Rename a disk program
     Insert disk"
2520 PRINT '''"ready?": PAUSE oo
2525 DIM x$(VAL "9"): DIM y$(VAL "9"): DIM
 z$(VAL "9"): DIM a(VAL "79")
2530 LET t=oa
```

```
2535 CLS : PRINT USR VAL "100": CAT "",
2540 INPUT "change name from "; LINE y$
2545 CLS : PRINT AT VAL "10",NOT PI;"NOTE:
Name search can take 2 min."
2550 GO SUB VAL "8010"
2555 POKE track,oo: RANDOMIZE USR settrack
: RANDOMIZE USR load
2560 FOR i=buffer+VAL "188" TO buffer+VAL
"4500"
2565 IF PEEK i=VAL "255" THEN  LET k=i+oa:
 FOR j=oa TO 9: LET x$(j)=CHR$ PEEK (i+j)
: NEXT j: IF x$=y$ THEN  INPUT "change na
me to "; LINE z$: GO TO VAL "2580"
2570 NEXT i
2575 CLS : PRINT AT VAL "10",oo;"Program n
ot found:""'"Press any key for menu": PAUS
E oo: GO TO VAL "5000"
2580 FOR i=oa TO VAL "9": POKE k,CODE z$(i
): LET k=k+PI/PI: NEXT i
2585 POKE track,oo: RANDOMIZE USR settrack
: RANDOMIZE USR save
2590 IF PEEK k<>VAL "253" THEN  LET k=k+oa
: GO TO VAL "2590"
2595 IF PEEK (k+oa)=VAL "249" OR t>VAL "79
" THEN  GO TO VAL "2610"
2600 LET k=k+oa: IF PEEK k>oo AND PEEK k<V
AL "80" THEN  LET a(t)=PEEK k
2605 LET t=t+oa: GO TO VAL "2595"
2610 LET t=oa: GO TO VAL "2620"
2615 IF a(t-1)+1=a(t) THEN  RANDOMIZE USR
nexttrack: RANDOMIZE USR load: GO TO VAL
"2625"
2620 POKE track,a(t): RANDOMIZE USR settra
ck: RANDOMIZE USR load:
2625 FOR l=oa TO 9: POKE buffer+1+l,CODE z
$(l): NEXT l
2630 RANDOMIZE USR save
2635 LET t=t+oa: IF a(t)<>oo THEN  GO TO V
AL "2615"
2640 CLS : PRINT  PAPER ob; INK 7;"Program
 has been renamed   "'"from ";y$;" to ";z
$
2645 PRINT #0;"Press ENTER to rename anoth
er    program, or ""m"" for Main Menu."
2650 PAUSE oo: IF INKEY$=CHR$ 13 THEN  GO
TO 2515
2655 GO TO VAL "5000"
2660 STOP
3000 REM  Dir Analysis Routine
3010 CLS : LET number=oo: GO SUB VAL "7500
"
3020 LET fn=oo: LET d=oo: LET n=buffer+VAL
 "188"
3030 PRINT #1;"z=copy  c=cont  m=menu  t=t
op"
3050 IF PEEK VAL "23689"<6 THEN  PRINT AT
oo,oo;a$: PRINT AT oo,oo;"* Directory / F
ile Data Report *"
3060 LET fn=fn+oa
3070 PRINT "Filename ";fn;":";TAB 13;
3080 LET n=n+oa
3090 IF PEEK n=253 THEN  GO TO VAL "3130"
3100 IF PEEK n=VAL "254" THEN  GO TO VAL "
3220"
3110 PRINT  PAPER 6;CHR$ PEEK n;: PAPER 7
3120 GO TO VAL "3080"
3130 PRINT TAB oo;"Tracks used: "
3140 LET ft=oo
3150 LET c=oo
3160 LET n=n+oa
3170 LET ft=ft+oa
3180 IF PEEK n=VAL "249" THEN  GO TO VAL "
3240"
3190 PRINT (PEEK n);",";
3200 LET c=c+oc
3210 GO TO VAL "3160"
3220 PRINT ;"Empty name cell"
3230 GO TO VAL "3260"
3240 PRINT 'ft-oa;" ";bufflenth;" Byte) Bl
ock(s)."
3250 PRINT "-----------------------------
--" AND (d=0)
3260 LET n=n+1
3270 IF n=buffer+lastcell THEN  PRINT '"La
st name cell.Press m for menu": PAUSE oo:
 GO TO VAL "5000"
3280 IF PEEK n=255 THEN  GO TO 3300
3290 GO TO 3260
3300 POKE 23658,0: LET i$=INKEY$
3310 IF i$="z" THEN  COPY
3320 IF i$="c" THEN  GO TO 3050
3330 IF i$="m" THEN  GO TO 5000
3340 IF i$="t" THEN  GO TO 3020
3350 GO TO 3300
4000 REM Track Manipulation
 Routines
4010 CLS : LET m=oo
4020 PRINT "Examine which track? (0-79)"
4030 INPUT number
4040 GO SUB VAL "7500"
4050 GO TO VAL "4080"
4060 BORDER 7: PRINT  INK 7;#0;" alter bac
k cont dire etc jump5  load  menu name sa
ve top"
4070 RETURN
4080 CLS
4090 GO SUB 4060
4100 FOR n=buffer TO buffer+bufflenth STEP
 17
4110 PRINT AT 0,6; PAPER 6;"Reading track
No ";number
4120 PRINT AT 1,29;"  ";AT 1,1;"addr    ma
p    byte  chr$"
4130 FOR x=n TO n+16
4140 PRINT x;TAB 9;x-buffer;TAB 17;PEEK x;
"  ";TAB 25;: PRINT CHR$ PEEK x AND (PEEK
 x>31 AND PEEK x<123);"*" AND (number=0 A
ND PEEK x>248 AND PEEK x<256);: PRINT "
"
4150 NEXT x
4160 GO TO 4180
4170 NEXT n
4180 POKE 23658,0: LET i$="": LET i$=INKEY
$
4190 IF i$="c" AND n<=50111 THEN  GO TO 41
70
4200 IF i$="l" THEN  GO TO 4000
4210 IF i$="a" THEN  GO TO 4310
4220 IF i$="s" THEN  GO TO 4630
4230 IF i$="m" THEN  GO TO 5000
4240 IF i$="n" THEN  LET n=buffer+171: GO
TO 4170
4250 IF i$="e" AND n<=50094 THEN  LET n=n+
17: GO TO 4170
4260 IF i$="b" THEN  LET n=n-34: GO TO 417
0
4270 IF i$="j" AND n<=49924 THEN  LET n=n+
187: GO TO 4170
4280 IF i$="d" THEN  LET n=buffer+diskname
: GO TO 4170
4290 IF i$="t" THEN  LET n=buffer-17: GO T
O 4170
4300 GO TO 4180
4310 IF INKEY$<>"" THEN  GO TO 4310
4320 POKE 23658,0: LET i$="": PRINT AT 20,
0;"Change Name or Track (N/T)? ": LET i$=
INKEY$
4330 IF i$<>"n" AND i$<>"t" THEN  GO TO 43
20
4340 IF INKEY$<>"" THEN  GO TO 4340
4350 PRINT AT 19,0;"    Select Start addre
ss              Use keys Q, A, and ENTER"
4360 LET y=x-17: LET z=2
4370 PRINT AT z,16; FLASH 1;">"; FLASH 0
4385 IF INKEY$="a" AND z<18 THEN  LET y=y+
1: PRINT AT z,16;" ";AT z+1,16; FLASH 1;"
>"; FLASH 0: LET z=z+1
4390 IF INKEY$="q" AND z>2 THEN  LET y=y-1
: PRINT AT z,16;" ";AT z-1,16; FLASH 1;">
"; FLASH 0: LET z=z-1
4400 IF INKEY$=CHR$ 13 THEN  LET addr=y: G
O TO 4430
4410 GO TO 4380
4420 IF addr<buffer OR addr>bufflenth THEN
  PRINT AT 20,0;"Address out of range"'"P
ress any key to re-enter": PAUSE 0: GO TO
 4390
4430 IF i$="t" THEN  PRINT AT 19,0; INK 2;
"                                      En
ter (2068) to stop     "
4440 IF i$="n" THEN  INPUT "Enter Prgm Nam
e: ";n$: GO TO 4530
4450 IF i$="t" THEN  INPUT "Enter Track nu
mber: ";z: GO TO 4470
4460 IF i$<>"t" AND z$<>"n" THEN  GO TO 44
30
```

```
4470 IF z=2068 THEN  GO TO 4580
4480 IF PEEK addr=255 OR PEEK addr=253 THE
N  PRINT AT 20,1; INK 2;"Overwriting a ma
rker at ";addr;TAB 32'"Do you wish to con
tinue? (y/n)": INK 0: PAUSE 0
4485 POKE 23658,0: IF INKEY$="n" THEN  GO
TO 4580
4490 POKE addr,z
4500 IF z=249 THEN  GO TO 4580
4510 LET addr=addr+1
4520 GO TO 4450
4530 FOR a=1 TO LEN n$
4540 IF PEEK addr=255 OR PEEK addr=253 OR
PEEK addr=249 THEN  PRINT AT 20,1; INK 2;
"Overwriting a marker at ";addr;TAB 32'"
Do you wish to continue?(y/n)": INK 0: PA
USE 0
4545 POKE 23658,0: IF INKEY$="n" THEN  GO
TO 4580
4550 POKE addr,CODE n$(a)
4560 LET addr=addr+1
4570 NEXT a
4580 CLS : GO SUB 4060: GO TO 4110
4590 IF LEN p$>=1 THEN  LET p$=p$( TO LEN
p$-1)
4600 PRINT AT lin,23;p$;i$;" "
4610 IF INKEY$<>"" THEN  GO TO 4610
4620 GO TO 4580
4630 IF INKEY$<>"" THEN  GO TO 4630
4640 POKE 23658,0: LET i$=INKEY$
4650 INPUT "Save to Same or New track? ";i
$
4660 IF INKEY$<>"" THEN  GO TO 4660
4670 IF i$="s" THEN  GO TO 4700
4680 IF i$="n" THEN  GO TO 4720
4690 IF i$<>"s" OR i$<>"n" THEN  GO TO 464
0
4700 GO SUB VAL "7600"
4710 GO TO VAL "4080"
4720 INPUT "New track number? ";number
4750 IF INKEY$<>"" THEN  GO TO VAL "4750"
4760 IF number>VAL "79" THEN  GO TO VAL "4
080"
4770 GO SUB VAL "7600"
4780 GO TO VAL "4080"
4790 LET p$=p$( TO LEN p$-1)
4800 PRINT AT VAL "18",VAL "20";p$;i$;" ";
 PAPER VAL "7"
4810 GO TO VAL "4730"
4820 COPY
4830 GO TO VAL "4180"
5000 REM   Menu
5010 BORDER ob: INK oo: PAPER 6: CLS
5020 PRINT "          DISK DOCTOR      v2.
25"
5030 PRINT "        For the TS 2068
       and LARKEN DSK400
  G.F. Chambers"
5040 PRINT AT VAL "5",VAL "4";"1. Tracks U
sed/Free"
5050 PRINT '"    2. Program Header Reader"

5060 PRINT '"    3. Directory Analysis"
5070 PRINT '"    4. Examine/Modify a Track
"
5080 PRINT '"    5. Rename a Program"
5090 PRINT '"    6. Format a Disk"
5100 PRINT '"    7. Select Drive (Now Drv
"; FLASH oa;dr; FLASH oo;")"
5110 PRINT '"    8. Exit to Main Disk Menu
"
5120 PRINT '"      >";"PRESS A KEY": PAUSE
 oo
5130 PAPER 7: LET i$=INKEY$
5140 IF i$<"1" OR i$>"8" THEN  GO TO VAL "
5130"
5150 IF i$="1" THEN  GO TO VAL "1000"
5160 IF i$="2" THEN  GO TO VAL "2000"
5170 IF i$="3" THEN  GO TO VAL "3000"
5180 IF i$="4" THEN  GO TO VAL "4000"
5190 IF i$="5" THEN  GO TO VAL "2500"
5200 IF i$="6" THEN  GO TO VAL "6000"
5210 IF i$="7" THEN  PRINT AT 21,oo;a$( TO
 32): GO SUB VAL "8010": GO TO VAL "5000"

5220 IF i$="8" THEN  GO TO VAL "8000"
5300 STOP
6000 GO SUB VAL "8010": PAPER 7: CLS : PRI
NT "FORMAT PROGRAM for LARKEN DSK400"
6010 PRINT : PRINT ;"ENTER DISK NAME OR DI
SK DESCRIP-TION UP TO 1000 CHARACTERS .
   PRESS DELETE TO BACKSPACE       PRESS
'STOP' TO END DISK NAME  "
6020 PRINT  INK 1;"You are in Drive ";dr:
INK 0
6030 PRINT "Disk Name : ";
6040 LET m=VAL "41000"
6050 LET n$=INKEY$: IF INKEY$="" THEN  GO
TO VAL "6050"
6060 IF CODE n$=oo THEN  GO TO VAL "6050"
6070 BEEP .02,oa
6080 IF CODE n$=12 THEN  PRINT CHR$ 8;CHR$
 32;CHR$ 8;: LET m=m-oo: GO TO VAL "6130"

6090 IF CODE n$=VAL "226" THEN  POKE m,oo:
 GO TO VAL "6150"
6100 POKE m,CODE n$
6110 PRINT n$;
6120 LET m=m+oa
6130 IF INKEY$<>"" THEN  GO TO VAL "6130"
6140 GO TO VAL "6050"
6150 POKE m,oo: PAUSE VAL "100": CLS
6190 POKE VAL "43312",oo: REM  6=0 12=1 20
=2  30=3
6200 INPUT "Head speed 6 12 20 30 MS ? ";h

6210 IF h=6 OR h=12 OR h=20 OR h=30 THEN
GO TO VAL "6230"
6220 GO TO VAL "6200"
6230 IF h=6   THEN  LET a=oo
6240 IF h=12 THEN  LET a=oa
6250 IF h=20 THEN  LET a=ob
6260 IF h=30 THEN  LET a=oc
6270 POKE VAL "43312",a
6280 INPUT "Number of Sides ";a
6290 IF a=oa THEN  GO TO VAL "6320"
6300 IF a=ob THEN  GO TO VAL "6320"
6310 GO TO VAL "6280"
6320 POKE VAL "43310",a
6330 PRINT AT 8,8;a;" Sided"
6340 INPUT "Number of Tracks ";a
6350 POKE VAL "43311",a
6360 PRINT AT 9,8;a;" Tracks"
6370 INPUT "Ok to Format ";n$
6380 IF n$<>"y" THEN  CLS : PRINT "OK TO R
UN": STOP
6390 RANDOMIZE USR VAL "40000"
6400 CLS : BEEP .2,10: BEEP .2,-2
6410 GO TO VAL "5000"
7000 STOP
7500 REM Load a Track
7511 POKE track,number: RANDOMIZE USR sett
rack: RANDOMIZE USR load
7550 RETURN
7600 REM Save a Track
7601 CLS : PRINT AT 10,6;"Writing to track
 ";number'"  Remove WRITE PROTECT sticke
r,      then press any key"
7603 IF INKEY$<>"" THEN  GO TO 7603
7604 PAUSE oo
7611 POKE track,number: RANDOMIZE USR sett
rack: RANDOMIZE USR save
7650 RETURN
8000 STOP
8010 INPUT "Drive 0 to 3 ? ";h
8020 LET dr=h
8025 LET h=h+oa
8030 LET h=2^h
8040 POKE drive,h
8050 RETURN
9500 RANDOMIZE USR VAL "100": LOAD "menu.B
1"
9900 SAVE "doctor.B1"
```

Note: This excellent LARKEN program cannot be included in July/April Issue disk because the disk space is maxed out and there is no space left. But not to worry! This and another absolutely GREAT Disk Manager program (for next issue) will be included in the next issue disk, along with some Larken programs by Larry Kenny.

The BUDGET SOFTWARE, PART II

Continuation from the April UP-DATE issue

This BIG software began in the April issue. Some repetition is included for convenience of reading. There are several ways of attacking this group of programs. You can key in the listings, which will give you a operating BUDGET and SPREADSHEET program "in basic". Then you can use the TIMACHINE compiler program to convert the two Basic programs into fast running Machine Code. This will require you to have or to buy the Timachine software from Novelsoft or a distributor. You could use a different Basic to MC Compiler, but if you do you'll be on your own! Of course, the easy way and likely the most practical is to order the April Issue disk to get the fully operational software (and a bunch of other programs and utilities).

The listings are given for those who want to go the educational way of compiling the basic programs with Timachine, which will be the equilivant of a college semester on programming, except there are no tests (but no Girls either!). At my age the latter poses no problem of social signficance. If going this route, use the Check Sum tables given in the April and this issues for the two programs that are to be compiled. There are two things to watch for. One is the similarity of the characters "i" (eye), "l" (el), and "1" (one). Take a look on page 9-18 of the April issue, Line number 40. Notice the difference in the number 1 (one) and lower case "L". There is a difference but it is small. Also notice in line 320, that "i" can be mistaken for lower case "L".

There is one more key-in error to watch for. Several LET and PRINT statements must be entered in the INVERSE mode (HOLD CAPS SHIFT and touch 4) after the first quote, then (CAPS SHIFT and touch 3) before the last QUOTE. These highlight the menus. Line numbers that require this INVERSE print are the LET and PRINT lines of the BUDGET.BJ program: lines 470, 480, 490, 2250, 2620, 2930, 3080, and 3240. Each failure to use INVERSE in these lines will result in the check sum table being off 4 bytes for each omission. The check sum table is given in ten line groups. Type CLEAR ENTER, then PRINT FREE. The check sum will not be accurate unless you started the key in project with a cleared computer (OFF and then ON) and 38652 FREE bytes. Loaded Code programs, such as a printer driver code, will throw off the check sum figures. Solve this by OFF, then RE-LOAD.

The "SPREAD.BJ" program (given in this issue) is the second BASIC program listing which already includes the Timachine commands needed for COMPILING the Basic program into Machine Code. The Compiler commands begin with (REM!) and can be entered directly in each line as given. Once finished with the program Key-in, SAVE it to Disk or Cassette. Then LOAD the TIMACHINE program and pick up the instructions given in Bob Mitchell's Documentation in this issue "Part II". I would advise that you NOT make any program changes for your first attempt at keying in and compiling the SPREAD.BJ program. Later, after the whole integrated program group is checked out, you can then feel confident to go back and make custom changes to the two Basic programs "BUDGET.BJ and SPREAD.BJ", and re-compile these custom programs which includes your changes.

Now to discuss the additions made by UP-DATE for the three Disk Operating Systems. This involves the two ancillary LOADER and DISK MENUS. For identification, Bob calls "budget.Bx" a "loader / manager". UP-DATE's listings names this "BUDGET.B1" and gives three different DOS versions. It was found that some conflicts existed when a previously loaded CODE program had not been eliminated before loading the budget and spread code programs from disk. Therefore, the "Cclear" CODE program was added to clear memory when the budget program is first loaded. See "Cclear explanation in this issue. Also, the re-set of RAM TOP was not included. The large budget and smaller spread codes are loaded without change of RAM TOP. This eliminated a chancy stoppage report of "out of memory" with one of the DOS versions.

The DISK MENUS: Bob made mention of adding a disk menu, and UP-DATE added one for each of the three DOS versions. These disk menus can easily be adapted to other uses for coordinating a different set of programs. In this instance, each of the Disk Menus give more options and allows you to place the Budget software on stand-by to do other computer functions. Beware of the fact that there isn't much FREE memory to add other programming lines. However, the menus are designed so that if you should happen to overwrite the Budget Code, it can be brought back in to resume. You would just lose the data "in the program that was in memory". Start up can be by loading either BUDGET.B1, or Menu.B1 first. The Issue Disk uses a jazzy kaleidoscope auto boot utility, which also can be re-used for other effects.

The Budget Program ensemble takes up a lot of space in UP-DATE, is NOT a simple "week end key-in project", but does provide an extensive and complete software system to explore over a long period of time. Please feel free to call or write if you get stuck on something. For those of you who really want to delve into the programming, sufficient documentation is given to use your imagination and make this BUDGET software into a variety of different Spreadsheet applications. Be Bob Mitchell's guest! By the way, I changed to a new print wheel that gives a better Character "i" (eye), but alas, the ones (1) and lower case "L" look the same. So, I will circle lower case "L" in the listings when not obvious. If used alone and not circled, its a "1" (one).

The Aerco prcode is used for the LARKEN DOS and the AERCO FD-68 DOS versions. For Larken DOS, this is the early cassette code as supplied with the Aerco printer interface (starting address 64000 and 800 bytes). For the FD-68 DOS, the code is that supplied with the Aerco FD-68 and Jack Dohany's "RALOADER" program, which is re-locatable (Starting address is 64500 and is 867 bytes). For both see the utility program listing "INIT" given in this issue.

Bob Mitchell's documentation which follows contains a notation at the top left corner of pages. This is the TASWORD file program which contained the text sent to UP-DATE by Bob. The print outs of the documentation were done with the utility "MS-TAS" (Listing is given in this issue). Next will be given the "SPREAD.CJ" basic listing to be compiled by use of TIMACHINE, and following that will be Bob Mitchell's "Part II" documentation, and finally the Basic loader "BUDGET.B1" and the Disk Menu "Menu.B1" will be given. The utilities "MS-TAS" and "INIT" are included in the April Issue Disk", along with the complete BUDGET ensemble and several more programs. See the April Issue Catalog given elswhere.

Code pgm, "bujJ3.CT"33280,11200

DOCUMENTATION FOR BUDGET
PART II
By Bob Mitchell, Willowdale, Ontario,
Provided to TS Update Newsletter
(Revision dated 880414)

========== WHAT UTILTIES YOU WILL NEED ==========

1. A copy of the BASIC compiler "Timachine" (the amended version that handles the *D and *E routines properly) to compile "budget" and "spread".

2. A Centronics printer interface code; "budget.Bx" as shown uses the "aerco" prcode or, alternatively, the Larken extended BASIC printer code contained in its EPROM cartridge.

3. Copies of the Tasman utility "taswide" and the "tasword" word processor program are optional. With these you can type up some help pages and recall them when needed to refresh your memory while using "budget". More on that later.

========== THE MAIN PROGRAMS ==========

4. There are three programs:

a) budget.Bx — expanded "loader/manager" or input/output program; this will NOT be compiled.

b) budget.BJ — for compilation; change the category names in line 590 to suit your needs, before you do the compilation. Pad out each name in this line with spaces to a total length of ten.

c) spread.BJ — also for compilation; change line 8250 to match the names in line 590 in your version of budget.BJ. No extra spaces will be needed in line 8250. This program will run slowly in BASIC providing you are using the Larken cartridge or Aerco "prcode.Cl".

========== THE LOADER/MANAGER ==========

5. TYPE IN THE LOADER, "budget.Bx". Check your typing carefully and SAVE it to disk (GO TO 310); then SAVE it to tape. You may want to make the loader into your autostart program. This option is not on the menu but is located at line 400 (for Larken).

========== BUDGET COMPILATION ==========

6. Now restart your 2068 (turn power off, then on -- disk drive gate OPEN!) and LOAD TIMACHINE. Remember! You must have the corrected version of Timachine. Then LOAD the TAPE version of "budget" that you saved earlier (the one with no budget values). Make the change to line 590 explained above (para 4b). Compile the program; you will have to SAVE IT TO TAPE in the two parts (*D and *E) mentioned earlier . Make sure your 2040 printer is turned on so that you get a printout of the runtimes and variables generated by Timachine during the compilation process.

7. Save this Timachine printout (with variable m/c addresses) for future use. Restart the 2068 and LOAD the TWO parts to the CORRECT ADDRESSES as instructed by the Timachine prompts. RUN the program by entering RANDOMIZE USR 32000. If you do not initialize via 32000 you will get strange results later as some variables will not be set up. If all is well, you should have a pull down menu on the screen. Enter the Date and budget Year to complete this menu.

8. LOAD "budget.Bx" from disk and check the USR addresses and the SAVE code length in it to ensure they are what you got from your Timachine print-out. You will have two entry point addresses, one of which is 32000 and the other which is generated by the second REM ! OPEN # which should be 37166. If it is a different address, then you have altered the length of the program and you will have to change <37166> in loader lines 190 & 260 to your new second entry point address. Ensure that the code length in loader line 40 for the SAVE represents the total of the code lengths from the *D and *E parts plus the length set aside for machine code variables. The latter is the area that will hold the budget amounts. Now ReSAVE "budget.Bx" to disk and just to be safe, to tape,too.

9. Now you are ready to enter your budget amounts for the entire year. It is best to work these out on paper first! Next, enter the actuals for the year so far. Check out the various displays at this point and if you are satisfied, SAVE the compiled program with your budget amounts to disk using option 3 in the loader. A SAVE to TAPE would also be wise. Test it all out by powering down and LOADing "budget.Bx" and then LOAD "budget.CJ" using option 2.

10. When you QUIT "budget" you return to the loader; you can get back to the budget program by using the restart option 4. Nothing will happen when you do this unless budget is loaded. Next time you LOAD budget from the disk, use option 2 to dump all the data for viewing or further entries. You can start afresh (with no data) using Option 1. If you don't want to use help routines take out lines 185 and 186; and change "6" to "5" in line 150. If you are using Larken DOS then delete line 330; remove the token <REM> from lines 65 and 333 and change line 196 to read <IF k$="5" AND PEEK 32000=205 THEN CLS: RANDOMIZE USR 58400: GO TO 60>.

========== SPREADSHEET ==========

11. Key in the spreadsheet program "spread.BJ". Change the DATA in line 8250 to match your budget program category names. No extra spaces are needed here.

12. If you have changed the length of "budget" at all, some changes will be needed. Examine your Timachine printout for "budget" and locate the arrays indicated in lines 8340 to 8370. Change variable <first> to the address of each array in turn. Change variable <last> to the address of the next variable on the printout minus 1. Next, locate the arrays shown in lines 8590 and 8600. Change the variable <i> in each case to the address for that array minus 60. Now, SAVE "spread" to tape and to disk (GO TO 9000 for the latter).

13. Spreadsheet is now ready for use. If you intend to compile it with Timachine, jump to para 14 below and disregard the rest of this para. You may merge "spread.BJ" with the loader "budget.Bx" and use it as a BASIC program entirely. You will have to change <RANDOMIZE USR 58400> TO <GO SUB 8000> in line 195 of the loader. Change line 8620 of "spread.BJ" to read <GO TO 60> in lieu of <STOP>. There will be some other minor changes to do after the MERGE: DELETE line 335; CLEAR 29000 in line 400 will not be high enough if you want to SAVE this as an AUTOSTART version (Larken); try 30000 or higher. Delete line 9000. The spreadsheet will print out quite slowly so be patient.

========== SPREADSHEET COMPILATION ==========

14. Of course, compilation is the way to go! LOAD Timachine and then "spread" from your tape. Do the compilation and SAVE it according to Timachine instructions. Restart the computer and LOAD "spread.CJ"CODE 58400. SAVE it immediately to your disk as "spread.CJ"CODE 58400,length; where the latter is the length shown in the Timachine SAVE instruction at the end of your printout: <SAVE "m/c" CODE address,length>.

15. Now restart the computer and try what you have done. LOAD the loader "budget.Bx" and use option 2 to LOAD your budget code "budget.CJ"; make sure your printer is ready and then use option 5 to print out the spreadsheets. Each of the four sheets should print out in about two minutes.

Code pgm, "bujJx.CT"33280,2944

BUDGET LOADER -- PROGRAM BREAKDOWN

40	SAVE budget with data from menu option 3. The code length 25923 is the sum of the *D and *E parts and the bytes for m/c variables from the Timachine compilation.
60	The menu (Budget Input/Output) starts here; ON ERR RESET restores the BREAK facility which ON ERR GO TO activates elsewhere ; CLOSE #2 restores channel 2 to normal; OUT 127,18 restores the wide printer font to normal PICA size; POKE 26703/4 restores Lprint to the 2040 printer.
65	Larken extended BASIC command to restore Lprint to normal 2040 printer. This line is inactive due to <REM>.
180	SAVE will not take place unless Budget is loaded due to PEEK 32000 which must be 205.
185 & 186	Print "help" routine called here. You must have taswide on your disk as well as the help program. If you BREAK, action goes to line 186 due to ON ERR in line 185, then restores font to normal Sinclair and prints a bottom line prompt "ENTER".
190	Restarts "budget" complete with all data; this will not work if budget is not loaded. Check that the address 37166 is still the correct second entry point from your Timachine operation.
195	This line is set up to reload the aerco prcode which will be destroyed only if the help routine is called. If you are not incorporating the help routine, this line can be changed to remove the commands to reload prcode and do the POKES shown.
220-240	Initializing routine.
250-270	Load budget with all saved data.
290	Save routine starts; goes to line 40.
310	Save the loader.
330	Load the Aerco print code.
333	Optional use of Larken print code.
335	Load the spreadsheet compiled code.
400	Autostart save (Larken).

-30-

Code Pgm, "bujJs.CT"33280.2048

SPREADSHEET -- PROGRAM BREAKDOWN

8000-8030	Timachine instruction.
8050-8210	Print Using routine.
8250-8260	DATA for t$(19,10). The 19th is for "TOTALS>>"
8270	Set the parameters for Print Using routine. The value of u$ is set to cater to numbers as large as 999999.99 which in the budget program will handle up to $99999.99 and still leave a space between columns (months) to make the spreadsheet more readable.
8280-8620	The Print Spreadsheet routine. This routine uses two FOR...NEXT loops <i> and <m> which, for each of the four types of spreadsheets, look at the machine code from <first> to <last> in STEPs of 60 and PEEK out the five bytes of each floating point number (lines 8410-8450);it then moves to the sub routine at 8630 to turn these into a printable decimal number (ie, with dollars and cents). Note the boolean logic used in lines 8310 and 8490.
8630-8780	Coversion of floating point numbers to decimals for the printout.

-30-

Bob Mitchell 880415

Code pgm, "help.CT"32280,3968

BUDGET CODING SHEET dated 880411

Expenditures by Category

Capital:	Major items such as furniture, appliances, renovations. Others in Misc.
Clothing:	Including accessories: purses, wallets.
Dental:	Do not include costs paid by insurance,etc. Insurance premiums in Insurance.
Electric:	Including water heater rental or repair.
Gas:	Including any maintenance.
Gifts:	Including cards, wrapping, charitable donations, Christmas cards and wrap.
Groceries:	All food and non-food items (except liquor, beer and wine whether purchased at supermarkets, drug stores or other grocery outlets,and including health care items (but not patent medicines).
Housing:	Mortgage or rental costs, home insurance, repairs to major appliances (but furnace repairs in Gas).
Insurance:	Premiums for home, car, medical, dental, life, unemployment and other insurance policies.
Leisure:	Dining out, take-out food, movies, VCR movies, movies, Cable TV rental, entertainment equipment and entrance fees.
Liquor:	Used at home.
Medical:	Prescriptions drugs (less costs paid by insurance); patent medicines, doctor and hospital bills (less costs paid by insurance).
Misc. :	General items such as magazines, newspapers, stamps, Lucaya fees, small household items (apply sales tax rebates here).
Personal:	Magazines, hairdo's, perfume and other cosmetics, hobbies (computers, photography, sports).
Phone:	Including purchase of telephones, recorders, etc., used with telephone services.
Prop.Taxes.	Realty taxes.
Transport:	Car costs including insurance, parking costs, public transit and taxis, car and drivers licences, air,train and bus tickets.
Vacation:	Telephone, lodgings and food on trips, home-sitting, (car costs in transportation)

-30-

THE TABLES

Table 1 is a break-down of the "budget" program identifying its major parts along with some useful sub routines which might be incorporated in other programs.

Table 2 is a break-down of the "spread" program. This program contains a PRINT USING routine and a conversion routine to change floating point numbers to their decimal equivalents.

Table 3 is an optional HELP routine which Larken users may add to your disk and called up from "budget.Bx"; this gives suggestions on allocating different types of expenditures to the 18 categories. You should tailor your own help routine to suit your chosen category names. I used Tasword to type in this table and Taswide to call it up in 64 characters-per-line format; the Larken command PRINT USR 100: PRINT "help.CT" allows the help routine to be displayed on the screen without interfering with the budget program. See "budget.Bx" lines 185 and 186 for details.

Table 4 is a break-down of the loader program.

Bob Mitchell 880415.

-30-

Code pgm, "bujJb.CT"33280,8320

BUDGET -- PROGRAM BREAKDOWN

80	First entry point with Timachine (used only when initializing a new year budget.
110-180	Add zeros.
200-280	Menu outline; sub of menu routine at 300.
300-400	Print menus (370-400 controls menu cursor).
420	Set array sizes.
430-550	Set variables.
560-570	Load b$() with category names.
580-630	DATA lines for some menus.
590	NOTE SPACES TO PAD OUT ALL DATA ITEMS TO TEN CHARACTERS IN LENGTH.
650-690	Move menu cursor down.
700-740	Move menu cursor up.
750-820	Data input totaling.
830-850	Data totaling for display.
860-870	Calculate totals for graphs.
900-910	Calculate k() array for totals (planned or actuals).
920-1310	Right justify screen display of integer values of arrays and totals.
1060-1070	Print negative values in FLASHing RED; others in normal colours.
1320-1440	Change PLANNED values, one by one.
1460	Second Timachine entry point. Used after initializing to preserve all data.
1470-1630	Main menu.
1640-1700	Menu for input of PLANNED data.
1710-1760	Inpute values for whole year.
1770-1800	Totaling PLANNED data for each category.
1810-1870	Grand-totaling PLANNED data for all categories.
1880-1910	Set budget year.
1920-2040	Menu for Display routines.
2050-2400	Display PLANNED data.
2410-2730	Display COMPOSITE data.
2740-3000	Display ACTUALS data.
3010-3360	Display CATEGORY data.
3370-3790	Input ACTUALS data.
3800-3970	Display Menu for Graphs.
3980-4750	Pie chart routine.
4760-4960	Table displays.
4970-5400	Bar chart routine.
5410	End Timachine compilation area.

VARIABLES

Numeric arrays

a(18,12)	PLANNED values per category per month.
e(18,12)	ACTUALS values per category per month.
m(18,12)	TOTALS YTD PLANNED per month per column (category).
t(18,12)	TOTALS YTD PLANNED per category per line (month).
u(18,12)	TOTALS YTD ACTUALS per category per line.
y(18,12)	TOTALS YTD ACTUALS per month per column.
k(18)	TOTALS (PLANNED or ACTUALS) for graph routine.
o(18)	Pie chart array used to calculate sectors.
q(18)	Bar chart array used to calculate bars.
w(18)	Pie chart array used to calculate sectors.
z(18)	Pie chart array used to calculate sectors.

String arrays

b$(18,10)	Category names.
c$(32)	32 spaces for erasing.
m$(12,3)	Month names (3-letter abbreviations).
t$(3,11)	1=PLANNED; 2=ACTUALS; 3=COMPOSITE (all in INVERSE VIDEO)

Strings

e$	Menu item from DATA lines (Line 300).
g$	Temporary store for t$() in graph display.
k$	INKEY$.
n$	File name.
o$	Menu item from DATA lines (Line 560).
q$	Input control.
s$	Justification workspace.
x$	Date.
y$	Year " YY ".
z$	Input control.

Numeric (excluding FOR...NEXT)

bal	Worksheet balance (actuals input).
cf	Continue flag used in display menu.
cf2	Continue flag used in inputting Planned.
c	Column number.
cr	Counter.
d	Category number.
e	Length e$ used in menus.
exch	Currency exchange rate.
exp	Expense during inputting Actuals.
gmax	Max value for graph.
gc	Graph category (PLANNED or ACTUALS).
gt	Graph type (PIE or BAR).
gf	Graph flag (flags are used in branching).
k	Keypress value in menu operation.
k1	Derived from k in cursor move routine.
l	Month number.
lo	Flag used in justification routine.
loc	Location in printing bar charts.
ml	Derived from rl.
min	Minimum in graphs.
n	Month number used in YTD totaling.
o	Value used in DRAWing menu outline.
ol	Ditto.
p	Temp store for month number.
pe	No. of items for menu.
py,px	Y and X coords for printing menu.
pw	Max item width for given menu.
pp	Print position for graphs.
pc	Integer of percentage.
rl	Radius of pie chart.
s	Totaling (summation).
sa	Totaling actuals.
sa2	Totaling for actuals graphs.
sa3	Totaling for planned graphs.
sb	Totaling planned.
sca	Scale used in bar graph.
sf	Totaling planned for year.
ss	Integer value of array item.
sss	Integer YTD PLANNED percentage.
st	Array type to be generated as s$.
st3	Integer of t3.
t3	Derived from sa2 or sa3.
tp	Totaling percentages.
x1,y1	Coordinates for PLOT in pie chart.
x2,y2	Derived from x1,y1 to print category codes.
z	Input control.
zz	Integer used in printing bar chart.

The "spread.BJ" basic program for compiling with TIMACHINE

After Key-in make sure that it works. Load and initialize a printer port code, then type RUN, and enter "P" for planned

```
8000 REM !LEN $<=32
8005 REM ! LPRINT
8010 REM ! LIST
8020 REM !USR 58400
8030 REM ! OPEN #
8040 GO TO 8270
8050 LET il=LEN u$
8060 LET id=0
8070 FOR j=1 TO il
8080 LET j$=u$(j)
8090 IF j$=CHR$ 46 THEN LET id=L
EN u$-j
8100 NEXT j
8110 LET x$=SIR$ (SGN num*(INT (
ABS num*10©id+.55)))
8120 LET xd=LEN x$-id
8130 FOR j=0 TO -xd
8140 LET x$="0"+x$
8150 NEXT j
8160 LET xd=LEN x$-id
8170 IF id>0 THEN LET x$=x$( TO
xd)+"."+x$(xd+1 TO )
8180 FOR j=1 TO il-LEN x$
8190 LET x$=CHR$ 32+x$
8200 NEXT j
8210 RETURN
8220 RESTORE 8250
8230 DIM t$(19,10): FOR i=1 TO 1
9: READ t$(i): NEXT i
8240 RETURN
8250>DATA "CAPITAL","CLOTHING","
DENTAL","ELECTRIC","GAS","GIFTS"
,"GROCERIES","HOUSING","INSURANC
E","LEISURE","LIQUOR","MEDICAL",
"MISC.","PERSONAL","PHONE","PROP
.TAXES","TRANSPORT","VACATION"
8260 DATA "TOTALS>>"
8270 GO SUB 8220: LET u$="######
.##": OUT 127,15: POKE 23658,8:
INPUT "Select Spreadsheet..."'"P
=Planned     A=Actuals"'"Q=Plann
ed YTD B=Actuals YTD"; LINE i$
8280 LET tflag=0
8290 PRINT #0;"CALCULATING/PRINT
ING SPREADSHEET[ENTER]=STOP"
8300 LET num0=0
8310 LPRINT ("PLANNED" AND i$="P
")+("ACTUALS" AND i$="A")+("PLAN
NED YTD" AND i$="Q")+("ACTUALS Y
TD" AND i$="B")
8320 IF i$="P" OR i$="A" THEN LP
RINT "                JAN       FEB
      MAR       APR       MAY
 JUN       JUL       AUG       SEP
     OCT       NOV       DEC      T
OTAL"
8330 IF i$="Q" OR i$="B" THEN LP
RINT "                JAN       FEB
      MAR       APR       MAY
 JUN       JUL       AUG       SEP
     OCT       NOV       DEC      "
8340>IF i$="P" THEN LET first=50
534: LET last=51613: REM a( )
8350 IF i$="Q" THEN LET first=54
044: LET last=55123: REM t( )
8360 IF i$="A" THEN LET first=51
614: LET last=52693: REM e( )
8370 IF i$="B" THEN LET first=55
124: LET last=56203: REM u( )
8380 LET cat=1
8390 FOR i=first TO last STEP 60
8400 LPRINT t$(cat);: LET tot=0:
 FOR m=0 TO 55 STEP 5
8410 LET a=PEEK (i+m)
8420 LET e=PEEK (i+m+1)
8430 LET d=PEEK (i+m+2)
8440 LET c=PEEK (i+m+3)
8450 LET b=PEEK (i+m+4)
8460 GO SUB 8630
8470 GO SUB 8050
8480 IF INKEY$=CHR$ 13 THEN GO T
O 8620
8490 IF cat=19 AND (i$="Q" OR i$
="B") THEN LET num1=num: GO TO 8
750
8500 LET tot=tot+num
8510 POKE 23692,-1: LPRINT x$;
8520 NEXT m
8530 IF i$="P" OR i$="A" THEN LE
T num=tot: GO SUB 8050: LPRINT "
 ":x$
8540>IF i$="Q" OR i$="B" THEN LP
RINT "          "
8550 IF tflag THEN RETURN
8560 LET cat=cat+1
8570 NEXT i
8580 LET tflag=1
8590 IF i$="A" OR i$="B" THEN LE
T i=57314: REM z( )
8600 IF i$="P" OR i$="Q" THEN LE
T i=53804: REM o( )
8610 LET cat=19: GO SUB 8400
8620 LPRINT '''': STOP
8630 IF a<>0 THEN GO TO 8680
8640
8650 LET num=256*c+d
8660 IF e=255 THEN LET num=num-6
5536
8670 GO TO 8730
8680 LET exp=a-128
8690 LET sign=-1
8700 IF e<128 THEN LET sign=1: L
ET e=e+128
8710 LET mant=e/256+d/65536+c/16
777216+b/4294967296
8720 LET num=sign*2©exp*mant
8730 LET x$=SIR$ num
8740 RETURN
8750 LET tflag=1
8760 LET num0=num1+num0
8770 LET num=num0
8780 GO SUB 8050: GO TO 8510
8790 REM ! CLOSE #
9000 CLEAR : RANDOMIZE USR 100:
SAVE "spread.BJ": LIST
```

LINE	Notes
8050	variable is IL
8070	j=one to IL
8110	the © symbol is my printers EXP (sym shf H) arrow
8180	j=one to IL
8230	FOR I=one to nineteen
8300	is a zero- any variable O would be in lower case.
8380	LET cat=one
8400	is zeroes
8420	I+M+one
8490	variable is one
8510	minus one
8560	cat+one
8580	tflag=one
8640	is just the line number with one space after
8700	sign=one
8720	the © is my printers EXP (sym shift H) arrow
8750	tflag=one
8760	the variable is NUMzero. Count the O as having a slash

LINE- FREE CHECK

START=38652
8100=38492
8200=38275
8300=37732
8400=38032
8500=36796
8600=36539
8700=36031
9000=36057

CHECK SUM to help verify your key-in accuracy. Start with a NEWED computer that has FREE Memory of 38652 bytes. Figures given are what you must get with PRINT FREE after the line number has been entered.

<CLEAR> before <PRINT FREE>.

After making sure that the Check Sum is right, change the last line to suit your DOS syntax. Then COMPILE with TIMACHINE.

"BUDGET.Bl" for Aerco FD-68 DOS

Start line is 20.

```
  20 LET base=64500: IF PEEK bas
e<>24 THEN  OUT 244,1: CAT "Ccl
ear.bin",: CAT "Cprcode.bin",
  30 GO TO 60
  40 OUT 244,1: MOVE "budgetCY.b
in",32000,25923
  60 ON ERR  RESET : LET base=64
500: BORDER 1: PAPER 1: INK 7:
CLS : RANDOMIZE USR (base+2): P
RINT AT 4,0;"=======BUDGET INPU
T/OUTPUT======";''TAB 7;"1> Bud
get [Set-Up]"''TAB 7;"2> Budget
 [LOAD]"''TAB 7;"3> Budget [SAV
E]"''TAB 7;"4> Budget [re-Start
]"''TAB 7;"5> PRINT Spreadsheet
s"''TAB 10;"on Wide Printer"''TA
B 7;"6> Disk   [Menu]"''"======
====Select Option========="
 130 PAUSE 0: LET k=CODE INKEY$-
48: IF k<1 OR k>6 THEN  GO TO 1
30
 155 IF k=6 THEN  OUT 244,1: CAT
 "MenuBl.bas",
 160 IF k=1 THEN  GO TO 220
 170 IF k=2 THEN  GO TO 250
 180 IF k=3 AND PEEK 32000=205 T
HEN  GO TO 40
 190 IF k=4 AND PEEK 37166=205 T
HEN  RANDOMIZE USR (base+2): CL
S : LPRINT : ON ERR  GO TO 60:
RANDOMIZE USR 37166: GO TO 60
 195 IF k=5 AND PEEK 32000=205 T
HEN  RANDOMIZE USR base: CLS :
POKE (base+4),131: LPRINT : OUT
 244,1: CAT "spreadCJ.bin",: ON
 ERR  GO TO 60: RANDOMIZE USR 5
8400: POKE 23658,0: GO TO 60
 210 GO TO 60
 220 RANDOMIZE USR (base+2): IF
PEEK 32000<>205 THEN  OUT 244,1
: CAT "budgetCJ.bin",
 230 RANDOMIZE USR 32000
 240 GO TO 60
 250 RANDOMIZE USR (base+2): OUT
 244,1: CAT "budgetCY.bin",: ON
 ERR  GO TO 60: RANDOMIZE USR 3
7166
 260 GO TO 60
 310 OUT 244,1: MOVE "BudgetBl.b
as",20: GO TO 60
 350 RANDOMIZE USR base: CLS : I
NPUT "Enter line width: ";11: P
OKE (base+4),11-1: BEEP .5,12:
INPUT "Does printer need a line
 feed?  <1> Yes - <2> No ";lf:
IF lf=1 THEN  POKE (base+5),10
 360 CLS : PRINT AT 10,2;"Ready
for Centronics printer"''TAB 2;
"To resume BUDGET, <GO TO 20>":
 STOP
```

SEE FOOT NOTES under the Aerco FD-68 Disk Menu program listing.

"Disk Menu" for Aerco FD-68 DOS
Start line is 10

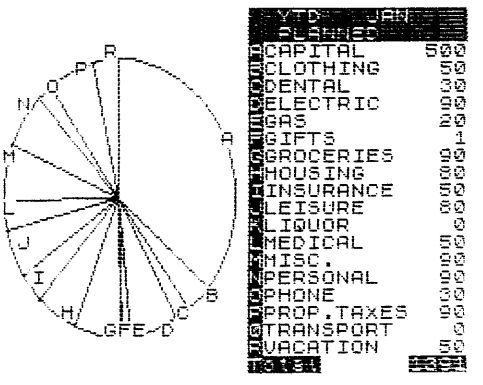
```
  10 INK 3: BORDER 0: PAPER 0: C
LS
  20 FOR n=0 TO 31: PRINT AT 2,n
;"*";AT 20,n;"*": NEXT n: FOR n
=3 TO 19: PRINT AT n,0;"*";AT n
,31;"*": NEXT n: INK 7
  30 PRINT AT 4,7;"The Disk Mana
ger";AT 7,5;"1> Load Your Budge
t";AT 9,5;"2> Load Spread Sheet
";AT 11,5;"3> EXIT Budget To BA
SIC";AT 12,8;"[for printing]";A
T 14,5;"4> Save this Budget Dat
a";AT 16,5;"5> Go to Main Menu"
;AT 18,5;"6> Re-Set Computer"
  40 PAUSE 0: LET aa=CODE INKEY$
-48: IF aa<1 OR aa>6 THEN  GO T
O 30
  60 OUT 244,1: IF aa=1 THEN  CA
T "budgetCY.bin",: GO TO 10
  70 IF aa=2 THEN  CAT "spreadCJ
.bin",: GO TO 10
  80 IF aa=6 THEN  OUT 244,0: RA
NDOMIZE USR 0
  90 IF aa=4 THEN  MOVE "budgetC
Y.bin",32000,25923: GO TO 10
 100 IF aa=5 THEN  CAT "BudgetBl
.bas",
 110 IF aa=3 THEN  CLS : OUT 244
,1: CAT "Cprcode.bin",: INPUT "
Select:
 1> For TS-2040 -- <2> Centroni
cs";p: IF p=1 THEN  RANDOMIZE U
SR 64502: STOP
 120 IF p=2 THEN  RANDOMIZE USR
64500: INPUT "Enter Line width
";w: POKE 64504,(w-1): INPUT "E
nter ""0"" for no Line Feed or
        ""10"" for Line Feed ";
f: POKE 64505,f: LPRINT ': PRIN
T AT 10,2;"Ready to LPRINT"''TA
B 2;"To resume BUDGET, type <RU
N>": STOP
 130 GO TO 10
```

Foot Notes: This Disk Menu and the Aerco Budget Loader have a relocatable Aerco printer code as given in Jack Dohany's RALOADER program. The code is located at 64500,867. The top 67 bytes are reserved for VARS that need to be preserved from one Basic program to another. POKE the vars starting at 65300 before a new program is loaded or before a new VAL program is loaded. Then PEEK the vars back after the LOAD.

Both this Menu and the BUDGET Loader have a Stand By printer code initialization routine to use when you need to put the Budget program on stand-by to do some direct printing.

"Disk Menu" for Larken DOS
Start Line is 10

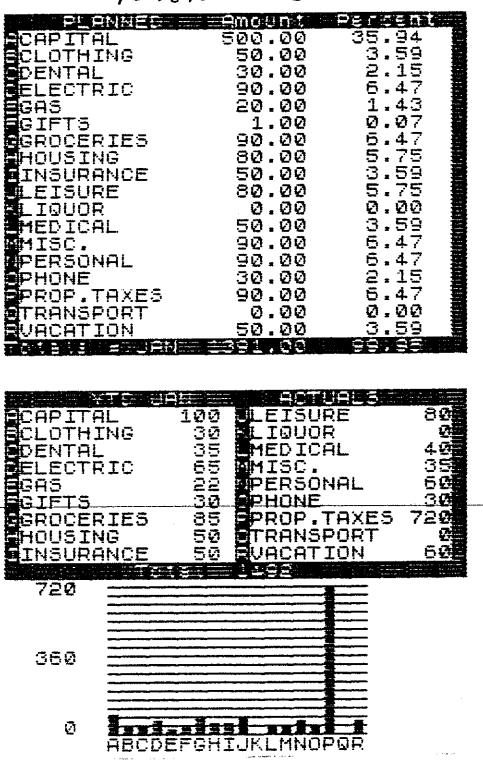
```
  10 INK 3: BORDER 0: PAPER 0: C
LS
  20 FOR n=0 TO 31: PRINT AT 3,n
;"*";AT 21,n;"*": NEXT n: FOR n
=3 TO 21: PRINT AT n,0;"*";AT n
,31;"*": NEXT n: INK 7
  30 PRINT AT 5,7;"The Disk Mana
ger";AT 7,5;"1> Load Your Budge
t";AT 9,5;"2> Load Spread Sheet
";AT 11,5;"3> Load Printer Code
";AT 12,8;"To Print Centronics"
;AT 14,5;"4> Save this Budget D
ata";AT 16,5;"5> Go to Main Men
u";AT 18,5;"6> EXIT Budget Prog
ram"
  40 PAUSE 0: LET aa=CODE INKEY$
-48: IF aa<1 OR aa>6 THEN  GO T
O 30
  45 IF aa=6 THEN  RANDOMIZE USR
 0
  50 RANDOMIZE USR 100: OPEN #4,
"dd"
  60 IF aa=1 THEN  PRINT #4: LOA
D "budget.CY"CODE
  70 IF aa=2 THEN  PRINT #4: LOA
D "spread.CJ"CODE
  80 IF aa=3 THEN  CLS : GO TO 1
20
  90 IF aa=4 THEN  PRINT #4: SAV
E "budget.CY"CODE 32000,25923
 100 IF aa=5 THEN  PRINT #4: LOA
D "BUDGET.Bl"
 110 GO TO 10
 120 CLS : PRINT AT 10,5;"Please
 Stand by . . .": LET base=6400
0: PRINT #4: LOAD "Cclear.Cl"CO
DE : PRINT #4: LOAD "prcode.Cl"
CODE : RANDOMIZE USR base: POKE
 26703,205: POKE 26704,251: POK
E 64451,254: POKE 64457,0: POKE
 64458,0: CLS : INPUT "Line Wid
th? ";w: POKE 64459,w-1: INPUT
"Printer need Line feed?
  <1> yes or <2> No ";lf: IF lf
=1 THEN  POKE 64460,10
 130 IF lf=2 THEN  POKE 64460,0
 140 IF lf<1 OR lf>2 THEN  GO TO
 120
 150 LPRINT : CLS : PRINT AT 10,
5;"READY to PRINT"''TAB 5;"When
 finished, type <RUN>.": STOP
```



TS-2040 Samples

"BUDGET.B1" for Larken DOS
Starting Line is 60.

```
  30 GO TO 60
  40 RANDOMIZE USR 100: SAVE "bu
dget.CY"CODE 32000,25923
  60 ON ERR  RESET : CLOSE #2: O
UT 127,18: POKE 23703,0: POKE 2
6704,5:: BORDER 0: PAPER 0: INK
 7: CLS : PRINT AT 4,0;"=======
BUDGET INPUT/OUTPUT======"
  65 REM  RANDOMIZE USR 100: CLO
SE #3
  70 PRINT '"  1> Budget [set up
]"
  80 PRINT "  2> Budget [load]"
  90 PRINT "  3> Budget [save]"
 100 PRINT "  4> Budget [restart
]"
 105 PRINT '"  5> Print Spreadsh
eets"'"     on Wide Printer"
 110 PRINT '"  6> Disk Menu"
 115 PRINT '"  7> HELP!"
 120 PRINT '''=========Select op
tion=========="
 130  PAUSE 0
 140 LET k$=INKEY$
 150 IF k$<"0" OR k$>"7" THEN  G
O TO 150
 160 IF k$="1" THEN  GO TO 220
 170 IF k$="2" THEN  ON ERR  GO
TO 60: GO SUB 250
 180 IF k$="3" AND PEEK 32000=20
5 THEN  GO TO 40
 185 IF k$="7" THEN  ON ERR  GO
TO 186: RANDOMIZE USR 100: LOAD
 "taswi.Cx"CODE : RANDOMIZE USR
 64300: PRINT CHR$ 3;: PRINT US
R 100: PRINT "help.CT"
 186 IF k$="7" THEN  ON ERR  RES
ET : POKE 23607,60: PRINT #1; F
LASH 1;"[ENTER]": PAUSE 0: GO T
O 60
 190 IF k$="4" AND PEEK 37166=20
5 THEN  ON ERR  GO TO 60: RANDO
MIZE USR 37166
 195 IF k$="5" AND PEEK 32000=20
5 THEN  CLS : RANDOMIZE USR 100
: LOAD "prcode.C1"CODE : POKE 2
6703,205: POKE 26704,251: POKE
64459,131: POKE 64451,254: POKE
 64460,0: POKE 64456,1: POKE 64
458,0: POKE 64457,0: PRINT #4:
LOAD "spread.CJ"CODE : RANDOMIZ
E USR 58400: GO TO 60
 200 IF k$="6" THEN  RANDOMIZE U
SR 100: LOAD "Menu.B1"
 210 GO TO 60
 220 RANDOMIZE USR 100: LOAD "bu
dget.CJ"CODE
 230 RANDOMIZE USR 32000
 240 GO TO 60
 250 RANDOMIZE USR 100: LOAD "bu
dget.CY"CODE
 260 RANDOMIZE USR 37166
 270 GO TO 60
 310 RANDOMIZE USR 100: SAVE "bu
dget.B1" LINE 60: REM **  Direc
t SAVE of this Basic Program
 320 STOP
 350 RANDOMIZE USR 100: LOAD "pr
code.C1"CODE : POKE 26703,205:
POKE 26704,251: POKE 64459,131:
 POKE 64451,254: POKE 64460,0:
POKE 64456,1: POKE 64458,0: POK
E 64457,0: CLS : INPUT "Line Wi
dth?";11: POKE 64459,11-1: LPRI
NT : STOP : REM ** To enable Ce
ntronics Printer.
```



Disk Menu for Oliger DOS
Start Line is 10.

```
  10 INK 3: BORDER 0: PAPER 0: C
LS
  20 FOR n=0 TO 31: PRINT AT 2,n
;"*";AT 20,n;"*": NEXT n: FOR n=
3 TO 19: PRINT AT n,0;"*";AT n,3
1;"*": NEXT n: INK 7
  30 PRINT AT 4,7;"The Disk Mana
ger";AT 7,5;"1> Load Your Budget
";AT 9,5;"2> Load Spread Sheet";
AT 11,5;"3> EXIT Budget To BASIC
";AT 13,5;"4> Save this Budget D
ata";AT 15,5;"5> Go to Main Menu
"
  40 PAUSE 0: LET aa=CODE INKEY$
-48: IF aa<1 OR aa>6 THEN GO TO
30
  60 IF aa=1 THEN LOAD /"budget.
CY"CODE
  70 IF aa=2 THEN LOAD /"spread.
CJ"CODE
  80 IF aa=3 THEN CLEAR 65535: C
LEAR 65367: LOAD /"DOSDEX MGR"
  90 IF aa=4 THEN SAVE /"budget.
CY"CODE 32000,25923
 100 IF aa=5 THEN LOAD /"BUDGET.
B1"
 110 GO TO 10
```

Foot notes

Upon finishing with the BUDGET program, a selection of <3> will boot in the DOSDEX MGR program for selection of another Basic program to load (assuming that the either this same disk has the DOSDEX MGR program. If not then you can install a disk that does.

"BUDGET.B1" for the Oliger DOS
Starting Line is 60

```
  10 REM ****BUDGET LOADER****
  20 REM ** See Cclear ** LOAD /
"Cclear"CODE
  30 GO TO 60
  40 SAVE /"budget.CY"CODE 32000
,25923
  60 ON ERR RESET : CLOSE #2: BO
RDER 0: PAPER 0: INK 7: CLS : LE
T /p=t: PRINT AT 4,0;"=======BUD
GET INPUT/OUTPUT======"
  65 REM CLOSE #3
  70 PRINT '"  1> Budget [set up
]"
  80 PRINT "  2> Budget [load]"
  90 PRINT "  3> Budget [save]"
 100 PRINT "  4> Budget [restart
]"
 105 PRINT '"  5> Print Spreadsh
eets"'"     on Wide Printer"
 110 PRINT '"  6> Disk Menu"
 120 PRINT '''=========Select op
tion=========="
 130 PAUSE 0
 140 LET k$=INKEY$
 150 IF k$<"0" OR k$>"6" THEN GO
 TO 150
 160 IF k$="1" THEN GO TO 220
 170 IF k$="2" THEN ON ERR GO TO
 60: GO TO 250
 180 IF k$="3" AND PEEK 32000=20
5 THEN GO TO 40
 190 IF k$="4" AND PEEK 37166=20
5 THEN ON ERR GO TO 60: RANDOMIZ
E USR 37166
 195 IF k$="5" AND PEEK 32000=20
5 THEN CLS : LET /p=o: POKE 2332
4,0: POKE 23323,132: LOAD /"spre
ad.CJ"CODE : RANDOMIZE USR 58400
: GO TO 60
 200 IF k$="6" THEN LOAD /"Menu.
B1"
 210 GO TO 60
 220 LOAD /"budget.CJ"CODE
 230 RANDOMIZE USR 32000
 240 GO TO 60
 250 LOAD /"budget.CJ"CODE
 260 RANDOMIZE USR 37166
 270 GO TO 60
 310 SAVE /"budget.B1" LINE 330
 320 GO TO 60
```

Foot Note: There are two code budget programs. One is budget.CJ and has no data. budget.CY is created (line 40) when you save your first data program.

Selecting <4> will clear out the data of an existing program and start over.

<GO TO 310> will SAVE a back up of this basic loader.

"BUDGET.B3" Alternate Loader for "MAX DOS" (Stacked Larken and Oliger DOS

Starting Line=330

```
  10 REM ****BUDGET LOADER****
  20 CLEAR 31999
  30 GO TO 60
  40 SAVE /"budget.CY"CODE 32000
,25923
  60 ON ERR RESET : LET /p=o: CL
OSE #2: REM ** OPTION FOR STACKE
D DOS
  61 REM ** OUT 127,18: POKE 237
03,0: POKE 26704,5: REM * FOR AE
RCO CODE
  63 ON ERR RESET : BORDER 0: PA
PER 0: INK 7: CLS : PRINT AT 4,0
;"=======BUDGET INPUT/OUTPUT====
=="
  65 REM RANDOMIZE USR 100: CLOS
E #3
  70 PRINT '"  1> Budget [set up
]"
  80 PRINT "  2> Budget [load]"
  90 PRINT "  3> Budget [save]"
 100 PRINT "  4> Budget [restart
]"
 105 PRINT '"  5> Print Spreadsh
eets"'"     on Wide Printer"
 110 PRINT '"  6> Disk Menu"
 115 PRINT '"  7> HELP!"
 120 PRINT '''=========Select op
tion=========="
 130 PAUSE 0
 140 LET k$=INKEY$
 150 IF k$<"0" OR k$>"7" THEN GO
 TO 150
 160 IF k$="1" THEN GO TO 220
 170 IF k$="2" THEN ON ERR GO TO
 60: GO SUB 250
 180 IF k$="3" AND PEEK 32000=20
5 THEN GO TO 290
 185 IF k$="7" THEN ON ERR GO TO
 186: CLS : RANDOMIZE USR 100: L
OAD "taswi.Cx"CODE : RANDOMIZE U
SR 64300: PRINT CHR$ 3;: PRINT U
SR 100: PRINT "help.CT"
 186 IF k$="7" THEN ON ERR RESET
 : POKE 23607,60: PRINT #1; FLAS
H 1;"[ENTER]": PAUSE 0: GO TO 60
 190 IF k$="4" AND PEEK 37166=20
5 THEN ON ERR GO TO 60: RANDOMIZ
E USR 37166
 195 IF k$="5" AND PEEK 32000=20
5 THEN CLS : LET /P=O: RANDOMIZE
 USR 100: OPEN #4,"DD": POKE 233
23,132: PRINT #4: LOAD "spread.C
J"CODE : RANDOMIZE USR 58400: GO
 TO 60: REM ** For Stacked DOS
 196 REM for Aerco code- IF k$="
5" AND PEEK 32000=205 THEN CLS :
 RANDOMIZE USR 100: OPEN #4,"DD"
: LOAD "prcode.C2"CODE : POKE 26
703,5: POKE 26704,251: POKE 6425
9,131: PRINT #4: LOAD "spread.CJ
"CODE : RANDOMIZE USR 58400: GO
TO 60
 200 IF k$="6" THEN RANDOMIZE US
R 100: LOAD "Menu.B1"
```

Cont

An Interesting Productivity Program for MAX DOS

Are you ready for this? Well, some day you will be. This is a Budget Loader program to be used with "MAX DOS", Larken and Oliger DOS "stacked". It works like a charm at UPDATE. Now this will repeat some of the information contained in the article "MAX DOS", but will serve as an example of a useful program. The senario is: Larken LK cartridge in the Dock port and the full Oliger DOS in place. RGB is via the Oliger hardware. Printer support is via the Oliger SAFE DOS. LET/p=t enables the TS-2040 printer, or LET/p=o enables the Centronics printer. Maximum line width is controlled via POKE 23323,width.

Note line 330, which is the Auto Run Line. Both Oliger and Larken DOS are initialized in this line. <LET/d=1> assigns disk #1 to the Oliger DOS. <PRINT #4: GO TO 0> assigns disk zero to the Larken DOS. From then on, the disk syntax used results in automatic switching of the disk drives as needed. This allows 1.6 mgbts of disk space under direct program control (two 80 track drives). Other combos of drives are acceptable. As programmed, disk "PROGRAM" SAVE and LOADs are to the LARKEN disk in disk zero, and Disk "DATA" saves and LOADS are to the Oliger Disk in drive #1. But this can easily be changed as desired. The convertable line numbers are: 40, 185, 196, 200, 220, 250, 310, and 335. All you need do is change the LOAD SAVE syntax and the right drive will be auto selected.

Note lines 40 and 250, which uses the Oliger SAVE and LOAD of the BUDGET DATA. When these lines execute, the drive is switched to drive #1. All other LOADS and SAVES use LARKEN DOS syntax, which switches to drive #0. A good operating practice with MAX DOS is to use one DOS format for PROGRAMS and the other for DATA. The drives are switched automatically. MAX DOS allows the user to FORMAT Oliger disks directly or to load the Larken tool kit disk to FORMAT Larken Disks. Disks may be prepared for other users who operate with either of the two single DOS, Larken or Oliger.

There are two separate operating protocols. Normally one would think of the "stacked DOS" as mostly having to do with operations to and from the disk drive. Actually the most significant advantages of MAX DOS are the "cumulative" Extra Basic functions which can be used without regard to which disk format is being used. During operation one does not pay any attention to "which disk is being used", as this does not matter. One just uses the programming syntax to do the function needed. In this program, menu option <7> HELP, results in the exercise of a Larken function which brings up a "disk file to screen" for viewing. It is the "taswi.Cx"CODE file, which happens to be a TASWIDE 64 Column "Help File", which is printed to screen.

Other Extra Basic functions of either DOS are "at home" in the MAX DOS environment. Programs that contain "mixed" DOS syntax are acceptable for whichever disk format will be used for SAVE or LOAD. The Oliger "Fast FOR counter" is at home in a Larken Disk program, and the Oliger Printer Driver works as well in a Larken format as it does in a Oliger Format. A single drive can be used as easily as one normally does with either DOS. Program prompting to "install Larken Disk or Oliger Disk" before a disk command would be appropriate for single drive systems, but not necessary with a two drive system.

Since the two EPROMS of Oliger and LARKEN DOS operate in different expansion banks, the extra basic functions that they contain can be expanded. These groups of new functions are additions to the TS-2068 ROM, but are different in concept, ie, they are "whole routines of productive functions", rather than "loops to support the TS-2068 Basic". The extra basic functions add substantial computing power to the TS-2068.

```
 210 GO TO 60
 220 RANDOMIZE USR 100: LOAD "bu
dget.CJ"CODE
 230 RANDOMIZE USR 32000
 240 GO TO 60
 250 LOAD /"budget.CY"CODE
 260 RANDOMIZE USR 37166
 270 GO TO 60
 280 STOP
 290 GO TO 40
 300 GO TO 60
 310 RANDOMIZE USR 100: SAVE "BU
DGET.B1" LINE 330
 320 STOP
 330 LET /p=o: LET /d=1: RANDOMI
ZE USR 100: OPEN #4,"dd": PRINT
#4: GO TO 0: REM ** Initializes
Dual DOS, Larken disk in drive 0
, Oliger in drive 1, TS-2040 pri
nting
 331 REM ** RANDOMIZE USR 100: L
OAD "prcode.C2"CODE : REM aerco
prcode
 333 REM RANDOMIZE USR 100: OPEN
 #3,"lp"
 335 RANDOMIZE USR 100: LOAD "sp
read.CJ"CODE
 340 POKE 23658,0
 350 GO TO 10
 400 CLEAR 29000: RANDOMIZE USR
102: GO TO 330
```

Alternate BUDGET.B1 (LARKEN)
By Bob Mitchell. Includes the Taswide Help File option, and a different Aerco prcode handler.

```
  10 REM ****BUDGET LOADER****
  20 CLEAR 31999
  30 GO TO 60
  40 RANDOMIZE USR 100: SAVE "bu
dget.CY"CODE 32000,25923
  60 ON ERR RESET : CLOSE #2: OU
T 127,18: POKE 23703,0: POKE 267
04,5:: BORDER 0: PAPER 0: INK 7:
 CLS : PRINT AT 4,0;"=======BUDG
ET INPUT/OUTPUT======"
  65 REM RANDOMIZE USR 100: CLOS
E #3
  70 PRINT '"  1> Budget [set up
]"
  80 PRINT "  2> Budget [load]"
  90 PRINT "  3> Budget [save]"
 100 PRINT "  4> Budget [restart
]"
 105 PRINT '"  5> Print Spreadsh
eets"'"     on Wide Printer"
 110 PRINT '"  6> Disk Menu"
 115 PRINT '"  7> HELP!"
 120 PRINT '"=========Select op
tion=========="
 130 PAUSE 0
 140 LET k$=INKEY$
 150 IF k$<"0" OR k$>"7" THEN GO
 TO 150
 160 IF k$="1" THEN GO TO 220
 170 IF k$="2" THEN ON ERR GO TO
 60: GO SUB 250
 180 IF k$="3" AND PEEK 32000=20
5 THEN GO TO 290
 185 IF k$="7" THEN ON ERR GO TO
 186: CLS : RANDOMIZE USR 100: L
OAD "taswi.Cx"CODE : RANDOMIZE U
SR 64300: PRINT CHR$ 3;: PRINT U
SR 100: PRINT "help.CI"
 186 IF k$="7" THEN ON ERR RESET
 : POKE 23607,60: PRINT #1; FLAS
H 1;"[ENTER]": PAUSE 0: GO TO 60
 190 IF k$="4" AND PEEK 37166=20
5 THEN ON ERR GO TO 60: RANDOMIZ
E USR 37166
 195 IF k$="5" AND PEEK 32000=20
5 THEN CLS : RANDOMIZE USR 100:
LOAD "prcode.C1"CODE : POKE 2670
3,5: POKE 26704,251: POKE 64259,
131: PRINT #4: LOAD "spread.CJ"C
ODE : RANDOMIZE USR 58400: GO TO
 60
 200 IF k$="6" THEN RANDOMIZE US
R 100: LOAD "Menu.B1"
 210 GO TO 60
 220 RANDOMIZE USR 100: LOAD "bu
dget.CJ"CODE
 230 RANDOMIZE USR 32000
 240 GO TO 60
 250 RANDOMIZE USR 100: LOAD "bu
dget.CY"CODE
 260 RANDOMIZE USR 37166
 270 GO TO 60
 280 STOP
 290 GO TO 40
 300 GO TO 60
 310 RANDOMIZE USR 100: SAVE "BU
DGET.B1" LINE 330
 320 STOP
 330 RANDOMIZE USR 100: LOAD "pr
code.C1"CODE : REM aerco prcode
 333 REM RANDOMIZE USR 100: OPEN
 #3,"lp"
 335 RANDOMIZE USR 100: LOAD "sp
read.CJ"CODE
 340 POKE 23658,0
 350 GO TO 10
 400 CLEAR 29000: RANDOMIZE USR
102: GO TO 330
```

The Aerco prcode, and possible Problems

There are at least four versions of the Aerco printer code, and each of these may have three variations of configurations set up during the user's adaptation to his own printer equipment. The Budget program LOADER, and the two programs "MS-TAS" and "BRIEF" may not print right with your equipment and with the Aerco prcode programming (POKES) given in the program listings. But try it anyway. My Epson compatable printer and my Queme compatabke Daisy printer both work well with the code used in the Budget, INIT, and BRIEF programs. Hopefully, yours will also. If not, then use the code that you normally use with the these three programs, and change the POKES in the listings to that which you normally use successfully with your other programs.

8970 REM

** TS-2068 UPDATE **

-MS-TAS-

A utility to PEEK and PRINT TASWORD and MSCRIPT Code Data files from Disk or Cassette.

* Does not require the software to be loaded.

```
8972 LET x=9610: LET y=9600: CLO
SE #2: BORDER 0: PAPER 0: INK 7:
 CLS : PRINT '''TAB 8;"TS-2068 U
PDATE"''''* Utility to PEEK and
Print out MSCRIPT and TASWORD Da
ta Files  from Disk."''"* Does n
ot require Word Proc to be loade
d."''"* DOES Require;"''"1. Aerc
o or Oliger Printer Intfc  to b
e installed."''"2. Centronics Pr
inter -ON- READY"''''"SET EQUIPME
NT, THEN ENTER": PAUSE 0: CLS
8974 PRINT AT 10,2;"Set Printer
ON and READY"'TAB 2;"Then [ENTER
]": PAUSE 0: CLS
8976 PRINT AT 10,12;"Select-"''
TAB 5;"<1> TS-2040 Printer"'TAB
 5;"<2> Centronics Printer": GO
SUB y: LET pr=z: IF pr=1 THEN GO
 TO 8980
8978 CLS : PRINT AT 10,1;"Does P
rinter need a Line Feed?-"''''TAB
 4;"<1> Yes - or - <2> No": GO S
UB y: CLS : LET lf=z
8980 PRINT AT 10,2;"Input DOS or
 Cassette"'TAB 5;"<1> Cassette"
''TAB 5;"<2> Oliger DOS"'TAB 5;
"<3> Larken DOS"'TAB 5;"<4> Aer
co FD-68 DOS": GO SUB x: LET tm=
z: IF z<1 OR z>4 THEN GO TO 8980
8982 GO SUB (tm=1)*9200+(tm=2)*9
360+(tm=3)*9310+(tm=4)*9300
8984 IF tm=1 AND pr=1 THEN GO TO
 9000
8996 GO SUB (tm=2)*9325+(tm=3)*9
330+(tm=4)*9320
9000 LET Lc=1: INPUT "Input File
 Starting Address ";sa: INPUT "I
nput File Byte length ";bL: INPU
T "Format Line Length? ";LL
9002 IF tm=2 THEN POKE 23323,LL
9004 IF tm>2 THEN POKE 64458,LL-
1
9006 CLS : OPEN #2,"p": FOR n=sa
 TO sa+bL STEP LL
9008 FOR y=n TO n+LL-1
9010 LET c=PEEK y
9012 IF c=13 THEN PRINT '
9014 PRINT CHR$ c;
9016 NEXT y
9018 IF tm>2 THEN PRINT
9020 LET Lc=Lc+1: IF Lc>55 THEN
PRINT : CLOSE #2: LET Lc=1: OUT
127,12: BEEP PI/oj,ol: CLS : PRI
NT AT oj,od;"Adjust Page, then E
NTER": BEEP PI/oj,ol: PAUSE 0: C
LS : OPEN #2,"p"
9030 NEXT n
```


A Tutor in Basic Programming Techniques

This is one of them utilities designed for a job at hand. Friends used to send me letters on cassette or disk and I was kinda like the poor guy who couldn't read writing. I'd have to fake it and call them to discuss what they'd said, because the data file was in Code files. Mostly these files were either Mscript or Tasword files. Both are good word procesors, but that didn't matter as I dont use either and it was Greek to me! I use Smart Text which saves data in Character arrays.

So, that sets the senario. Then when Bob Mitchell sent me 10 pages of Tasword code files, that did it! "MS-TAS" was the result. Now take time out to look at Bob's documentation in this issue. All of it was printed out with this utility MS-TAS, without a word processor. The print product was good enough that all I had to do was run it through the copier. "MS-TAS" grabs Code files out of Disk (or Cassette) and prints them to paper. This particular program works with LARKEN DOS, OLIGER DOS, AERCO FD-68, or CASSETTE. And it works with either CENTRONICS or TS-2040 printer.

I'll now donate MS-TAS to public domain so that the Clubs can have at it. (Would be nice if Credits were mentioned). Now since MS-TAS will work with WHATEVER configuration one has, we will use MS-TAS as this issue's Programming Tutor. You Lads and Lassies who dont need no tutoring can just start keying in and skip the prose.

MS-TAS looks like a lot of programming, but it trims down to a few lines when the extra stuff for other Disk Systems and Cassette is eliminated. The plan is to use menus to make you create variables to pin down your equipment types (artificial intelligence). Then Sub Routines are used for Cassette, for Larken Disk, for Oliger Disk, for Aerco FD-68 Disk, for Centronics Printer, or for TS-2040 printer.

Once your equipment is identified, then a disk catalog will be given. A prompt for INPUT of a CODE FILE NAME will be presented at the bottom of the Catalog. Then, while the cat is still visable, other prompts ask for Code Start address, and byte length. Any "friend" who sends you Code data will tell you these two essential bits of information. But if using the Oliger V2-52, the info will be on screen. The last prompt will be for LINE LENGTH for printing. The disk Code File will then LOAD, and begin printing. If the line length input isn't right, the print out wont look so good, and you'll have to stop and start over with the right line length that you figured out from the wasted page.

If you BREAK during printing, you'll need to <<CLOSE# 2>>, because if you dont, any PRINT or LIST will print to paper instead of the screen. Since all variables are identified within the program lines, you can repeatedly use RUN to start the program. The program remains in memory while the Code files are printing, so you can print out one code file after another until everything has been printed. Now thats the way the program operates. We will now investigate the programming.

8972 and 8974 sets the vars used for sub-routine <GO TO> or <GO SUB>, then the screen colors, then gives information.

8976 begins the artificial intelligence gathering by assigning var "pr" to the type of printer in use. pr=1 for TS-2040 or pr=2 for Centronics.

8978 gathers more intelligence and assigns var "lf". lf=1 if your printer needs a line feed, else lf=2. Might mention the <GO SUB y>. "y" was set to =9600 in line 8972. Look at line y (9600). This is a INKEY$ routine which waits for a key to be touched and then assigns that key to variable "z". Note the "-48" which makes z=the numbered key struck. AND, there is a

```
9035>PRINT : CLOSE #2
9040 PRINT AT 10,6;"<1> for Next
 File"''TAB 6;"<2> To QUIT": INP
UT "Choice?";a: IF a<1 OR a>2 TH
EN GO TO 9040
9045 IF a=2 THEN STOP
9050 GO TO 8996
9200 PRINT AT 10,2;"Advance Tape
 to Index spot,"''TAB 2;"Then [E
NTER]": PAUSE 0: CLS : PRINT AT
15,6;"CODE LOADING": CLS : RETUR
N
9300 OUT 244,1: CAT "prcode.bin"
,: GO SUB 9350: RETURN
9310 RANDOMIZE USR 100: OPEN #4,
"dd": PRINT #4: LOAD "prcode.C1"
CODE : GO SUB 9340: GO SUB 9350:
 RETURN
9320 OUT 244,1: CAT "",: GO SUB
9370: CAT "z$",: RETURN
9325 CAT : GO SUB 9370: LOAD /z$
CODE : RETURN
9330 RANDOMIZE USR 100: OPEN #4,
"dd": PRINT #4: CAT "",: GO SUB
9370: LOAD z$CODE : RETURN
9340 INPUT "Input Line width ";l
w: RETURN
9350 POKE 23703,205: POKE 23704,
251: POKE 64451,254: POKE 64456,
1: POKE 64457,0: POKE 64459,Lw-1
: POKE 64460,0: POKE 64458,(10 A
ND Lf>1)+(0 AND Lf<2): LPRINT :
RETURN
9360 LET /p=o: GO SUB 9340: POKE
 23323,Lw: POKE 23324,(0 AND Lf>
1)+(10 AND lf<2): LPRINT : RETUR
N
9370 INPUT "ENTER TITLE of CODE
to LOAD     [everything between
Quotes]      ";z$: RETURN
9600 PAUSE 0: LET z=CODE INKEY$-
48: IF z<1 OR z>2 THEN GO TO 960
0
9602 RETURN
9610: PAUSE 0: LET z=CODE INKEY$
-48: RETURN
```

-MS-TAS-

Abbreviated for Oliger DOS

```
8990 CLOSE #2: BORDER 0: PAPER 0
: INK 7: CLS : PRINT '''TAB 8;"T
S-2068 UP-DATE"''''"* Utility to
PEEK and Print out MSCRIPT and T
ASWORD Data Files  from Disk."''
"* Does not require Word Proc to
 be loaded."''"* DOES Require;"'
'"1. Aerco or Oliger Printer int
fc   to be installed."''"2. Cent
ronics Printer -ON- READY"''"SE
T EQUIPMENT, THEN ENTER": PAUSE
0: CLS
```

trap here which refuses to accept any key except "1" or "2". Also note the RETURN at line 9602. Subroutines such as this one <<GO SUB y>> are used repeatedly from any line in a program, thus saving many program lines.

8980 gathers some more intelligence, this time "what kind of DOS?" Variable "tm" is set accordingly. Example- if Oliger Disk then tm=2.

Now the "program knows" everything about your TS-2068 system! The variables are set so that the right sub-routines can be called for Oliger Disk, Larken Disk, Aerco FD, or Cassette. And for Centronics or TS-2040. And for giving or not giving a line feed. All of that only took a few seconds and we're now ready for some action.

8982 Here's the first action. We use BOOLEAN logic to determine which GO SUB routine to use to CAT the directory. "tm" is the variable set back at line 8980. The logic is actually four IF THEN conditionals. I'll give you the first and you can visualize the others. <GO SUB (tm=1)*9200> means <IF tm=1 THEN GO SUB 9200>. Next, IF tm=2 THEN GO SUB 9360.

Lets look at 9360. That is the Oliger DOS syntax to initialize the Centronics printer. Then there a <GO SUB 9340> which is to INPUT line width. This same subroutine (9340) is used with whichever DOS elected. After this we are back at 9360), the line width is POKED. The LPRINT is to return the printer carriage. Then we RETURN to line 8984.

8994 diverts to line 9000 "if using cassette and TS-2040"

8996 is another BOOLEAN logic having three possible routes for the GO SUB. We will take one route. If tm=3 then Larken DOS has been elected back at line 8980, so the <GO SUB 9330> will execute. At line 9330 we see the Larken syntax for CAT. The Catalog is put on screen. Then there isa <GO SUB 9370>. Follow that and we find a dual use prompt to INPUT the "title of the Code program to load". Then its back to 9330 again where the Larken LOAD is accomplished. Now we have the code file that we want to print out loaded. The RETURN puts us back to line 9000.

9000 At this time the Catalog is still on screen. The Code file has been loaded. The program asks for INPUT of "starting address" and then "byte length" of the code file. Any "friend" who sends text in code will tell us that. But if using the Oliger catalog, it will show this information. Next, we need to INPUT the format line length. Guess at 64 if you dont know.

9002 here we poke the line length to the Oliger control address, and at line 9004 we poke the line length-one to the Aerco prcode control address.

9006 here we <OPEN# 2,"p"> to send PRINT to the printer instead of to the screen. Then we start the counter "n" to count from "Code Start address" to "sa+byte length", in STEPs of the line length.

9008 Now we start counter "y" to count from "n" to the end of the line which will be "n+ll-1".

9010 here we PEEK the code file at the "y" address, and assign the Code of the peeked character to var "c".

9012 mscript uses ENTER to skip a line. ENTER is code 13. So here we cause a line feed each time code 13 is peeked from the code file.

9014 here the PEEKED character is printed, one at a time, as the y counter counts. Note the ";" which avoids a carriage return and keeps the next print position on the same line.

9016 next y gets the next peeked character until y counts

```
9000 LET /p=o: POKE 23324,0: LET
 lc=1: LPRINT : CAT : INPUT "Inp
ut Disk File Name";z$: INPUT "In
put File Starting Address";sa: I
NPUT "Input File Byte Length";bl
: INPUT "Format Line Length";ll:
 POKE 23323,ll: LOAD /z$CODE : O
PEN #2,"P": FOR n=sa TO sa+bl ST
EP ll
9005 FOR y=n TO n+ll-1
9010 LET c=PEEK y
9015 IF c=13 THEN PRINT '
9020 PRINT CHR$ c;
9025 NEXT y: LET lc=lc+1: IF lc>
55 THEN PRINT : CLOSE #2: LET lc
=1: OUT 127,12: BEEP 1,20: CLS :
 PRINT "Adjust Page, THEN ENTER"
: PAUSE 0: OPEN #2,"P"
9030 NEXT n
9035 CLOSE #2
9040 PRINT '''"<1> For next file"
'''"<2> To Quit": INPUT "Choice?
";a: IF a<1 OR a>2 THEN GO TO 90
40
9045 IF a=2 THEN STOP
9050 GO TO 9000
```

"INIT.BAS" for Aerco FD-68 DOS

```
9900 INK 7: PAPER 0: BORDER 0: C
LS : LET base=64500: POKE 23731
,INT ((base-1)/256): POKE 23730
,(base-PEEK 23731*256-1): OUT 2
44,1: CAT "Cclear.bin",: CAT "C
prcode.BIN",: CLS
9910 CLS : PRINT AT 3,5;"Printer
 Status is:"''
9920 PRINT TAB 2;"Printer in use
 is: ";: IF PEEK (base+803)>2 T
HEN  PRINT "NOT SET"
9922 IF PEEK (base+803)=1 THEN
PRINT "TS-2040"
9924 IF PEEK (base+803)=0 THEN
PRINT "Centronics"
9926 PRINT ''TAB 2;"Line Width i
s: ";PEEK (base+4)+1
9930 PRINT ''TAB 2;"Line Feed Sw
itch is: ";: IF PEEK (base+5)=1
0 THEN  PRINT "-ON-"
9931 IF PEEK (base+5)=0 THEN  PR
INT "-OFF-"
9933 PRINT ''TAB 5;"Your Choice?
"''TAB 5;"1> RESET PARAMETERS"'
'TAB 5;"2> Leave AS IS": INPUT
"Choice? ";b: IF b<1 OR b>2 THE
N  GO TO 9933
9934 IF b=2 AND PEEK (base+803)=
1 THEN  RANDOMIZE USR (base+2):
 GO TO 9995
9935 IF b=2 AND PEEK (base+803)=
0 THEN  RANDOMIZE USR base: GO
TO 9995
9936 CLS : PRINT AT 10,2;"SELECT
:"''TAB 2;"<1> Centronics, or"'
'TAB 2;"<2> TS-2040": INPUT a:
LET a=a-1: IF a>1 THEN  GO TO 9
937
```

up to the line length, upon which time line 9018 will execute.

9018 if tm>2 then we are using the Aerco printer code for Larken or Aerco DOS. The PRINT is to account for a built in glitch in the Aerco prcode.

9020 var lc is the line counter. lc is incremented. If lc reaches 56 then a page end is directed. CLOSE# 2 allows a prompt to be put on screen. Line count is put back to one, <OUT 127,12> causes the printer to feed the form to the next page beginning. The prompt is give to adjust the page (single sheet printer). Then when ENTER is touched <OPEN# 2,"p" puts PRINT back to the printer.

9030 next n gets the next line of characters PEEKED from the Code file.

9035 will only execute after all of the characters in the code file has been peeked and printed. We're finished. So we CLOSE# 2, and give options on screen.

9050 if we elected to get the next file, the whole process starts over, but without the intelligence gathering. The Catalog is gotten to load in the next code file to print.

8982 somehow I skipped part of the action of this line for the Aerco FD-68 and Larken DOS. If either of these are in use, then the Aerco prcode is needed for Centronics printing. If "tm=3" (Larken DOS) then the GO SUB is to line 9310 to LOAD and initialize the Aerco prcode. But if the Aerco FD-68 is in use, the GO SUB is to line 9300 to load and initialize the Aerco prcode. If the Oliger DOS is in use there is no need to load a code table, as the printer code is contained in the SAFE EPROM.

While printing, you may BREAK and stop the printing. When you do that do a <CLOSE# 2> in order to re-direct PRINT to the screen. This will be when you elect a line length that is different from that formatted by the author of the Code data. You can determine the correct line length to input by looking at the aborted print out. Then just type <RUN ENTER>. Print out is fairly fast, as fast as my daisy wheel printer can print. But it wont be as fast as a dot matrix printer can print in the draft (fast) mode.

```
9938 POKE (base+803),a: IF a=1 T
HEN  RANDOMIZE USR (base+2): GO
 TO 9995
9942 CLS : PRINT AT 10,0;"Does P
rinter need a LINE FEED  with
each Carriage Return?"''''"<1> Y
es - OR - <2> NO"
9943 PAUSE 0: LET a=(CODE INKEY$
-49): IF a<0 OR a>1 THEN  GO TO
 9943
9944 POKE (base+5),(NOT a)*10: P
OKE (base+805),(NOT a+10)
9960 CLS : PRINT AT 10,0;"Key De
sired Printer line length:": IN
PUT "Input line length: ";ll: I
F ll>255 THEN  GO TO 9960
9964 POKE (base+4),ll-1: POKE (b
ase+804),ll-1
9968 CLS : GO TO 9910
9995 CLS : PRINT "Copy these imp
ortant notes:"''"1. Always init
ialize the var  ""base"" in
 your programs- ie, <LET base
=64500>."''"2. The Prcode does
not work  until it is TUR
NED ON, by:  <RANDOMIZE USR
 base>. This  is for Centro
nics Printers."''"IT should be
TURNED ON before  doing any PO
KES."''"3. For TS-2040 use, <RA
NDOMIZE  USR, (base+2)."''"4
. There are 64 bytes ABOVE this
  Code, (base+804 to 867) that
  can be used for hiding Vars
  that need to be carried fo
rth  to a new BASIC program."'
'"The UDG area 65368-65535 is l
eftintact."''"5. To turn ON lin
e feed, <POKE  (base+5),10>"
''"6. To Turn OFF LINE FEED, <P
OKE  (base+5),0"
9996 PRINT ''"7. To set printer
Width,  <POKE (base+4)
,63> for 64 chr  lines. (width
 minus 1)"''"8. To TURN ON prin
ter code,  <RANDOMIZE USR
 base>"''"9. To Turn OFF printe
r code,  <RANDOMIZE USR (b
ase+2) >  ie, to use TS-20
40 printer"''"10. This code is
saved at 64500,  867. You ma
y use the  RALOADER pg
m (Jack Dohany)  to RE-LOCA
TE the code"''"ENTER when Ready
."
```


Cont'd Page

```
9997 PAUSE 0: CLS : PRINT AT 6,0
;"The Printer driver code IS no
w  initialized."'"NEXT, a CAT
will be done with  a  prompt t
o load a program."'"During ope
ration, if  the code  is overwr
itten, you may re-load this pro
gram to re-set  it into memory.
"'"ENTER WHEN ready.": PAUSE 0
: CLS
9998 IF a$<>"" THEN  OUT 244,1:
CAT "",: INPUT "Enter a Title t
o LOAD (all bet- ween quotes)";
a$: CAT "a$",
9999 DELETE 9900,
```

"INIT", Larken DOS, Aerco prcode

Utilities "INIT" and "Cclear Code"

The "INIT" utilities for the three Disk Operating Systems are slightly different. Each listing may be used to pick up another little twist for another DOS. The Larken INIT uses the prcode as is supplied by Aerco on cassette for the Aerco printer interface. INIT for the Aerco FD-68 DOS uses the prcode as supplied on disk by Aerco and modified to be re-locatable by Jack Dohany in his program RALOADER. For the latest improvements in RALOADER call or write Jack. His address appears in the January UPDATE under "TS-2068 Suppliers". Both "INIT" and "Cclear Code" are explained in this issue's editorial pages.

```
9997 CLEAR 64455: LET PRINTORG=6
4461: POKE 26704,INT (PRINTORG/
256): POKE 26703,PRINTORG-(INT
(PRINTORG/256))*256: POKE 64456
,1: POKE 64458,0: POKE 64457,0:
 POKE 64459,79: BORDER 0: PAPER
 0: INK 7: CLS : RANDOMIZE USR
100: OPEN #4,"dd": PRINT #4: LO
AD "prcode.Cl"CODE 64456,1111:
CLS : BEEP PI/PI,PI*PI: INPUT "
INPUT LINE WIDTH LIMIT";Lw: POK
E 64459,Lw-1: BEEP PI/PI,PI*PI:
 PRINT AT 20,1;"Install PROGRAM
 DISK, ENTER": PAUSE 0: CLS : R
ANDOMIZE USR 100: OPEN #4,"dd":
 PRINT #4: CAT "",: INPUT "Inpu
t Title of Basic Program,  ""[
All Between Quotes]"" ";a$
9998 IF a$<>"" THEN  PRINT #4: L
OAD a$
9999 DELETE 9997,
```

"INIT" for Oliger DOS

```
9980 CLEAR 65535: CLEAR 65367: B
ORDER 0: PAPER 0: INK 7: CLS
9982 INK 5: FOR n=0 TO 31: PRINT
 AT 6,n;"*";AT 16,n;"*": NEXT n:
 INK 7: PRINT AT 9,7;"PRINTER SE
T-UP";AT 12,7; FLASH 1;"TYPE of
PRINTER": INPUT "<1> Centronics,
 or <2> TS-2040";pr
9988 IF pr=2 THEN LET /p=t: GO T
O 9994
9990 LET /p=o: PRINT AT 12,7; FL
ASH 1;"SET LINE LENGTH": INPUT "
Line Length? ";ll: POKE 23323,ll
: PRINT AT 12,7;"Printer Need Li
ne Feed?": INPUT "INPUT: <1> Yes
 or <2> No :";lf: POKE 23324,(10
 AND lf=1)+(0 AND lf=2): CLS
9994 FLASH 0: LOAD /"DOSDEX MGR"
```

SOFTWARE REVIEW
BY Tex Faucette

MONEY MACHINE II (T/S 2068)
ABBA Software Productions

Does Vanna know about Banna? And in these days of intensive search for true Artificial Intelligence there arises an even more important question: Is Banna aware of Vanna?

Stay tuned for the next exciting episode of this digital drama!

Vanna stars in a letter-turning role on what is probably the most popular game show on todays boob-tube. While viewing times may be different for the various "market areas", a few of us lucky ones may even be able to observe this shapely lady twice daily!

For those unfortunate ones who are able to observe Vanna infrequently or not at all, but are nevertheless equipped with a T/S 2068, there remains the option of MONEY MACHINE II. This option not only permits discretionary observation of Miss Banna Brite (as shapely an assembly of pixels as one could wish for), but also permits one to compete in the actual game! Winnings are of course inaginary, but with Banna turning the letters, who cares?

MONEY MACHINE II has performed flawlessly in several critical tests including exposure to unsupervised grandkids and their cronies. If there is a tougher test for 2068 software it has escaped my attention. Full set of instructions for playing the game are included. Auxillary tapes containing additional word puzzles are available, also a "Master Puzzle Maker" program for do-it-yourself enhancements.

MONEY MACHINE II is available on casette from Herb Bowers, Sr., 2588 Woodshire Circle, Chesapeake, VA 23323. Current price is listed as $12.00 (US), a bargain you shouldn't miss.

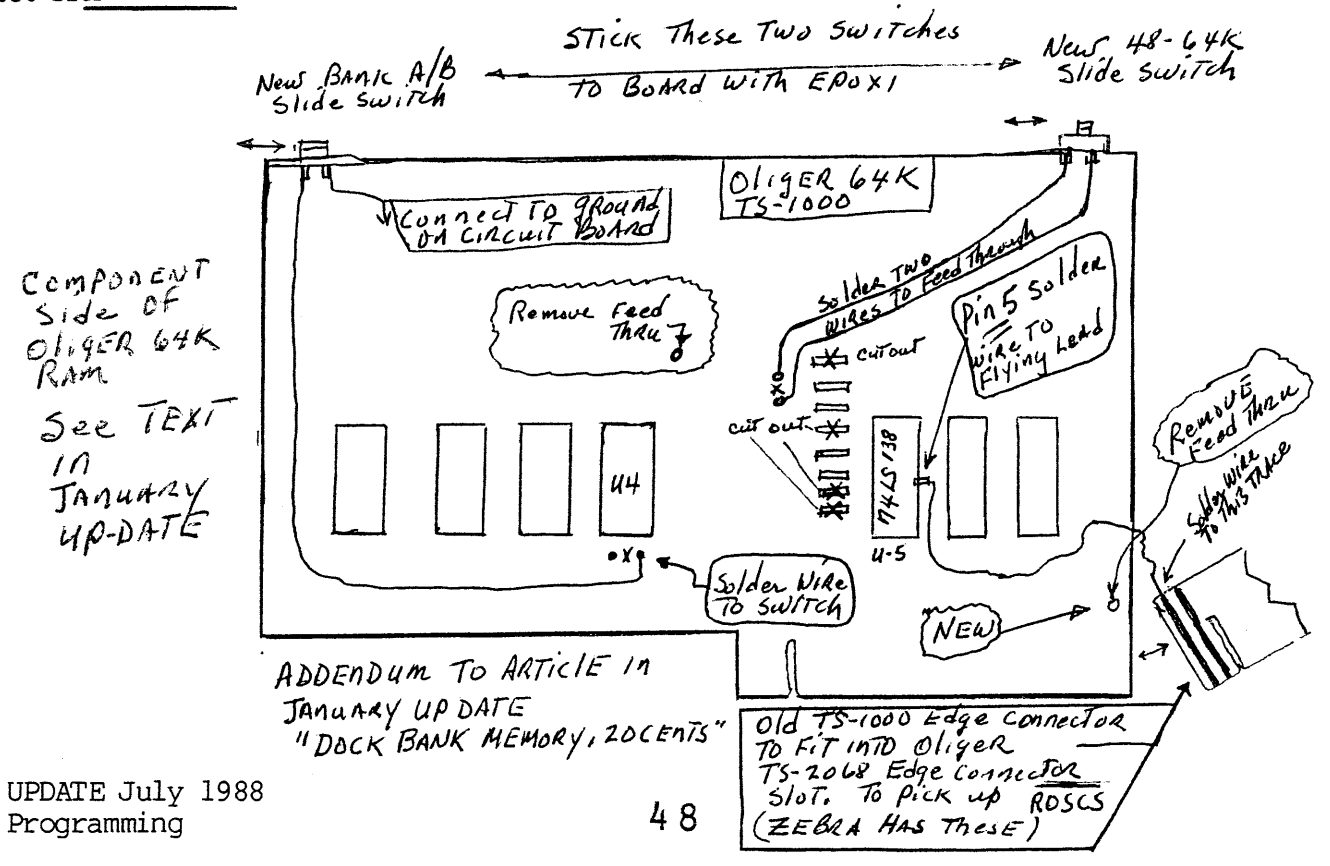
```
9000 REM ** PEEK VARS **
9002 BORDER 0: PAPER 0: INK 7: C
LS
9004 PRINT "The BASIC Program Le
ngth is     ";(PEEK 23627+256*PE
EK 23628-1)-(PEEK 23635+256*PEEK
 23636);" Bytes"''
9006 PRINT "Systems Variables:"'
'
9008 PRINT "ERR SP=";PEEK 23613+
256*PEEK 23614'"LIST SP=";PEEK 2
3615+256*PEEK 23616'"NEWPPC=";PE
EK 23618+256*PEEK 2369'"NSPPC=";
PEEK 23620'"VARS=";PEEK 23627+25
6*PEEK 23628'"PROG=";PEEK 23635+
256*PEEK 23636'"E LINE=";PEEK 23
641+256*PEEK 23642
9010 PRINT "WORKSP=";PEEK 23649+
256*PEEK 23650'"STKBOT=";PEEK 23
651+256*PEEK 23652'"STKEND=";PEE
K 23653+256*PEEK 23654'"S TOP=";
PEEK 23660+256*PEEK 23661'"UDG="
;PEEK 23675+256*PEEK 23676'"MEMB
OT=";PEEK 23698+256*PEEK 23699
9018 PRINT "RAMTOP=";PEEK 23730+
256*PEEK 23731'"P-RAMT=";PEEK 23
732+256*PEEK 23733'"MSTBOT=";PEE
K 23744+256*PEEK 23745
9025 PRINT '''"For a VARS area Pr
int out,     Key <CONT ENTER>":
 STOP
9035 FOR n=(PEEK 23627+256*PEEK
23628) TO (PEEK 23641+256*PEEK 2
3642-1)
9040 PRINT n;"=";PEEK n: NEXT n
9050 STOP
```

The "PEEK VARS" Utility

The PEEK VARS utility is designed to be MERGED to a Basic program either under construction, or one that is being modified. Use it as a "dangling sub-routine" to be consulted when facts about the System Variables are needed. <<GO TO 9000>> gets a Screen print of the System Variables that may be needed for program analysis. Then after the System Vars are known, <<CONT ENTER>> screen prints the whole Variable Storage are of memory, disassembled by memory address and character code (in decimal).

Type <<CLEAR>> before SAVING the utility to Disk, and do not use a Auto RUN line. The utility is exactly 1110 bytes in length. Keep this off-set figure in mind to extrapolate the ACTUAL System VARS when the utility is deleted. These MOVING system vars are VARS and E LINE. Note that the only variable used in the PEEK VARS utility is "counter n". The use of "n" as a counting variable will extend the TOP of VARS and E LINE exactly 19 bytes-- "IF n has not been initialized in the main program". However, if n IS used in the main program, then the PEEK VARS utility will have no affect at all upon the SIZE of the vars file.

Note that the n counter in line 9035 counts from "VARS to E LINE minus one", which encompasses the current Vars File in memory. If a Counter n has not been used in the main program, then EDIT and make the count "TO E LINE+18" in order to account for the 19 bytes within the vars file that is costed by initializing the n counter. If you CLEAR and RUN the utility, then CONT ENTER to print the Vars File, the print out will be in error by 19 bytes. This is because Counter n had not been initialized when the System Vars was printed. A <GO TO 9000> will then print an accurate set of vars. You may add more PEEKS before line 9025 to get the status of the various FLAGS. A study of the TS-2068 User Manual, Appendix C and D, may help.



The JULY/APRIL Issue Disk

Consolodated into ONE

The July and April Issue Diskettes have been consolodated into one giant assemblage of programs and utilities. The consolodation was necessary because the Budget program, by Bob Mitchell, was split between the two issues. I have not ever seen such a combined useful and educational works presented in a computer magazine. The combined April/July Issue Disk packs in a virtual plethora of programs and utilities, almost filling a 40 track diskette. The April/July issue disk is available in all three DOS formats, Oliger DOS, Larken DOS, and Aerco FD-68 DOS, and are recorded on 5 1/4 inch Double Side Double Density, 40 or 80 track.

The sales proceeds of Issue Disks are shared between UPDATE and the Authors in a diabolic scheme designed for both to become rich quickly at the expense of the Subscribers. UPDATE takes $8.00 from each disk sale for the tiny amount of work to assemble the programs in three disk formats, plus cost of the disk and mailing. The (already rich) Authors make a killing sharing the remaining $8.00. The accounting firm, Sly, Slickingheimer, and Dolittle, assists UPDATE in disbursing the huge profits to the authors and helping us to evade taxes. This requires us to hold quarterly business meetings in Monaco aboard the consortium's 300' Yacht.

ORDERING: Send $16.00 (check or M/O) for the April/July Issue Disk. Please specify for WHICH DOS (Oliger, Larken, or Aerco FD-68), and for which type of Disk Drive (40 or 80 track). A Catalog of the disk is given below.

FILENAME	TYPE	CYLS	SIZE	START
code.Cl	BYTES	1	1826	63417
DOSDEX MGR	BASIC	2	6007	2
LOAD	BASIC	1	87	10
BUDGET.Bl	BASIC	1	986	60
Menu.Bl	BASIC	1	702	10
budget.CJ	BYTES	6	25923	32000
spread.CJ	BYTES	1	3904	58400
budget.BJ	BASIC	5	24554	10
spread.BJ	BASIC	1	2855	8000
BRIEF.Bl	BASIC	2	5445	5
MS-TAS.Bl	BASIC	1	924	8000
PEEK VARS	BASIC	1	196	
budget.CY	BYTES	6	25923	32000
SEARCH	BASIC	1	4978	1
FILE_DEX	BASIC	1	4962	1
MARK-MOVE	BASIC	1	2092	1
VERI_DISK	BASIC	1	4496	1
Chopin	BASIC	2	7584	220
Btls	BASIC	3	14780	9900
Mzrt	BASIC	2	7674	1
Spell	BASIC	2	7411	100
Ohm's Law	BASIC	1	1257	100
Multiply	BASIC	2	6038	10
Type Style	BASIC	1	2055	4608
Piano	BASIC	1	4747	10
budgetD	BYTES	1	652	64716
bujE.Cl	BYTES	4	17642	47047
bujJb.CT	BYTES	2	8320	33280
bujJs.CT	BYTES	1	2048	33280
help.CT	BYTES	1	3968	33280
bujJ3.CT	BYTES	3	11200	33280
bujJx.CT	BYTES	1	2944	33280
OLI-MS-TAS	BASIC	1	1512	8990
OLI CODE	BASIC	1	1841	
prod	BYTES	1	256	23296
Cclear	BYTES	2	5367	60000
Super MS-T	BASIC	1	3060	8972
REM	BASIC	1	3815	
file_dex	BASIC	1	1118	
1	C ARR	1	3565	
Ll MCLOAD	BASIC	1	712	
INIT.Bl	BASIC	1	543	9980
ms-tas	BASIC	1	3045	8972

TOTAL FILES: 43

The OCTOBER and JANUARY Issue Disks

Now available for All Disk Operating Systems

The OCTOBER 87 and JANUARY 88 Issue disks are now available for three Disk Operating Systems; Oliger DOS, Larken DOS, and Aerco FD-68 DOS. Both October and January Disks contain a major software plus several useful utilities making their value at least twice the price. See the diskette Catalogs below.

ORDERING: Sent $16.00 Check or money order for the disk of your choice. Please specify for which DOS, and whether for 40 or 80 track drive.

The October Issue Disk

JLO SAFE V2.52 1988, J. Oliger

DISK NAME: OCT UPDATE
FORMATTED @80 TRACKS, 2 SIDE(S)
CAPACITY: 159 CYLS/795K BYTES
FREE: 132 CYLS/660K BYTES

FILENAME	TYPE	CYLS	SIZE	START
SDOS MMRG	BASIC	3	12180	2061
Mail Merge	BASIC	2	10099	2061
J-UTIL	BASIC	2	9441	9800
ALLEGRO	BASIC	1	1576	1
pseudo	BASIC	1	1805	5
pseu	VRBLS	1	564	
PRT USING	BASIC	1	1373	10
PROMPTS	BASIC	1	1066	90
MENAGERIE	BASIC	1	4671	2070
PTR STATUS	BASIC	1	684	1
fm	BYTES	2	6912	16384
ml	BYTES	2	6912	16384
pg	BYTES	2	6912	16384
cpr	BYTES	2	6912	16384
fo	BYTES	2	6912	16384
mrg	VRBLS	1	612	
LEADER	BASIC	1	1766	150
	BASIC	1	4485	152

TOTAL FILES: 18

The January Issue Disk

JLO SAFE V2.52 1988, J. Oliger

DISK NAME: U/D JAN 88
FORMATTED @40 TRACKS, 2 SIDE(S)
CAPACITY: 79 CYLS/395K BYTES
FREE: 70 CYLS/350K BYTES

FILENAME	TYPE	CYLS	SIZE	START
DISK MGR	BASIC	2	5988	2
Purity	BASIC	1	4512	152
LIST LOOK	BASIC	1	2259	2
DOSDEX	BASIC	1	5057	2
mterm2.	BASIC	1	2756	
PROG	BASIC	1	955	9999
prog	BASIC	1	955	
PGM-MOVE	BASIC	1	1709	10

TOTAL FILES: 8

OLIGER 2068 DISK SYSTEM PRICES AND BASIC INFORMATION

DISK BOARD "A"
Bare pc only: $17.95pp
Kit of board with parts: $55.95pp
Assembled & tested: $66.95pp
Two drive data cable for above, 3 foot long total: $16.95pp
Four drive data cable for above, 4 foot long total: $26.95pp
WD1770PH-00 disk controller chip: $19.95pp (spare or replacement-limit 1 per order)

DISK BOARD "B" W/NMI SAVE
Bare pc with JLO SAFE Disc Basic eprom: $26.95pp
Kit of board with parts: $45.95pp
Assembled & tested: $63.95pp

PACKAGE OF BOTH DISK BOARDS "A" & "B" W/NMI SAVE
Bare pcs only with JLO SAFE Disk Basic eprom: $43.95pp
Kit of both boards with parts: $99.95pp
Both boards assembled & tested: $127.95pp
Both boards, assembled & tested w/2-drive data cable: $139.95pp
The DiskWorks! Both bds assd w/2-drv data cable & assd 2068 Expansion Bd: $189.95pp

---

2068 PARALLEL PRINTER PORT Bare board w/cassette:$16.95 Kit of board w/parts:$24.95
Assembled & tested:$30.95 Bare board w/cass (gold):$19.95 Gold board w/parts:$27.95
Assd & tested (gold):$33.95 Printer cable for this port:$14.95 (standard Apple II to Epson type,3ft.)
Port software cassette only: $6.95pp (Use as alternate Aerco port driver)
This very small but very capable Centronics type printer port for your 2068 will drive most any printer that has a Centronics type parallel interface. The port uses the same port and BUSY bit assignment used by the Aerco 2068 printer I/F, and thus is compatable with the vast amount of 2068 software that supports this interface. This port is fully decoded and contains a "PRINTER READY" LED to show your printer's current status. Software on cassette included with this port allows use of the standard LPRINT and LLIST commands and use of AT, TAB, and "," the full width your printer is capable of. Also included are hi-res screen dumps (via RAND USR call) for the Okidata dot addressable graphic printers, the Olivetti PR2300 ink jet printer, the Gemini 10X printer, and the Gorilla Banana printer. An ASCII screen dump of ASCII characters is also included for use with ANY printer. Unlike most other 2068 printer driver software, THIS software mainly resides in the 2040 printer buffer, using absolutely none of the ram normally used by Basic and well out of the way for opening DFILE 2 on the 2068.

2068 EXPANSION BOARD Bare pc board:$14.95 Board w/parts:$43.95
Assembled and tested Oliger 2068 Expansion board $54.95.
Also available with gold plated rear edge traces. Add $3.00 per board to above prices for these. This compact (3 3/4" * 4 3/8") expansion board is designed to plug into the rear expansion connector of your 2068 computer. All of the 2068 boards from John Oliger Co., except the 2068 User Cart board are designed to plug into this expansion board, (the boards have male edge traces, and plug into this board similar to the way new boards are plugged into the Apple computer) so this one is a must if you plan on adding more circuits to your 2068 with John Oliger Co. products. The board includes one "spare" 20 pin ic socket for a user circuit, a rear edge connector feedthrough for attaching the 2040 printer, modem, etc., and the nec. circuitry for an ultra stable RGB monitor interface. (NOTE: The use of the RGB circuitry requires opening your 2068, cutting one trace, and installing one jumper.) The board's purchase includes assembly instructions and a schematic of the RGB interface. The parts kit for this one includes five precut & slotted edge connectors, 2 ic sockets, 1 74HC00 IC, and 1pc 47K Ohm 1/4W resistor.

2068 EPROM PROGRAMMER Bare board:$12.95 Board w/pts (no eproms):$25.95
2068 Pgmr asmbld & tstd:$33.95 Add $3.00 per board for gold plated edge traces.
This board adds the capability of programming 2764 & 27128 eproms to your 2068 system. This one is a must for serious use of the 2068 User Cart board. This board comes with complete documentation on its assembly, use, theory of operation, tips, and details on replacing the Home rom of the 2068 w/a 27128 eprom. This programmer requires the Vpp Power Supply (below, recommended) or a 4-22VDC regulated adjustable power supply w/voltmeter.

2068 USER CARTRIDGE BOARD Bare board:$11.95 (or 3@$31.95) Board w/parts (ex eprom):$15.95
3pcs board w/parts:$45.95 Assembled & tested:$20.95ea or 3 for $59.95 post pd (no eproms)
Also available w/gold plated edge traces. Add $3.00 per board to above prices for these gems.
ATTN: Software Producers & Users Groups: Write for a quote on quantities of 10pcs. or more.
This board is designed to plug into the cartridge port of the 2068 computer. The board can hold 2-2764 or 2-27128 eproms in any combination. It allows the 2068 computer owner, when used with the 2068 or TS1000 programmers, to store his own Basic programs in cartridge form for fast and easy access. The board's purchase includes assembly instructions, schematic, and instructions for its use with Basic programs.

Vpp POWER SUPPLY Bare board:$4.49 Board w/bd mounting pts:$9.95
Assembled and tested, less transformer:$15.95 (TS2040 Printer's power pack may be used as substitute for transformer with suitable connector). This small single sided board provides 4.4, 21, and 25VDC for use with any programmer offered by The John Oliger Co. This circuit is designed to put as little stress on the eprom as possible during programming and is recommended for use with these programmers. The board is supplied with schematic, assembly instructions, and use instructions. Parts kit does NOT include transformer or power cords/alligator clips.

For detailed product list, write--------

THE JOHN OLIGER CO.
11601 Whidbey Dr.
Cumberland, IN 46229

THE 1988 MIDWEST REGIONAL TIMEX-SINCLAIR CONFERENCE

Saturday & Sunday August 27 & 28, 1988
Cleveland, Ohio

hosted by The Greater Cleveland Sinclair Users Group

co-sponsored by: Zebra Systems, Syncware News
Lemke Software, Sharps Inc, Cuyahoga Valley Software
Budget Robotics, Indianapolis, CATS, SAF Michigan
and the SW Pennsylvania Users Groups

LOCATION: BECK CENTER FOR THE ARTS
(a community theatre/arts complex)
17801 Detroit Avenue
Lakewood, Ohio (1st suburb W. of Cleveland)

Admission: $4.00 each day, or
$7.00 for 2 days (includes Sat. night hospitality)
(exhibit table rentals include 2 admissions)

+ +

Suggested Hotels:

near I-90; Ramada Inn, 20375 Center Ridge Rd (US Rt 20)
Rocky River, Ohio 800-272-6232. Closest Hotel.

Red Roof Inn, I-90 & Crocker Road, Westlake Oh.
800-848-7878

near I-71; Days Inn, 4181 W 150th, Cleveland 216-252-7700

near I-480; Scottish Inn, 22115 Brookpark Road, Fairview Park,
216-734-4500

Airport; Sheraton Hopkins Airport, 216-267-1500

Restaurants: There are many fine reasonably priced family restaurants in the area. A list will be available.

+ +

Travel Routes:

Lakewood Ohio is the first suburb west of Cleveland along Lake Erie. Interstate access is via I-90.
From I-90 in Lakewood, exit at the McKinley Ave or Warren Road, and head north to Detroit Ave, then west.
From I-90 in Rocky River(next suburb west), exit at Detroit Road, and head east. Detroit Road is US Rt 6.
From I-71, exit at W. 150th St. and head north. W 150 will become Warren Road. Continue north into Lakewood to Detroit Ave. and turn west.
These routes are shown on the AAA Ohio map.

Warning! Dont start this one if you're sleepy. It requires concentration. -ed.

PEEKING AND POKING ABOUT

by Bob Mitchell
Willowdale Ontario

One of the fun things about using the TS2068 is the ability to PEEK and POKE about in amongst the machine code and particularly when using TIMACHINE. This is the first in a series of articles I'll write for TS2068 Up-Date on this subject. Bouquets or brick-bats are welcome!

****

For starters, let's look at a scenario using the compiler Timachine. You have compiled a data file and entered about 18000 bytes of data. Then you decide you wish you had built in a routine to print the data to your big printer. It was only designed to print to the 2040 printer.

You have a choice: You can redesign the BASIC and recompile in which case you have to re-enter all the data (well, not really but I'll come back to that); or you can PEEK out the data from the compiled code and using a simple routine in BASIC send the data to the big printer.

Once again, you will need the REM! LIST info which you should have kept when you compiled your program. If you didn't do it at the time and you kept a copy of your BASIC program then go through the motion of compiling to get the runtime and variable lists and addresses.

Find the start address of the array or string containing the data. Remember the data really starts two bytes further on. For argument's sake let's say your array was d$(600,32) and let's say that the start address listed was 40000 so that means that the data starts at 40002. In your BASIC program assign this to variable s1. By the way, don't forget that Timachine limits arrays to two dimensions. Tough but that's the way it is.

If all 600 subscripts are not full, then you should know how many are used so far and this value should be in some variable, say, n. Look for n in your variables list and let's hope your made it an integer when you compiled. If not the value will be in floating point notation. All is not lost if it is but it gets more difficult and I shall not go into that solution here. Assuming the value is indeed in integer notation, then n will be located at the address 33335 and the actual value of n will be at addresses 33335 and 33336. Use LET v1= PEEK 33335 + 256 * PEEK 33336 in BASIC to get this value.

Now build a FOR-NEXT loop to LPRINT the data to the big printer.

```
300 FOR k = s1 TO s1+(v1 * 32) STEP 32
305 FOR m = 0 TO 31
310 LPRINT CHR$ PEEK (k + m);
313 NEXT m
320 LPRINT
330 NEXT k
```

This will not be as fast as a compiled routine would be, but it would be acceptable.

As I said above, you don't really have to re-enter all the data after you have recompiled a program. Here's how I did it.

1. Note the value of n (records entered so far).

2. SAVE the data in your original compiled program by:
   SAVE "prog.C1"CODE start,len
   where start is the array start address + 2
   and len is the total length of the array
   (600 * 32 = 19200).

3. Restart the TS2068 and
   LOAD your newly recompiled program and make sure it is initialized by running it from the first entry point. ie, the first REM ! OPEN #.
   LOAD "prog.C1"CODE,newstart
   where newstart is the new start address + 2
   POKE the value n into the correct address.

4. Proceed as you did in your earlier version.

On a different tack, here is a way to adapt the LPRINT routine supplied with the Hacksel printer interface hardware to the Spectrum. This works on my emulator which is switched on by using OUT 244,3; whether it works on a real Spectrum, I know not. This technique should be adaptable for other CPI LPRINT routines.

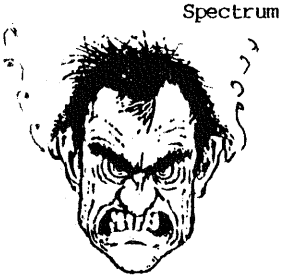
1. Change two ROM CALLS in the LPRINT code:
   CALL 0745h to CALL 0C10 (at 65096 or FE48h)
   CALL 2009h to CALL 1F54 (at 65117 or FE5Dh)
   (eg, CD4507 becomes CD100C and you POKE the decimal equivalents into the right places).

2. Now SAVE this and then CLEAR 64999 and LOAD the revised code into Spectrum.

3. POKE 23749,0: POKE 23750,254
   (These are the locations in the Spectrum Channel Info area corresponding to 26703 and 26704 in the TS2068.)

4. POKE 65535 with 1 or 0 as required to turn the Sinclair character set off and the ASCII set on, and vice versa as instructed in the instructions for LPRINT. In other words, this requirement is the same for the Spectrum and TS2068.

WANTED! Capture Alive. Desparado last seen in Portland, OR. Left no forwarding address. Name: Syd Wyncoop, a Guru. Need to contact to get UDATED. DONT SHOOT, but contact UPDATE with new location. We will apprehend.



If you dont subscribe then may your SCLD Chip get fried, your tax return be audited, the bank lose your deposit, and someone use your credit card.

Bill Jones
Editor



****

IF YOU OWN A 2068, AND BUY A Z-88--YOU WILL HAVE ENOUGH COMPUTER
POWER TO RUN *ALL* THE PROGRAMS NORMALLY USED..*EASILY*
IF YOU NEED TO OWN A PORTABLE COMPUTER THAT HAS THE EASIEST WORD PROCESSOR AND SPREADSHEET CAPABLE OF SERIOUS BUSINESS AND PERSONAL APPLICATIONS; THE Z-88 IS THE ONE!

1. *WILL TALK TO ANY COMPUTER SERIALLY*...Italics
2. WILL PRINT TO ANY PRINTER.....Regular
3. NOISELESS TYPING IF NEEDED.....Bold
4. ADD UP TO 3 MEGABYTE MEMORY.....Underline
5. CAN REACH PRINTER GRAPHICS — Superscript....$e=mc^2$
6. MAKE YOUR OWN PROGRAMS — Subscript.....H_2O
7. *DIARY,CLOCK,CALENDAR,ALARM*.....Bold-Italics
8. DIFFERENT FONTS AVAILABLE — |
9. USER DEFINED CODES — |2
10. PORTABLE MEANS 4 DOUBLE AA BATTERIES!!!.....CONDENSED
11. This page typed and printed by Z-88

IMMEDIATE DELIVERY; PAID FOR BY ME; DELIVERED BY UPS--UPON RECEIPT OF $550 MONEY ORDER OR CLEARING OF CHECK MADE OUT TO MIKE FINK

SUGGESTED EXTRA OPTIONS: A. PARALLEL PRINTER CABLE.....$65
B. AC ADAPTER.....$20

MIKE FINK 355 W 39 ST.
N.Y.CITY 10018-1401
212 695 8598

****

Editor's Note: Both Mike Fink and Sharp's Inc. are UPDATE subscribers. Mike was so enthused about the new Sinclair Z88 that he pulled out all stops to become a Z88 Dealer. Sharp's Inc. is also a dealer for the Z88. This new Lap Top computer is on its take off roll and will surely fly to high altitude. UPDATE will report its progress.

Sharp's, Inc.

Rt. 10, Box 459
Mechanicsville, VA 23111
(804) 746-1664 or 730-9697

QL Computer

$129.95 includes PSION s/w

We carry **ALL** hardware and software lines for the Sinclair QL. Compare our prices. VISA and MASTERCARD accepted (3 % surcharge).

WRITE FOR FREE CATALOG