UPDATE! $4.95 July 1990

ComputerSystems

Take it to the limit!

A quarterly Magazine for the users

EXPLORING THE POWER



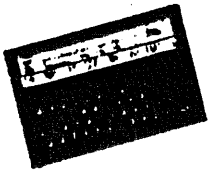
OF

SINCLAIR COMPUTERS

THE CAMBRIDGE Z88
Portable Power

THE SINCLAIR QL
32 Bit Power House

THE SINCLAIR TIMEX TS-2068
Versatile Desk Topper



In this issue. . .



UPDATE MAGAZINE CHANGES PUBLISHER

UPDATE MAGAZINE
513 E. Main Street
Peru, IN 46970

NEWS AND NEW PRODUCTS

LKDOS SOFT-WARE



MAXCOM 300/1200 baud Terminal/BBS

This full feature modem software lets you effectivly link the modem to the disk, so you can upload or download files larger than 100K.This lets you send NMI saves and large text files as well as Basic or Code files, which other modem programs couldn't send because of memory limitations. Also files can be sent with or without a information header (Spec-term compatible) .

It also has a 64 column display and is the only 2068 modem program that will run at full 1200 baud without missing characters. Other features of the terminal mode are IBM graphic characters, auto-dialer, macro keys, 110K disk buffer for captured text, auto repeat keys.

The BBS is the most elaborate available for the 2068 and allows the remote user full access to the disk drives similar to a CPM BBS. Features of the BBS are - 300/1200 baud, operation with fast reponce even at 1200 baud, Passwords, priority levels, Use time limits, Multiple Message Bases, Uploads and Downloads are not limited by Memory, Dual Window Monitor screen, Interupt driven time out and Hangup protection, Also Keeps a User log of callers Names and Activity on disk so a printer isnt needed and its easily customised and expanded.

MAXCOM can be used with the 2050 modem or with the Z-SIO RS232 Interface and a 300/1200 baud modem. (a version for the Aerco RS232 is in the works).

PRICE: $24.95

LARKEN DISK EDITOR

This program lets you examine or change any byte on a LKdos disk. It is useful in repairing damaged or glitched disks, changing the disk Name or head speed etc. You can also examine the track map and disk info headers.

The documentation for the disk editor also includes information on how to access LKdos from Machine Code and information on the structure of the disk catalog and data blocks.

PRICE: $15.00

SEQUENTIAL/RANDOM Access FILES

This utility is a ram resident extension to the LKdos operating system and lets you create, read from, or write to very large data files kept on disk. It uses the commands OPEN#, CLOSE#, PRINT#, INPUT#, and INKEY$# to access the files from BASIC. There is also a high speed search command for finding a text string within the data file. Using the TAB command, you can randomly access directly any record in a file.

The utility is easily used within your basic programs and has a lot of uses, from storing large data files, creating large spell checker dictionarys or even converting you basic programs to an ascii text file .

This utility lets you take the limits off of the data capacity of the 2068 .Files on disk can be huge !

A simple data base program is included on the disk to demonstrate the utility.

PRICE: $ 15.00

Coming Soon from Larken Electronics

** The Larken Desk Top Publisher ** Produce high quality printing (like this page) with an epson compatible printer and LKdos. The entire Hi-res graphic page is kept on the disk and scrolling though the page is as easy as clicking the joystick or mouse on the icon type menu. Text and Graphic editing with multiple sized fonts will also be featured.

** Spell Checker for Tasword and M-Script ** This utility program will let you have your wordprocessor text files be checked by a 350K dictionary.

All Prices $US - Specify type of drive when ordering.

LARKEN ELECTRONICS RR#2 NAVAN, ONTARIO CANADA, K4B-1H9
(613)-835-2680

SUBSCRIPTION RENEWAL

Please tear out and send in for your subscription renewal for 1991.

This is the LAST ISSUE of the 1990 subscription year.

UPDATE Subscription Renewal form and User Data Base.

1. Please enclose $18.00 for renewing subscription for year 1991.
2. Mail to: Update Magazine, P.O. Box 1095, Peru, IN 46970.
3. Your Computer:_____QL _____TS-2068 _____Z88
4. If TS-2068 please check type of DOS: ____LKDOS ____Oliger ____FD-68 ____TOS ____SPDOS
5. If QL please check: ____Trump ____Sandy ____CST intfc. ____Unexpanded QL _____Addl Ram
6. Please check type of disk drive: ____5 1/4" ____3 1/2" _____SS ____DS ____360K 40 track ____720K 80 track
7. Brand and model printer_____
8. Please print your name and mailing address or affix a mailing label.

_____

_____

_____

_____

$18.00 a year

PLEASE PULL OUT AND SEND IN

WITH YOUR SUBSCRIPTION

RENEWAL

9. Update welcomes your constructive comments.

_____

_____

_____

Please send your renewal to
UPDATE MAGAZINE, P.O. BOX 1095, PERU, IN 46970
Publishers- Frank and Carol Davis

The Sinclair Desktop Publishing Journal's TS2068/Spectrum.....

DESKTOP PUBLISHING SURVEY

	PIXEL PRINT 4.X	PIXEL PRINT 5.0	PRINT FACTORY 1ST ED/PRESS	WORD-MASTER FULL PKG.
Machine or OS	2068	2068	2068/Spec.	Spec.
Disk I/O	By User	By User	By User	By Dealer
Printer I/O	By User	By User	w/in Program	By Dealer
Aerco Compatible	✓	✓	✓	✓
64 Column Display	NO	NO	Windowed	✓
What You See is What You Get	✓	✓	✓	NO
Works w/Entire 8 1/2 x 11" Page	Half Page at a Time	✓	✓	✓
Loadable Fonts	✓	✓	✓	✓*
8 Point	✓	✓	✓	✓*
16 Point	✓	✓	✓	✓*
Loads Zebra Icons	✓	✓	Translated	NO
Loads SCREEN$s	✓	✓	PRESS Only	✓
Resizes SCREEN$s	NO	NO	PRESS Only	✓
Word Processor/ Text Editing	Via Tasword	Via Tasword and MScript	NO	Via Word-Master
SCREEN$ Editor	NO	Overlay	Companion Prg	Extention
Page Formator	NO	✓	NO	✓
Block Functions: Cut, Paste, Move, Delete	✓	✓	NO	✓
Memory Usage	Almost All some left for I/O	All, plus a 32K extent, or RAM Disk	Almost all, some left for I/O	All, due to memory management

* WORD-MASTER COMES WITH IT'S OWN FONTS AND FONT PACKAGES. IT WILL NOT LOAD THE STANDARD 8 POINT FONTS FOUND IN PIXEL PRINT OR THE PRINT FACTORY.

ADDRESSES:

PIXEL PRINT 5.0
LEMKE SOFTWARE DESIGN
2114 WHITE OAK
WICHITA, KS 67207

THE PRINT FACTORY
BYTE POWER MAGAZINE
1748 MEADOWVIEW AVE
PICKERING, ONT L1V 3G8
CANADA

WORD-MASTER
C/O JACK DOHANY
390 RUTHERFORD
REDWOOD CITY, CA 94061

Interested in Desktop Publishing?

Would you like more information about the fascinating world of Timex and Sinclair Desktop Publishing?

The Sinclair Desktop Publishing Journal IS A QUARTERLY NEWSLETTER PUBLISHED AND DISTRIBUTED THROUGH **UPDATE Computer Systems** MAGAZINE. ALL AND ANY DESKTOP PUBLISHING, WORD PROCESSING OR GRAPHICS PROGRAM FOR THE TS2068 AND SINCLAIR SPECTRUM COMPUTERS ARE DISCUSSED, REVIEWED AND EXAMPLIFIED WITH IN THE JOURNAL.

FOR MORE INFORMATION ON **The Sinclair Desktop Publishing Journal and UPDATE Computer Systems** WRITE TO:

UPDATE Computer Systems
P.O. Box 1095
Peru, IN 46970

And PLEASE enclose a Self Addressed Stamped Envelope for your response. Thank you.

Bill Jones
1317 Stratford Ave.
Panama City, FL 32404
904 871 4513

Dear Friend,

'Be Seein You

I dont mean Good-bye, for I'll be seeing you. My hobby is Sinclair Computer Programming. I've had Apples, Commodores, a Coloco, an Atari, and an IBM AT 286. I've always returned to the Sinclairs, I guess because of "perestroka", or the openness of the Sinclairs to the user and programmer. It's just more fun to work with Sinclairs. Now that Frank Davis will be publishing Update Magazine, I'll be free to devote more time to programming both the TS-2068 and the QL. Of course there's no money to be made, but that aspect does not matter. It's the satisfaction of creating and sharing with the user group that keeps me motovated.

I've made many friends during the three years of publishing Update magazine. Money cannot buy friends, which is the most treasured of assets. And I'd like to keep in touch. Two friends are in trouble and I'd like to encourage all to drop cards. Bill Pederson (WIDJUP) suffered a stroke that left him physically impaired. Bill has contributed greatly to Sinclair Computing. Drop him a GET WELL card at: Bill Pederson, 1120 Merrifield S.E., Grand Rapids, MI 49507. Then, Don Lamen, a long time contributor of Sinclair Programs and a main-stay of the SINCUS Club, is recuperating from a tripple by-pass operation. I'm sure that Don would appreciate some encouraging cards: Don Lamen, RR-3 Box 3404, Windsor, NY 13865.

I will continue to submit programming articles to Update Magazine in hopes that it will help Frank Davis to keep the depth of content that has been the tradition built over the past three years. But there has been an even more important tradition established which I hope you have appreciated, and I hope will be continued. That is a sense of un-selfish giving and sharing amongst the Sinclair User Group in North America. Update Magazine has sponsored a spirit of cooperation between the users, the Clubs, the programmers, and the suppliers. We are all bound together by a keen interest in hobby computing and the need for cooperation to further our own hobby interests. Besides, it is enjoyable to communicate and to share.

Now I'll end this epistle with a laugh. Pere' wrote from the hinterlands saying, "I regret your demise--", followed by some appreciated laudable comments. Well, Pere', let us know your secret of communicating with the departed. We all need that information! I really do appreciate all of the friendly letters received and hope to keep in touch with all. Be ah seein you in the pages of Update Magazine. Good Luck, Good Health, and God Bless!

Sincerely,

Bill Jones

Bill Jones

REPORT ON THE SINCLAIR EXPO IN MILWAUKEE

As part of the team assuming the editing and publishing of UPDATE, I thought I would take a few minutes to do two things. One, I want to tell you about our recent trip to attend the Sinclair Expo in Milwaukee, and second , give you a glimpse of what is in the future for this magazine.

Carol and I had a rather hectic and traffic filled drive past Chicago to get to the Sinclair EXPO, put on by SMUG. We got there and with the help of Bill Angel got our station wagon unloaded. After hurriedly changing clothes we barely got to the banquet at the starting time. We were pleased to see so many familiar faces, after all many of these people have made it to all of the Sinclair Fests in the past, from Cincinnati, to Indy, to Cleveland, to Wash. D.C. , and to Florida and the Great Northwest Shows! The people at SMUG were so right to call this a family reunion. Many thanks to Neal, Bill, Dick and the rest of this fine users group. Sinclair users make good friends.

The banquet offered variety to eat and was done in a timely manner by the hotel. And...of course we had pleasant company for the banquet (actually the entire show, I met no Sinclair user that I feel the need to complain about).

Carol and I picked up some really nice t-shirts, momentos of the Sinclair Expo, which was truly a "family reunion". There were seminars on Forth, Robotics, Modems, Computer Intelligence, Quill-how to use it, Archive Program DBADDRES and Desk Top Publishing. The one on desk top publishing was done by Mike Felerski and Frank Davis covering the TS2068 and the QL.

There were some vendors that apparently felt they had no need to attend the show for some reason or other. I wish they had been there, but those who did make it had a profitable time. Among those who were there were as follows: EMSOFT (Peter Hale of Boston), MECHANICAL AFFINITY (Frank Davis and Paul Holmgren of Indiana), WOOD AND WIND COMPUTING (Bill Cable of New Hampshire), ZEBRA SYSTEMS INC. (manned by members of SMUG), Mike Hammer doing Sinclair repairs and offering refurbished TS2068s, and the following user groups...CATS, CATUG, DMA, GCSUG, ISTUG, SMUG, SNUG. There were a number of people doing a flea market of used hardware and software. Eric Johnson, who would have been offering refurbished TS1500s and TS2068s and repair parts, had major auto trouble part way there and did not show. Paul Holmgren, the QUANTA librarian was selling library disks. Al Feng of Platypus Software was demoing his software (which is quite good and worth the price) and Bob Swoger was offering Logicall, a must for Larken users. There is talk of doing the next show in Chicago. If that is true, perhaps ISTUG and SMUG can help out as co-sponsors and make it a really big show.

For those interested in finding out more about QUANTA, QL Users and Tinkerers Association, I suggest you contact their North American Librarian. He can provide you with details about their extremely large and useful library and how you can become a member of QUANTA. Send a SASE to Paul Holmgren, 5231 Wilton Wood CT., Indianapolis, IN 46254. He can also be reached most evenings (at a reasonable hour) by phoning 317-291-6002. In addition to the library they put out a monthly magazine, from England, of about 30 to 40 pages. And it is all QL. Back issues are available by mail from the U.K. and at last report the price for membership was about 17 English pounds.

What can we look forward to in the next year or two for UPDATE? I for one have been proud of what has been done by Bill Jones, so I am not going to go into phrases like "I hope to improve" or " It will be hard to follow in the footsteps of". Instead I want to say that Carol and I are prepared to deliver UPDATE to you in a timely manner. It will remain at least the same size, with a few format changes as we experiment with various desktop publishing programs. Barring death or severe physical impairment...this is what we want to bring to your mailbox 4 times a year. With the help of many this will be done.

While we were at the Sinclair Expo we got a chance to speak with Peter Hale, Bill Cable and Al Feng. They have all written for the magazine in the past and have stated that they will continue to support it. They are all writers and programmers that I was glad to have support from. I have also heard by mail from Bob Hartung, who will also continue to write for us, and even sent to us updated copies of his programs. We have already received articles and programs from other users who have written for UPDATE before and from some new ones. It does not look like we will be hard up for good material. Bill Jones has told me he still will contribute material to the magazine on the TS2068. That will, of course, be welcome. With the help of these people we will continue the issue disk program and give our North American programmers a chance to be heard from and make a few dollars at the same time. I am a great believer in exchange.

What will we be adding to the magazine that is new? A new section dealing with Desk Top Publishing. This will be primarily handled by Mike Felerski of Bottlecap Software, of Cincinnati, Ohio. You will be pleased with his work. Also helping out in this area we have had the offer of assistance from Bill Ferrebee of Mountaineer Software. Bill has also started doing a column for Vulcan Computer Monthly on Sinclair Computers. I will be reviewing some European hardware and software.

Carol and I have already picked up some new subscribers to the magazine and have started a vigorous campaign for more. Now we need to have all those who have not done so renew. A growing readership is a good guarantee that a magazine will be there waiting for you in the future. As long as the magazine pays for itself, it will be there. I am looking forward to being of service to you. I can often be reached evenings at 317-473-8031. When not there, I will have one of those nasty answering machines there to take a message. Please don't ask me to return calls at my expense, but do leave a number and an address.

Ruminations From The QL Editor

This will be the last rumination for me. I will continue the Cable Column but give up the QL Editor position for UPDATE when Frank Davis takes over as UPDATE publisher next issue. I was glad to give a hand to Bill Jones for the past year but I feel that someone else could do a better job. I am a bit isolated here in the hills of New Hampshire and my specialty with the QL is primarily database programming with ARCHIVE. The QL is a complex computer that has many facets. It will be beneficial for all if someone else with a different background takes over as QL editor. Although Frank may not continue having a QL editor for UPDATE, I hope he does. It is an important focus point because there is not much QL specific coverage left and having someone sound off on the QL scene keeps the vision alive. I have great confidence in Frank Davis and will help any way I can with the new UPDATE.

Peter Hale of EMSOFT and I made a quick trip by car to the Sinclair Exposition in Milwaukee on June 1-3. As usual Peter was very organized and I was a bit disorganized. We both had vendor tables. We both did just a little better than at the CAT's Fest last year although the crowd was a little smaller. It was estimated that 120 people attended. Being tied to my booth most of the time I didn't get to attend any of the seminars.The subjects covered included FORTH programming, MODEMS, Desk Top Publishing (Frank Davis), Computer Intelligence, QUILL and my DBAddres program. Bill Heberlein, SMUG editor, gave the seminar on DBAddress. I put many powerful features into my program and I was impressed at what he was doing with it. I got to meet old friends and make some new ones. Frank Davis and his wife were there and I got to talk with them about UPDATE. I also met Paul Holgren who is the new Quanta Librarian for North America and head of SNUG. I was impressed by his enthusiasm. Frank and Paul will be putting a lot of energy in keeping Sinclair computers viable. Please support them financially by subscribing to UPDATE and SNUG. I had a nice visit with Al Feng, a fellow QL programmer. We are offering his just released program ,QLUMSI, as an Issue Disk in this issue. Peter and I were the only QL software vendors. Paul Holmgren and Frank Davis had a large booth and were selling used hardware and software. Much of it part of Tom Bent's collection. There was a large SNUG (Sinclair North America User Group) on Saturday night. There was lots of energy and the leadership seems dedicated and able to really get the ball rolling with this organization. QL and 2068 people seemed to get along better than ever. Maybe I was imagining things but there seems to be a large increase in QL owners who were using ARCHIVE and liking it.

This issue of update has the SECOND installment of Timothy Stoddard's TRUMP CARD MANUAL. He does a great job at explaining the directory structure features of TK2. Peter Hale has some tips for the home office. I begin a new series on ARCHIVE by tackling database concepts. Al Feng lightens us up with a backdoor approach in his ARCHIVE: The FINAL FRONTIER. David McCall's last Window Defining article and Peter Hale's 2nd TEXT87 article will appear in a later issue.

ARCHIVE: The FINAL FRONTIER
by Al Feng

All right. I'm probably the last QL user to work with the ARCHIVE/QLDB data base program. The "problem" with ARCHIVE is that the HELP files really don't "help" you with the program's syntax. Until you know, you don't know. And, you really can't "fake it" the way you can with the other three Psion business programs. ARCHIVE's syntax is very specific.

CREATE, END CREATE. Are we playing God with the Universe? If you've gotten far enough to see the blank, central window (i.e., the "void") with a red DOS-like prompt in a lower COMMAND WINDOW, you already know that you can't just begin typing in any of the "words" seen in the PROMPTs window at the top of the screen at the COMMAND LINE. You have to know exactly what-and-how you want to define ALL the FIELDs before you start to work, otherwise you're only going to have trouble later on when the time comes to fill in your RECORDs. Perfection counts.

The following, alternate method for setting up an ARCHIVE data base should give all novices a better (i.e., EASIER) idea of what is going on. Those who consider themselves to be "power users" may wish to take note as they may not be aware of know this flexible method for CREATing a data base's structure.

First, forget everything you've heard about a data base being like a set of file cards. You just cannot scribble-and-erase here, there, or anywhere. We may be better prepared if we think of a data base's format as being akin to a governmental form (i.e., "Rigid").

Unlike a word processor where you can twiddle-and-tweak with ideas for paragraphs, all the FIELDs of a data base must be pre-DEFined before you begin putting in any of the RECORDs.

Power users too familiar with setting up their data bases are probably saying: "Big deal!" But, a computer is supposed to make things easier. Why should one first need to consolidate (m)any randomly jotted notes onto a single sheet of paper prior to beginning to ensure that everything that must be included is? It would be better if any data base could be constructed on-the-fly.

Now, consider your data base as being a matrix. Uh oh! You may recall that calculus was a prerequisite for the course on matrices. Don't worry. We've all seen examples of matrices since we were in grade school. The "multiplication table" and the "mileage between cities" chart(s) are common examples of matrices. In a meaningful matrix, the data in the middle of the grid can be correlated to the labels of the 'X' or 'Y' axis. For the data base, the 'X axis' represents the FIELDs, and the 'Y axis' represents the RECORDs.

Because of the structure of the Psion programs, you can first set up your data base in ABACUS by constructing a valid matrix and then IMPORTing it

into ARCHIVE. For some of you, a light should have gone on in your head. For the rest, read on.

Just LOAD the ABACUS/QLSS spread sheet program into your QL and start typing in the label names for items you think you may want to track in your data base. Use the following example as a model, beginning in CELL 'A1'.

	A	B	C	D
1	Star	RtAccnsn	declintn	magnitude
2	Achernar	1h 35m	S 57 35	1
3	Acrux	12h 22m	S 62 42	1
4	Agena			

All the CELLs are "text" with the exception of 'D2' & 'D3' which are "numerical" values.

While you can change the names in the columns after you've input all the FIELD label names, some foresight helps (corrections can be made by adding columns and inserting the forgotten, or misplaced, label in the correct column). The better organized you are, the quicker the entire process will be.

Depending on the RECORDs, you may wish to do the bulk of input from the matrix. This is especially applicable if you are still using microdrives for storage as this will circumvent repeated media access during initial data INPUT.

After setting up the matrix's framework, you may decide that you would rather sub-divide "declination" into two parts:

	A	B	C	D	D	E
1	Star	RtAccnsn	declH	declD		magnitude
2	Achernar	1h 35m	S	57	35	1
3	Acrux	12h 22m	S	62	42	1
4	Agena	13h 58m	S	60	2	1
5	Aldebaran	4h 31m	N	16	22	1
6	Altair	19h 47m	N	8	40	1
7	Antares	16h 25m	S	26	16	1
8	Arcturus	14h 12m	N	19	32	0
9	Betelgeuse	5h 51m	N	7	23	1
10	Canopus	6h 22m	S	52	39	-1
11	Capella	5h 11m	N	45	55	1
12	Deneb	20h 39m	N	45	1	1
13	Fomalhaut	22h 53m	S	29	59	1
14	Pollux	7h 30m	N	32	2	1
15	Procyon	7h 37m	N	5	20	0
16	Regulus	10h 4m	N	12	18	1
17	Rigel	5h 11m	S	8	16	0
18	Sirius	6h 42m	S	16	37	-1.6
19	Spica	13h 21m	S	10	47	1
20	Vega	18h 34m	N	38	43	0

This was done by using:

```
<F3>        (g)rid
            (i)nsert
            (c)olumn, at C
```

to insert a 'column C'. The labels, as well as data, in columns 'C' & 'D' are then amended to suit your current specifications.

All right. You've gotten this much done. Now what? Simply follow the standard EXPORT sequence:

```
<F3>        (F)iles
            (E)xport
            <ENTER> (for data)
range (<ENTER> for default)
            (c)olumns
to ___ (e.g., 'STARdata').
```

Note: the default '_exp' extension is automatically added to the EXPORTed file by the ABACUS program.

So, now that you've created an IMPORT/EXPORT file you'll want to switch to ARCHIVE. But, before you QUIT ABACUS, SAVE the file as an '_aba' file. Go through the SAVE sequence and call the file 'STARdata(_aba)'.

Now, QUIT the ABACUS program and LOAD ARCHIVE. Then type:

```
Import "flp2_STARdata_exp" to "STARdata"
                    <ENTER>
```

The QL will CREATE the file on your default medium. When the cursor returns, type:

```
display     <ENTER>
```

and [ta-DAH!], you should see your newly created data base file (see below):

```
Logical name : main
Star$        : Achernar
RtAccnsn$    : 1 h 35 m.
declH$       : S
declD$       : 57 35
magnitude    : 1
```

Now type: next <ENTER>

and, you should see the second record (i.e., "Acrux"). If you type:

```
last        <ENTER>
```

You should see the last record which you typed in (in this example: "Vega"). Now, if you type:

```
DIR "mdv2_"   <ENTER>
```

you will get a DIRectory of your data disk. You should see files named:

```
'STARdata_exp'
'STARdata_aba'
'STARdata_dbf'
```

If you ever need to add more FIELDs to your data base (eg., "comment1$"), either go back to the original 'nnnn_aba', or EXPORT the current 'nnnn_dbf' to 'nnnn_imp' (where 'nnnn_ext' is the name of the appropriate file).

From here on, using ARCHIVE is pretty straight forward. To add new data, type "alter" to enter the correct mode for 'ALTERing' (amending) the RECORDs in the data base.

Remember to CLOSE the RECORD file before turning off your QL (ARCHIVE automatically CLOSEs the current file when you QUIT the program).

HAPPY TRAILS, AND COMPUTING, TO YOU. . .

The C.A.T.S. Newsletter

The following three articles are re-printed from the CATS NEWSLETTER. To Join CATS and receive this nice newsletter, send $15.00 US Check or MO to: CATS Newsletter, PO Box 11017, Takoma Park, MD 20913. There are other benefits. _BJ-

The C Column

By Tim Swenson

As promised last month, I'll talk about how C accesses files.

The simplest way to discuss how C handles files is to take a look at a C program that uses it. Look at program number 5. This program asks the user for a file name. It then opens the file, reads in a character and prints it out the screen. This goes on for the entire file.

The program declares a character array of size 20 to be hold the file name. It then declares the integer fd to hold the file ID. When a file is opened, C assigns a number to it. This number is returned from the fopen command to the variable fd.

Note the syntax of the fopen command. The file name is sent into the function along with a string that specifies if the file is for reading or writing. If we wanted to open a file to write to, we would have used "w" instead of "r".

The next couple of lines checks to see if the file was opened sucessfully. If the file was not opened (there was some error like a bad file name) fd will be returned as NULL (a predefined constant).

Instead of using getch the function getc is used. Where getch defaults to the keyboard, getc needs a file ID to know where to get it's input. In our case we use fd.

We exit the while loop when the program reaches the end of the file (EOF). After the program exits the while loop, we use the fclose function to close the file. We must tell fclose which file to close. If we do not close the file, we may loose the file.

```
/* prog 5 */
#include <stdio_h>

  main() {
    char  file[20];
    int   c, fd;

    printf("Enter file: ");
    gets(file);

    fd=fopen(file,"r");
    if (fd == NULL) {
       printf("File open Error");
       abort(1);
    }
    while ((c=getc(fd))!=EOF) {
       putchar(c);
    }
    fclose(fd);
  }
```

QL Small-C Companion

By Tim Swenson

I have been using the Small-C compiler on the QL for some time now. I have found the documentation severly lacking. Knowing that other users are probably having the same problem, I decided to document the solutions to some of the problems I ran across. I also looked into other versions of Small-C for other computers. A copy of this document is available from the QL library. The library has a hard copy of the document and a Quill file. The hard copy was printed out using a laser printer and looks good. I've utilized multiple fonts to enchance it's looks.

When looking at the CP/M version of Small-C, I found that the QL version did not come with all of the functions found in most Small-C compilers. I typed in the functions that I could convert and created a new library. I tested the library generator with this new library. The library is also available from the QL library. The functions are documented in the Companion documentation.

I'm all for computer dating,
But I wouldn't want one to marry my sister.

Additions to the QL Library

By Tim Swenson

Over the years I have written a number of usefull programs for the QL. Some of them I have used in articles for numerous newsletters. I have donated them to the club's QL library. I thought I would let users know about the more interesting and usefull.

Lister - This program reads in a SuperBasic, Pascal, or C program (actually any text file will work). It prints the program to the printer with page breaks and the file name at the top of each page. It even has options for NLQ, bold, etc, type print. It is set up for an Epson compatable printer.

QLcryptII - This programs reads in any file and encrypts it. The user enters a password that controls how the file is encrypted and decrypted. The output file may look like garbage, but when decrypted the original file is returned. Even works on executable files. It is very secure.

Calc - This is a small calculator program that works in decimal, hexidecimal, and binary. Lets you add any of the three bases and convert between them. Easy way to convert between bases. If compiled, it can be a nice pop up type calculator.

Maillist - A menu driven Archive program for maintaining a mailing list. User can add, edit, delete, and print entries. Has a nice data entry screen.

All of the programs have a Quill file that describes the programs fully. All of the documentation is 1 to 5 pages long. Maillist has longer documentation. Be aware that the software has my old address in California. I've been too lazy to update them.

See Tom Robbins for getting copies of the QL Public Domain library.

More On Benchmarks

By Herb Schaaf

Duane Parker had the time down to 23 seconds using Forth in his January classic on benchmarking with Pascal, QL-SuperBasic, etc.

The Small-C QL combo does it in 16 seconds(14 with Lightning installed). An IBM PS/2-30 and the same (well, nearly the same) source code using Turbo-C took 6 seconds. Mix-C & IBM did it in 5. A Toshiba T5200/100 & Turbo-C took 2 seconds.

When ported over to the VAX-3650 and timed it came back as 0.0 seconds, so I extended the run down to 1; then it took 0.2 seconds. Times to go down to 1 are:

Turbo-C & T5200	22 secs.
Mix-C & IBM	42
Turbo-C & IBM	55
Small-C & QL	155
with QL &	
Conqueror Mix-C	1390
Turbo-C	2070

Now as to costs:

```
Hardware----
  QL systems $$ hundreds
  IBM systems $ thousands
  T5200 $ ten thousand
  VAX $$hundreds of Thousands

Software----
  Small-C $ 1 (cost of disk)
  Mix-C  $ 20 (BEST BUY!!)
  Turbo-C $ 100 (discount)
  UNIX C $ a lot more
```

```
/*  dpbench7_c      May 13 1990  10:30am
**  Small-C benchmark program by Duane H. Parker
**  Ref:  CATS Newsletter Jan 1990  pages 6-8.
*/

#include <stdio.h>      /* standard input/output header      */
#define MAX 32767           /*  limit of signed integers     */

main()
{
char c, string[20];     /* characters and/or strings         */
int start, stop, index, i, j; /* handy integer values        */
int testfactor, tfsq;       /* testfactor and its square     */
int secs, d1[2], d2[2];     /* used in timing                */
int minval, pos;          /* more integers                   */
minval = 0;
pos = (MAX/100);
testfactor = 181;         /* int(sqrt(MAX))                  */
tfsq = 32761;
    while( ((minval>0) + (minval<MAX) + (index == 1)) != 3)
    {
    stop = 0;             /* initializing for clean start    */
    index = 1;
    start = 1;
    cls(2);
    printf("This is a Small-C program\n");
    printf("of a benchmark test by DH Parker\n");
    printf("see CATS Newsletter of January 1990\n");
    printf("\nStop calculation at what value less than %d ?\n",MAX);
        gets(string);           /* when to stop ? */
        j = strlen(string);
 strncpy(c,string,1);
        start = atoi(c);      /* makes string into integer  */
 for(i=0 ; i<j; i++)    /* checks for correct input    */
            {
            c = string[i];
     stop = isdigit(string[i]); /* is it a digit ?    */
            index = stop * index;
     }
 minval = atoi(string);
 if( j>5 )                  /* are there too many digits ?  */
 {
 index = 0;
     }
 if((j == 5) && (index == 1) && (start>3) )
     {
              index = 0;
            }
 if((minval>=MAX) || (minval<1) || (index == 0))
        {
        printf("\n\nI N V A L I D  ! !"); /* error trapping */
        printf("\n\nany key to try again\n q to quit");
        c =  getchar();
        switch (c)
     {
     case 'q' :  xcit();  /* 'fancy' exit    */
     }
 }
    }
    cls(stdout);      /* clear screen */
    printf("Ready to find primes from 32767 to %d\n",minval);
    printf("any key to begin\n"); /* inform and wait for */
    getchar();                    /* trigger   */
    {
                  /* start clock     */
    *d1 = date();     /* starting time  */

    for(index = MAX; index >= minval ; index -= 2)
        {
 while ((pos*100)> index) /* make newline to    */
            {                       /* group by 100's     */
     --pos;
     printf("\n");
            }
         while( index < tfsq )    /* reduce tfsq and */
     {                      /* testfactor when */

    for( j = 0; j < 4 ; j++)    /* possible  */
       {
       tfsq -= testfactor;
       tfsq++;
       }
            testfactor -= 2;
    }
    if(testfactor == 1 && index != 1 )
    {                       /* small primes */
    printf("%6d",index);
    }
       for( j=3; ( j <= testfactor ); j += 2 )
       {
       if((index % j)==0) /* not prime       */
         break;
       if(j >= testfactor) /* is prime       */
   printf("%6d",index);
       }
       }
 if(minval < 3)      /* to catch 2,     */
  {                  /* the even prime */
  printf("%6d",2);
       }
    }
    *d2 = date();             /* checks time of finish    */
    secs = d2[1] -d1[1];    /* simple difference          */
    printf("\nRuntime = %d seconds in Small-C\n",secs);
    printf("\nany key to exit");
    getchar();
    xcit();
}
xcit()          /* a fancy (?) way to exit     */
{               /* a function is like a defined procedure */
   mode(8);
   cls(2);
   at(stdout,1,9);
   ink(stdout,1);
   printf("It's been good to");
   csize(stdout,3,1);
   at(stdout,3,11);
   ink(stdout,2);
   flash(stdout,1);
   printf("'C'");
   flash(stdout,0);
   csize(stdout,0,0);
   at(stdout,12,15);
   ink(stdout,4);
   printf("you");
   at (stdout,17,5);
   flash(stdout,1);
   ink(stdout,6);
   printf("any key to exit . . ._");
   getchar();
   mode(4);
   flash(stdout,0);
   exit(2);
}
/* The end of the small-c source code for dpbench7_c May 13, 1990 11am  */
```

Dealing With The BBS

By Mark Fisher

GETTING ON THE AIR

What you'll need:

This will take your computer, a terminal program, and a MODEM. For the 2068, the most common setup is the 2050 (Anchor) MODEM, along with SmartTerm II. For the QL, the horizon is wider - any "external" MODEM will work at 1200 BAUD. For terminal programs, again, the choice is wider, with QL Term, QLink, QL-52 being the main contenders. (If you want 300 BAUD compatibility, you'll have to do more work; i.e. the Modaptor and QCode software)

Now you'll have to connect the pieces:

2068 --- Plug the 2050 into the expansion bus in the back of the computer, and plug its adapter into the wall.

QL --- Get a connecting cable to bridge the gap between the DB9 connector on the QL and th e DB25 on the MODEM.

Using the software:

Load the software. If you have the SmartTerm II cartridge, that's already happened by the time you see the first screen. QL programs are nearly that simple.

Terminal programs must make a distinction between input that controls the operation of the program, and input that is intended to be sent out the port of your computer. The usual way to describe these choices is the term "mode". "Command mode" controls the operation of the terminal program itself, while "terminal mode" sends your input out the serial port. When you first get on, make sure your terminal program is saving everything to its buffer. This will let you review what happened after you disconnect. You may want to save the (I)nformation files as permanent references.

Depending on your particular

setup, you may dial the BBS via your phone, or through the auto-dial option of your terminal program. Many external MODEM's can be dialed by typing ATDP5880579 from terminal mode.

When you connect with the BBS, you will see a prompt to set the date (this is an addition I've made to get around the lack of a clock on our machine). If you get something wrong, you'll be prompted again. Don't worry; the computer has plenty of time, and you can go around as much as you like. Next, you will be asked for your name. If I've guessed the spelling of your name correctly, you'll be asked for the password. The default password is VERNON, but you should change that as soon as possible. (To encourage you to do so, I've also set your message reading width to 20. You can change both preset values from the (U)pdate record menu choice.)

EXPLORING

Once you get past this, you'll be IN the CATS BBS. There are no time limits here. The BBS willI not penalize you for making a mistake. The worst thing that can happen is that you'll have to 'unplug' your computer. The BBS will detect this, and re-set itself for the next caller.

If you think of the BBS as a house, there are three main wings. They are the Main menu, the Mail menu, and t he files system. Entry into each is heralded by a specific phrase. You can think of each phrase as being carved over the entrance to the wing (one of the weaknesses of this metaphor is that the actual location of the phrase will be in the bottom left corner of your screen, not top center).

The Main Menu Wing:

"Over the Door", you'll see:

PBBS Menu, Commands are ....

This is the general purpose section of the BBS. Here, you'll find out about general (I)nformation, recent uploads, other users, and your personal customization values. You can also do some basic chores, such as toggling the full display of the menu or the [more] prompt, (Y)akking with me, (J)umping to CPM, switching to the (M)ail menu, or saying (G)oodbye.

The Mail Menu Wing:

Over the door of this wing is :

Folder: General
MAIL Menu: Commands are ....

This wing offers ways to delve through the message base. The most basic are (E)nter a message and (R)ead one. In addition, you can scan the headers, either for a given subject, or a list of all subjects. As the message base grows, it will get harder to scan everything. To deal with this, the message base is divided into 9 FOLDERS. By switching between the folders, you can view all "FOR SALE" messages, all 2068 messages, etc.

There are sub-menus that appear as you go through the more complex operatons: for (E)nter, there is a menu that appears when you press <ENTER> twice. It offers you the option to (L)ist, (E)dit, or C)ontinue entering the message, or (S)ave the message as public or (P)rivate.

The CPM Attic:

Over the door is:

B3>

The last wing of the "PBBS house" is the files area. The rules here are different. You can think of it as the "attic" of the system, storing all those things you MIGHT use. There are no menus here - instead there are a number of discrete commands you can invoke. CPM interprets the word following the B3> prompt as the file name of a machine code program. There is room after the name for further data - see the examples below. When you are adding that extra data, separate it from the command with ONE SPACE. The most basic are:

"BYE" Hangs up the phone and resets the BBS f or the next caller.

"PBBS" Returns to the Bulletin board program, so you can leave a message about what you found up in the CP M attic.

"D" This is the Directory command - producing an alpha sorted list of files.

"USER xx" This changes the user area (C PM's sub-directory method)

"TYPE-DN (your filename here)" This will type to the screen the contents of the file you specify. If you have cleared your capture buffer and opened it, you will end up with a copy of the file you requested. If you use TYPE-DN on COM or DOC files you will see some interesting screen art, and possibly hang up your computer, but it won't hurt the BBS.

"TYPE-UP (your filename here)" This program will create a new file on the BBS computer, and store whatever you send to it there. You can use your terminal program's ASCII file transfer routines to send your ASCII files to the BBS. It will be closed when you send a control-Z, and after a few seconds, you'll see the B3> again.

"XMODEM (your filename here)" The is the big gun of the files area. This program can work with the Xmodem part of your terminal program to transport ANY file to or from the BBS. Typing XMODEM alone will give you a short list of your options. To use it, type "XMODEM (your filename here)" at the B3> prompt. You will then enter command mode of your terminal program, and start up its xmodem file transfer procedure. The two programs will work together to shift the file between the machines. XMODEM knows that you will take a while to get your end going - there's a delay built in. If you take too long it will abort the transfer, but you can always try again. The transfer may also be aborted if there are too many transmission errors - often a second try will work fine.

"DEMO CPM" To see several of these commands in action, type DEMO CPM. There is an automated demo of the basics (I've put in a place for your input - don't just sit back!).

T R Y U S O U T !

These articles from the CATS Newsletter were photo reduced in order to get as much of a sample of the depth of articles given in each NL issue.

The Newsletter is printed in a nice size readable print.

All QL and TS-2068 users would do themselves a favor by subscribing to the CATS Newsletter.

THE Sinclair Z88 Small But Powerful

BILL JONES HAS ASKED ME TO WRITE A COLUMN DEMONSTRATING HOW EASY IT IS TO USE THE Z-88'S FABULOUS WORD PROCESSOR TO MAKE MULTIPLE COLUMNS AND TO MIX TEXT AND SPREADSHEET SLOTS IN THE SAME LETTER OR EVEN ON THE SAME LINE. IT IS EASY, EASY, EASY!!

FIRST, I CHOSE A TYPE WIDTH, ELITE, THAT LETS ME MATCH UPDATES 2 COLUMN PRINTING AND THEN MODIFIED THE COLUMN WIDTHS IN PIPEDREAM TO MATCH. SECOND, I DIVIDE EACH COLUMN WIDTH TO ALLOW MIXING TEXT AND SPREADSHEETS LATER.

THEN, IN THE COLUMN BETWEEN THE TWO COLUMNS, A MARK, OR LETTER IS PLACED TO ACT AS A WALL FOR JUSTIFYING THE COLUMN TEXT. COPY THIS WALL AT THE END OF THE SECOND COLUMN. AFTER THE TEXT FOR BOTH COLUMNS IS DONE, AND YOU HAVE JUSTIFIED THE TEXT, YOU ERASE THE WALL 'MARKS'. THIS IS EASIER OF COURSE, IF ALL YOU ARE DOING IS JUSTIFYING 2 COLUMNS, NONE OF THIS EXTRA PLANNING IS REQUIRED. TO MAKE THE BOTTOM PART OF THE LETTER INTO A SPREADSHEET, YOU CHOOSE WHICH SLOTS TO MAKE EXPRESSION SLOTS AND THEN MIX TEXT AND/OR NUMBERS, AT WILL!!

NORMALLY THE MONTHS WOULD BE ON THE LEFT, BUT, SINCE THIS IS DESIGNED TO FIT IN THE SPACE FOR 'UPDATE', I MOVED THAT COLUMN, TO THE COLUMN BETWEEN THE 2 COLUMNS. I THEN MADE UP SOME NOS. FOR THE SPREADSHEET. THERE IS NO LIMIT TO THE WAYS YOU CAN MODIFY A PAGE TO SUIT YOUR NEEDS. YOU CAN EVEN *ITALICIZE* AND/OR UNDERLINE YOUR TEXT WORDS. YOU CAN ALSO DO H_2O X^4, lower case, AND double width as **shown here!!**

The column between the 2 columns and the wall on the right, were made in 5 seconds with the replicating command, and erased in 5 seconds when no longer required. This could also be done manually without being fancy. I can put the RENT from the spreadsheet inside my text here if i choose to;

RENT = $2200 by using the 'field' marks around the slot. If the rent were to change in the expression slot it would automatically change in my text above!! Any mistakes, changes in the text can be re-justified by using 'diamond R', at the beginning of each paragraph. There are so many word processing commands that usually there is more than just one way to accomplish your task. There is no training required for any novice to sit down and type an ordinary letter!

Just use it like a typewriter that thinks for you.

The slot commands for USABLE PROFIT slot J48, are B48+C48-SUM(D48H48).

This tells the computer to add the amount in Jan sales, and subtract the values in the slots D thru H, and print the answer in slot J48.

Replicating automatically takes care of the remaining months, as many as you have.

You can pull any text or number from any expression slot and have it print itself anywhere in your letter as long as you set your field spaces properly. FANTASTIC COMPUTER!!

DOMINO CUBES 355 W.39 ST
NEW YORK 10018-1401 N.Y.
MIKE FINK 212 971 5638

==========

SOFTWARE SALES	Z-88 SALES	RENT	ADV'G COSTS	MOS	WAGES	SALARY	INVEN-TORY	USABLE PROFIT
$250	$1200	$2200	$300	JAN	$0	$200	$2000	$-1250
$500	$2400	$2200	$900	FEB	$800	$500	$2400	$-1500
$1000	$6000	$2200	$1250	MAR	$1200	$1500	$2900	$850

COLUMN LABELS AND WIDTHS ARE SHOWN BELOW ONLY FOR CLARITY (A omitted)

.....B.....C.....D.....E...F.....G.....H.....I.....J

ED NOTE: *QL and TS-2068 Users: This is the best article that I've ever read about Computers in general and their best uses. Two Data Base softwares for the TS-2068 are given in this issue. This article will help to understand these and all other types of data that a computer can be used to manage. _BJ_*

CABLE COLUMN

By Bill Cable

ARCHIVE SERIES - PART 1 : DATABASE CONCEPTS

Starting with this UPDATE issue I am going to begin a multi-part series that will explore database concepts and uses with particular emphasis on the ARCHIVE Database System on the QL. This will include developing useful programs to illustrate techniques of ARCHIVE programming and database techniques. I firmly believe that ARCHIVE is one of the best database systems available on any computer and potentially the most useful program for the QL. An active group of ARCHIVE users/programmers will go a long way towards preventing the QL from becoming history.

I will try to keep each column somewhat self-contained so that a reader will be able to jump in at any point. I will not explain in detail how to use the ARCHIVE programming editor each time. Please refer to the April 89 Cable Column for my tutorial on using it. My other UPDATE articles on ARCHIVE will be handy references also. If you don't have these back issues of UPDATE you can order them from Bill Jones, 1317 Stratford Ave., Panama City, FL 32404 (($18.00 for Year Group of 4 issues, 1988, 1989, 1990)). These back issues of Update Magazine are chock full of goodies and well worth buying. I will also put all my ARCHIVE stuff together. You can order the collection directly from me for $10.00. Specify whether you want it in printed form or on DOC files on 5.25 DSQD (720k) or 3.5 DSQD (720k). I will also include some extra goodies on the disks.

The path and duration of this series depends to a large extent on my sense of what the readership needs and desires. So feel free to write and help give me some guidance. I also want to emphasize that this won't quite be a formal course in database theory. Although I have access to a few books on the subject for reference, the ideas expressed will be my own opinions and may not totally coincide with an academic treatment of the subject. In fact much of what I am going to say is ad lib for the purpose of stimulating interest and may strain the limits of credibility.

Many QL users have used the ARCHIVE Database System and liked it but the vast majority have not tried it or have abandoned it at an early stage. I think there are 3 stumbling blocks keeping people from using a database system such as ARCHIVE. First, many people don't know what a database is or how it can be used. I believe that all but the most casual computer user can have fun with and save a lot of time using computer databases. I will try to convince you of this in this series. The second stumbling block is that to use a powerful database system like ARCHIVE in its most basic and flexible form requires learning how to use its language. And ARCHIVE's language has a lot of commands and functions. This is both good and bad. Lots of commands means lots of power but it also means more to learn. Fortunately it is a straight forward language similar to but probably easier than SUPERBASIC. This series will explore the main features of the language. The third stumbling block is that it takes serious work to type in all the information in a typical database. Luckily we have one database, gazet_dbf, given to us to play with. We will also explore ways to capture data from others sources. It is sad but true that for most databases getting the data into the database is the hard part and there are no shortcuts.

I will begin at the beginning : what are databases and why do we use them.

1. What is a database?

My defintion of a database is: a collection of useful data that is given a coherent structure and kept in a convenient location so it can be used in a way that saves time and effort.

I think this is a pretty clear definition that doesn't use any obscure computer lingo. The terms 'useful data' and 'coherent structure' may be a little vague. The examples below will help illustrate what I mean by them. Data may be textual in form or numeric or both. The definition stresses the main points for using databases, namely to save time and effort when examining data. As you will see, I have chosen my words so that practically everything is a database when viewed with this definition in mind.

2. What are some common databases?

The alphabet should be considered our most primal database at least for written information. It consists of the 26 single characters (really 52 considering upper/lower case) we use to spell the words of our language. The coherent structure of the alphabet database is its order. Without that order we would have a hard time listing them or keeping track of them. And we wouldn't have a convenient way of ordering words either. Because we use this database so much we have memorized it so we don't really need to keep it anywhere except in our heads.

The number system is another primal database and is peculiar because it has infinite amount of data. Numbers have very specific rules of construction and order. We can construct them using 0-9 and ".". We have memorized these rules and don't need to store the number system anywhere except our heads. Most other databases we use are are constructed from the data of these 2 databases.

A phone book is a database. It is a collection of data (names, addresses, phone numbers) listed in a book. They are listed with alphabetic order of last name, first name. If the listing was random the phone book would be almost worthless. Even so, there would still be a structure because name and address would be in one column and corresponding phone number would be in another column. If we took away even the column structure so they were butted up against one another then it would start losing its database property. All the data would be there but having no structure it would be virtually useless.

A dictionary is a database. It consists of all words of our language with their definitions listed in alphabetical order. It allows us to quickly look up the meaning of a word.

A book index is a database. It consists of important

words and terms used in a book and the page number where they occur. They are also ordered alphabetically.

A checkbook is a database. It hopefully consists of records of each check with the number, who it was written to, the amount, the date, and perhaps a description of its purpose. It is a database that has both textual and numeric data.

Any graph is a database. It consists of a set of pairs of data displayed in a visual format. At least one of the components is numeric data. A graph of the daily stock market performance is a familiar example. It consists the numeric value of the daily DOW index on specific days.

A mathematical formula is essentially a shorthand form of a database. It specifies the structure of the database and the data elements of the database are implicit numbers. Thus, y=sin(x) gives us a database of infinite size over the real number system.

Even this Cable Column or any written information that has meaning to the reader could be called a database because it is a collection of words (data) put together with a coherent structure of meaningful sentences and paragraphs. We generally write things down to provide a quick and easy way to examine the information contained therein.

Each example above is a non computer database that we use in our daily lives. They are databases stored in our head or written down. We don't usually think of them as databases but they satisfy my definition. These examples have many implicit rules that we never give a second thought. When we pick up a phone book we already know what it is and how to use it from past experience.

Our brain is the model and interpreter of database systems. It has some inborn ablility for organizing our perceptions when we are born and our culture provides the other necessary tools, especially a language, for each of us to develop a biochemical database handler in our head. Everything our senses encounter automatically gets processed by our brain and stored in our own brain databases. Its structure and rules are not really known to us although we know how to use it. It works best at recalling general information and usually has trouble recalling large amounts of data with exact precision. That is one reason we need external databases. Another is that there is more data out there than we have time to personally absorb so we need a way to examine only what we need quickly. A database is a good way to share information with others.

Using something as a database just means seeing its data from a certain prespective. Trying to make use of its underlying structure. When we design databases we are essentially trying to make external extensions of our brain to make life a little easier. I want you to appreciate that working with databases is an exciting area of endeavor. It is closely tied to how our brain works and what intelligence is. It's a field that has been greatly expanded by computers with lots of new frontiers to explore.

4. Computers and Databases:

Database systems have been around a long time in our heads and in printed form. But humans are always trying to do things faster and with less effort. When electronic technology reached a necessary threshold the computer was created as an electronic time and effort saving device. I would argue that by my definition of a database, computers are really database machines. Computers store data with a coherent structure so that it can be examined in a way that saves time and effort. At first computers were used for mostly numerical applications but remember, these are really databases with a special structure. As we have become more experienced with computers we have been able to also apply them to textual data.

Why are computers so useful for databases applications? The computer compliments our brain perfectly. It can store vast amounts of data with any structure we design. It can access that data exactly and with great speed. So the computer becomes an extension of our brain in handling data much as an automobile is an aid to our feet in covering distance. In fact many things we do with computer databases save so much time that it would be totally impractical to do them without a computer. Some of the more obscure ways our brain is able to organize data still has not been duplicated on a computer. I suppose that when we can duplicate all of the data handling functions of the brain we will be close to having a thinking machine.

I should also point out a common abiguity when talking about computer databases. Besides using the term database to refer to the collection of structured data in a file, we often use it to refer to the database system that manages that data. I often refer to ARCHIVE as a database. But I should refer to it as a Database System which includes a program to handle a certain type of database and a language for writing even more specialized programs (procedures) to manipulate those databases. Not all computer database systems include a language. Unlike a written database like a phone book in which the rules for using it are in our head, a computer database is managed by a special program which undertands how the data is structured. We can't do much with the data without that program. I will try to be clear about whether I am talking about a database (the data in a file), a database system (the main executable program), or a database program written in the language of the database system.

All the databases mentioned above could be made into computer databases. There are many different computer database systems that could be used to handle these databases. Some database systems can take advantage of the coherent structure of the data better than others. Computer database systems usually have ways to order and select on the data. There are usually ways to search the data and visually display it. There is no set rule on what a database system has to do but they all have one thing in common. They handle the data in a time and effort saving way. When viewed with this in mind, some old familiar programs take on new lives as database systems.

*5. **Some Familiar Computer Database Systems:*** Each of the PSION programs is actually a database system. The ability to transfer data between the 4 programs indicates this commonality. By viewing them as such I hope to illustrate some advantages and disavantages of various kinds of databases systems. A database system may only have commands and not an explicit language but since a language is basically just a string of commands that can be stored as a program we can use the Toolkit ALTKEY function to store sequences of commands and give

even QUILL a quasi-language (also called macros).

QUILL

Using my rather loose definition of a database, I can call QUILL, the most used QL program, a database system and the documents created with it databases. QUILL could be described as a visually oriented database system designed to work with textual data in the form of words, lines, paragraphs, and pages. It does not recognize numbers except as text. The type of structure given to this type of data is usually extremely complex. The rules for putting words into sentences and sentences into paragraphs defies any easy explanation. Although work is being done in this area we are a long way from having computers deal with this type of data except for its physical structure (appearance) and spell checking. QUILL is not designed to take advantage of any simple structure that might be given to the data. I could create a database with QUILL that has a simple structure like a phone number list made of lines with columns for last name, first name, town, phone number. The structure rules would be in my head and not known to QUILL because QUILL doesn't manipulate columns. There would be some advantage to having the phone list as a QUILL document. I could locate names with the search command. I could also locate phone numbers so I could look up a person by his phone number. These searches would take a long time if the phone list were large because all text would have to be searched sequentially until a match were found since QUILL would not know that I wanted to search only information in a certain column. If I wanted to print the phone list I could basically only print it out in the same order that I typed it in. I could print out just a portion of it by making pages and printing just a page. If I wanted to print out the phone list ordered by phone number with phone number in the first column I couldn't do it automatically with QUILL. If I wanted to group together everyone who lives in a certain town I couldn't do that either. I could find them but I couldn't automatically group them together. So QUILL has some useful database features but it is somewhat limited in manipulating simple structured data and it has no explicit language. But it is the database system of choice for dealing with data constucted of words, sentences, paragraphs, and pages and being able to highlight information as in bold and underline. This column is a good example of a database that works best with QUILL

EASEL EASEL is a visual database that deals with pairs of data. One of the data elements must be numeric the other is usually text. It can handle more than one set of data (figures) at a time but is limited in its data handling capabilities. It can create new data from old by using formulas which is a handy function. Its primary use is really for graphical display of numeric data from other sources. It is not a good database for storing data in its own right since you can't easily inspect individual bits of data but have to work with the whole set (figure) at once.

ABACUS ABACUS can be called a database system also. It is primarily a numerically oriented database system but is not bad as a textual database system also. It is arranged so that all data is structured in rows and columns. It is visually oriented database system and it's easy to get at any of the data. It is actually a pretty good database system for handling many common databases that have simple structure. It handles a phone list database very well because a phone list can be given a column format. We could put the last name in column A, the first name in column B, the town in column C, and the phone number in column D. The phone number is really not a number because it has space or dash as separators. So we wouldn't be using ABACUS's numeric powers to keep a phone list database in it. Still we could do searches. We could order the rows using any of the columns. So we could order by phone number. We can move columns so we could put the phone number in the first column and print out our reverse phone lookup. We could also group by town. So ABACUS has powerful database features. If we were to use it on a database that has numeric data like a check book database it is very handy because we can easily sum the check amounts to get totals. ABACUS is sometimes superior to even ARCHIVE as a database system. When I was making a brochure of class schedules for a health spa I found ABACUS to be the easiest program to set it up on. The data needed to be arranged in cells for the 7 days of the week (columns) and by time of day (rows). Periodically the schedule had to be updated and printed out again. This is very hard to do with QUILL and with ARCHIVE it would take a lot of special programing. With ABACUS it was simple even though all the data was textual. It is often easier to export data from an ARCHIVE database to ABACUS to do numerical calculations than to write the necessary ARCHIVE procedures to do it. What are the short comings of ABACUS as a database system? It doesn't have enough commands or an explicit language to handle a complex series of data manipulations. There are many things we want to to with databases that can't be done. Suppose we had an address database with columns last name, first name, address, city, state, zip, phone, and key. And in the key column we had whether the address was for a friend, customer, or vendor. To locate an address might work by using the 'Go To' command. But that is a very limited search command and would be very slow in a large address database because all columns of text would be searched. Suppose we wanted to send a letter to all our customers. We could order by the key column to group our customers together. That might not work perfectly because we might have a customer who was also a friend. If we had 'friend,customer' as our key value then the address would sort with friends and not customers. We couldn't print out mailing labels from ABACUS because we have no language to pick out the data elements and put them in an address label form and print them. Also since ABACUS keeps everything in memory when our database reached a certain size it would fill memory and we would have to break it up into smaller databases. ABACUS is a pretty good database for simple structured data but lacks the ability to do many handy things.

ARCHIVE ARCHIVE is actually designed as a database system so it has many data handling features that QUILL, ABACUS, and EASEL do not. It is what is known as a multi-file relational database. This means that it can access multiple databases simultaneouly and that

the data is arranged in a table format. The table format is similar to the column and row format of ABACUS but a closer analogy is a box of index cards. The relational database term for card is record. ARCHIVE data is in a file of records instead of a box of cards. In a relational database the record is divided into parts so a particular kind of data is in a particular location of the record. The area where the data is expected is labeled so we know what type of data goes there. This labeled area is called a field. This is just the same as taking an index card, labeling it to keep a certain kinds of information in particular areas of the card. If we reproduced the card and put them in a box ready for us to begin filing them out we would be close to having an exact equivalent to creating an ARCHIVE database ready to hold data.

So relation databases such as ARCHIVE are based on very traditional concepts used by libraries for centuries. Take a card (record) and figure out what type of information you want where on the card (fields). A library index card would want types of information (fields) like author, title, subject, publisher, number of pages, year of publication. Fill out the cards with information about the books (enter data) and store them in order by the most important item (field). If we want to look up by author this would be the authors name. If we want to look up by title it would be the book title.

If we were to create an ARCHIVE equivalent to a library index card we would defeine the record to have a field structure something like this :

(field names)	(field values)
author$	: Jones,Jan
title$	: QL Superbasic The Definitive Hbk
subject$	: computer,Sinclair QL,SUPERBASIC
publiser$	: McGraw-Hill Book Company
isbn$	: 0-07-084784-3
year	: 1985
pages	: 257

Each record (card) would appear on the screen as it is shown above. The field names indicate the kind of information and the field values are the information for in a particular record in this case a library book. Besides a field value and a field name, a field also has a type and a number. It is either a numeric field or a text field. When an ARCHIVE database is created the last character given to the field name determines whether it is a numeric or text field. Just like in BASIC, if the field name ends in a '$' then it is a text (string) field and if it doesn't it is a numeric field. Notice that in the above card index example there are 7 fields in all of which 5 are text fields and 2 are numeric. The field number (rank,order) is determined at the database creation time just like the field name and type. The first field defined is field number '0' and the next is field '1' and so on.

To review, when a database is first set up with ARCHIVE (created) we must define what consitutes a record by specifying the fields of the record. The fields have the following properites :

1. A field name
2. A field type (ends in '$' or not)
3. Total number of fields (numfld())
4. Each field's number (0 to numfld()-1)

Each ARCHIVE database has these qualities defined at creation time and they never change. It is like creating our box of index cards with the labels on them ready to be filled out. This is the basic coherent structure of our database that ARCHIVE understands. There is no data in the database until we start inserting records (values for the fields).

If having a endless box of labeled file cards ready for you to put your information on appeals to you, you will love ARCHIVE. ARCHIVE is a file card genie. Its language is specifically designed to manipulate file cards (records/fields). It can inspect and rearrange those cards in almost any imaginable way. True you sacrifice some flexibility by having to structure your data into fields and records. But I think you will find that in many cases it actually helps put data in that form. It makes you really look at your data and think about what you want to do with it rather than putting it in a file a disorganized way as you might do with a word processor. As the saying goes, garbage in - garbage out. Structure makes data more useable.

Our sacrifice in flexibility is made primarily so ARCHIVE can give use 3 wonderful commands :

1) ORDER : ARCHIVE will order our records (cards) by any field, 2 fields, or 3 fields in ascending or descending order. Within a few seconds we could change a library card database from being ordered by author to being ordered by title. We wouldn't have to have 2 separate file card databases like a library does. We can change our database to suit our needs instantly.

2) LOCATE : This command works off our current order. It allows us to find any record within 1 second no matter how large our database (100 or 1000 or 10000 records). If we had our library card database ordered by author we could just type : locate "Jones,Ja" to find Jan Jones's book instantly.

3) SELECT : This command allow us to work with subgroups of our database. We can temporarily eliminate records we don't want for quicker access to those we do. For our library card database we could type :

```
select instr(subject$,"computer") and year<1986
```

to get all records which deal with computers and were written before 1986. Select has a very flexible format making it very powerful.

The ARCHIVE language has many other useful commands and functions but these 3 (order, locate, select) are the key to its usefulness. They are the reason why relational database were developed.

Next Issue we will look a little bit into the inner working of ARCHIVE. It will help us understand some of its strong points and weak points. We will also look at these 3 commands in detail and begin some examples.

Bill Cable
Wood and Wind Computing,
RR-3 Box 92,
Cornish, NH 03745

TS-2068 Program "Dbx.B6"
A software for RECORD Keeping

Key in Project and Tutorial- By Bill Jones

""Dbx.B6"" is a New "Record Data Base" Software that takes advantage of the MERGE function to conserve FREE MEMORY and thus provide the largest data base ever before created with the TS-2068. In addition, there is a word processor included, "Search and Find", and the most versatile of Sorting choices ever before offerred. There are four programs to key in. Dbx.B6 is the main program. As it operates two other programs are automatically MERGED when needed by the function in progress. "sort.B6" MERGES when SORT is selected at menu. "Init.B6" is MERGED when the Dbx.B6 program is first RUN. "ml6.B6" is the MENU that is saved to Disk to be loaded each time the program needs the menu. A fifth program "var.B6" is given to be used only once to initialize a set of variables used in the main program.

Actually, Dbx.B6 evolved from an earlier program called "Mail Merge". Most of the Mail Merge functions were upgraded and expanded and more lattitude of sorting was added. The addition of more sorting choices makes Dbx.B6 a software that is adaptable for just any type of records that one may need to create and manage. Some typical types of records are: Telephone lists, Mailing Lists, Inventory lists, Disk Library, Audio Tape Library. It can even be used for such diverse purposes as Invoices, or Daily Diary. If you can think of any type of records needing from one to 50 lines or more, this program will create and manage the records. The data capacity is enoromous by TS-2068 standards. A Telephone list having single line records of 28 chrs can be built to over 1000 names.

Other examples: A Seven Line Record for use as a combination Mail File with 2 lines for notes can be built up to over 130 records. A smaller Mail File having 4 lines of 31 characters each can reach over 240 records in length. A 5 line invoice of 60 chrs per line can be as large as 100 customers. A 3 line by 20 chr INVENTORY FILE can hold up to 500 records. The SIZE of the RECORDS determines the maximum number of records, and the user sets up the Record dimensions. SORTING CHOICES are by any line, and by "First or Last" group within the line. When the records are Sorted each record remains intact and the records are lined up by the sorted line. Example: a mail file can be sorted by either LAST NAME or by Zip Code. A Stock List can be sorted by NOUN or by Stock Number. IF there are 10 record lines there are 20 sorting choices.

Print Out of the records is controlled by menu choice. Whole records may be printed or partial records. For example, the Mail List that is used for Update Magazine has 7 lines per record. The first four are Name and Address, the fifth is the person's first name as "Dear John," and lines 6-7 are information lines. To print LABELS, we elect to print-out the first four lines. Then prompts ask for "TAB and Spaces Between". Such control allows any size label to be printed. For just paper print of listings prompts guide the set-up of printing so that pages of listings are printed without over-running the page separations.

The Menu offers a "SEARCH" option. The "Search Word" is input, and prompts ask for the "Line to Search". Input "Danka" and the first record to have Danka is presented on screen. We can continue for more Dankas, or elect to Quit to Menu. Dbx.B6 has a management protocol that lets the user use any number of disk drives from one to four. Prompts allow different drives to be used for data files and the program itself. Data files are SAVED and LOADED by discrete "numbered files" for best orientation of record files management.

Dbx.B6 is programmed 100% in BASIC. If mistakes are made during operation the program interrupts with a REPORT CODE at the bottom of screen. There are two "GO TO" solutions to operator error. The one to use during a SORT operation is <GO TO 1974>. Line 1974 DELETES the sorting program (if on board) and re-constructs the main "Dbx.B6" program, then presents the main Menu. If a report code comes during operation in any mode EXCEPT SORTING, <GO TO 2000> resumes operation. These two corrective <GO TOs> should be posted somewhere until you have operated the program enough to remember them. Just remember that REPORT CODES come only after a MISTAKE in operation, mostly from entering the wrong DISK Number. A line number where the mistake occurred will be given in the report code.

Most often, after a report code, you can LIST the line number and see that a <GO TO> the previous line number will resume operation at that point. A study of the main Menu given at the end of this article will be helpful at this point. *** See Menu *** We have dwelled upon Report Codes too much really. The program is FRIENDLY and easy to operate. After Key-in of the five

1 of 7 pgs

programs Do NOT RUN the program to get it started, Instead <GO TO 1804>. The Dbx.B6 program SHOULD NOT be Saved to Auto Run unless you want to create a separate Auto Run Loader. This is because during operation the main program is actually Destroyed when SORT is ordered by menu selection. Then when SORTING has completed the sort program is DELETED, and the Dbx.B6 program is MERGED BACK. If the Dbx.B6 program were saved to Auto Run this process would be thwarted.

A typical AUTO LOADER that you might use to Load and Run this program is: <10 RANDOMIZE USR 100:LOAD "Dbx.B6": GO TO 1804>. Then SAVE this one line auto loader by the direct command: <RANDOMIZE USR 102: GO TO 10>. When the music plays touch s. This is for LKDOS Systems. All of the SAVE and LOAD functions are given in LKDOS syntax. For other systems the SAVE, MERGE, and LOAD lines should be edited and changed in syntax to that needed by the DOS in use. Some DOS does not support MERGE. For those systems ALL of the lines having DELETE and MERGE should be edited to remove the DELETE and MERGE. But KEEP the GO TOs. This way will result in a larger program that remains in TS-2068 memory and has a reduction in maximum RECORD capacity of about 15%.

The Key-in Tutorial

We'll use the key in project as a tutorial. DID YOU KNOW that Sinclair Computer have "BUILT IN SORTING" capability? Well, dont be ashamed as most users dont know that. The Sinclair ROM has a sophisticated SORTING function built in. All we need to do is manage it with Basic Programming. Lets learn to do this in a simple fashion. Type these direct commands from the keyboard. 1. <LET A$="ABC Benjamin Franklin and Joe Smutz" ENTER>. 2. <LET B$="Ben Franklin" ENTER>. Now we have two lines of data entered that can be SORTED. It would appear that "A$ is LARGER then B$", now wouldn't it? But instead, "B$ is LARGER". At least it is "to the Sinclair ROM", because A$ begins with lower number characters than does B$. Lets prove it. 3. Type: <IF A$<B$ THEN PRINT A$ ENTER>.

The data in A$ will print to screen because the ROM has SORTED the two data lines. So despite the fact that the A$ data is much "longer" than the B$ data, "A$ is Smaller". If this is news to you then you should monkey around entering different data in A$ and B$ and see how the ROM evaluates the two data lines. Actually you can see HOW the Rom COMPARES data by looking at the user manual pages 239-245. Each Character is assigned a numbered CODE for the ROM to evaluate. "PRINTABLE characters range between CODE 32 (space) to CODE 122 (z). The ROM sorting priority (order) is the order of the character CODE Numbers. Since we wont be sorting "non printable" characters, we will trap out all codes BELOW 32 and ABOVE 122, and just ask the ROM to sort PRINTABLE CHARACTERS by comparing one data line with another.

Dbx.B6 will assume that no record line should begin with an empty space and discard the line if it does. We will set up a SORTING system that lets us choose to sort by GROUPS WITHIN A LINE-first group in the line, or LAST group within the line. Then we will choose which line to sort the records with. Lets look at a couple of examples: A Telephone List would be a set of records having only one line, such as, "Mr. George W. Washington Jr. 942 466 9223". We could sort by "First Group", but that wouldn't do anything intelligent to sort by "Mr.". Or, we could sort by "LAST GROUP" and sort by tel numbers, another exercise in folly!.

Lets fix the line so that it can be sorted. Change it to: "Washington,Jr Mr. George W." See what we have done? We put LAST NAME FIRST in the lines to be sorted and the telephone list can be sorted by entering "Sort By First Group". Then if we enter all of the names "last name first" we can let the ROM compare all "First Groups" of the lines and sort them. Also, in managing the records you'll want to keep the lines as short as practical. So some abbreviations of long names may be in order, as "Washington,Jr Geo W. 6494448222". That will trim the line to 31 characters. By shortening the lines and keeping within the bounds you can make much longer record files.

Another Example: A Mail file of records is constructed using 5 lines. Four lines are used for name and address, the fourth line containing "STATE and Zip Code". The fifth line will be the person's FIRST NAME to use for picking off in a letter (Dear John,). You will want the first line of Mail files to sort, and if you are doing Bulk Mail, you'll also want to sort byZip Code. A typical First Line: "Mr. George W. Bush,Jr.". Note that the only practical sort for the first line would be "LAST GROUP" or <Bush,Jr>. Also note that the space has been removed between "Bush Jr.". We wouldn't want to sort "Jr.", eh?. Sorting by LAST GROUP could also be accomplished with LINE 4 to sort by ZIP CODE. Canadians use a

2 of 7 pgs

hyphen between the twin postal codes. To sort by Zip Code you would enter <Line 4> <LAST GROUP>.

An EXAMPLE of an INVENTORY Record FILE: Line 1 could be "noun and Stock Number". Line 2 could be "Dollar value and ON HAND number". Sort by first group of line 1 for a line up by noun. Sort by Last Group of line 1 for a stock number sort. Sort by first group of line 2 to get Inventory Value, or last group of line 2 to get On Hand Levels (for re-order information). As you can see, there is a certain amount of discipline required to be consistant with the format for entering LINE data in a record data base.

The speed of Sorting is (in my opinion) not a critical criteria for a useful data base program such as Dbx.B6. One spends days working on a large Record Data Base. Sorting is needed only once, when the record file input has been finished for the day. Sorting with BASIC is indeed slow. But by comparison with the data input chore, sorting is fast. As a bench mark, I sorted a record file having 100 records of 7 lines and 31 characters per line. It was a Zip Code Sort, which requires more time than a "first group" sort. The sort was accomplished in 57 minutes. The length of time required to sort depends upon the variable factors of Total number of Records, and the number of fields (lines) in the records, and whether sorting by "First or Last" group within the sort line.

Later, in another issue of Update Magazine I will give you a fast Machine Code program to key in that will do the sort function in about 10 seconds. But for now lets do it in BASIC and learn something about sorting techniques. The act of keyin in the program should be educational, and when finished you'll have the best RECORD DATA BASE program ever done for the TS-2068.

Also a Daisy.B6 Annex Program

The Dbx.B6 program and its Sort annex is operated as satelite programs from the Daisy.B6 software. Dbx.B6 is MERGED to Daisy by Menu elective to create and manage record data bases to be integrated within Text print-out products. Also, Daisy uses the Dbx.B6 data base files (without Dbx.B6 being on-board) for Mail Merge purposes to print personalized form letters to the mailing list. Other types of Dbx.B6 data base record files can be print processed within Text files (or alone) by the Daisy software. The maximum size data base that can be created by Dbx.B6 (when it is MERGED into the Daisy environment) is reduced by the size of other Daisy data bases that may be present. ** Other Basic Programs may be MERGED to the Dbx.B6 program and operated, as long as the line numbers of the Merged program do not overlay the Dbx.B6 program lines.

Typically, a letter data base can be present in the daisy program and still allow about 100 mail files to be created, or loaded from disk. When creating records with the Dbx.B6 program to be later used with the Daisy software, one should bear in mind that the files must be kept within maximum bounds of total Memory used. A good rule of thumb would be to keep the Record Dimensions within a boundary of about 10K of memory usage, which would accommodate a mail file of about 60 records of 5 lines and 31 characters. Then such a record data base could "co-reside" in the Daisy program with other files to be "mix printed".

The Key in Project

Start the key-in project by FORMATTING a new Disk to have ready for the SAVES. Key in each of the five programs and save each one separately with the file name given. Take your time in doing the key in. Dont just have the goal of finishing, but study the program lines as they are keyed in. Such study is more valuable than a manual to learn what will be taking place as the program operates. And, you might learn some of my unorthodox methods of programming while you are at it. It will likely take several days of work to get it all keyed in, so SAVE your work often. Now we will discuss each of the five programs individually.

The "Dbx.B6" Program

The Dbx.B6 program contains the Menu call and all of the functions EXCEPT SORTING. It is also a CONTROL program that DELETES itself before MERGING the Sort Program. It also MERGES the Init.B6 program as needed to set the LKDOS system and Set the Printer to one of three print styles. Dbx.B6 is organized to keep FREE MEMORY at maximum by using DELETE and MERGE.

The "Init.B6" Program

The "Init.B6" program MERGES in when needed to provide the initialization of the LKDOS system for the type of interface in use. It also

contains a menu for PRINTER STYLE set-up. The Printer Style changes are programmed for an Epson Compatable Printer, the Panasonic KXP 1124. The program line # 1996 will likely need editing and change of the OUT commands for a different type of printer than the KXP 1124. A temporary measure can be taken to by-pass the PRINTER SET function and just use MANUAL Settings at the printer itself. To do so place a REM after line numbers 1996 and 1997. Then you can set your printer to the Print Style needed and the software wont command the printer at all.

Two of the programs are not used at all by the Dbx.B6 operating program, but are necessary to key in and use "BEFORE" running the Dbx program. These are:

The "var.B6" Program

Key in the var.B6 program and save it with the name "var.B6". Then LOAD the "Dbx.B6" program from disk and do the following: Type <GO TO 1976 ENTER>. This results in the var.B6 program MERGING, the vars being initialized, and then the var.B6 program is DELETED, never to be needed again. But later, if you want to add more functions to the Dbx.B6 program and use more variables you can add new vars to the var.B6 program and type <GO TO 1976> to add the vars to an operating program.

The "Menu.B1" Program

Type in the "Menu.B1" program. This program's only purpose is to CREATE the Menu in a Disk Screen File, to be loaded to the Dbx.B6 program when the menu is needed. This menu would normally take up about 900 bytes of FREE MEMORY, reducing the total data capacity by several records. When the Menu is saved to disk it is called up as needed using only 10 bytes of programming. After keying in the Menu.B1 program type RUN. The menu will appear on screen. Compare it with the printed Menu SAMPLE for correctness. If the menu is the same as the Sample, then type <CONTINUE ENTER> and it will be SAVED to disk. The following help is given for keying in the graphics characters.

In line 9998 the first group is gotten (after the Quote) by using CAPS SHIFT 9 to shift into GRAPHICS, then SHIFT 3 three times, followed by SHIFT 2, then SHIFT 9 to get out of Graphics. Then later in the same line, the second group of graphics characters are entered by, using CAPS SHIFT 9 for graphics, then SHIFT 3 three times, followed by SHIFT 2, then SHIFT 9 to get out of graphics. The third graphics group is gotten by SHIFT 9 to start graphics, then SHIFT 1, followed by SHIFT 3 nine times. Then SHIFT 9 to get out of graphics.

Challange to Math Programmers

This suggestion is for our MATH Expert programmers (How about it Warren!"). Such a data base program as Dbx.B6 can be ammended to have quite a powerful "VIEW CALC" annex far beyond the capabilities of the old PSION View Calc Program. The DEF FN function could be used for base formulas. The annex program could be MERGED as needed to use the DEF FN formulae to calculate from figures given in one of the lines of the data base. The MERGE annex could be arranged to do the following: 1. Elect the record line to exercise a formula. 2. Elect the record line to RECEIVE the results of the Calc. 3. Exercise the formula and deposit the results in the receiving line.

Example: a DEF FN formula has been added to the main program to "calculate the value of "B" when "A" is given". Line #4 of the data base contains the value of "A". The program picks off the value of "A" from line 4, performs the calc and then deposits the value of "B" in line 5. Perhaps this will give someone ideas. As for me, I'm not sharp enough on mathematics to create such an annex.

The Dbx.B6 is offerred as an ISSUE DISK in LKDOS 40 or 80 Track Format (5 1/4" Disk) ready to operate. But I encourage all who want to learn something about Basic Programming to key the programs in for the educational value. Then in a latter issue we will add the faster MC sorting functions. _BJ-

The ISSUE DISK

To Receive the Dbx.B6 ISSUE DISK, send $22.00 to: Update Magazine, 513 E. Main St., Peru, IN 46970. Available for LKDOS in either 3 1/2" or 5 1/4 inch diskette, formatted either 40 or 80 tracks. Specify Drive Size and Format.

4 of 7 Pgs

"Dbx.B6" The Main Data Base Program

```
1800 REM ** "DBX.B6" **
             A RECORD DATA BASE PROGRAM
1801 REM ** A SHARE WARE PROGRAM
1802 REM ** An Update Magazine Issue Disk **

1803 REM ** By Bill Jones, 1317 Stratford Ave,
            Panama City, FL 32404

1804 BORDER oo: PAPER oo: INK og: CLS : GO TO VAL "
1992"
1976 RANDOMIZE USR VAL "100": MERGE "var.B6"
1978 DELETE VAL "1977",VAL "1977": GO TO VAL "1992"

1982 CLS : LET a=LEN o$(1,1): INPUT "Input SEARCH G
roup";m$: LET b=LEN m$-1: LET c=a-b: INPUT "Record
Line to Search?";so: LET n=1
1983 PRINT "Searching "; INVERSE oa;m$; INVERSE oo:
 FOR n=n TO m5: ON ERR  GO TO VAL "1986": FOR y=1 T
O c: IF o$(n,so)(y TO y+b)=m$ THEN  ON ERR  GO TO V
AL "1990": STOP
1984 NEXT y
1985 NEXT n
1986 ON ERR  RESET : CLS : PRINT  FLASH oa;AT oj,of
;m$;" Not Present";#oo;"<1> Another Search <2> Quit
": PAUSE oo: LET m=CODE INKEY$-48: IF m<>oa THEN  G
O TO VAL "2001"
1988 GO TO VAL "1982"
1990 ON ERR  RESET : PRINT AT of,oo;"Record # ";n:
FOR y=oa TO 12: PRINT o$(n,y): NEXT y: LET n=n+oa:
PRINT #oo;"1. Cont 2. Search New Gp 3. Quit": PAUSE
 oo: LET m=CODE INKEY$-VAL "48": CLS : GO TO (m<oa
OR m>oc)*VAL "1990"+(m=oa)*VAL "1983"+(m=ob)*VAL "1
982"+(m=oc)*VAL "2001"
1992 RANDOMIZE USR m1: MERGE "Init.B6"
1999 DELETE VAL "1993",VAL "1998": GO TO VAL "2001"

2001 RANDOMIZE USR m1: LOAD "m16.C1"SCREEN$ : PRINT
 AT op,oy;("Pica" AND ps<ob)+("Elite" AND ps=ob)+("
Micron" AND ps=oc)+("Cond" AND ps>oc);AT ot,oi;pg;"
 ";AT ot,oq;nn;" ";AT ob,ou; FLASH oa;("No " AND LE
N o$(oa,oa)<ob)+("Yes" AND LEN o$(oa,oa)>oa); FLASH
 oo
2002 PRINT AT oc,ox; FREE ;" ";AT oj,ox;("Limit" AN
D LEN o$(oa,oa)<ob)+("Size " AND LEN o$(oa,oa)>oa);
AT ok,ox;(11 AND LEN o$(oa,oa)>oa)+(INT (( FREE -k2
)/124) AND LEN o$(oa,oa)<ob);AT ol,ox;"Records";AT
 om,ox;"(";: PRINT ("4,31" AND LEN o$(oa,oa)<ob);: I
F LEN o$(oa,oa)>oa THEN  PRINT 12;",";13;
2003 PRINT ")": PAUSE oo: LET z=CODE INKEY$-VAL "48
": GO TO (z<oo OR z>oi)*VAL "2003"+(z=oa)*VAL "2004
"+(z>oa AND z<oe)*VAL "2007"+(z=oe)*VAL "2040"+(z=o
f)*VAL "2056"+(z=og)*VAL "1982"+(z=oh)*VAL "2053"+(
z=oi)*VAL "2049"+(z=oo)*VAL "2057"
2004 CLS : PRINT AT oj,of; FLASH oa;"Confirm START
NEW"; FLASH oo;''TAB of;"<y> Yes or <n> NO": PAUSE
oo: LET g=CODE INKEY$: CLS : IF g=VAL "89" OR g=VAL
 "121" THEN  INPUT "Input Nbr of Files to set up?";
11: LET mx=11: INPUT "Input Nbr of Lines";12: IF mx
*12*VAL "31"+k2> FREE  THEN  PRINT #oo;"TOO BIG": B
EEP ob,oh: GO TO k2+od
```

5 of 7 Pgs

```
2005 CLS : IF g<>VAL "89" AND g<>VAL "121" THEN  GO
 TO VAL "2001"
2006 DIM o$(11,12,VAL "31"): LET in=oa: LET ms=oa:
CLS : GO TO k2+oi
2007 IF z>oa AND z<of THEN  LET in=oa: INPUT "Enter
 Start Number";ms: INPUT "Enter END Number";mx: CLS

2009 FOR o=ms TO mx: IF o>mx THEN  GO TO k2+ot
2010 IF z<oc THEN  GO SUB VAL "2022"
2011 IF z=od THEN  GO SUB VAL "2030"
2012 IF z=oc THEN  GO SUB VAL "2034"
2018 NEXT o: GO TO k2+ot
2019 LET o=mx+oa: GO TO VAL "2001"
2022 CLS : PRINT AT oo,oj;"Record #";o: FOR n=1 TO
12: PRINT AT ou,oo;"INPUT LINE "; FLASH oa;n; FLASH
 oo;" Or ` to QUIT": INPUT m$: IF m$="`" THEN  LET
o=mx+oa: RETURN
2024 PRINT AT ou,oo;"                              ";AT
n*ob,oe;"Line #";n'm$;#oo;AT oo,oo;"Touch <`> To Co
rrect Else ENTER ": PAUSE oo: LET m=CODE INKEY$: IF
 m=96 THEN  PRINT AT n*ob+oa,oo;"
            ": LET n=n-oa: GO TO VAL "2026"
2025 LET o$(o,n)=m$
2026 NEXT n
2027 CLS : PRINT AT oe,oj;"Record # ";o': FOR n=oa
TO 12: PRINT o$(o,n): NEXT n: PRINT #oo;AT oo,oo;"E
NTER if OK Other Key to Correct": PAUSE oo: LET m=C
ODE INKEY$: IF m=om THEN  RETURN
2028 LET n=n-oa: GO TO VAL "2022"
2030 CLS : PRINT TAB oj;"RECORD #";o'': FOR n=1 TO
12: PRINT AT n*2,oe;"Line # ";n'o$(o,n)': NEXT n: P
RINT ''"O.K.? <y> Yes or <n> No": PAUSE oo: LET g=C
ODE INKEY$: IF g<>VAL "121" AND z<>VAL "89" THEN  I
NPUT "Enter Line # to Correct";n: PRINT AT ov,oo;"E
nter new Line ";n: INPUT o$(o,n): GO TO VAL "2030"
2032 RETURN
2034 PRINT TAB oj;"Record # ";o': FOR n=oa TO 12: P
RINT o$(o,n)
2035 NEXT n
2036 IF o=mx THEN  PRINT "ENTER WHEN READY": BEEP o
a/od,oy: PAUSE oo
2037 RETURN
2040 CLS : PRINT #oo;"Set Printer, ENTER": PAUSE oo
: INPUT "Enter TAB";tb: BEEP oa/of,ox: INPUT "ENTER
 Start Rec #";ms: BEEP oa/of,ox: INPUT "ENTER END R
ec #";mx: BEEP oa/of,ox: INPUT "ENTER # of Lines (4
 for Labels";ly: BEEP oa/of,ox: INPUT "ENTER #Space
s Between Rec";ss: BEEP oa/of,ox: INPUT "Page Contr
ol? <1> Yes <2> No";pc
2041 IF pc=oa THEN  LET ln=VAL "56": LET nn=oa
2042 FOR n=ms TO mx: FOR y=oa TO ly: LPRINT TAB tb;
o$(n,y): LET nn=nn+oa: NEXT y: IF pc>oo AND ss+nn>l
n THEN  OUT VAL "127",ol: LET nn=oa
2044 IF ss>oo THEN  FOR g=oa TO ss: LPRINT : LET nn
=nn+oa: NEXT g
2046 NEXT n
2048 GO TO VAL "2001"
2049 CLS : INPUT "Enter <1> SAVE <2> LOAD";m: BEEP
oa/of,xo: INPUT "ENTER DATA Disk #";dd: RANDOMIZE U
SR m1: GO TO dd: RANDOMIZE USR m1: CAT ".A$",: BEEP
 oa/of,oq: INPUT "ENTER File name + Ext";m$: IF m=o
a THEN  RANDOMIZE USR m1: SAVE m$ DATA o$(): GO TO
k2
2050 RANDOMIZE USR m1: LOAD m$ DATA o$(): LET 13=LE
N o$(oa,oa): FOR n=1 TO k2: ON ERR  GO TO VAL "2051
": LET 15=(n AND o$(n,oa)(oa)<>" ")+(15 AND o$(n,oa
```

```
)(oa)=" "): NEXT n
2051 ON ERR  RESET : LET l1=n-oa: FOR n=oa TO VAL "
500": ON ERR  GO TO VAL "2052": LET x=LEN o$(oa,n):
 NEXT n
2052 ON ERR  RESET : LET l2=n-oa: BEEP oa/oj,zo: IN
PUT "Enter Program Disk #";pd: RANDOMIZE USR ml: GO
 TO pd: GO TO VAL "2001"
2053 CLS : INPUT "Enter Record line# to Flip";so: I
F so>l1 OR l5<2 THEN  GO TO k2
2054 CLS : FOR n=oa TO l1: PRINT TAB od; INVERSE oa
;"Record No. ";n;" Line# ";so; INVERSE oo'o$(n,so):
 NEXT n
2055 PRINT '' FLASH oa;"Enter When Ready": PAUSE oo
: GO TO VAL "2001"
2056 DELETE VAL "1804",VAL "2055": CLS : PRINT #oo;
"MERGING . .": RANDOMIZE USR ml: MERGE "sort.B6": G
O TO VAL "1810"
```

"sort.B6" The Sorting Program

```
1810 CLS : PRINT "Finding DB Dimensions": FOR n=1 T
O k1: ON ERR  GO TO VAL "1812": LET l3=LEN o$(n,1)
1811 NEXT n
1812 ON ERR  RESET : LET l1=n-1: FOR n=1 TO ml: ON
ERR  GO TO 1814: LET l3=LEN o$(1,n)
1813 NEXT n
1814 ON ERR  RESET : LET l2=n-1: FOR n=l1 TO oa STE
P -oa: IF o$(n,oa)(oa)<>" " THEN  LET l5=n: GO TO V
AL "1818"
1815 NEXT n
1818 PRINT "The DB is: O$(";l1;",";l2;",";l3;")"
1820 BEEP oa/of,zo: INPUT "Enter Record Line # to S
ort";so: BEEP oa/of,zo: PRINT #oo;"Sort: <1> First
Gp or <2> Last  or ENTER to Quit": PAUSE oo: LET sl
=CODE INKEY$-VAL "48": IF sl=om THEN  GO TO k2
1821 IF so>l2 OR so<oa OR sl<oa OR sl>ob THEN  GO T
O VAL "1820"
1824 GO SUB VAL "1900"
1826 IF sl=ob THEN  GO SUB VAL "1920"
1832 GO SUB VAL "1954"
1834 IF sl=ob THEN  GO SUB VAL "1938"
1836 GO TO VAL "1974"
1900 CLS : PRINT AT oj,oo;"Eliminating Repeats"'TAB
 of;"% Done": FOR n=1 TO l5: FOR y=n+oa TO l5: PRIN
T AT ok,od;INT ((n/l5)*100): IF o$(n,oa)=o$(y,oa) A
ND o$(n,oa)(oa)<>" " THEN  FOR g=oa TO l2: LET o$(y
,g)="": NEXT g
1902 NEXT y: NEXT n
1904 CLS : PRINT AT oj,od;"Eliminating Skips": LET
n=oa
1905 FOR n=n TO l5: PRINT AT l1,l0;INT ((n/l5)*100)
;" % Done": IF o$(n,oa)(oa)=" " THEN  GO TO 1910
1906 NEXT n
1907 FOR n=1 TO l1: IF o$(n,oa)(oa)=" " THEN  LET l
5=n-1: RETURN
1908 NEXT n
1909 RETURN
1910 BEEP oa/oz,zo: FOR y=n+1 TO l5: IF o$(y,1)(1)<
>" " THEN  FOR g=1 TO l2: LET o$(n,g)=o$(y,g): LET
o$(y,g)="": NEXT g: LET n=n+1: GO TO 1905
1911 NEXT y
1912 GO TO 1906
1914 ON ERR  GO TO VAL "1918": FOR n=h+oa TO m: IF
o$(n,ob)(oa)=" " THEN  LET o$(n,oa)=""
1916 NEXT n
1918 ON ERR  RESET : RETURN
1920 CLS : PRINT AT oj,oa;"Line Group Reverse": FOR
 n=oa TO l5: PRINT AT ol,ol;INT ((n/l5)*100);"% Don
e"
1922 FOR y=l3 TO 1 STEP -1: IF o$(n,so)(y)<>" " THE
N  GO TO 1926
1923 NEXT y
1924 NEXT n: RETURN
1926 FOR g=y TO 1 STEP -1: IF o$(n,so)(g)=" " THEN
 LET o$(n,so)=o$(n,so)(g+1 TO y)+" "+o$(n,so)( TO g
): GO TO 1924
1928 NEXT g: GO TO 1924
1938 CLS : PRINT AT oj,oa;"Replace First Group, Lin
e ";so: LET b$="": FOR n=oa TO l5: PRINT AT 12,12;I
NT ((n/l5)*VAL "100");"% Done": IF n>l5 THEN  RETUR
N
1940 FOR y=1 TO l3: IF o$(n,so)(y)=" " THEN  GO TO
1946
1942 NEXT y
1944 NEXT n
1945 RETURN
1946 FOR g=l3 TO y STEP -1: IF o$(n,so)(g)<>" " THE
N  LET o$(n,so)=o$(n,so)(y+1 TO g)+" "+o$(n,so)( T
O y): GO TO 1944
1948 NEXT g
1950 GO TO VAL "1944"
1954 CLS : PRINT AT oj,og;"SORTING, Line ";so: DIM
p$(12,VAL "31"): LET n=oa
1956 FOR n=n TO l5: PRINT AT 12,12;INT ((n/l5)*VAL
"100");"% Done": IF n=l5 THEN  RETURN
1958 FOR y=n+1 TO l5
1960 IF o$(n,so)>o$(y,so) THEN  FOR g=1 TO l2: LET
p$(g)=o$(n,g): LET o$(n,g)=o$(y,g): LET o$(y,g)=p$(
g): NEXT g: GO TO 1956
1962 NEXT y
1964 NEXT n: RETURN
1974 ON ERR  GO TO VAL "1975": DELETE VAL "1810",VA
L "1964": STOP
1975 ON ERR  RESET : RANDOMIZE USR ml: MERGE "Dbx.B
6": GO TO VAL "2001"
```

"var.B6" The Variable File Program

```
1977 CLEAR : DIM o$(1,1,1): DIM p$(1,1): LET so=1:
LET sl=2: LET l1=0: LET l2=0: LET l3=0: LET l5=0: F
OR n=1 TO 1: NEXT n: FOR y=1 TO 1: NEXT y: LET ml=1
00: LET m$="": LET ly=0: LET ss=0: LET ms=0: LET mx
=0: LET pc=0: LET m=0: LET oo=0: LET oa=1: LET ob=2
: LET oc=3: LET od=4: LET oe=5: LET of=6: LET og=7:
 LET oh=8: LET oi=9: LET oj=10: LET ok=11: LET ol=1
2: LET om=13: LET on=14: LET op=15: LET oq=16: LET
or=17: LET os=18: LET ot=19: LET ou=20: LET ov=21:
LET ow=22: LET ox=23: LET oy=24: LET oz=25: LET zo=
27: LET ps=1: LET pr=1: LET nn=0: LET ln=57: LET pg
=1: LET k1=1000: LET k2=2000
```

6 of 7 Pgs

"Init.B6" The LKDOS Initialization
And Print Style Menu Program

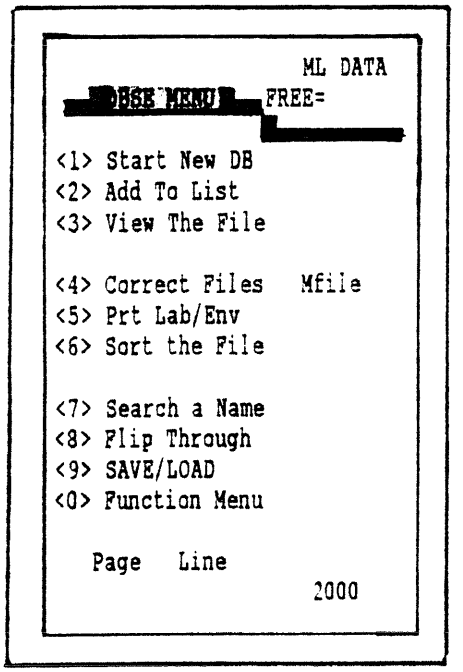
```
1993 BORDER NOT PI: PAPER NOT PI: INK VAL "7": CLS
: LET pg=NOT NOT PI: LET nn=NOT PI: PRINT AT og,od;
"Touch:"'''TAB od;"<1> For TS-2040 Printer"'''TAB od;
"<2> For Dot Matrix"'''TAB od;"<3> For Daisy Wheel";
AT os,og; INVERSE oa;"PRINTER MUST BE ON"; INVERSE
oo: PAUSE oo: LET z=CODE INKEY$-VAL "48": LET pr=z-
oa: IF z<oa OR z>ob THEN  GO TO VAL "1993"
1994 IF pr=oo THEN  RANDOMIZE USR ml: OPEN #og,"LP"
: RANDOMIZE USR ml: POKE VAL "16092",oo: GO TO VAL
"1999"
1995 CLS : PRINT AT oj,oc;"ELECT Type CPI:"'''TAB oc
;"[1] Aerco - Oliger"'''TAB oc;"[2] Tasman CPI"'''TAB
 oc;"[3] A & J CPI": PAUSE oo: LET a=CODE INKEY$-VA
L "49": PRINT #0;"Printer Need Line Feed? <y> <n>":
 PAUSE NOT PI: LET m$=INKEY$: LET b=(oj AND m$="y"
OR m$="Y")+(NOT PI AND m$<>"y" AND m$<>"Y"): RANDOM
IZE USR ml: OPEN #oc,"lp": RANDOMIZE USR ml: OPEN #
od,"dd": RANDOMIZE USR ml: POKE VAL "16096",a: RAND
OMIZE USR ml: POKE VAL "16090",VAL "138": RANDOMIZE
 USR ml: POKE VAL "16092",b: RANDOMIZE USR ml: POKE
 VAL "16094",NOT PI: OUT 127,24: LPRINT
1996 CLS : PRINT AT oj,oc;"SET PRINTER and Elect:"'
'TAB oc;"[1] Pica 10 Cpi"'''TAB oc;"[2] Elite 12 Cpi
"'''TAB oc;"[3] Micron 15 Cpi": PAUSE oo: LET ps=COD
E INKEY$-VAL "48": LET m=VAL "1997": GO SUB m: OUT
VAL "127",VAL "27": GO SUB m: OUT VAL "127",(VAL "8
0" AND ps=oa)+(VAL "77" AND ps=ob)+(VAL "103" AND p
s=oc): GO SUB m: OUT VAL "127",VAL "120": GO SUB m:
 OUT VAL "127",oa: GO TO VAL "1999"
1997 IF IN VAL "127"<>VAL "237" THEN  GO TO VAL "19
97"
1998 RETURN
```

"Menu.B1" The Program to Construct the Menu

```
9998 BORDER 0: PAPER 0: INK 7: CLS : PRINT AT 2,2
3;"ML DATA";AT 3,3;"[redacted]"; INVERSE 1;"DBSE MENU";
 INVERSE 0;"[redacted]FREE="'TAB 20;"[redacted]";AT 5,3
; INVERSE 1;"<1> Start New DB"; INVERSE 0'TAB 3;"
<2> Add Records"'TAB 3; INVERSE 1;"<3> View Recor
ds";AT 9,3;"<4> Correct File"; INVERSE 0;TAB 23;"
Mfile"'TAB 3;"<5> LPRINT File"'TAB 3; INVERSE 1;"
<6> Sort Records";AT 13,3;"<7> Search File "; INV
ERSE 0'TAB 3;"<8> Flip Through"'TAB 3; INVERSE 1;
"<9> SAVE / LOAD "; INVERSE 0'TAB 3;"<0> Function
 Menu"'''TAB 6;"Page";TAB 13;"Line": INK 5: PLOT 0
,0: DRAW 0,175: DRAW 255,0: DRAW 0,-175: DRAW -25
5,0: INK 3: PLOT 8,8: DRAW 0,159: DRAW 239,0: DRA
W 0,-159: DRAW -239,0: INK 7
9999 STOP : RANDOMIZE USR 100: SAVE "ml6.C1"SCREE
N$
```

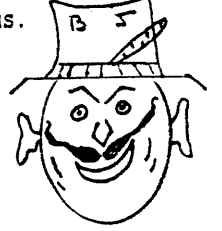
Key the two program lines in and then <RUN>. The Menu will present itself on screen and a STOP will occur. Examine the screen for balance and compare it with the printed screen below. If there is a report code or if the menu isn't shaped like the sample, then correct your work and <RUN> again. When satisfied, type <CONTINUE ENTER> and the menu will be SAVED to disk. After the SAVE you can delete the program, as we dont need it anymore. Or, if you wish you may SAVE it as a Basic help utitlity for constructing other menus.

The Menu Sample



The Menu Auto Loads as Needed

When we need the menu the program pulls it in from disk with a line command <RANDOMIZE USR 100: LOAD "ml6.C1"SCREEN$>. Such menus "left in RAM" would drastically depreciate the "record capacity" of the software. This one menu, if left in RAM, would reduce the number of records possible to generate by 24 records (A One line record of 31 characters). Some softwares NEED as many as 15 menus, which would cost up to 8K of memory to include. So, the practice in the past has been to "skimp on menus" and make the user REMEMBER what keys to use for a miriad of functions.



7 of 7 pgs

HOME OFFICE ON THE CHEAP

by Peter R. Hale

Cleaning computers, disk drives, printers and assorted electronic gear around the home office is recommended in every instruction manual but none describe how it to do it.

Any one using tractor feed paper in a printer knows how all those little paper dots flee to the most inaccessible spots. Well, the printer runs with all the garbage inside, so what?

In time, some one tells you that compressed air is just the thing, but not everyone keeps an air compressor in the home office.

When my salvaged Canon PC-25 photo copier began producing ever darker copies, I dreaded the need for major overhaul. I dropped by the local repair facility to learn there was dust on the most internal prismatic mirror and that for just $65.00 and a week in the shop it could be fixed.

Then I saw canned compressed air designed for just this and other similar purposes (like getting dust off the edges of the QL keyboard.)

Being a Sinclair computerist and by logical extension cheap, I was reluctant to buy it even after remembering the repair shop's estimate. I thought that $9.95 (plus tax) was a bit much for what is essentially a free item.

So I made an office air compressor from scrap.

You can, too. You need:

1) a strong, plastic bottle with a flip top cap, such as shampoo, baby oil, ketchup, etc., have.

2) the feed tube, such as comes with trigger operated cleaning sprays like 409 or Fantastic or similar containers. These are long, flexible and have a fine internal diameter.

3) a sharp pointed knife to ream out the hole in the cap so that the tube fits though the hole.

4) a bit of cement to seal the tube in the hole.

Squeeze the bottle. A powerful jet is created. The container can be held in any position and the tube guided to whereever it needs to go.

It blows dust from between keys, paper dots from corners of printers and it got the dust off that deeply hidden prismatic mirror that had been darkening my copies.

HOME OFFICE ON THE CHEAP, Part 2

by Peter R. Hale

The printer is one of the more difficult pieces of computer equipment to locate in the home office. Its too big to fit on most desks, and on other tables it wastes space.

There are printer cabinets in various sizes and shapes that let you place tractor feed paper on below the printer, but they are bulky and expensive, and there is a lot of waste space.

The typical home office printer does not need a lot of space. It's foot print is generally under 12" deep and is less than 18" wide.

An inexpensive compromize is to modify a 30" or 36", double-doored kitchen wall cabinet into a printer cabinet.

Such cabinets may be found cheap at bargain outlets, or even cheaper when a friend or neighbor remodels a kitchen.

The internal depth is generally a full 12" so that 11" tractor feed paper fits on a shelf. The cabinet may even have adjustable shelves.

Modifying the cabinet requires a 1x12 board as long as the cabinet and a piece 3/4" moulding of equal length. These should be attached to the top of the cabinet with screws or (heaven forbid!) nails. Pre-drill holes as the cabinet wood/particle board may otherwise split.

For mobility, four casters can be attached to the bottom, but this is gilding the lily as they will cost almost as much as the rest of the materials, including the cabinet.

Using a saber or key hole saw, cut a two inch wide slot about 12" long in the rear of the cabinet near the top for the paper to pass through to a rear feed tractor. Alternatively, a bottom feed printer can be accomodated with a similar slot in the top shelf. Maybe have both.

The shelf can accomodate a full 2750 sheet carton of tractor feed paper and there is space for all kinds of other items to be stored, out of sight and dust.

There are several advantages to this design. The movement of the printer head is side to side, so the 30" or 36" width cabinet provides great stability. Second, the depth of the cabinet is minimally intrusive into the room. Finally, it is very cheap!

Ed Grey
Enterprises
P.O. Box #2186, Inglewood, CA 90305
Info >> 213-759-7406 << Order

** SPECIAL THANKS **

TO
BILL JONES

for EXCELLENCE in UPDATE MAGAZINE

and Best Wishes to

FRANK DAVIS

In This NEW Challenge!

****
****

Timex/SINCLAIR & CP/M Products

Both NEW and USED Software and Hardware

For current CATALOG send SASE (with 45 cents postage) and $1.00 for each Timex/SINCLAIR and/or CP/M catalog.

Call The Grey Matter BBS & RCP/M (213-971-6260)

PRINTING MACROS

Printers are all different, but most have more capabilities than we are usually capable of taking advantage of. My KXP-1124 has hundreds of functions that can be programmed. I've been looking for a practical use of the "Superscript and Subscript" modes of my printer and have decided that these modes need some user adjustments. This article is done with such adjustments of the subscript style. Actually it is the "ELITE" style that gives 96 characters per page width PLUS Subscript. Now that would give you characters that are Smaller, but with no increase in the normal amount of text on page. Not much use for that, eh?

To take advantage of a printer's capability it is useful to set-up "macros" of programming that can be MERGED to printing softwares when needed. This is a group of four macros to produce a large amount of printing on a page in a "macro" style that produces fairly comfortable reading. We have the following set-up done in a program line: SET ELITE style + SET SUBSCRIPT. The result is a 2/3 print HEIGHT that has NORMAL LINE SPACING, which is too much spacing between lines for the reduced height of the characters". So, we add programming to reduce the LINE SPACING from the normal "3/16 inch" to "one-tenth inch spacing. This "Sounds" all too complicated, but it isn't.

Next, we need the ability to print Larger Headers. A header in small print is not very attention getting. So, for Headers we create a little utility that: EXPANDS HEIGHT and EXPANDS WIDTH. All we want is a "one time" expansion for headers as needed. Then the utility should RETURN to the pre-established print style. AND, the header should be CENTERED over the PRINT COLUMN.

ALSO, we need to SCHEDULE the number of lines to print on a page. Since we have reduced the spacing between lines, more lines can be printed on a page. If my math serves me right, a normal line of 60 lines per page (at 3/16" spacing) should produce 112 lines at .1" line spacing. So, substract about 6 lines for top and bottom margins and we get a figure of 100 lines for a formatted page.

The Variables used will be: LX=Maximum characters per line: LL=Column Line Length: TB=LEFT TAB: LN=Lines per page. Now follows are the four coordinated utilities that can be exercised by GO SUB from within an operating program.

The ENTRY POINTS for GO SUB: <GO SUB 2610> gets Elite STYLE, plus SUBSCRIPT. <GO SUB 2612> Sets .1" line spacing. <GO SUB 2616> is for larger HEADERS, setting EXPAND HEIGHT and EXPAND WIDTH. <GO SUB 2630 within the line> reduces QQ column center and prints the Header. Line 2610 re-sets QQ and CANCELS Double Height and Double Width. This column of printing is the result of using these utilities with the Daisy.B6 Software. The reproduction is in actual size. My next Article "DbseX.B6" will be done with Daisy and a different "Print Macro".

Many subscribers do not like such small print. Then, Just use the EXPANDED heigth and width as is done in line #2616. This would give an improvement of "Readers Digest Large Print edition". Individual "macros", such as these can be Saved to disk and MERGED as needed to set up special printing styles and formats. Name this set "Micro Columns".

Micro Columns

Four Printing Macros

```
2610 LET a=VAL "2620": GO SUB a:
 OUT 127,27: GO SUB a: OUT 127,7
7: OUT 127,27: GO SUB a: OUT 127
,83: GO SUB a: OUT 127,1: REM se
t ELITE+Subscrpt
2612 GO SUB a: OUT 127,27: GO SU
B a: OUT 127,43: GO SUB a: OUT 1
27,36: LET tb=6: LET ln=100: LET
 lx=42: LET ll=42: LET qq=28: RE
M set .1" line space - 100 Lines
 per page + vbls
2614 RETURN
2616 LET qq=12: LET a=VAL "2620"
: GO SUB a: OUT 127,27: GO SUB a
: OUT 127,87: GO SUB a: OUT 127,
1: GO SUB a: OUT 127,27: GO SUB
a: OUT 127,119: GO SUB a: OUT 12
7,1: GO SUB 2624: REM set Expand
 height+Expand width+Col Center+
get header print.
2618 LET qq=28: GO SUB a: OUT 12
7,27: GO SUB a: OUT 127,87: GO S
UB a: OUT 127,0: GO SUB a: OUT 1
27,27: GO SUB a: OUT 127,119: GO
 SUB a: OUT 127,0: REM re-set Co
l center, cancel dbl hi, cancel
dbl wide.
2619 RETURN
2620 IF IN 127<>VAL "237" THEN
GO TO a: REM check READY status
of printer, wait until ready.
2622 LPRINT TAB qq-LEN m$/2;m$:
LET nn=nn+1: RETURN : REM print
the header
2624 INPUT m$: RETURN : REM inpu
t the header
```

DONATED TO PUBLIC DOMAIN
Bill Jones

********** CORRECTION **********
********** OTHER USES FOR LARKEN RAMDISK **********
********** ARTICLE APRIL 1990 **********
****

Most humble apologies to anyone who tried to create a BASIC AROS by following my instructions. The following corrections will remove some of the frustration: 1. The LOAD statements in two of the sections of the article will need to have RANDOMIZE USR 100 added. 2. In the "CREATING AN AROS" section: line 30 should have FOR X= 32768 not 62768. The instruction in para 6 should read: OUT 244, 240: RANDOMIZE USR 100: LOAD "test.C1" CODE 32776: OUT 244,0. Once again, sorry!! Larry C

****

SCREEN MACROS
The "macro.B6" Program

A "Macro" is defined as a special bit of programming that provides an often needed function. Generally the term "macro" is used with Desk Top Publishing and are associated with specially constructed screens. In this case we use a macro for a special text file. We can create as many macros as we want to and at NO COST of Free Memory. AND, we can print the macros any time we want to by LOADING or MERGING this program.

The program uses managed INPUTs of up to 22 Screen Lines to allow you to input formatted lines. Then when the last line is input there are choices to: PRINT, SAVE, or QUIT. The Screen is SAVED as a special Macro Screen with a file name. Later, at any time, you can re-load the screen (with the "scopy.B6" program for print out. Or, this program will itself allow you to PRINT the macro, either individually, or mixed within a text file that you may be printing.

The Screen holds up to 704 characters, which is a nice size group. Screens are SAVED to disk, using only two blocks of disk space. When they are recalled for print out there is no cost at all of Free Memory.

The screen can be considered as a "data base" contained in a "high resolution String". The screen is printed out in two ways, by High Res isolation of "pixel" data, and in this case, by low resolution isolation of screen Characters. The data is organized "as in a character array" like <DIM A$(22,31) >. But the screen coordinates begin at zero. So to print a screen character in the top left corner: < PRINT SCREEN$ (0,0) >. We can pick off and LINE PRINT to paper all characters in a screen with counters, as: < For N=0 TO 21: FOR Y=0 TO 31: LPRINT SCREEN$ (n,y);: NEXT Y: LPRINT: NEXT N >.

Given below the listing is a sample print out of one of the Screen Macros used with the Daisy.B6 software.

Line #2444 contains a suggested way for page control while the macro is being printed. Var "nn" counts the lines and IF nn>=LN a page end routine is called by <GO SUB pe>. "LN" in this case is the pre-set number of lines per page. A typical Page End routine would be: >Line pe< LPRINT'''TAB 40;pg: OUT 127,12: LET nn=0: LET pg=pg+1: RETURN >. Enjoy! Bill Jones.

"macro.B6" Create / PRINT / SAVE Screen Macros

Donated to Public Domain by Bill Jones

```
2442 CLS : INPUT "[1] Create Macro [2] PRINT MACR
O[3] SAVE [4] Quit";z: GO TO (z<1 OR z>4)*2442+(z
=1)*2448+(z=2)*2444+(z=3)*2460+(z=4)*2462
2444 INPUT "Enter TAB";tb: FOR n=0 TO 21: LPRINT
TAB tb;p$(n+1): REM  LET nn=nn+1: IF nn>=ln THEN
 GO SUB pe
2446 NEXT n: GO TO 2442
2448 CLS : PRINT AT 5,2;"Build a Screen Macro"'''
TAB 2;"You may input up to 22 Lines."'TAB 2;"The
 Screen File will be SAVED"'TAB 2;"with the file
name that you     input, with a ""."C4"" extension
."'TAB 2;"ENTER WHEN READY": PAUSE 0: CLS
2450 DIM p$(22,31): FOR n=1 TO 22: PRINT AT 21,0;
 INVERSE 1;"Input Line ";n; INVERSE 0: INPUT p$(n
): PRINT AT 21,0;"
";AT n-1,0;p$(n)
2452 IF n=22 THEN  GO TO 2458
2454 NEXT n
2458 FOR n=1 TO 22: PRINT AT n-1,0;p$(n): NEXT n:
 PRINT #0; INVERSE 1;"Touch a Key for Menu": PAUS
E 0: GO TO 2442
2460 INPUT "DD # to Save to?";z: RANDOMIZE USR 10
0: GO TO z: INPUT "Input Name of Macro";w$: LET w
$=w$+".C4": RANDOMIZE USR 100: SAVE w$SCREEN$ : C
LS : GO TO 2442
2461 DIM p$(1): INPUT "Enter Program Disk #";z: R
ANDOMIZE USR 100: GO TO z: GO TO VAL "2462"
2462 STOP : REM ** Install your GO TO Line here.
```

1. This is a SCREEN MACRO
2. Selectable at the Quickie and Office Tools Menus.
4. Up to 24 Lines may be input
5. Macros are saved as SCREENS
6. The ".C4" Extension is used
7. A Macro Printer is included
8. (The "scopy.6" program.
10. Printing is to TAB control
11. Macro Printing with scopy can be scheduled to print the macros mixed with text.
13. Line length limit is 32, but the next line can be used for wrap around.
16. The macros are not in RAM memory, so you may create as many as you like in disk
20. A Screen holds up to 704 characters, which is a nice little data chunk.

Copy Screens with "scopy.B6"

Donated to Public Domain

By: Bill Jones

Now that we have a nice program to create Screen Macros, lets key in another to PRINT SCREENS "From Disk TO PAPER". ""scopy.B6"" is a program to do that. "scopy.B6" is a utility used in the Daisy software to selectively print Screen files in disk to be included "within text file print-out". You can use scopy.B6 to do that, or to load and use to print any disk Screen File. The program ignores any graphics or drawings on screen and prints just text data, making it useful to insert "screen macros" into a text file being printed.

There are two modes controlled by variable "gg". One mode is to Print Disk Screen Files from a Menu presented at the bottom of a Catalog. The other mode is to Automatically print a disk file "within the print out of another text file". If gg=0 the menu is presented. But if gg=1 the menu is by-passed and the screen file name that is placed into "Z$" is gotten from disk for print out.

The way to print a macro "automatically" from disk "withing a text file" is to insert the programming: <LET gg=1: LET z$="name of file in disk": LET tb=TAB: GO SUB 7003>. The lines of the screen file will print to the "existing TAB" setting of the host software, using "tb=TAB". Example: if your text were being printed at TAB 10, then LET tb=10. You can insert other controls in the Command Program given above, such as to count lines and execute a page end if the text reaches the last formatted line of the page.

Example: Line # 7070 contains the programming: LET nn=nn+1: IF nn>= LN THEN GO SUB pe. In this case "nn" is a line counter, "LN" is the LINES PER PAGE, and "pe" is a PAGE END routine to end the page and start a new page.

If just wanting to print one or more Screen Files (macros) then the way to do that is: Type LET gg=0: GO TO 7003. Then the prompts take over. First you are prompted for the Disk # to reach the Screen Files. Then a Catalog is placed on screen with a Prompt at bottom to "Input File Name". The file is gotten from disk and printed to paper. Then you are prompted to do another or to Quit.

The "scopy.B6" Listing

The line and page controls are closed by REM, Line #7070

```
7003 IF gg=1 THEN  GO TO VAL "7010"
7004 CLS : PRINT AT 10,0;"This is a Screen Copy U
tility"'''"Prints Screen Files from Disk"'''''"<1>
Print a Screen File"'''"<2> Quit": INPUT a: IF a=2
 THEN  GO TO VAL "7092"
7006 CLS : INPUT "Input left Margin?";tb
7008 INPUT "ENTER Screen Disk #";dd: RANDOMIZE US
R 100: GO TO dd: RANDOMIZE USR 100: CAT ".C↑",: I
NPUT "Input Whole Title of Screen File";z$
7010 RANDOMIZE USR 100: LOAD z$SCREEN$
7020 FOR n=0 TO 21: LPRINT TAB tb-1;" ";
7030 FOR y=0 TO 31
7040 LET a=CODE SCREEN$ (n,y)
7050 IF a>122 OR a<32 THEN  LPRINT " ";
7060 IF a<122 AND a>31 THEN  LPRINT CHR$ a;
7070 NEXT y: LPRINT : LET nn=nn+1: IF nn>=ln THEN
  GO SUB pe
7080 NEXT n
7082 INPUT "<1> Another Screen                <2> Q
uit";a: IF a=2 THEN  GO TO 7092
7090 PRINT #0;"Adjust Printer, then ENTER": PAUSE
 0: GO TO 7008
```

(TAB MUST Be 1 OR more)

Sample Screen Print Out, of one of the Daisy Menus.

```
        QUICKIE MENU
        Touch Select

  1. Typing  2. Re-ed 3. Store
  4. HELP    5. DELETE BUFFER
  6. View/Edit Array Cells
  7. Word Replace, Global
  8. Input to L$ for PASTE-UP
  9. Cycle Caps 0 Edit Seq DB
  [:] Zap a DB  [;] MACROS
  [<] LOAD DB     [=] SAVE DB

            Info
  Buffer=       Cell Limit=
  H-0  FREE=          L$=
  I-0  Touch Key to Select
```

April 23, 1990

John E. Juergens
18 Bryce Canyon Way
Pacifica, CA 94044

Interesting Letter with a way to SLAVE A QL to A QL -

Bill Jones
c/o UPDATE Magazine
1317 Stratford Avenue
Panama City, FL 32404

Dear Bill,

It was good to hear from you again. Sorry to hear about whatever caused your hospitalization but hope that it's over for good. Believe it or not did not even particularly notice UPDATE's lateness. That's surely a sign of addle-patedness or very busy-ness on my part.

Your call for co-processing and Tim Stoddard's Trump Card's Toolkit 2 article combined in my mind to produce some experimentation. I can't recall seeing an article on the QL network device other than Tim Stoddard's recent treatment. The results, attached, may or may not be of interest to the bretheren (and cistern ?) I persist in this despite numerous beatings about the ears by our many female offspring. They would call that remark sexist; I think it's funny and harmless. When will we ever be able to agree again?

As to slave co-processing on the QL, ie. beyond the few transparent efforts built-in and assigned to the MAB whats-its-number, termed a "redirectable I/O system:" Personally, I've yet to see a need for more. You, on the other hand, sound as if you want a Cray on the desk.

Do even a small number of computerists (pretty high-toned, huh?) wish to have a dialogue with Compuserve while doing a document? And, if so, why not use Taskmaster or the poor-man's multi-tasker that you sent me?

The end result of the notes, attached, sort of point in the direction of slave co-processing but, for the life of me, doubt that I can come up with a practical use for same. Perhaps what we (I) need is a seminar in "Uses". If there's a "use" I can sometimes come up with a "way."

Have often thought about the ZX-81 and T/S 1000 sitting in a box in our garage, along with the 2-64K and 1-16K memory paks, a Byte Back RS 232 and Hunter board, and wondered what could they help me do that I cannot already do reasonably well with just the QL? That question has been unanswered for the many years now that we've had our QL(s).
The connection(s) is/are there: the RS 232 ports! All that remains is the programming and that is doable. But that statement, for me at least, still evades the first and necessary question: What to do with it once it's hooked up?

Even with my limited programming skills, was able to re-do Psion's VU-Calc for the ZX-81 and make it print out the whole spreadsheet to an 8 1/2 by 11 sheet of paper, 12 labled columns across by 26 labled rows, to the Gemini 10X. And, for a number of years, this worked well for our only use for same, a rather complete Federal and State estimated and final tax program; even with the 40 formulae limitation imposed by VU Calc. However, once I temporarily got my hands on Lotus 1-2-3 (in Rom on a PC Jr yet), my VU-Calc-is-just-fine mentality changed forever. Later, when I found I could get a QL AND an almost-Lotus-alike for less than the price of Lotus itself, we made the investment.

But my point is, finally, that our QL purchase obsoleted (if that really is a verb) our ZX-T/S and related equipment as far as our use is concerned. After considerable (and continuing) thought I simply cannot find a use for the old stuff that cannot be done better or at least as well by/on the QL. Nor can I envision an auxilliary (slave and/or co-processing) use that would NOT be more trouble than it's worth. Don't wish to be heard to say there isn't one, just that I've been unable to come up with one. Hence, my suggestion, above, for a "Use" seminar.

Bill, the attached is pretty wordy, "Botts-like" if you will, so wont take it amiss if you don't use it.

The QL Network device without ToolKit 2
(an exercize in patience, perseverence and humility)

John Juergens

For the several or more years we've owned a QL I've been aware of the fact that it's supposed to be able to "network." However, because we had not perceived a use for network-ing and the paucity of articles about it the device was never explored. Tim Stoddard, via his article on the Trump Card Users' Guide, (UPDATE, Jan 1990), and Bill Jones' "slave co-processing" article of the same issue, prompted my curiosity. Stoddard mentioned that a number of bugs in the QL Rom affect network-ing adversely; he neglected to mention that there are a number of bugs in the User Manual as well, equally adverse-affecting. However, if one does not know how a thing works to begin with bugs are not the big problem. Anyway, my curiosity continued and resulted in the following:

First, some QL User Guide references used: INTRODUCTION, p. 11; KEYWORDS: Copy, Load, Net, and Save (which is erroneous and should read SAVE neto_3); and CONCEPTS, pgs. 17 and 34.

What can be done with NETWORK without ToolKit 2? Probably, much more than follows but at least one may, to a limited degree, manipulate one QL from another.

Conditions: QL#1 is the master, ie. the keyboard of which you are operating, and QL#2 is the slave. Both QL#1 and QL#2 are connected via the net plugs, power is on and running in Superbasic ie. F1 or F2 have been pressed with NO cartridges or disks in drive #1. The following commands have been given to QL#2: Type net 2 : lrun neti_1 and press ENTER. Now, put cartridges and/or disks in QL#2 and go to the keyboard of QL#1.

To Print to the monitor or TV attached to QL#2 type the following program into QL#1:

```
9  REMark  This is Program 1
10 a$="ABCDEF"
20 FOR i=1 to 100
30 PRINT a$;
40 NEXT i
50 LRUN neti_1

100 NET 1:SAVE neto_2, 10 to 50
```

Now type into QL#1: run 100 and watch "ABCDEF" appear 100 times on the left side of the monitor attached to QL#2 or the single screen of a TV attached to QL#2.

What has happened? First, when lrun neti_1 was typed into QL#2 you may have noticed the screen move as if clearing. You told QL#2 to "listen" for a program coming in FROM net 1. You could instead have commanded QL#2 to lrun neti_2, ie. "listen" for its own station number, NET 2, and the result would have been the same.
Secondly, the small program, statement lines 10 to 50, was NOT intended to be

RUN on QL#1 even though it appears on the screen attached to QL#1. It was intended to be RUN on QL#2 and was sent to QL#2 via the SAVE statement on line 100. The SAVE statement used could as easily have SAVEd lines 10 thru 50 on QL#1 to a microdrive cartridge or a floppy disk if line 100 were changed to: 100 SAVE mdv1_Test, 10 to 50. Notice one difference however: The original Line 100 did NOT use nor require a file name while SAVEing to mdv or flp does. Why? I'm not certain but it seems reasonable that using NET is NEVER a permanent storage facility, it's more like an intermediate, temporary transmitting device than a tape or disk, each of which is designed to store data later to be retrieved.

Why did QL#2 treat lines 10 to 50 on QL#1 as a program and RUN it? Recall the original command given to QL#2: lrun neti_1. That told QL#2 to listen to the net, (neti(n)), particularly for something coming from NET 1, treat it as a program and LOAD and RUN (lrun) it, which it did. If you had commanded QL#2 to listen for its own station number, NET 2, eg. lrun neti_2, it would have responded in exactly the same way BUT it would NOT have had to know in ADVANCE WHICH station was sending to it in order to get the message; in this case, lines 10 thru 50. The usefullness of listening to its own station number becomes apparent with more than 2 QLs on the network.

It should be mentioned here that the Users' Guide, CONCEPTS, p. 34, talks about neto_0 (ZERO) and neti_0. I could not get either to work.

Why put in statement line 50? Because we wish to continue to keep QL#2 in abject slavery to QL#1. If line 50 were not there QL#2 would have printed "ABCDEF" 100 times and then gone off the network. To keep it listening, always put in the statement contained in line 50. At this point go to the QL#2 keyboard and press and hold CTRL while pressing the SPACE bar, then release both; this is a BREAK and will release QL#2 from listening to the network. You may have to type cls #2:edit 10:ENTER:ENTER to see lines 10 to 50 displayed on the screen attached to QL#2's but they are there, meaning that QL#2 actually did LOAD (and run) the program originally typed into QL#1. The program in QL#2 should differ from the one in QL#1 only in that line 100 should NOT be present in QL#2. Why? Because line 100 only transmitted (SAVEd) lines 10 thru 50 to QL#2.

Is there another way to accomplish the same task? Not quite the same but similar. It is mentioned only to demonstrate another point. Since the CTRL-SPACE keypresses have interrupted QL#2's net listening, type load neti_2 and ENTER it into QL#2. Note that Net 2 does not require re-entering as once done it remains until a power-off or reset occurs. Now, type into QL#1 the following program:

```
9  REMark  This is Program 2
10 OPEN #5,neto_2 : REMark  Lines 10, 20, 40 & 60 are for QL#1 only NOT QL#2
20 for i=1 to 100
30 print #5,'print"ABCDEF";' :  REMark  Only print "ABCDEF" goes to QL#2
40 NEXT i
50 print #5,'LOAD neti_2'  :    REMark  Only LOAD neti_2 goes to QL#2
60 CLOSE #5
```

and type RUN and ENTER it.

The main point to the above is that QL#2, having been given the LOAD command, load neti_2, is put into a condition in which its expected input has been switched from the keyboard (type print "ABCDEF" and it prints it to the right screen, channel #1 if viewing on a monitor or the only screen if viewing with a TV but still channel #1) to another source of input; namely, a file or in this case what may be coming down the network.

When, for example, we have a program called BOOT on microdrive or disk, and either is in drive 1, and we type, in the Command mode (our typing appears at the bottom of the screen) load mdv1_BOOT (or load flp1_BOOT) we expect to see, after the drive stops and typing edit 1 and ENTER, the program BOOT on the screen with the statement line numbers arranged in ascending order.

What was NOT immediately apparent was that the input into the QL was turned over momentarily FROM the keyboard TO what was contained in the designated file, mdv1_BOOT. If BOOT had contained NO statement numbers (and we can prepare such a program in Quill, SAVE it to mdv1 as Test_doc, and LOAD it via load mdv1_Test_doc) the QL would have RUN the non-numbered statement lines just as if they had been individually typed-in from the keyboard in command mode and ENTERed. Remember, do NOT LRUN such a program, only LOAD it. The RUNing will be automatic.

This type of input has its limitations. If there are no statement line numbers there cannot be any GoTos, GoSubs or, probably, any PROCedures. And, if one decides to use it one must be careful that each line statement is correct and stands alone. However, that being said, the QL WILL execute such a "program" and when it's done leave nothing to be observed in the way of a program on the screen or in memory except for variables which may have been initialized, AND it is very fast-running.

Back to Program 2: Lines 30 and 50 sort of require a split personality to understand. However, they represent one of the two ways to use the network facility. The first "print" in lines 30 and 50 tells QL#1 to send the remainder of the statement line only to QL#2, ie. don't print to QL#1's screen. When QL#2 gets these it acts on them just as if they were typed into QL#2's keyboard in the command mode. The difference between Program 1 and 2 is that Program 2 requires QL#1 to do the FOR i loop and to send "ABCDEF" 100 times down the network to QL#2. Don't bother with a BREAK, CLS #2, EDIT 1 routine on QL#2 because there's no program behind the 100 "ABCDEF"s; not even Program 1 because a LOAD or LRUN clears out any existing program. Take a peek if you wish but there should not be anything there.

OPEN a channel and DIR a drive ON QL#2 FROM QL#1:

Send a Program to QL#2
first ENTER lrun neti_2 on QL#2

```
10 REMark  This is Program 3
20 OPEN #5,neto_1
30 DIR #5,mdv1_ (or flp1 or ram1)
40 CLOSE #5
50 LRUN neti_2
100 SAVE neto_2, 10 to 50
110 OPEN #5,neti_2
120 REPeat loop
130 INPUT #5,a$:PRINT a$
140 IF EOF(#5) THEN EXIT loop
150 END REPeat loop
160 CLOSE #5
```

Send ONLY instructions to QL#2
first ENTER load neti_2 on QL#2

```
10 Remark  This is Program 4
20 OPEN #5, neto_2
30 PRINT #5,'OPEN #6,neto_2'
40 PRINT #5,'DIR #6,mdv2_'
50 PRINT #5,'CLOSE #6'\'load neti_2'
100 RUN 20 to 60
110 CLOSE #5:OPEN #5, neti 2
120 REPeat loop
130 INPUT #5,a$:PRINT a$
140 IF EOF(#5) THEN EXIT loop
150 END REPeat loop
160 CLOSE #5
```

Start both Program 3 & 4 with the command: GoTo (or RUN) 100
NB. Program 4, line 50, the backslash outside the single-quote marks sends a Line Feed to QL#2 just as in an ordinary PRINT statement in Superbasic. The LF is needed by QL#2 to execute the CLOSE #6 statement. This is similar to the ENTER if you were typing in the line on the keyboard of QL#2. In the cases of lines 30 thru 50, the fact that a semi-colon is missing from the end of each statement line also ensures that a Line Feed will be sent after each line.

Load and Run QL#1 with a BOOT program residing on a drive in QL#2:
ENTER on QL#2: load neto_1 if it is not so running.
ENTER on QL#1: open #5,neto_2:print #5,'copy_n mdv1_ to neto_1':print #5,'load neto_1:close #5:lrun neti_1
NB. If copy is used instead of copy_n, QL#1 will stop with a "not found" error report. Copy_n gets rid of the file header sending only the file data.

LOAD QL#1 with a Psion Program (QLWP, QLSS, QLDB or QLBG) residing on a drive controlled by QL#2:
ENTER on QL#2: load neti_2 command
ENTER on QL#1: open #5,neto_2:print #5,'copy mdv1_QLSS to neto_1': print #5,'load neto_1': close #5: exec_w neti_1

Once running in either Abacus or Quill commands may still be sent to QL#2 provided that you "left it listening", either with lrun neti_2 or load neti_2, by typing in the instructions to Quill or a cell in Abacus and printing them to neto_1, eg. F3,P,ENTER for display,ENTER range,N for no border,F for file, neto_2 for Abacus. Remember, no margins, headers, footers, etc. in Quill and printing only the range in Abacus which contains the instructions. The other condition/limitation is that you must use the specially prepared printer driver explained by Tim Stoddard in his article. I cannot see much use for this but perhaps someone else may as I've not been able to figure a way to bring in data to Quill or Abacus other than to load a file on a drive attached to QL#2.

Am sure the above are not the only ways in which network can be used and I'm equally certain that they may not even be the best. However, they are a starting point for those interested in going further.

D. - base-1 FOR TOS

```
6800 CLEAR 63423: LET AB=2: LET AD=4: LET AE=5: LET AF=6: LET AG
=7: LET BO=10: LET DH=48: LET YF=256: LET M1=6804: LET M2=7020:
LET D1=7232: LET D2=7256: LET TS=9920: LET ZB=9922
6802 POKE 23730,191: POKE 23731,247: GO SUB 9920: DIM N$(5,31):
INPUT "LINE 1 OF YOUR LETTER HEADING";N$(1): INPUT "YOUR STREET
ADDRESS";N$(2): INPUT "YOUR CITY AND ZIP CODE";N$(3): INPUT "YOU
R NAME";N$(4): LET N$(5)="Sincerely,": ON ERR GO TO 6804: DELETE
 6800,6802
6804 ON ERR RESET : POKE VAL "23730",VAL "191": POKE VAL "23731"
,VAL "247": GO SUB TS: BORDER SIN PI: PAPER SIN PI: INK AG: CLS
: PRINT AT AB,VAL "12";"DBASE-1 MENU"''TAB AD;"<1> CREATE FILE"'
'TAB AD;"<2> ADD RECORD"''TAB AD;"<3> VIEW OR PRINT RECORDS"''TA
B AD;"<4> CORRECT RECORDS"''TAB AD;"<5> SAVE FILE"''TAB AD;"<6>
LOAD FILE"''TAB AD;"<7> SORT AND MANAGE"''TAB AD;"<8> TO BASIC"'
'TAB AD;"<9> LETTERHEAD OR SIGN OFF";#SIN PI;AT SIN PI,SIN PI;"T
OUCH SELECTION": PAUSE SIN PI: LET Z=CODE INKEY$-DH
6808 GO TO (Z<SGN PI OR Z>VAL "9")*VAL "6804"+(Z=SGN PI)*VAL "68
12"+(Z=AB OR Z=AD)*VAL "6824"+(Z=INT PI)*VAL "7440"+(Z=AE)*VAL "
7264"+(Z=AF)*VAL "7272"+(Z=AG)*M2+(Z=VAL "8")*VAL "9999"+(Z=VAL
"9")*VAL "7530"
6812 CLS : PRINT AT BO,AF; FLASH SGN PI;"CONFIRM, START NEW"; FL
ASH SIN PI'';TAB AF;"<Y> YES -OR- <N> NO": PAUSE SIN PI: LET Z=C
ODE INKEY$-DH: IF Z=VAL "30" OR Z=VAL "62" THEN GO TO M1
6814 CLS : DIM O$(SGN PI,SGN PI): INPUT "NBR OF RECORDS(EVENLY D
IVISABLE BY 4)";L1: INPUT "NBR OF LINES";L2: INPUT "CHRS PER LIN
E";L3: IF L1*L2*L3+L1/AD*L2*L3+VAL "600"> FREE OR INT (L1/AD)<>L
1/AD THEN BEEP VAL ".5",VAL "30": GO TO VAL "6814"
6818 DIM O$(L1,L2,L3): LET M1=SIN PI: GO TO M1
6824 CLS : INPUT "START AT RECORD #?";MS: INPUT "END RECORD #?";
MX
6830 FOR O=MS TO MX
6834 IF Z=AD THEN GO TO VAL "6850"
6838 IF Z=AB THEN GO TO VAL "6864"
6840 NEXT O: IF O>MX THEN GO TO 6804
6842 STOP
6850 ON ERR GO TO VAL "7032": PRINT AT AB,SIN PI;"RECORD # ";O':
 FOR N=SGN PI TO L2: PRINT "LINE ";N'O$(O,N)': NEXT N: PRINT ''"
O.K. ? <1> YES <2> NO <3> QUIT": PAUSE SIN PI: LET Q=CODE INKEY$
-DH: IF Q=AB THEN GO TO VAL "6858"
6854 IF Q=INT PI OR Q<SGN PI OR Q>INT PI THEN GO TO M1
6856 IF Q=SGN PI THEN CLS : GO TO VAL "6840"
6858 INPUT "ENTER LINE NBR TO CORRECT";QL: INPUT "KEY IN THE NEW
 LINE              ";Y$: LET O$(O,QL)=Y$: CLS : GO TO VAL "6850"
6864 CLS : ON ERR GO TO VAL "7032": FOR N=SGN PI TO L2: IF N>VAL
 "9" THEN INPUT "RECORD NBR "+STR$ O'+"KEY IN LINE # "+STR$ N+"
OR ZZ TO QUIT ";Y$: GO TO VAL "6867"
6866 INPUT "RECORD NBR "+STR$ O'+"KEY IN LINE # "+STR$ N+" OR ZZ
 TO QUIT  ";Y$
6867 IF Y$="ZZ" OR Y$="zz" THEN LET N=L2: LET O=MX: GO TO M1
6868 LET O$(O,N)=Y$
6872 PRINT AT VAL "19",SIN PI;"LINE ";N'O$(O,N)': RANDOMIZE USR
VAL "2361"
6876 NEXT N
```

1 of 6

REVISED BY EARL DUNNINGTON FROM THE LKDOS VERSION BY BILL JONES, PUBLISHED IN THE JULY 89 ISSUE, PAGE 27. THIS VERSION REQUIRES THE TOS DISK OPERATING SYSTEM, THE OLIGER OR AERCO CENTRONICS INTERFACE, AND THE ZEBRA OS-64 CARTRIDGE.

At this point, you should read the article by Bill Jones that accompanied his program.

Due to the many options available in this program, I have spent a great deal of time debugging it. I hope that I have removed all of them, but this I cannot guarantee. In some cases only use by many people over a long period of time, can uncover all of the bugs. This version of the program was designed for a Smith-Corona Fastex 80 dot matrix printer using Pica pitch. I do not think that this program will require any changes due to the use of a different printer.

In order to conserve memory, all REM statements have been removed and several sections were reprogrammed. The same series of line numbers for each task were used. Hidden variables and strings are used. Due to reprogramming of the ADD A RECORD section, there is no longer any prohibition against having more than 12 lines in a record. This version of the program will handle a 80,7,31 file, however the number of records in the file must be evenly divisable by four. This is so that the file can be divided into four parts for use in other programs such as DAISY.B6.

All screen displays are 32 character and are switched to 64 characters for LPRINT operations using Bill Jone's switching subroutines in lines 9920 and 9922.

```
6880 GO TO VAL "6840"
7020 BORDER SIN PI: INK AG: PAPER SIN PI: CLS : PRINT AT AB,SIN
PI;"FILE MANAGEMENT OPTIONS"''"1> REMOVE REPEATS"''"2> REMOVE SKI
PS"''"3> MOVE ZIP TO LINE 4"''"4> SORT DATA BASE"''"5> BREAK LG FI
LE TO 4 FILES        AND SAVE"''"6> COMBINE 4 SMALL DISK FILES
   TO SORT"''"7> SAVE A SORTED FILE"''"8> LOAD A LARGE FILE"''"9>
CLEAR MEMORY"''"0> PRESENT KEY RECORDS                (LINE 7 KEY)"
''":> VIEW OR PRINT RECORDS"";> TO DBASE1 MENU": INPUT "INPUT CH
OICE";Z$: LET Z=CODE Z$-DH: IF Z>VAL "11" THEN GO TO M2
7024 IF Z=AF OR Z>AG THEN GO TO VAL "7050"
7028 ON ERR GO TO VAL "7032": IF LEN O$(1,1)>1 THEN  ON ERR GO T
O VAL "7036": STOP
7032 ON ERR RESET : CLS : PRINT AT BO,SIN PI; FLASH SGN PI;"NO D
ATA PRESENT": BEEP SGN PI,VAL "20": GO TO M2
7036 ON ERR RESET : GO SUB VAL "7162"
7050 GO TO (Z=SGN PI)*VAL "7110"+(Z=AB)*VAL "7132"+(Z=INT PI)*VA
L "7070"+(Z=AD)*VAL "7054"+(Z=AE)*VAL "7240"+(Z=AF)*VAL "7280"+(
Z=AG)*VAL "7264"+(Z=VAL "8")*VAL "7272"+(Z=VAL "9")*VAL "7126"+(
Z=BO)*VAL "7440"+(Z=SIN PI)*VAL "7400"+(Z=VAL "11")*M1
7054 CLS : PRINT AT VAL "14",SIN PI;"THERE ARE ";L2;" LINES THAT
 CAN BE SORTED.": INPUT "INPUT LINE# TO SORT";SO: PRINT AT VAL "
18",SIN PI;"YOU MAY SORT BY FIRST GROUP OF LINE ";SO;" OR LAST G
ROUP:": INPUT "INPUT <1> FIRST OR <2> LAST";SOL: IF SOL<SGN PI O
R SOL>AB OR SO<SGN PI OR SO>L2 THEN CLS : GO TO VAL "7054"
7058 LET B$="": GO TO (SOL=SGN PI)*VAL "7364"+(SOL=AB)*VAL "7294
"
7074 CLS : PRINT "ZIP CODES TO LINE 4": FOR N=SGN PI TO L5: IF O
$(N,AD)(SGN PI)=" " THEN LET M$=O$(N,INT PI)
7078 IF O$(N,AD)(SGN PI)<>" " THEN GO TO VAL "7094"
7082 IF M$(LEN M$)=" " THEN LET M$=M$( TO LEN M$-SGN PI): GO TO
VAL "7082"
7086 FOR G=LEN M$ TO SGN PI STEP -SGN PI: IF M$(G)=" " THEN LET
O$(N,AD)=M$(G+SGN PI TO ): LET O$(N,INT PI)=O$(N,INT PI)( TO G):
 GO TO VAL "7094"
7090 NEXT G
7094 IF N>L5 THEN GO TO M2
7098 NEXT N
7102 GO TO M2
7110 CLS : PRINT "REMOVING REPEATS": FOR N=SGN PI TO L5: FOR Y=L
5 TO N+SGN PI STEP -SGN PI: IF O$(Y,SGN PI)=O$(N,SGN PI) THEN FO
R G=SGN PI TO L2: LET O$(Y,G)="": NEXT G
7114 NEXT Y
7118 NEXT N
7122 GO TO M2
7126 DIM O$(SGN PI): DIM V$(SGN PI): DIM K$(SGN PI): LET XX=SIN
PI: GO TO M2
7132 LET G=SGN PI: LET Y=SGN PI: CLS : PRINT "REMOVING SKIPS": F
OR N=SGN PI TO L1: LET X=SIN PI: IF O$(N,SGN PI)(SGN PI)=" " THE
N LET X=SGN PI: FOR G=AB TO L2: LET O$(N,G)="": NEXT G: FOR Y=N+
SGN PI TO L1: IF O$(Y,SGN PI)(SGN PI)<>" " THEN FOR G=SGN PI TO
L2: LET O$(N,G)=O$(Y,G): NEXT G: FOR G=SGN PI TO L2: LET O$(Y,G)
="": NEXT G: GO TO VAL "7142"
7138 IF X=SGN PI THEN NEXT Y
7142 NEXT N
```

The hidden variables are assigned in line 6800 and the hidden string variables in line 6802. These lines are deleted in the working copy of this version of the program. An alphabetical list of the hidden variables is as follows:

AB=2	INT PI=3
AD=4	M1=6804
AE=5	M2=7020
AF=6	SGN PI=1
AG=7	SIN PI=0
BO=10	TS=9920
D1=7232	YF=256
D2=7256	ZB=9922
DH=48	

You will note my use of the CAPS LOCK when programming. This makes it easier to determine, in dot matrix printouts, if a letter is a lower case L, a lower case I, or a numerical one. The larger capitalized program printout is also much easier on the eyesight when typing programs into the computer. With my printer, this is an upper case o (O) and this is a zero (0).

After typing in the program be sure to SAVE it as your Master Copy before attempting to RUN it.

When you RUN the program the first time, you will be asked to supply the data for a letter heading and signature block. This is for use in making out invoices or even using the program as a simple word processor with which to write a note or letter. I used it to write and print this article.

When the DBASE 1 Menu appears on the screen, chose option 8. This puts you back in BASIC in the 64 character mode so that you can SAVE the program this time as your working copy.

```
7146 GO TO M2
7162 CLS : PRINT "FINDING DIMENSION OF O$"'': LET L3=SIN PI: LET
 L2=SIN PI: LET L1=SIN PI: FOR N=PEEK VAL "23627"+YF*PEEK VAL "2
3629" TO PEEK VAL "23641"+YF*VAL "23642": IF PEEK N=VAL "207" TH
EN LET L1=PEEK (N+AD)+YF*PEEK (N+AE): LET L2=PEEK (N+AF)+YF*PEEK
 (N+AG): LET L3=PEEK (N+VAL "8")+YF*PEEK (N+VAL "9"): GO TO VAL
"7210"
7166 NEXT N
7210 CLS : PRINT AT BO,O;"THE DATA BASE IS O$(";L1;",";L2;",";L3
;")"
7214 FOR N=SGN PI TO L1: ON ERR GO TO VAL "7218": IF O$(N,SGN PI
)(SGN PI)<>" " THEN LET XX=N
7216 NEXT N: ON ERR GO TO VAL "7218": STOP
7218 ON ERR RESET : LET L5=XX
7220 PRINT "THE LAST FILLED RECORD IS: ";L5
7224 PRINT '''PLEASE MAKE NOTE AND <ENTER>": PAUSE SIN PI: RETUR
N
7232 CLS : INPUT "RE-INSTALL PROGRAM DISK, AND ENTER DRIVE LTR "
;D$: GO TO *D$D: RETURN
7240 CLS : GO SUB D2: INPUT "ARE THESE ALPHA OR ZIP FILES? <1> A
LPHA - <2> ZIP?";A: INPUT "ENTER STARTING FILE NBR?";S4
7244 LET XX=SGN PI: FOR N=SGN PI TO L1 STEP L1/AD: DIM V$(L1/AD,
L2,L3): LET M$=("A"+STR$ S4+".DAT" AND A<AB)+("Z"+STR$ S4+".DAT"
 AND A>SGN PI): FOR Y=SGN PI TO L1/AD: FOR G=SGN PI TO L2: LET V
$(Y,G)=O$(XX,G): NEXT G: LET XX=XX+SGN PI: NEXT Y: SAVE *M$ DATA
 V$(): LET S4=S4+SGN PI: NEXT N: CAT *"+.DAT": PRINT #SIN PI;AT
SIN PI,SIN PI;"NOTE THE SAVED FILES,'''<ENTER>": PAUSE SIN PI: G
O SUB D1: GO TO M2
7256 INPUT "INSTALL DATA DISK, THEN ENTER DRIVE LTR";D$: GO TO *
D$D: CAT *"+.DAT": RETURN
7264 ON ERR GO TO VAL "7032": CLS : GO SUB D2: INPUT "<1> ALPHA
SAVE- OR <2> ZIP SAVE ";AZ: INPUT "ENTER FILE NBR?";A: LET M$=("
A"+STR$ A AND AZ<AB)+("Z"+STR$ A AND AZ>SGN PI): LET M$=M$+".DAT
": SAVE *M$ DATA O$(): GO TO M2
7272 CLS : GO SUB D2: INPUT "<1> ALPHA LOAD- <2> ZIP LOAD";A: IN
PUT "INPUT FILE NBR TO LOAD ";M$: LET M$=("A"+M$ AND A<AB)+("Z"+
M$ AND A>SGN PI): DIM O$(SGN PI,SGN PI,SGN PI): DIM V$(SGN PI,SG
N PI): LET M$=M$+".DAT": LOAD *M$ DATA O$(): GO SUB D1: GO SUB V
AL "7162": GO TO M2
7280 CLS : GO SUB D2: INPUT "MERGE <1> ALPHA OR <2> ZIP FILES?";
AZ: INPUT "INPUT START FILE NBR TO MERGE";M: LET M$=("A"+STR$ M
AND AZ<AB)+("Z"+STR$ M AND AZ>SGN PI): DIM O$(SGN PI): LET M$=M$
+".DAT": DIM V$(SGN PI): LOAD *M$ DATA O$(): CLS : GO SUB VAL "7
162": DIM O$(L1*AD,L2,L3): LET XX=SGN PI
7282 FOR Y=SGN PI TO AD: DIM V$(SGN PI): LET M$=("A"+STR$ M AND
AZ<AB)+("Z"+STR$ M AND AZ>SGN PI): LET M$=M$+".DAT": LOAD *M$ DA
TA V$(): LET M=M+SGN PI: FOR G=SGN PI TO L1: FOR I=SGN PI TO L2:
 LET O$(XX,I, TO L3)=V$(G,I, TO L3): NEXT I: LET XX=XX+SGN PI: N
EXT G: NEXT Y: DIM V$(SGN PI): BEEP SGN PI,VAL "11": CLS : LET X
X=SGN PI: GO SUB VAL "7162": CLS : PRINT "FILES ";("A" AND AZ<AB
)+("Z" AND AZ>SGN PI);M-AD;" TO ";M-SGN PI;" ARE COMBINED AND R
EADY TO SORT.'''<ENTER>": PAUSE SIN PI: GO SUB D1: GO TO M2
7294 CLS : PRINT "REVERSING GROUPS IN LINE ";B0: FOR N=SGN PI TO
 L5: LET B$=O$(N,B0)
```

3 of 6

As this now contains hidden variables and strings, you must not use RUN or CLEAR. Save the program on a different disk using SAVE*"DBASE1" LINE 6804. This then becomes your working copy. be sure that any changes are also made in your Master Copy.

Dbasel Evolution

In this issue is another TREMENDOUSLY VERSATILE data base program called "Dbx.B6". Both this TOS Version by Earl, and the Dbx.B6 program, are off-shoots of the program "Dbase-1" given in the July 89 issue. Since Earl left some margin space I'll propound upon these programs and give the readers some insight to more that is to come.

It is intriguing that Earl printed his text with this Dbasel program, which is really a "Record" Data Base program. The Records are "user flexible" so that the Dbase can be used for any purpose. The most common use is a Mail File, a Telephone Listing, or an Inventory Data Base. But, as you can see, Earl found a way to use the program as a Word Processor as well. T.O.S. Disk users are fortunate to have the services of Earl Dunnington to provide this and other programs in a form that can be used with TOS Disk.

The original article about Dbasel in the July 89 issue created much interest in using the TS-2068 Vars area for managing data bases. And Why Not! Sinclair Computers manages the vars area much better and provide much more capability THAN ANY OTHER COMPUTER marketed now or before! IBM computers, for example, limit the LEN (length) of string arrays to 255 characters while Sinclair Computers have no practical limit. This whole group of paragraphs can be tucked into a character array (and it is). We dont have to worry about FINDING the

```
7300 IF B$(LEN B$)=" " THEN LET B$=B$( TO LEN B$-SGN PI): IF LEN
 B$>AE THEN GO TO VAL "7300"
7304 FOR G=LEN B$ TO SGN PI STEP -SGN PI: IF G=SGN PI THEN GO TO
 VAL "7316"
7308 IF B$(G)=" " THEN LET O$(N,SO)=B$(G+SGN PI TO LEN B$)+" "+B
$( TO G-SGN PI): GO TO VAL "7316"
7312 NEXT G
7316 IF N>=L5 THEN GO TO VAL "7364"
7320 NEXT N
7324 GO TO VAL "7368"
7332 FOR N=SGN PI TO L5: LET M$=O$(N,SO)
7336 IF M$(LEN M$)=" " AND LEN M$>SGN PI THEN LET M$=M$( TO LEN
M$-SGN PI): GO TO VAL "7336"
7340 IF LEN M$=AE THEN NEXT N: GO TO VAL "7352"
7344 FOR Y=SGN PI TO L3: IF M$(Y)=" " THEN LET O$(N,SO)=M$(Y+SGN
 PI TO )+" "+M$( TO Y): GO TO VAL "7352"
7348 NEXT Y:
7352 IF N>=L5 THEN GO TO VAL "7396"
7356 NEXT N: GO TO VAL "7396"
7364 PRINT "SORTING, LINE ";SO
7368 DIM V$(L2,L3): LET N=L5: LET I=N-SGN PI: FOR J=SGN PI TO I:
 LET K=J+SGN PI: FOR L=N TO K STEP -SGN PI
7372 IF O$(L,SO)(SGN PI TO AD)="    " THEN GO TO VAL "7384"
7376 IF O$(L,SO)>O$(J,SO) THEN GO TO VAL "7384"
7380 IF O$(L,SO)(SGN PI TO AD)<>"    " THEN LET B$=O$(L,SO): FOR
 X=SGN PI TO L2: LET V$(X)=O$(L,X): NEXT X: FOR X=SGN PI TO L2:
LET O$(L,X)=O$(J,X): NEXT X: FOR X=SGN PI TO L2: LET O$(J,X)=V$(
X): NEXT X
7384 NEXT L
7388 NEXT J
7392 IF SOL=AB THEN GO TO VAL "7332"
7396 CLS : PRINT AT BO,AD;"SORT COMPLETE": BEEP 1,18: PAUSE VAL
"200": GO TO M2
7400 CLS : PRINT AT BO,SIN PI;"INPUT KEY WORD": INPUT Y$: INPUT
"<1> VIEW OR <2> PRINT OUT";Z: IF Z=AB THEN INPUT "TAB?";TB: INP
UT "NBR LINES PER PAGE?";LN: INPUT "PAGE NBR?";PG
7404 LET NN=SIN PI: FOR N=SGN PI TO L5: IF O$(N,AG)( TO LEN Y$)=
Y$ THEN GO SUB VAL "7412"
7408 NEXT N
7412 IF Z=SGN PI THEN PRINT "RECORD# ";N
7416 IF Z=AB THEN GO SUB ZB: LPRINT TAB TB;"RECORD# ";N: LET NN=
NN+SGN PI
7420 FOR Y=SGN PI TO L2: IF Z=SGN PI THEN PRINT O$(N,Y)
7424 IF Z=AB THEN LPRINT TAB TB;O$(N,Y): LET NN=NN+SGN PI: IF NN
>=LN THEN LPRINT '''TAB VAL "40";PG: LET PG=PG+SGN PI: LET NN=SI
N PI: GO SUB VAL "7436"
7428 NEXT Y
7432 NEXT N: IF Z=SGN PI THEN PRINT #SIN PI;"<ENTER> FOR MENU":
PAUSE SIN PI
7434 GO SUB TS: GO TO M2
7436 GO SUB TS: PRINT #SIN PI;"SERVICE PRINTER THEN <ENTER>": PA
USE SIN PI: GO SUB ZB: RETURN
7440 LET SPC=SIN PI: LET RET=SIN PI: LET LN=VAL "54": LET NN=SIN
 PI: CLS : PRINT #SIN PI;AT SIN PI,SIN PI;"VIEW OR PRINT OUT?"'"
```

data for editing. The Sinclair Rom does that for us. We can shift data about using simple basic commands <LET A$=H$(3)>, etc.

Back in the July 89 issue I promised to bring in a set of Machine Code utilities to provide faster SORTING of the Dbasel Records. These sorting utilities have been done and will be presented in a follow on issue of Update. Don Lamen is the author of these MC utilities. When the MC utilities are given they will be adaptable to Earl's TOS program given here, and for the Sort functions of the Dbx.B6 program given elsewhere in this issue. Then an Update of the original Dbasel program that uses the MC sorting will be given also.

Dbasel goes beyond any software that I am aware of to "Sort and Manage Disk Data Files". Lets set up a straw man for talking purposes. Lets say that we have set up several different "Record Files" using either this program, or the "Dbx.B6" program. One file is a simple one liner Telephone list created with Dbx.B6. It has 1000 "one line" records, a typical record being "McManus, Robert A. 425 872 6657". Since we are highly socialized we have 4 such telephone files in disk (4000 names and tel nbrs). Another type of Record Data base is a Mail File. It is similiar to our Telephone list but has 5 lines in each record, typically: Line 1= "Mr. John A. McManus": Line 2="4416 Main St.": Line 3="Peroria, IL 49665-2416": Line 4="49665": Line 5="John,".

Now the versatility provided by "sorting choices" will come into focus. We could sort by "First Group" to sort the Telephone listing. "McManus" would sort. For the Mail List, we could Sort BY LAST GROUP OF LINE #1 of the records. Then "McManus" would also Sort. Or, we could SORT BY LINE 4 LAST GROUP, which would sort by Zip Code. Any line can be sorted by either "first group of last group" within the line. This is true for both Dbasel and Dbx.B6. But

```
<1> VIEW <2> PRINT OUT": PAUSE SIN PI: LET X=CODE INKEY$-DH: IF
X>AB THEN GO TO VAL "7440"
7442 IF X=AB THEN INPUT "ENTER STARTING RECORD";MS: INPUT "ENDIN
G RECORD?";MX: INPUT "ENTER NBR OF LINES TO PRINT     (4 FOR LAB
ELS)";LP: IF LP=AD THEN INPUT "RETURN LABELS TOO?              <
1> YES OR <2> NO";RET
7444 IF x=AB THEN GO TO VAL "7480"
7448 PRINT #SIN PI;AT SIN PI,SIN PI;"<1>WHOLE RECORDS -OR-"'"<2>
SINGLE LINES?          ": PAUSE SIN PI: LET R=CODE INKEY$-DH: IF
 R=AB THEN CLS : INPUT "ENTER RECORD LINE NBR TO VIEW ";RL
7454 CLS : PRINT #SIN PI;AT SIN PI,SIN PI;"TO DISCONTINUE, JAB <
BREAK>": PAUSE VAL "200": CLS
7460 ON ERR GO TO VAL "7468": IF R=SGN PI THEN FOR N=SGN PI TO L
1: PRINT N: FOR G=SGN PI TO L2: PRINT O$(N,G): NEXT G: PRINT : N
EXT N: STOP
7464 IF R=AB THEN FOR N=SGN PI TO L1: ON ERR GO TO VAL "7468": P
RINT N'O$(N,RL): PRINT : NEXT N
7468 ON ERR RESET : PRINT "<ENTER> FOR MENU": PAUSE SIN PI: CLS
: GO TO M1
7480 CLS : INPUT "TAB?";TB: INPUT "DO YOU WANT AN OUTLINE FORMAT
?  <1> YES OR <2> NO";TB2: INPUT "INPUT NBR SPACES BETWEEN RECOR
DS?";SP: INPUT "DO YOU WANT PAGE CONTROL?        <1> YES OR <2>
NO";PGC: IF PGC=SGN PI THEN INPUT "PAGE #";PG: INPUT "SET PRINTE
R AND INPUT # LINES PER PAGE?";LN
7482 PRINT AT VAL "8",AB;"YOUR SET-UP IS:"'TAB AB;"PRINT RECORD
S ";MS;" TO ";MX'TAB AB;"PRINTING LINES 1 TO ";LP'TAB AB;"AT TAB
 ";TB'TAB AB;"SPACES BETWEEN RECORDS- ";SP': IF PGC=SGN PI THEN
PRINT TAB AB;"PAGE=";PG'TAB AB;"LINES PER PAGE=";LN
7483 IF RET=SGN PI THEN PRINT TAB AB;"RETURN LABELS WILL PRINT"
7484 IF TB2=SGN PI THEN PRINT TAB AB;"OUTLINE FORMAT"
7486 INPUT "CORRECT? <1> YES OR <2> NO";Z: LET TB2=(TB+INT PI AN
D TB2<AB)+(TB AND TB2>SGN PI): IF Z=AB THEN GO TO VAL "7440"
7488 ON ERR GO TO VAL "7498": GO SUB ZB: LET NN=SIN PI: FOR N=MS
 TO MX: FOR Y=SGN PI TO LP: LPRINT TAB (TB AND Y<AB)+(TB2 AND Y>
SGN PI);O$(N,Y): GO SUB VAL "7510": NEXT Y: GO SUB VAL "7516"
7490 IF PGC=SGN PI AND NN>=LN THEN GO SUB VAL "7500"
7494 NEXT N: IF PGC=SGN PI AND NN>SIN PI AND NN<LN THEN GO SUB T
S: INPUT "PRINT LAST PAGE # ?"'"<1> YES OR <2> NO THEN <ENTER>";
Z: IF Z=SGN PI THEN LET SP=LN-NN: GO SUB ZB: GO SUB VAL "7517"
7495 IF RET=SGN PI THEN CLS : GO SUB TS: PRINT AT BO,SIN PI;"SER
VICE PRINTER FOR RET LABELS"'"THEN PRESS <ENTER>": PAUSE SIN PI:
 GO SUB ZB: FOR Y=SGN PI TO MX: FOR N=SGN PI TO INT PI: LPRINT N
$(N): NEXT N: LPRINT : LPRINT : LPRINT : NEXT Y
7496 GO TO M1
7498 GO SUB TS: GO TO VAL "7032"
7500 GO SUB ZB: LPRINT '''TAB VAL "50";PG: LET PG=PG+SGN PI: LET
 NN=SIN PI: GO SUB TS: PRINT AT BO,AB;"SERVICE PRINTER, THEN <EN
TER>": PAUSE SIN PI: GO SUB ZB: RETURN
7510 IF PGC=SGN PI THEN LET NN=NN+SGN PI: IF NN>=LN THEN GO SUB
VAL "7500"
7512 RETURN
7516 IF PGC=SGN PI AND NN>=LN THEN GO SUB VAL "7500": RETURN
7517 GO SUB ZB: FOR G=SGN PI TO SP: LPRINT : IF PGC=SGN PI THEN
LET NN=NN+SGN PI: IF NN>=LN THEN GO SUB VAL "7500": RETURN
```

Dbasel carrys it further and allows you to SORT DISK FILES. You can sort the 4 "Telephone Lists" that are in disk. Or you could sort any number of Mail Files that are in disk. The sorting is "against each file in disk", providing a re-save of all files in numeric order.

Both programs use similiar protocol. When a file is LOADED from disk the program finds the "Dimension" of the file for management needs. When SORT is ordered, if sorting by "last group" the program first reverses the sorting line. Then a sort is done by sorting the first group within the line. When Sort is finished the sorting line is restored to its original form. Example: Mr. John McManus would be changed to McManus Mr. John. When sort is finished the line would be restored to Mr. John McManus. If sorting by Zip code, and the line were "Panama City, FL 32404" the sort would reverse to put the zip code first, then sorting would occur, then all zip codes would be returned to last group in the line.

The Machine Code utilities will greatly speed up the operation. Right now if you want to sort several disk files having many records it becomes an over-night "sort while you sleep". You just wouldn't believe the number of operations that take place to sort several disk record files! It is in the millions of operations! Included in the sorting is "Removing Repeat Records" in all files, and "Reordering" each file so that there are no Blank Records in the interior of the files.

So, the coming Machine Code utilities will make Dbasel very fast in handling disk data files. In the meantime, we can get accustomed to operating a very versatile data base program for creating and managing large Record files for any purpose.

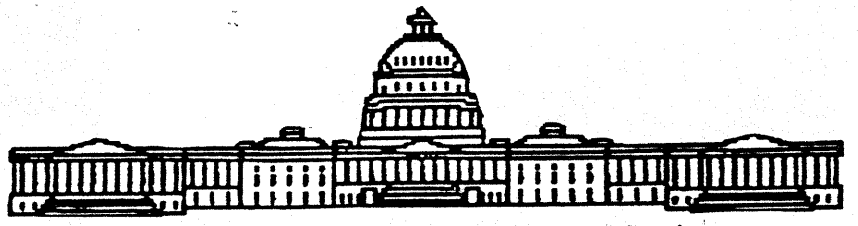
596

```
7518 NEXT G: RETURN
7530 CLS : INPUT "<1> LETTERHEAD OR <2> SIGN OFF?";LS: IF LS=SGN
 PI THEN INPUT "<1> SINGLE OR <2> MULTIPLE?";SM
7531 GO SUB ZB
7532 IF LS=SGN PI AND SM=AB THEN FOR Y=SGN PI TO MX
7534 IF LS=SGN PI THEN FOR N=SGN PI TO INT PI: LPRINT TAB VAL "2
8";N$(N): NEXT N: IF SM=AB THEN OUT 127,12: NEXT Y
7536 IF LS=AB THEN LPRINT '''TAB VAL "30";N$(AE)'''TAB VAL "30";
N$(AD)
7538 GO TO M1
9920 CLS : OUT VAL "255",SIN PI: OUT VAL "244",SIN PI: BORDER SI
N PI: PAPER SIN PI: INK AG: CLS : RETURN
9922 CLS : OUT VAL "255",VAL "62": OUT VAL "244",INT PI: INK AG:
 POKE VAL "65525",VAL "80": RANDOMIZE USR VAL "15623": RETURN
9999 GO SUB ZB: STOP
```

6 9 6

The OS-64 Screen

I notice that this program for TOS, by Earl, is for use with the Zebra OS-64 cartridge that implements the 64 column Screen. For those who do not have the OS-64, careful editing can be done to eliminate the 64 column screen calls and use the standard 32 column screen. Earl is invited to follow up in the next issue with guidance as to which lines to modify for 32 column screen. For those who DO have the OS-64, I have an article in works that is a complete word processor for use with the OS-64 cartridge. Look for it in a coming issue of Update. _BJ-



CATS

The Official Journal of the Capital Area Timex/Sinclair Users Group

NEWSLETTER

Newsletter

Memberships cost $15 per year, are good for 12 months, and include all privileges (access to libraries, group buys, etc.). A newsletter only subscription is available for persons living outside the Washington Metro area and is $12 per year.

CATS BBS: (301) 588-0579. 8-N-1. 300/1200 baud. News and general information, 24 hours a day.

CATS Newsletter
P.O. Box 11017
Takoma Park, MD 20913

TRUMP CARD USER's GUIDE
by
Tim Stoddard

INTRODUCTION

I have been using three TRUMP cards now for about a year and a half now, and I have noticed that there is no real 'USER GUIDE' available. The very small manual that comes with the TRUMP card outlines the syntax (sometimes incorrectly) but gives very few examples. These series of articles will bring to light some of the very powerful features of the TRUMP CARD and TOOLKIT II via real usable examples. The articles **may** be assembled into a user guide at some future time, if enough interest is shown.

I welcome all critiques and suggestions. I want this user guide as accurate as possible. Feel free to write via UPDATE MAGAZINE or my home. If you wish an answer please include S.A.S.E. envelope.

Tim Stoddard
85-48 66th Road
Rego Park, NY 11374-5212

You can also reach me via CompuServe (73127,2664), Genie (xtx53868), or Prodigy (BGCX06A).

A note before we begin: In this series I will sometimes use more than one consecutive underscore. These are hard to see, so please bear with my LQ500 printer and watch the length of the underscore! Examples:

_ (1 underscore)
__ (2 underscores)
___ (3 underscores)

DIRECTORY CONTROL

ONE of the more powerful UNIX-like features of QJUMP's TK2 supplied with the TRUMP CARD is it's handling of file names as a directory structure. For instance, consider the file name FLP1_EDT_BIN. BIN is an acronym for BINARY. Some software companies will also use EXE (an acronym for EXECUTABLE), or in Digital Precision's case: TASK. In any case, this extention is added on to indicate that this file is an executable program. To start this program without TOOLKIT II you would need to key in:

exec flp1_edt_bin <ENTER>

QJUMP's TOOLKIT II has commands that set default directories, and all file commands have been enhanced to use these defaults. You could use the default by first setting it using the command:

prog_use flp1_ <ENTER>

Then, all that needs to be entered to start EDT is:

exec edt_bin <ENTER>

Using QJUMP's short EXEC name of EX you only need to type in:

ex edt_bin <ENTER>

There are two other defaults that can be set: 1) where data is read from and 2) where data is written to. These two defaults are used by such commands that have a source and/or a target file spec. Examples would be:

copy source,target

dir source

load source

view source

The source file spec is set using the command:

data_use flp2_

The target file spec is set using the command:

dest_use flp2_

NOTE that these three defaults are looking for directories and not files. TK2 automatically adds an underscore to the directory named. For instance, executing:

data_use flp2_t
actually implies a path of flp2_t_, and you can verify that this is the case by executing:

dlist

which gives the current paths for all three defaults. You will see that the trailing underscore was added by TK2. As an example:

data_use flp2__doc

would seem to imply that you would 'select' all files ending in '_doc'. This is almost true! Remember, TK2 adds the trailing underscore to make the path actually 'flp2__doc_' and this would also select a file named 'doctor_txt'.

NOTE: The two underscores together imply a wildcard match with anything.

For example the following files are in RAM2_:

net_doc
doctor_txt
test_txt_doc

data_use ram2_ is executed.
dir doc is executed, implying a path of ram2_doc_ The single underscore between ram2 and doc implies NO wildcard match and so the selection MUST BEGIN with 'doc'. The following prints out:

doctor_txt

dir _doc is executed, implying a path of ram2__doc_ Use dlist to check the path. The double underscore will match a file spec with 'doc' anywhere in it. The following prints out:

net_doc
doctor_txt
test_txt_doc

The target default file spec, dest_use works the same way as the data_use but describes the target file spec. For

example executing:

dest_use flp2_ sets the target file spec to flp2_

wcopy (implies wcopy ram2_ to flp2_) will copy all our above files from ram2_ to flp2_. NOTE that I did not have to describe a source and target. WCOPY used the defaults we set with data_use and dest_use: ram2_ & flp2_.

Now adding a source modification:

wcopy doc (implies wcopy ram2_doc_ to flp2_) will copy only doctor_txt to flp2_ since only that file begins with 'doc'.

wcopy _doc (implies wcopy ram2__doc_ to flp2_) will copy all three example files to flp2_ since they all contain 'doc' in their file names. REMEMBER the double underscore signals a wildcard match!

Adding a target modification will allow coping and renaming:

wcopy doc to xxx (implies wcopy ram2_doc_ to flp2_xxx_) will copy ram2_doctor_doc to flp2_xxxtor_txt.

wcopy _doc to _xxx (implies wcopy ram2__doc_ to flp2__xxx_) copies all our files changing the 'doc' to 'xxx'.

wcopy _doc to xxx (implies wcopy ram2__doc_ to flp2_xxx_) copies all three files while renaming the 'doc' to 'xxx' AND because the target modifier has no double underscore (no wildcard): any of the filename to the left of the match will be truncated! Our three files would copy over as:

ram2_net_doc --> flp2_xxx
ram2_doctor_txt --> flp2_xxxtor_txt
ram2_test_txt_doc --> flp2_xxx

Note: this last operation would cause us to lose the file originating as ram2_net_doc since ram2_test_txt_doc would also be copied over as flp2_xxx overwriting the first copy over.

You are not limited to a one-for-one text match for example:

wcopy _doc to _x

causes the three files to copy over as:

flp2_net_x
flp2_xtor_txt
flp2_test_txt_x

And...

wcopy _doc to _1234567

causes....

flp2_net_1234567
flp2_1234567tor_txt
flp2_test_txt_1234567

Interestingly, although the manual does not cover this, adding further concurrent underscores does help us! For instance:

wcopy __txt (implying ram2___txt to flp2_) NOTE the THREE concurrent underscores above! This would copy over only files with 'txt' in the second position! These files would copy to ram3_:

doctor_txt
test_txt_doc

wcopy ___doc (implying ram2____doc to flp2_) NOTE FOUR concurrent underscores above! This would copy over only files with 'doc' in the third position! The only file to meet this requirement is:

test_txt_doc

Try some of the other file commands now and experiment!

dir
dir doc
dir _doc
dir __doc
dir __txt
dir ___doc
wstat __txt
wdel __doc

A variation of the default target file spec command dest_use is spl_use. Its only difference is that it does not add the trailing underscore to the default file spec. Its intended use is to set the target default to a device such as: ser1 or n2_ser1. You are not restrained from using a file name, however. For example:

spl_use flp2 Use command dlist to verify that there is no trailing underscore.

then....

wcopy Will produce error message 'not found' since a device MUST end with an underscore.

spl_use flp2_test
wcopy These two commands will now copy and rename all files originating in ram2_ to flp2_test!!!

AND the intended use....

spl_use ser1 Sets the destination to serial port #1.
wcopy copys all files to that serial port.

Navigation is straight-forward and as the manual states it. Those commands are:

ddown name Down one level to name. Use Dlist to verify.

dup Up one level.

dnext name	Up one level, then down to 'name'.
dlist	Prints the current defaults to channel#1.
dlist #channel	The same, but to a specified channel.
dlist \name	**DOES NOT WORK!!! Gives error message 'bad parameter'.**

The \name parameter, although it does not work with dlist, is a very handy parameter! Try this with a printer attached to ser1:

```
dir \ser1
wstat \ser1
jobs \ser1
```

and if you have the HOT KEY system.....

hot_list \ser1	Works great!!!

file names also work.....

dir \flp2_list	Copies the directory to file 'list' on flp2_.
dir \flp2_list,flp1_	Copies the directory of flp1_ (not default of ram2_) to file 'list' on flp2_. NOTE THE COMMA separating the name parameter from the requested device.

Finally, there are functions that allow your SuperBasic programs to use the set defaults.

datad$	returns the data default
progd$	returns the program default
destd$	returns the destination default

ED NOTE: Tim Stoddard's TRUMP CARD MANUAL began in the January 1990 Update Magazine with PART 1. Other parts will continue in coming issues of Update Magazine. I consider Tim's manual to be one of the most important series of articles ever done for the Sinclair QL. The Trump Card System has so very much power that is not articulated in the tiny manual supplied with the hardware. Even so, the tiny Trump Card Manual has much information, albeit brief. My Trump manual has been "blown up" to full page size, which is really the only way to digest the information. Tim's articles in January Update, this one, and the continuations, will open up the gate for us to learn much more about the use of the built-in tools of the Trump System. If you dont have the January issue of Update, you may order it along with the 1990 year series of 4 issues ($18.00 to: Bill Jones, 1317 Stratford Ave., Panama City, FL 32404)

ANNOUNCING: "Gulf Micro Electronic" *(Bill Jones' little software house)*
1317 Stratford Ave., Panama City, FL 32404

All I have right now is "DAISY.B6", The *biggest* and most *comprehensive* software *ever produced* for the TS-2068.

Daisy.B6 is for the LKDOS SYSTEM. It Manages the disk drive as if it were a part of the Computer's Memory. Daisy takes up 3 360K disks or two 720K disks. It is a complete system for TS-2068 LKDOS.

To give you a clue as to its vastness, the "Dbx.B6" software given in this issue is an *"annex" of the Daisy System*, included in disk #2. *AND so are the three "Macro" programs given in this issue! Plus MORE.*

This Daisy System started out as my Smart Text software and has evolved over the three years of publishing Update Magazine. Daisy is used for 90% of the data base and print management that you see in this magazine. If you can do better print management, tell me what you use and I'll buy it. If you have LKDOS you should have the Daisy System that really takes full advantage of LKDOS. If you dont have LKDOS it is worth while to BUY it to get the full functions of the Daisy System. Daisy + 80 page Manual- $29.95 post paid.

Watch for "mini daisy" for TS-2068 cassette systems *(a bit of reverse engineering)*. More good things coming as Frank takes over publishing Update! (and I have more time). *-Happy Computing! Bill Jones_*

For SALE: Timex Disk Drive System and extra Parts. Amdisk Dual Disk Drives. Call or write: Stan Light, PO Box 13, Ellenton, FL 34222. 813 747 5833

EMSoft
Software for the QL that works hard

Now has in-stock the very best true WYSIWYG QL word-processor: text^{87} v3.00 with spell check and several new features. (includes 1 hr USA help line). $94.95

Tax-time means	TAX-I-QL/89	$24.95
RECIPE	_dbf	$14.95
INVENTORY	_dbf	$9.95
TRUST_FUND	_dbf	$24.95
QLAND_LORD	_dbf	$24.95
MAILBAG	_dbf	$24.95
DBTutor	_dbf	$24.95
DBProgs	_dbf	$19.95
CDIR/BOOTS	SBas	$9.95

On 5.25" DD or QD disks. On 3.5" or microcartridge add $2.00.

FREE CATALOGUE

EMSoft
P.O. Box 8763
Boston MA 02114
(617)889-0830

WOOD AND WIND COMPUTING

A NEW SOFTWARE RELEASE !!!

QLERK : A powerful accounting program for the QL $75.00 on 2 disks

QLERK is an ARCHIVE based accounting program for home and business. It includes payable, receivable, and general ledger modules integrated with my address database program, DBAddres. QLERK has check writing and invoicing built in. It even keeps track of your VISA, Master Charge, or other charge cards. Bank balancing is easy with QLERK's help. You can display all your checks and deposits and compare them with your bank statements. You don't have to use a predefined account coding system. You can use the one you want. When you run the general ledger, each account is individually tallied making income tax time a snap. It makes a general ledger export file for ABACUS.

I have developed QLERK over the past 3 years working with several businesses. I have many,many hundreds of hours of development work in this program. I have been careful not to release it too soon but now feel it is ready. At $75.00 QLERK is a pretty expensive piece of software but for those who want to keep track of their finances it has no equal on the QL and will quickly pay for itself. It has many features that more expensive PC accounting software like Peachtree and DAC Easy fail to include like adding new vendors, customers, accounts or products on the fly. The address database features are second to none for being able to target subgroups for mailing labels or other purposes. A PC version of QLERK using PC ARCHIVE is also in the works.

Don't send money for QLERK. To order QLERK first send a request for a more detailed description and special order form that will ask for necessary information. Each purchaser of QLERK will get a custom copy of the software with their home or business embedded in the code.

Bill Cable : RR3 Box 92 : Cornish, NH 03745
603 675 2218

Two tough things to find for your Timex or Sinclair ZX/TS computer--

SOLID INFORMATION & GOOD CONNECTORS

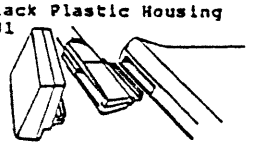
YOU KNOW WHERE TO GET THE BEST INFO FOR YOUR ZX/TS--SYNTAX, SQ, & THE FOYT BOOK CHECK IT OUT--ASK ANYONE--THEN GET 'EM ALL FOR

JUST $19.95 plus $10 S&H

BUS CONNECTOR & EXTENDER & HOUSING

LIKE A PRINTER CONNECTOR

Keyed Connector, Extender PC Board
PLUS Molded Black Plastic Housing
Fits: ZX80 & 81
TS1000 & 1500
PC8300
ZX/TS SUBSET
OF SPECTRUM
& 2068



Pkg of 3, ONLY $13.95 plus $2 S&H
SAVE Buy 2, get a FREE one
9 for $26.95 + $6 S&H SAVE

EMS, Box 8763, Boston, MA 02114-8763
617-889-0830 VISA & MC Add 4%

UPDATE ISSUE DISKS

All Issue Disks are undergoing up-dating by the authors and re-release to Frank Davis for release in the next (Oct) issue. Actually, the authors own these programs and Update Magazine acts as the distributor.

For now you may order Issue disks at the same price ($22.00), and receive either the original title of an up-dated version. (See listings of issue disks in past magazine issues).

Two new TS-2068 issue disks released in this month's magazine are: The "Dbx.B6" software for the LKDOS System and "Dbasel" for the TOS disk system.

The TS-2068 software "Daisy.B6" has been withdrawn as an Issue disk and is being offered separately by its author (Bill Jones). This is due to the extensive manual that is supplied and the need for direct coordination with the author for continuing service and up-dates as they occur. All other TS-2068 and QL issue disks have been given to Frank Davis for his coordination with the authors and re-release.

I believe that you can expect even more new software to be offered in the Issue disks in the future. Also, Frank Davis has full capability in ALL TS-2068 and QL DOS systems, plus both 3 1/2 and 5 1/4 inch disk drive support.

QL users can expect GREAT NEWS about the QUANTA LIBRARY. Paul Holgren, an associate of Update Magazine, is the new Quanta Librarian for North America. You may contact Paul through Update Magazine, or directly at 5231 Wilton Wood Ct., Indianapolis, IN 46254.

TS-2068 Issue Disks

(Specify type DOS, 40 or 80 track drive.)

1. "Dbx.B6" (see article this issue). For LKDOS ONLY. Will be converted to other DOS systems and announced.

2. "Dbasel" (See Article in this issue.) For TOS DISK ONLY. Order directly from: Earl Dunnington, 12198 Forrest Greens Drive, Boynton Beach, FL 33437.

SINCLAIR QL ISSUE DISKS (MDV)

The following issue disks are offered for QL systems of all configurations. The programs are on either 80 track Diskette or MDV cartridge. If ordering for MDV send a formatted MDV cartridge. All prices are $22.00 (post paid) and are guaranteed to satisfy.

---

1. "HARTUNG COLLECTION" DISK

Address and QSO file managers, plus a raft of disk utilities. All in Superbasic. Lots of tricks and hints for the programmers. Requires some knowledge of Superbasic or a yen to learn.

2, "FENG QLUTer" DISK

QLUTer - QL UTilities to un-clutter your disks and tapes. COPY, DELETE, FORMAT, PRINT, VIEW functions. TURBO compiled for SPEED. Multitasks and minimal key press. Also has-- SNAP SHOT - a directory column or condensed print out program. ALSO includes "VEGEMAT" - a CLONE making program.

---

3. "FENG QLUSTer" DISK

QLUSTer - very similiar to QLUTer (above) but has extended functions using some SUPER TOOLKIT II features. You must have the Toolkit (TK2_EXT) for this program. SNAP SHOT and VEGEmat also included.

---

4. CABLE ARCHIVE DISK

ARCHIVE utilities and exensive doc file instruction manuals. Programs are: REFIELD, SCAN, SPLIT, JOIN, DIR, FILER, FREQ. For Beginner to Advanced user. These programs GREATLY EXTEND the capabilities of ARCHIVE and the manuals lead one to complete understanding of Archive and these extensions.
If ordering for MDV send two mdv cartridges- too many programs to fit into one cart.

---

5. "CABLE HELPER2" DISK

A very useful TOOLKIT program for managing both DISKS and MICRODRIVE. HELPER lets you copy from one medium to another, Read or Copy doc files to paper or screen, repeat format mdv carts or floppy. Uses screen windows to see two directories from and to, copies over existing files in disk mdv, copy selected files. Does not conflict with TK2.

Order to UPDATE Magazine, address on cover.
Check or money order accepted.

> SHOW SPECIALS <

Even if you couldn't make it to Milwaukie-
Here are a few SPECIALS that we want to offer you:

SPECIAL #1:

RMG#Ø475 5.25" DSDD DISKETTES--REG $ 8.95 SALE $ 6.95+SH
(PAK OF 25 W/SLEEVES & WP TABS)
(ADD $3.5Ø SHIPPING)

SPECIAL #2:

RMG#Ø24Ø NOVELSOFT SUITE(2Ø68)-REG $19.95 SALE $16.95+SH
(ADD $4.ØØ SHIPPING)

SPECIAL #3:

RMG#Ø4Ø6 PIX-FX (2Ø68)---------REG $19.95 SALE $16.95+SH
(ADD $3.5Ø SHIPPING)

SPECIAL #4:

RMG#Ø682 CHOCOLATE SOLUTION (QL)---REG $24Ø SALE $19Ø+SH
(MSDOS EMULATOR W/DOS 4.Ø)
(ADD $5.ØØ SHIPPING)

SPECIAL #5:

RMG#Ø635 3.5" PREM DSQD DISKETTE-REG $1Ø.95 SALE $7.95+SH
(PAK OF 1Ø 8Ø TRACK)
(ADD $3.5Ø SHIPPING)

SPECIAL #6:

RMG#Ø165 MITSUBISHI VIDEO PHONE--REG $799.95/PR SALE $3ØØ/PR
THESE ARE * FULL-FEATURED * PHONES!
REDIAL-COMPUTER MEMORY/DIRECTORY-SEND/RECEIVE
PICTURES-SPEAKERFONE-MULTILINE CONNECTOR/
SWITCHBOX-WALL OR DESK MOUNT-BATTERY BACKUP
ADD $25 PER PAIR SHIPPING/INS.

If any of these appeal to you, just send your order to the address on the other side of this sheet or call us!

We accept MASTERCARD and VISA cards as well as C.O.D. ordes.
There is a 4% surcharge for bankcards and add $3.5Ø for C.O.D..

The above specials are for a LIMITED TIME ONLY!

ALL OFFERS EXPIRE AS OF 8/31/9Ø OR WHEN SUPPLIES ARE GONE!

HAVE A GREAT SUMMER!

RMG ENTERPRISES

1419 1/2 7TH STREET, OREGON CITY, OR 97Ø45
VOICE: 5Ø3/655-7484 NOON-1ØPM PT TUE-SAT FAX: 5Ø3/655-4116 24HRS
WE ACCEPT VISA/MASTERCARD