Micropendium
P.O. Box 1343
Round Rock, TX 78680

PERIODICALS

A 1/00 T
CHARLES GOOD
P.O. BOX 647
VENEDOCIA OH 45894

Covering the TI99/4A and Geneve home computers

MICROpendium

Volume 16 Number3 May/June 1999 $6

MICROPENDIUM CEASES PUBLICATION

CONTENTS

MICROpendium

MICROpendium (ISSN 10432299) is published bimonthly for $35 per year by Burns-Koloen Communications Inc., 502 Windsor Rd., Round Rock, TX 78664-7639. Periodical postage paid at Round Rock, Texas. POSTMASTER: Send address changes to MICROpendium, P.O. Box 1343, Round Rock, TX 78680-1343.

All correspondence should be mailed to MICROpendium at P.O. Box 1343, Round Rock, TX 78680.

Foreign subscriptions are $42.50 (Canada and Mexico); $40 surface mail to other countries; $52 airmail to other countries.

All editions of MICROpendium are mailed from the Round Rock (Texas) Post Office.

Mailing address: P.O. Box 1343, Round Rock, TX 78680.

Telephone & FAX: (512) 255-1512

Internet E-mail: jkoloen@earthlink.net

Home page: http://www.earthlink.net/~jkoloen/

John Koloen Publisher

Laura Burns Editor

COMMENTS

The end of the road

This is the final edition of MICROpendium.

We knew this day would come and frankly I thought it would come sooner. Although you our readers are loyal, there aren't enough of you to sustain us any longer. Laura and I have been underwriting MICROpendium for a while and can no longer afford it.

This publication has been a big part of our lives over the past 15 years, much bigger than we imagined when volume 1 number 1 was published in 1984. We thought then it might last three, maybe four years. I would never have predicted it would last until 1999.

But as the saying goes, all good things come to an end. Producing MICROpendium has become a physically challenging project as well as a personal financial drain. While we still have hundreds of loyal readers, Laura and I can no longer afford to devote ourselves to the publication. It's time to call it quits and move on to other things.

We do regret that we can't provide refunds for the balance of your subscriptions. There's just no money available to do this. If we'd stopped publishing a year ago, there would have been enough money in the checkbook to make token refunds. But we decided to keep MICROpendium going as long as we could. Thus the checkbook balance declined

Continued on page 4

November 6, 1999

The 17th Annual

Chicago TI International World Faire

SAVE THE DATE

COMMENTS

Continued from page 3

every month until reaching a point where it can decline no further.

I wish we had the foresight to plan for our demise, but it's not something that comes easily. I would have liked this final edition to have been a kind of "survival" guide, wherein we list all the resources, web sites, e-mail addresses, and other information that could be used by our readers to move forward into the 21st century. But I guess it was the denial of the inevitable that allowed us to continue publishing MICROpendium in the first place, and if we'd allowed ourselves to see what was coming we would have quit long ago.

Although we don't anticipate publishing another edition, we are offering back issues at 50 percent off the cover price (see the ad elsewhere in this issue). Funds that we raise now will be used to pay our accountant, hopefully cover most of our debts, and to close down the company. We will fill orders until July 31, 1999, after which we will be completely out of business (the phone will be disconnected). So, if there's anything you want, this is the time to buy it.

THE FUTURE OF TI/GENEVE USERS

My predictive powers are not what they used to be, but it's clear to me that the future for TI and Geneve users lies on the web. A number of user groups maintain useful and active web sites, not to mention publishing electronic newsletters. The Milwaukee User Group and Southwest Ninety-Niners are two that pop into mind. Individual users also maintain sites dedicated to the TI and Geneve. And you're missing the boat if you have access to the Internet and are not subscribing to the TI list-server (ti99@TheRiver.com) or accessing the TI newsgroup (comp.sys.ti).

The web is the one place where TIers can maintain contact, regardless of our marginal TI user groups and businesses become. I imagine that I will continue to use my TI: at least the PC99 version that never has problems with keys that have a habit of repeating or cartridges that overheat and cause the console to lock up.

It's been a great run for us and we are eternally grateful for your support.

— JK

MEETING

TI MUG'99 rescheduled to June 12

Due to circumstances, TI Multi-User Group '99 (TIMUG'99) has been rescheduled to Saturday, June 12. The preconvention get-together will be held 7-10 June 11 at Spang mansion has been cancelled, according to Glenn Bernasek of the TI-Chips user group.

The conference site and hours will remain as originally announced: 7 a.m. to 7 p.m. at Spang Mansion on Kolthoff Drive in Brookpark, Ohio. No smoking is allowed on site.

The TIMUG'99 grand raffle prize will be an Epson inkjet color printer. This will be a "break-even" raffle. Therefore raffle tickets will be sold only up to the cost of the printer.

For more details of TIMUG'99, use the following URL to access Harry Hoffman's web site: http://members.stratos.net/harryhoffy/newsletter/

User groups to be represented at TIMUG'99 include:

Computer Users of Erie
Hoosiers User Group
Lima User Group
Milwaukee Area User Group
Northcoast 99ers
TI-Chips

Schedule of speakers

Here is the schedule of speakers, in alphabetical order:

WHO	WHAT
Dave Connery	Tutoring for new Geneve users
Dan Eicher	GPL Development Kit demonstration
Charles Good	Demonstrating the TI-74
Bruce Harrison	"The End of an Era"
Lew King	Getting the TI-99/4A to communicate with a 56K baud modem
Tony Knerr	SGCPU & HSGPL card demonstrations (will also be available for Geneve consultation)
Ron Markus	Software and hardware offered by Ram Charged Computers
Tim Tesch and Ted Zychowicz	Co-moderating Open Forum for Geneve

There will be two conference rooms — one for TI99/4A sessions and one for Geneve sessions.

File transfer and printing with a PC, Hyperterminal, and a TI

BY ROGER PRICE

Was reading Charlie Good's assessment of transferring text files (D/V80) directly between TI-Writer and a PC using term.exe. He says that he was not able to use Hyperterminal to do the transfers and so suggests term.exe that no one has. Also, he says that he cannot send data from the PC to a 99/4A console as he does on a Geneve.

I decided to tackle this as I know I have used direct xmodem transfers from and to my PC many times via Hyperterminal without using a modem. Since many of the PC users have Hyperterminal this will also be the same setup as used with xmodem to download files for V9T9. One mistake maybe that he used 7 bit, odd, parity. Most of the time file transfers use 8-n-1. So, the setup for Funnelwriter (TI-Writer), I used is: RS232.BA=4800.PA=N.DA=8. You can experiment with various baud rates but 4800 is plenty fast. Make sure the baud is the same on the receiving end. Use LF then Enter, put the RS232 information in to send and receive in TI-Writer.

Hyperterminal: To find it, go to Start, Programs, Accessories, Hyperterminal. If you have never used Hyperterminal you will have to make a new setup. Do this by clicking on new connection. Use the same settings as revising a setup. Revise a set up by clicking on the bar under help on the icon that looks like a hand delivering a letter. It will show up as properties if you leave the pointer on it for awhile. Click on settings: I have it set on: Use terminal keys, Emulation: Auto detect, Backscroll buffer: 500.

The other screen you need to set only one item, very important. Click on the down arrow at Connect Using: Direct To COM1 Clicking on the down arrow gives several selections.

After making the selection the other items in this window turn gray showing that they are not in effect. Near the bottom click on configure.

Configure: Then Port Settings: 4800, 8, None, 1, Hardware. Then click OK at the bottom.

See the various selections by clicking on the down arrow at the right of the selection.

Emulation: Viewdata, I am not sure if all of the setup selections have an effect on the direct transfers.

Send From The TI: Have the file loaded in TI-Writer, select SF, then enter, put the RS232 information in if it does not already show up from previous use. Before you hit the enter key have the cable plugged in, start Hyperterminal with the setup as above. You should see the Hyperterminal window with a smaller black square box inside of it. Inside the black window you should see a cursor line blinking. This is where the text will appear.

To save the file to disk click on Transfer. Click on Capture Text. You will get a window to show where you want the information to be saved. I put: a:\filename.txt Then press the enter key on the TI and the text will show up in the black box as it is loaded and saved to the disk. When it seems as if the file is done, you must go back to Capture Text and click on "stop" to close the file and reset the PC or the file will not be usable and your next download will not work right.

If you want to just use your PC to print a file or a disk catalog, select Capture Print. When you click on it you will think that nothing is happening. It is waiting for the data. Start the TI data with the enter key and your PC will send the information to your printer to print out. When it does one page it will stop. Click on Capture Print again and it will print out another page. When done printing, click on Capture Print again to close the printer or your next print will not start (printer busy error). You can also save the text in the black window with copy and paste it into a word processor to view, print out, or save to disk.

Now the real bonus! You can use Archiver or a word processor that sends to RS232 to print out your disk catalog. If you never had a printer on your TI, but you do on your PC, you now have a printer for both. Print out those disk catalogs with Archiver. Just put in the same RS232 code as before (RS232.BA=4800.PA=N.DA=8.) and watch those disk catalogs come out just about as fast as you can load and change the disk. Remember to use the same Capture Print and make sure you click on it again after each print.

Send From PC To TI: Have the file on a 1.44mb floppy or a hard drive. You must prepare your file carefully or it will not load. One of the things you do not want to do is to use the enter key. This will really mess up the file. Save everything as ASCII or text. Just use word wrap and blank spaces to make the article. Also, if the first line loads and it seems to stop, do nothing to the PC, just type LF on the TI and enter and, since your RS232 info should be there, just hit enter again and the rest of the file will load. You will lose the first line or maybe several lines. There may be a way to append the file if it goes in two or three segments. I have not explored this yet. If the last line is not 80 characters it will not load either.

Start the TI setup first. LF, then enter, put the RS232 information if not there, then enter. The computer will seem to lock up, just setting there waiting for the data to flow to it.

Start Hyperterminal, click on Transfer, then Send Text File. You will then be sent to the browser to find the file you want to send. Click on

Continued on page 8

FILE TRANSFERS

Continued from page 7

3.5 floppy A, the files are shown on the menu. Click on the text file to select, then Open. The file will be shown inside the black PC screen as it is sent to the TI. A line counter will quickly increment on the top right side of the screen on the TI as the lines are loaded. One thing that will abort the process is data that the TI does not understand or expect to receive. A Wordpad file in rich text format (RTF) or Microsoft Word 6 format will not work. I use a DOS program called Easy Writer to type the text file on the PC.

Once the file is loaded onto the TI, press FCTN 4 (break), then press E (edit) enter to see the text on the TI screen. I spent several hours checking to find the combination to make this work right. I knew it should work if the setup was right. Pinouts for a laptop are different than a standard PC to a TI. Only four or five wires are actually needed. The TI computer RS232 serial manual shows the following:

1	ground
2	RD-data in
3	TX-data out
5	cts-clear-to-send
6	DSR-data-set-ready
7	SG-signal-ground
8	DCD-data-carrier-detect
20	DTR-terminal-ready

My laptop manual shows, 2-RD, 3-TX. The point is that Charlie has No. 2 going to No. 2 and No. 3 to No. 3. On a standard PC with a 25-pin connector, No. 2 is TX and No. 3 is RD. This is opposite of a laptop. The RD pin must go to the TX pin. So on a laptop you must have 2-3 and 3-2. I have a hard time finding pin 20 on my laptop as there are only nine pins. I think Charlie's pinouts are for a standard PC but he was talking about his laptop. Perhaps an old laptop may have a 25-pin connector.

My conclusion is that from a standard TI-99/4A console you can do a lot with transferring to a PC or a printer. File transfer from the PC to the TI by Hyperterminal is not as useful because of the critical file structure needed for it to work. I want to Thank MICROpendium and Charlie Good for the article that got me going on this. Hopefully this article will help others without a printer on their TI.

This is but one of at least three ways to transfer a file. You can also use xmodem or PC Transfer. Need help. Send me an e-mail. bumblebe@comteck.com or pricerogerl@hotmail.com.

K'town columnist Buehler dies

Bob Buehler of Knoxville, Tennessee, a member of the K'town 99ers, died March 27. He was 89. According to the Rev. John Bull of the K'town 99ers, he had been in poor health for about five years.

He was the author of the "Kinder Korner" and "Chatterbox" monthly columns in the group's newsletter, the K'town 99'er.

THE ART OF ASSEMBLY 76

New and Improved

BY BRUCE HARRISON

We've said before in this column that even after years of making various programs in Assembly language we are still able to learn new tricks and improve things. Today's column is about old versus new ways of doing various operations. Most are in the form of subroutines, so there's no complete program here, but it should be a learning experience and provide better ways to do some little things.

Screen Clearing

As with so many things, there are many ways to do the screen clearing operation. For example, one can use VSBW 768 times, incrementing R0 each time. That works, but it's the very slowest method. In many of our programs, we've used a line in the DATA section of the program called BLNKLN, which is 32 or 40 spaces (depending whether we're in 32- or 40-column mode) and is used with a loop using VMBW 24 times. This method also works, of course, and is much faster than the VSBW method, but it takes up more memory both in the code segment and in the DATA part.

NOTE OF CAUTION: The Boot Tracking shown here in Sidebar 76 works with floppy drives and Ramdisk drives **only**! It does **NOT** work for SCSI drives.

While working on the improved version of Load Master, we came up with the "new and improved" method that uses very little memory, requires no DATA, and is faster than any other method. It affects only registers 0, 1, and 2 of your workspace, and can be modified slightly to provide other services such as partial screen clearing operations. What we've shown in the sidebar is a very simple screen clear that can be used in the "normal" Editor/Assembler environment, plus slightly modified versions for use in the BASIC and Extended BASIC environments, and so on. Take your pick, but be assured any of these new subroutines will do the job as quickly as possible and in a minimal amount of memory. We've even included a version that can do an "HCHAR" type service many times faster than the XB version. Take care, though, that your calling code does not make this subroutine go beyond your screen area, as no provision has been made here for the "wraparound" to the top of the screen as HCHAR does.

Continued on page 10

THE ART OF ASSEMBLY 76

Continued from page 9

These routines work faster than the old ones mainly because we've taken full advantage of the auto-incrementing feature when writing to VDP RAM via >8C00. Thus once we've written the starting VDP address to >8C02, we don't have to write an address again, as was the case when we used BLWP to either VSBW or VMBW. Rather, every time we MOVB @>8C00, the VDP address increments all by itself. We also save some time by avoiding the BLWP "overhead" in favor of the less time consuming BL.

Slightly Better Keyscan

In our own "normal" key routine, we need somewhere in the DATA a byte of >20 (ANYKEY BYTE >20) so that we can compare the value at >837C (GPL Status byte) to ANYKEY, and repeat scan if that's not equal. Here's another method, which works just as well but doesn't need the byte of >20 in DATA. This, as shown in the sidebar, does the LIMI 2 and LIMI 0 operations, so that waiting for a keystroke won't stop anything that needs the interrupts, such as sprite motion or background music. That has the disadvantage of enabling the FCTN-= to get back to the title screen. If you don't need the interrupts serviced, just use this without the lines LIMI 2 and LIMI 0.

Slightly Better Speed

You can't do this in stuff that links from Extended BASIC, but in pure Assembly programs, speed of execution overall can be slightly improved by placing your workspace registers in the >8300 (RAM Pad) area. Since that part of memory is accessed on a 16-bit word basis rather than byte-by-byte, all register actions will happen slightly faster.

To give a concrete example, we recently took a modified (by Bob Carmany) version of Shawn Baron's AMSTEST program, which runs an exhaustive test on the AMS card, and instead of having the registers in the program code as BSS blocks, we put in EQUates so that the main program registers were at >8300 and the utilities' registers at >8320. With no other changes, the execution time for this program on our 256K AMS card went from 2 minutes 25 seconds to 2 minutes flat. That's not a big improvement, (about 20 percent) but points in the right direction.

Much Better Boot Tracking

Last but not least, for those using Assembly routines with Extended BASIC, here's a much improved method for finding out what disk drive an XB program loaded from, then modifying all occurrences of "DSK1" in that program to whatever device it loaded from. This will work for "non-DSK" devices as well, including those starting with WDS (old hard drives) and SCS (new SCSI hard drives.)

The impetus for this new version of our boot-tracking routine came from our experience in modifying Load Master. One of the things we did was to add

THE ART OF ASSEMBLY 76

boot-tracking to the LOAD program, so that Load Master could be started from any disk drive and would always "know" where the Load Master disk was located. In our early attempts, we found that while our boot-tracking worked just fine on our own system's drives, both the floppies and the RAMdisks, it would cause a lockup on Mickey Cendrowski's system. Mickey has a Myarc HFDC as her main disk controller, which handles both her floppy and hard drives. That old version of boot-track used a particular method of turning the ROM on and off, which worked okay for most disk controllers, but not for the Myarc HFDC.

In recent work, we'd done some things with CRU access on the AMS card, and in that case used SBO and SBZ operations to turn the card's ROM on and off. It looked as though that might be a good idea for the on-off switching required in the boot-tracking case, so we tried that. Yes, we now had a means of boot-tracking that worked with every kind of controller that it was tested on. This means it worked on TIs with Myarc disk controllers, on Geneves, and even with the SCSI controller on Charlie Good's Geneve. As you can see in the sidebar, once we've set R12 to the CRU address for the controller's ROM, we simply SBO 0 to turn the ROM on so we can read it, then SBZ 0 to turn the ROM off again.

This new version of boot-track also has the ability to work with programs that contain a zero byte in the middle of a program line, so long as that's after a >C9 or >C7 token. This is a bit esoteric, but hang in there a moment. Suppose a line contains IF XYZ THEN 40. When this is tokenized, the line number 40 becomes (in hex) C9,0,28. In early versions of our boot-track, that byte of 0 following the C9 would be mistaken for the 0 that ends a program line, so any DSK1 that came later in that line would get skipped. A similar problem can occur when a program sets a string variable to null value. Let's say the line includes A$="". That "" will become tokenized as >C7,0, which again raises the possibility of a false ending to the line.

In the version of boot-track shown in the sidebar, both of these problems are taken care of, so that if >C9 or >C7 is found, the comparison process will skip over that 0 byte and thus will correctly find DSK1 anywhere in any XB program line. As in older versions, wherever DSK1 is found, it will be changed to the device name found by the earlier part of the routine.

Using the Track6 Routine

Perhaps it's best to describe by example. In the case of Load Master, the LOAD program has this code embedded by Todd Kaplan's ALSAVE routine. Let's suppose that we've copied LOAD, LOADMASTER, DEFAULTS, OPTIONS, and the mysterious EA file into the root directory of a hard drive named SCS2 (a SCSI drive). We then get into Extended BASIC and type RUN "SCS2.LOAD"

Continued on page 12

Continued from page 11

<Enter>. When LOAD starts, it first dumps all of its Assembly routines, including TRACK, into low memory. There's a line in LOAD that says RUN "DSK1.LOADMASTER", but before that line executes, there's one that says CALL LINK("TRACK"). In that CALL LINK, the TRACK routine gets the device name SCS2 from the controller ROM, then searches the XB part of LOAD (in memory) until it finds DSK1. It changes DSK1 to SCS2, so that instead of RUN "DSK1.LOADMASTER", that line says RUN "SCS2.LOADMASTER". When that line executes, LOADMASTER is loaded and run from SCS2.

When LOADMASTER starts running, it too executes a CALL LINK ("TRACK"). This time the code in TRACK sees that it already has the device name, because its R0 is non-zero, so it jumps ahead to the part that scans the XB content of LOADMASTER, where it replaces all occurrences of DSK1 with SCS2. Now if the user selects OPTIONS, that program will be sought on SCS2 as well. OPTIONS, in turn, will also perform a CALL LINK("TRACK"), and thus will look for DEFAULTS on SCS2. When OPTIONS is done, it will save the revised DEFAULTS and then re-load LOADMASTER from SCS2, and so on.

There are places in the OPTIONS file where we want the four characters "DSK1" to stay "as-is" when TRACK is running. To make sure of that, we put DSK1 in as a string expression like this: D$="DSK"&"1". Having that ampersand in the middle ensures that TRACK won't find DSK1 in four successive bytes, and thus will not replace it with SCS2. This same trick can be applied in your own work if "DSK1" is supposed to stay that way. In this set of programs, only LOAD contains a CALL INIT, so that the code for TRACK remains in place and available while any of these programs is running. There are additional Assembly routines included in LOADMASTER, but those are embedded using Harry Wilhelm's High Memory Loader, and so don't affect the routines that LOAD placed in Low memory.

To summarize, use TRACK6 this way:

1. Embed TRACK6/O using ALSAVE in the "main" program.

2. Make sure CALL LINK("TRACK") is performed before any RUN "DSK1.XXXX" lines in that program.

3. In chained programs, (e.g. LOADMASTER) perform CALL LINK ("TRACK") early in the program. Make sure the chained programs don't perform CALL INIT.

For your convenience if you get MICROpendium on disk, we've included TRACK6/O as an object file in our. Otherwise, just type in the section of the sidebar that's source for TRACK6 and assemble that so it can be embedded with ALSAVE. If you need help with any of this, don't hesitate to call us any time from 9AM through Midnight Eastern time at (301) 277-3467.

Again our next topic is undecided. See you in two months.

SIDEBAR 76

```
* SIDEBAR 76
* NEW AND IMPROVED SUBROUTINES
* all Public Domain
*
* CLEAR THE SCREEN - SIX NEW VERSIONS
*
* 1. "PLAIN-JANE" VERSION FOR 32 CHAR SCREEN IN E/A
*
CLS     LI   R0,>0040      SET >40 IN LOW BYTE R0
        LI   R1,>2000      SPACE IN LEFT BYTE R1
        LI   R2,768        768 TIMES (32 * 24)
CLS1    MOVB R0,@>8C02     SEND HIGH BYTE
        SWPB R0            SWAP
        MOVB R0,@>8C02     SEND >40 BYTE
CLS2    MOVB R1,@>8C00     WRITE A SPACE
        DEC  R2            DECREMENT COUNT
        JNE  CLS2          IF NOT ZERO, REPEAT
        RT                 ELSE RETURN
*
* 2. "PLAIN-JANE" VERSION FOR 32 CHAR SCREEN IN XB
*
CLSXB   LI   R0,>0040      SET >40 IN LOW BYTE R0
        LI   R1,>8000      SPACE W/OFFSET IN LEFT BYTE R1
        LI   R2,768        768 TIMES (32 * 24)
CLS1    MOVB R0,@>8C02     SEND HIGH BYTE
        SWPB R0            SWAP
        MOVB R0,@>8C02     SEND >40 BYTE
CLS2    MOVB R1,@>8C00     WRITE A SPACE
        DEC  R2            DECREMENT COUNT
        JNE  CLS2          IF NOT ZERO, REPEAT
        RT                 ELSE RETURN
*
* 3. "PLAIN-JANE" VERSION FOR 40 COL SCREEN IN E/A
```

Continued on page 14

THE ART OF ASSEMBLY 76

Continued from page 13

```
*
CLS40   LI    R0,>0040      SET >40 IN LOW BYTE R0
        LI    R1,>2000      SPACE IN LEFT BYTE R1
        LI    R2,960        960 TIMES (40 * 24)
CLS1    MOVB  R0,@>8C02     SEND HIGH BYTE
        SWPB  R0            SWAP
        MOVB  R0,@>8C02     SEND >40 BYTE
CLS2    MOVB  R1,@>8C00     WRITE A SPACE
        DEC   R2            DECREMENT COUNT
        JNE   CLS2          IF NOT ZERO, REPEAT
        RT                  ELSE RETURN
*
* 4. "FANCY" VERSION FOR 32 OR 40 CHAR SCREEN IN E/A
* USE BL @CLS32 FOR 32, BL @CLS40 FOR 40
*
CLS32   LI    R2,32*24      768 FOR 32 CHAR
        JMP   CLS0
CLS40   LI    R2,40*24      960 FOR 40 CHAR
CLS0    LI    R0,>0040      SET >40 IN LOW BYTE R0
        LI    R1,>2000      SPACE IN LEFT BYTE R1
CLS1    MOVB  R0,@>8C02     SEND HIGH BYTE
        SWPB  R0            SWAP
        MOVB  R0,@>8C02     SEND >40 BYTE
CLS2    MOVB  R1,@>8C00     WRITE A SPACE
        DEC   R2            DECREMENT COUNT
        JNE   CLS2          IF NOT ZERO, REPEAT
        RT                  ELSE RETURN
*
* 5. "FANCIER" VERSION FOR 32 OR 40 CHAR SCREEN IN E/A
*  IN THIS VERSION, YOU CAN CLEAR ANY SELECTED AREA
*  OF SCREEN BY FIRST SETTING R0 TO THE START POINT
*  AND R2 TO THE NUMBER OF SPACES TO WRITE, THEN
*  BL  @CLS0A
*
CLS32   LI    R2,32*24      768 FOR 32 CHAR
```


THE ART OF ASSEMBLY 76

```
        JMP   CLS0
CLS40   LI    R2,40*24      960 FOR 40 CHAR
CLS0    CLR   R0
CLS0A   ORI   R0,>4000      SET >4000 BIT IN R0
        SWPB  R0
        LI    R1,>2000      SPACE IN LEFT BYTE R1
CLS1    MOVB  R0,@>8C02     SEND HIGH BYTE
        SWPB  R0            SWAP
        MOVB  R0,@>8C02     SEND >40 BYTE
CLS2    MOVB  R1,@>8C00     WRITE A SPACE
        DEC   R2            DECREMENT COUNT
        JNE   CLS2          IF NOT ZERO, REPEAT
        RT                  ELSE RETURN
*
*  6. FANCIER YET VERSION:
*  IN THIS VERSION, YOU CAN CLEAR ANY SELECTED AREA
*  OF SCREEN BY FIRST SETTING R0 TO THE START POINT
*  AND R2 TO THE NUMBER OF SPACES TO WRITE, THEN
*  BL  @CLS0A, BUT YOU CAN ALSO PUT ANY CHARACTER
*  ASCII YOU WANT IN R1'S LEFT BYTE, THEN THE STARTING
*  SCREEN ADDRESS IN R0 AND NUMBER OF REPEATS IN R2,
*  THEN BL @CLS0B TO DO A VERY SWIFT "HCHAR" TYPE
*  OPERATION.
*
CLS32   LI    R2,32*24      768 FOR 32 CHAR
        JMP   CLS0
CLS40   LI    R2,40*24      960 FOR 40 CHAR
CLS0    CLR   R0
CLS0A   LI    R1,>2000      SPACE IN LEFT BYTE R1
CLS0B   ORI   R0,>4000      SET >4000 BIT IN R0
        SWPB  R0
CLS1    MOVB  R0,@>8C02     SEND HIGH BYTE
        SWPB  R0            SWAP
        MOVB  R0,@>8C02     SEND >40 BYTE
CLS2    MOVB  R1,@>8C00     WRITE A SPACE
```

Continued on page 16

THE ART OF ASSEMBLY 76

Continued from page 15

```
       DEC  R2             DECREMENT COUNT
       JNE  CLS2           IF NOT ZERO, REPEAT
       RT                  ELSE RETURN
*
* IMPROVED KEY SCAN SUBROUTINE, DOESN'T REQUIRE
* EXTERNAL DATA FOR OPERATION
* DOES REQUIRE THAT KSCAN BE REF'D OR EQU'D
*
KEY    BLWP @KSCAN         SCAN KEYBOARD
       LIMI 2              ALLOW INTS
       LIMI 0              STOP INTS
       MOVB @>837C,@>837C MOV GPL STAT BYTE TO ITSELF
       JEQ  KEY            IF 0, SCAN AGAIN
       RT                  ELSE RETURN
*
* TRACK6/S  - SOURCE CODE FOR CHANGING
* DEVICE NAME IN AN XB PROGRAM
* FINDS DSK1 ANYWHERE IN XB PROGRAM
* AND CHANGES IT TO XXXX, WHERE XXXX IS THE
* DEVICE FROM WHICH XB PROGRAM WAS LOADED
* EVEN WORKS FOR NON-DSK DEVICES, e.g. WDS1, SCS1
* 18 DEC 1996
* PUBLIC DOMAIN
* CODE BY Bruce Harrison
*
       DEF  TRACK
TRACK  LWPI WS             USE OUR WORKSPACE
       MOV  R0,R12         USE OLD CRU ADDR IF AVAILABLE
       JNE  GET30          THEN JUMP AHEAD
GETD0  MOV  @>83D0,R12     GET THE CRU BASE IN R12
       JEQ  EXIT           GET OUT IF 0
       MOV  @>83D2,R9      GET THE ROM ADDRESS FOR DEVICE
       JEQ  EXIT           GET OUT IF 0
       SBO  0              ENABLE THE DEVICE ROM
       AI   R9,5           ADDING FIVE PUTS US AT DEVICE NAME
```


THE ART OF ASSEMBLY 76

```
       LI   R4,4           4 BYTES TO GET
       LI   R10,TEXT+1     POINT TO TEXT BUFFER+1
MOVIT  MOVB *R9+,*R10+     MOV ONE BYTE FROM ROM TO TEXT BUFFER
       DEC  R4             FINISHED?
       JNE  MOVIT          NO, DO ANOTHER BYTE
      SBZ  0             DISABLE THE ROM (R4 IS ZERO AT THIS POINT)
       MOV  R12,R0         SAVE R12 IN R0
GET30  MOV  @>8330,R13     PUT START OF LINE NUMBER TABLE IN R13
NEWLI  INCT R13           POINT TO BYTE CONTAINING ADDRESS OF LINE
       C    R13,@>8332     ARE WE PAST END OF LINE NUMBER TABLE
       JGT  EXIT           IF SO WE ARE FINISHED
      MOVB *R13+,R4      GET HIGH ORDER BYTE OF LINE ADDRESS IN R4
       SWPB R4             SWAP R4
       MOVB *R13+,R4       GET LOW ORDER BYTE OF LINE ADDRESS
       SWPB R4             SWAP SO R4 CONTAINS STARTING ADDRESS OF
A LINE
*
* AT THIS STAGE R4 POINTS TO THE BEGINNING OF A LINE IN THE XB
PROGRAM,
*
NEXT
       MOV  R4,R10         SET R10 EQUAL TO R4
CHECK
       LI   R9,DSK1        POINT AT TEXT 'DSK1'
       LI   R3,4           SET FOR 4 CHARACTER COMPARE
CMPB   CB   *R10,@CEE9     CHECK FOR C9 TOKEN
       JNE  CMPC7          IF NOT EQUAL, JUMP AHEAD
       INCT R4             ELSE MOVE R4 POINTER AHEAD TWO
       JMP  NOCMP          THEN JUMP AHEAD TO NOCMP
CMPC7  CB   *R10,@CEE7     IS THIS A BYTE OF >C7
       JNE  CMPC           IF NOT, JUMP
       INC  R4             ELSE INCREMENT POINTER BY ONE
       JMP  NOCMP          THEN JUMP
CMPC   CB   *R10,@ZERO     IS THE BYTE WE'RE LOOKING AT A ZERO?
       JEQ  NEWLI          IF SO, IT'S THE END OF A PROGRAM LINE
```

Continued on page 18

THE ART OF ASSEMBLY 76

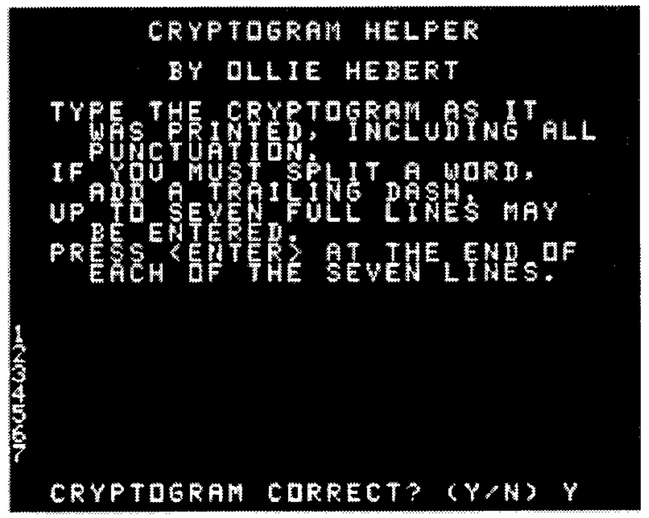
Continued from page 17

```
        CB    *R9+,*R10+    COMPARE BYTES AND INCREMENT
        JNE   NOCMP         IF NOT EQUAL, GET OUT
        DEC   R3            ELSE DECREMENT COUNT
        JNE   CMPB          IF NOT ZERO, REPEAT
        LI    R9,TEXT+1     DSK1 WAS FOUND. POINT TO BOOT TRACKED
DEVICE NAME
        MOV   R4,R10        R10 POINTS TO LOCATION WHERE "DSK1" WAS
FOUND
*
* THE LOOP AT MOV2 OVERWRITES "DSK1" IN THE XB PROGRAM LINE
* WITH DEVICE NAME FOUND IN THE BOOT TRACK PROCESS
*
        LI    R5,4          FOUR BYTES TO MOVE
MOV2
        MOVB  *R9+,*R10+    MOVE ONE, INCREMENT POINTERS
        DEC   R5            DECREMENT COUNTER
        JNE   MOV2          IF NOT ZERO, REPEAT
        MOV   R10,R4        START OF NEXT GROUP OF BYTES
        JMP   CHECK         JUMP BACK
NOCMP   INC   R4            GO START AT NEXT BYTE IN XB PGM LINE
        JMP   NEXT          AT LABEL NEXT
EXIT    LWPI  >83E0         LOAD GPL WORKSPACE
        B     @>006A        RETURN TO GPL INTERPRETER
WS      DATA  0             R0 STARTS AS ZERO
        BSS   30            REST OF WS
TEXT    BYTE  4             LENGTH OF DEVICE NAME
        BSS   5             BUFFER FOR DEVICE NAME
ZERO    DATA  0             ZERO BYTE FOR COMPARISON
DSK1    TEXT  'DSK1'        COMPARISON TEXT
CEE9    BYTE  >C9           TOKEN FOR LINE NUMBER
CEE7    BYTE  >C7           TOKEN FOR QUOTED STRING
        END
```


EXTENDED BASIC

Cryptograms

BY OLIVER HEBERT



Cryptograms are letter substitution puzzles: all occurrences of each letter (for example, C), have been replaced with another letter. In the example, any letter except C could be the replacement used for all of the Cs. The encryption XBQX might be decrypted as "THAT." You solve the puzzle by figuring out what letter each encrypted letter decrypts to.

Cryptograms are commonly found in newspapers near the other puzzles, such as crossword, scrambled words, and word search. They are also found in some word puzzle books, and as books containing only cryptograms. Cryptograms may or may not provide the first letter substitution as a clue. The screen dump shows a cryptogram that I was unable to solve before writing this program. Have a go at it!

This program won't decipher cryptograms for you, but it can be a timesaver (compared to the usual method of pencil in, then erase) in finding solutions to them. The Extended BASIC subprograms CHAR and CHARPAT are used as the heart of this program. The encrypted message (cryptogram) is entered from the keyboard using uppercase letters and punctuation. The decrypted message (solution) is made to be similar to the original cryptogram: punctuation remains unchanged, but the letters are changed to lowercase. All lowercase letters are then redefined as spaces so that you can't see them. Each time the user selects a letter and its decryption, CHARPAT and CHAR updates the screen.

Statements 100-120 are the set-up. DISPLAY ERASE ALL (which uses three bytes) only seems to be longer than CALL CLEAR (eight bytes), so five bytes were saved. With limited memory available, we should be aware of ways to conserve it.

The screen is set to dark blue with the characters (except the cursor) set to white. When ASSIGNING a logical TRUE or FALSE, either -1 (if TRUE)

Continued on page 20

CRYPTOGRAM

Continued from page 19

or 0 (if FALSE) is made. When TESTING for TRUE or FALSE, any non-zero number is treated as TRUE. If (J=0) is TRUE, it is assigned -1, so 16+14*(-1)=2 (black foreground) for character set 0 (cursor). If (J=0) is FALSE, it is assigned 0, so 16+14*(0)=16 (white foreground) for character sets 1-12.

The CALL KEY(3) seems to be unused, but it, until changed, causes the keying in of lowercase letters (in both ACCEPT and KEY inputs) to be automatically shifted to uppercase. It also eliminates five seldom-used function keys.

Array E$(0-6) contains the seven segments of the cryptogram as entered in uppercase by the user (ENcrypted). Array D$(0-6) is similar to E$(), but its alpha characters are in lowercase (DEcrypted). LE$ contains (in alphabetical order) all of the

Cryptogram helper

Type the cryptogram as it appears below, including all punctuation. If you must split a word, add a training dash. Up to seven full lines may be entered. Press Enter at the end of each of the lines.

```
1 BZLWI-IZCF PJCFMKXFLW ATU
2 TAVUJNHFSQF BZUTUAZTH
3 TWWZWITUAF BLJC ZU-HTNW MX
4 WTXZUQ: "IPTUVW T HJI!"
5
6
7
CRYPTOGRAM CORRECT? (Y/N) Y
```

For the following example, the display is in groups of two lines where the upper line is the letters that you choose and the lower line is the cryptogram as entered. The final group shows the letters used in the cryptogram and is followed by a message. To use, type a letter from the bottom row, then type the letter that you think it represents. The most frequently used letters are reported to be E, T, A, I, N, S, H, R, L, D, and U. It is not necessary to use the Enter key.

```
READY TO BEGIN? (Y/N) ::
F     -
BZLWI-IZCF PJCFMKXFLW ATU
----------F-----------------
TAVUJNHFSQF BZUTUAZTH
----------F-----------------
TWWZWITUAF BLJC ZU-HTNW MX
------: "-----------!"------
WTXZUQ: "IPTUVW T HJI!"
-----------------------------
-----------------------------
-----------------------------
 F
ABC  F HIJKLMN PQ STUVWX Z
ABCFHIJKLMNPQSTUVWXZ?
```


CRYPTOGRAM

ENcrypted letters used in the cryptogram along with spaces for any unused letters. TE$ (ENcrypted) is a duplicate of LE$ but all spaces have been removed. LD$ and TD$ are DEcrypted equivalents of LE$ and TE$. All other variables are of general use (and re-use).

Statement 120 finishes the prescan requirements, and prescan is turned off for faster program startup.

Solutions

```
FIRST-TIME HOMEBUYERS CAN
BZLWI-IZCF PJCFMKXFLW ATU
-------------------------
ACKNOWLEDGE FINANCIAL
TAVUJNHFSQF BZUTUAZTH
---------------------
ASSISTANCE FROM IN-LAWS BY
TWWZWITUAF BLJC ZU-HTNW MX
--------------------------
SAYING: "THANKS A LOT!"
WTXZUQ: "IPTUVW T HJI!"
-----------------------
-----------------------
-----------------------
-----------------------
CFM  E LTOURBW HG DANKSY I
ABC  F HIJKLMN PQ STUVWX Z
IS CRYPTOGRAM SOLVED? (Y/N)

         B  -
VTRZIVXSI AVW-FXZWKX DVIIXQ
      B                  B
YGKFB AX YDZTRVSW ZI IDX AZI
QVWDI SGO:
----------
----------
----------
----------
B
AB D FG I K   O QRST VWXYZ
ABDFGIKOQRSTVWXYZ?
```

SUBroutines: Line 130 converts R$ (uppercase and punctuation) into S$ (lowercase and punctuation). Characters from 65 (letter A) to 90 (Z) are changed to lowercase by adding 32 to their value (a=97, z=122). SUB 140 is called to supply a honk tone when an unacceptable KEY is pressed. SUB 150-170 flashes the cursor and gets, tests, and displays a keypress. If the key pressed isn't contained in R$ (a test for acceptable keypresses), you get a honk. The flashing cursor adds a nice look to the screen, but it does cause some keypresses to be missed. Without the flashing cursor, you probably wouldn't always know what the program is expecting, so it is an acceptable trade-off.

Statements 180-220 create screen one and give some instructions. This is where you fill E$(0-6) with the ENcrypted cryptogram. If you make a typo, you get to correct it (SUB 150) before proceeding. ERASE ALL in 180 seems unnecessary, but this screen is accessed from other places in the program, so is needed.

Statements 230-330 create screen two: more instructions. There is a delay while the cryptogram is being pre-processed.

In 270, all seven input lines are put into variable R$. In

Continued on page 22

CRYPTOGRAM

Continued from page 21

280, if R$="" (all lines were blank), you are returned to screen one. Variable LE$ is set to 26 spaces. Lines 290-300 fill LE$ with each of the letters used in the cryptogram. These letters are positioned alphabetically (A would be leftmost and Z rightmost). LE$ is sent to SUB 130 to change it's letters to lowercase for use as LD$.

In line 310, TE$ (which starts out as =LE$) is stripped of any spaces. TD$ is set to the quantity of spaces equal to the number of characters in TE$. Line 320 sends uppercase E$(0-6) to SUB 130 which returns lowercase for D$(0-6). Characters 96-123 (lowercase) are then set to spaces. Line 330 either clears the screen or restarts the program dependant upon your response.

Statement 340 creates screen three, where the cryptogram is to be solved. For the seven segments of the cryptogram, each upper row, D$(), shows only the punctuation (as it hasn't yet been DEcrypted), and each lower row, E$(), shows the ENcrypted version. The third from the bottom row displays LD$ (DEcrypted letters, which are still spaces). The next to the bottom row displays LE$ (ENcrypted letters used), spaced appropriately.

Statements 350-390 process your keystrokes as you work toward a solution. Line 350 displays (on the bottom row) the letters used in the cryptogram (TE$), a question mark, and SUB 150 supplies the flashing cursor. After an acceptable keypress (the characters in TE$), variable L stores the keystroke for future use. Variable C (column) is set according to the position of S$ (the letter keypressed) within LE$ (letters and spaces).

Line 360 sets R$ (acceptable keypresses) to be a space and all uppercase alphas. The space is so that you can erase DEcryptions that seem to be incorrect. In TD$, the character in the position being worked on (variable M), is replaced with a space. This prevents a rejection if you decide that the previous DEcryption really was correct, and re-use the same letter.

Line 370 gets a keypress, and LD$ overwrites the keypress because further testing must be done to prove it's acceptability. If the DEcrypted letter is the same as the ENcrypted letter (J=L) or if the DEcrypted letter has already been used (TD$ contains S$), you get a honk (SUB 140) and must enter a different DEcrypted letter.

The character numbers for lowercase letters are numbered 32 higher than uppercase, so, to convert to lowercase, we add 32 to uppercase.

Statement 380 is the heart of the program. If a DEcrypted space (32) was entered, ENcrypted character L+32 is set to a space. Otherwise, the CHARPAT is done on variable J and used to CALL CHAR(L+32, which instantly updates all occurrences of that character that are displayed on the screen.

CRYPTOGRAM

Line 390 updates TD$ by putting S$ in at the proper place, then TD$ is tested for a space character. If one is found, go back to 350 for another set of keypresses, else the puzzle is complete as far as the rather ignorant program knows. Complete, however, doesn't necessarily mean solved, so 400 asks the human user if it really is solved, and takes appropriate action. Line 410 asks if you want to do another cryptogram, and either ends or restarts the program.

CRYPTOGRAM

```
100 ! CRYPTOGRAM                v1.0
  Oliver D. Hebert              PUBLIC
  Rt. 4, Box 23                 DOMAIN
  Brewton, AL 36426             XBASIC
  Ph: 205 867-7193
110 DISPLAY ERASE ALL :: CAL
L SCREEN(5):: DIM D$(6),E$(6
):: FOR J=0 TO 12 :: CALL CO
LOR(J,16+14*(J=0),1):: NEXT
J :: CALL KEY(3,K,L)
120 GOTO 180 :: CALL CHAR ::
 CALL CHARPAT :: CALL SOUND
:: CALL VCHAR :: C,LD$,LE$,M
,R,R$,S$,TD$,TE$ :: !@P-
130 S$="" :: FOR K=1 TO LEN(
R$):: L=ASC(SEG$(R$,K,1)):
S$=S$&CHR$(L-32*(L>64 AND L<
91)):: NEXT K :: RETURN
140 CALL SOUND(2E2,220,5)::
RETURN
150 CALL SOUND(2E2,14E2,5)
160 M=30-2*(M=30):: DISPLAY
AT(R,C):CHR$(M);:: CALL KEY(
3,J,K):: IF K THEN S$=CHR$(J
)ELSE 160
170 IF POS(R$,S$,1)THEN DISP
LAY AT(R,C):S$;:: RETURN ELS
E GOSUB 140 :: GOTO 160
180 DISPLAY AT(1,6)ERASE ALL
:"CRYPTOGRAM HELPER": :"
  BY OLLIE HEBERT": :"TYPE T
HE CRYPTOGRAM AS IT     WAS
PRINTED, INCLUDING ALL  PUN"
190 DISPLAY AT(7,6):"CTUATIO
N.":"IF YOU MUST SPLIT A WOR
D,     ADD A TRAILING DASH."
:"UP TO SEVEN FULL LINES MAY
    BE ENTERED.":"PRESS <EN"
200 DISPLAY AT(12,10):"TER>
AT THE END OF":"  EACH OF TH
E SEVEN LINES.": : : : : : :
 : : : :"CRYPTOGRAM CORRECT?
 (Y/N) Y" :: FOR J=0 TO 6
210 CALL VCHAR(J+16,1,J+49):
: DISPLAY AT(J+16,1):E$(J)::
 NEXT J
220 FOR J=0 TO 6 :: ACCEPT A
T(J+16,1)SIZE(-28)BEEP:E$(J)
:: NEXT J :: R=24 :: C=27 ::
 R$="NY" :: GOSUB 150 :: IF
J=78 THEN 220
230 DISPLAY AT(1,1)ERASE ALL
:"FOR THE FOLLOWING PAGE, TH
E   DISPLAY IS IN GROUPS OF
    TWO LINES WHERE THE UPPE
R   LINE IS LETTERS THAT YO"
240 DISPLAY AT(4,26):"U":"
CHOOSE & THE LOWER LINE IS
THE CRYPTOGRAM AS ENTERED.":
 :"THE FINAL GROUP SHOWS THE
     LETTERS USED IN THE"
```

Continued on page 24

CRYPTOGRAM

Continued from page 23

```
250 DISPLAY AT(10,3):"CRYPTO
GRAM AND IS FOLLOWED":"  BY
A MESSAGE.": :"TO USE: TYPE
A LETTER FROM":"  THE BOTTOM
 ROW, THEN TYPE   THE LETTE"
260 DISPLAY AT(15,12):"R THA
T YOU THINK":"  IT REPRESENT
S.": :"THE MOST FREQUENTLY U
SED":"  LETTERS ARE REPORTED
 TO BE  E,T,A,I,N,S,H,R,L,D"
270 DISPLAY AT(20,22):",U...
": :"<ENTER> IS UNNECESSARY
NOW.": :"PLEASE WAIT WHILE I
 COMPUTE." :: R$="" :: FOR J
=0 TO 6 :: R$=R$&E$(J)
280 NEXT J :: IF R$="" THEN
180 ELSE LE$=RPT$(" ",26)
290 FOR J=1 TO LEN(R$):: S$=
SEG$(R$,J,1):: K=ASC(S$):: I
F K>64 AND K<91 AND POS(LE$,
S$,1)=0 THEN LE$=SEG$(LE$,1,
K-65)&S$&SEG$(LE$,K-63,26)
300 NEXT J :: R$,TE$=LE$ ::
GOSUB 130 :: LD$=S$
310 M=POS(TE$," ",1):: IF M
THEN TE$=SEG$(TE$,1,M-1)&SEG
$(TE$,M+1,26):: GOTO 310 ELS
E TD$=RPT$(" ",LEN(TE$))
320 FOR J=0 TO 6 :: R$=E$(J)
:: GOSUB 130 :: D$(J)=S$ ::
NEXT J :: R$=RPT$("0",49)::
FOR J=96 TO 120 STEP 4 :: CA
LL CHAR(J,R$):: NEXT J
330 C=23 :: R$="NY" :: DISPL
AY AT(R,1):"READY TO BEGIN?
(Y/N)" :: GOSUB 150 :: IF J=
78 THEN 180 ELSE DISPLAY ERA
SE ALL
340 R$=RPT$("-",28):: FOR J=
0 TO 6 :: DISPLAY AT(J*3+1,1
):D$(J):E$(J):R$ :: NEXT J :
: DISPLAY AT(22,1):LD$:LE$
350 C=LEN(TE$)+2 :: R$=TE$ :
: DISPLAY AT(R,1):R$;"?" ::
GOSUB 150 :: DISPLAY AT(R,1)
:S$;"=?" :: L=J :: R=22 :: C
=POS(LE$,S$,1)
360 R$=" ABCDEFGHIJKLMNOPQRS
TUVWXYZ" :: M=POS(TE$,S$,1):
: TD$=SEG$(TD$,1,M-1)&" "&SE
G$(TD$,M+1,26)
370 GOSUB 150 :: DISPLAY AT(
22,1):LD$ :: IF J=L OR J<>32
 AND POS(TD$,S$,1)THEN GOSUB
 140 :: GOTO 370
380 IF J=32 THEN CALL CHAR(L
+32,"0")ELSE CALL CHARPAT(J,
R$):: CALL CHAR(L+32,R$)
390 M=POS(TE$,CHR$(L),1):: T
D$=SEG$(TD$,1,M-1)&S$&SEG$(T
D$,M+1,26):: R=24 :: IF POS(
TD$," ",1)THEN 350
400 C=28 :: R$="NY" :: DISPL
AY AT(R,1):"IS CRYPTOGRAM SO
LVED? (Y/N)" :: GOSUB 150 ::
 IF J=78 THEN 350
410 C=23 :: DISPLAY AT(R,1):
"DO ANOTHER ONE? (Y/N)" :: G
OSUB 150 :: IF J=89 THEN 180
 ELSE DISPLAY ERASE ALL :: E
ND
```


HORIZON RAMDISK

How to make a bootable 800K Horizon RAMdisk

BY TONY KNERR

The following article appeared in the newsletter of the Milwaukee TI user group.—Ed.

The boot EPROM on the Geneve only has routines for up to quad-density disks, so we will have to fool it when reading our "high-density" RAMdisk so it can find the proper sectors where SYSTEM/SYS is located.

Format the RAMdisk with Form3meg, choose Y for "Set disk to boot MDOS" and Y for "Load SYSTEM/SYS from disk to RAMdisk" and go ahead and load the file on the RAMdisk.

Now that SYSTEM/SYS is on the RAMdisk, you will need to find the File Descriptor Record (FDR) for it. Load Disk Utilities or other sector editor. The sector will begin with the filename and will probably be sector >4 or >8. Write this sector down.

You'll also need to find the first actual data sector of SYSTEM/SYS. This sector includes the string "reassembling". Write this sector down also. It will probably be >200.

Now for the thinking part. We will be editing bytes >1C, >1D, and >1E of the sector with the FDR — either >4 or >8. You did write that down, didn't you? We will be interested in the six nybbles, or digits, of these three bytes. Let's number them one through six starting at the left digit of byte >1C.

Here's what the Nybbles mean:

Nybbles 4, 1, and 2, in that order, are the starting sector of the file.

Nybbles 5, 6, and 3, in that order, are the number of sectors of actual data in the file.

So if bytes >1C, >1D, and >1E are >40, >F0, and >1F, the FDR is telling us we have a file that starts at sector >040 and the data in it is >1FF sectors long (for any MDOS higher than version 2.50, at least). Well, the length is OK but the Geneve will load the wrong data, because it thinks the file starts at sector >040 instead of sector >200! You'll get a lock-up, because the wrong data was loaded in RAM where SYSTEM/SYS is supposed to be.

So lets fix it. If we edit Bytes >1C, >1D, and >1E to be >00, >F2, and >1F, the we are telling the EPROM that the file starts at sector >200 and is >1FF sectors of data in length, where the EPROM will find the correct data.

If you've done everything right, the Geneve will now boot from the RAMdisk. If not, you'll have to disable the RAMdisk via the on-off switch, reboot from floppy, turn the RAMdisk back on, then start over and do it correctly the next time. It does work. I've been doing it for years.

Continued on page 26

HORIZON RAMDISK

Continued from page 25

For a 1.44mb floppy, copy SYSTEM/SYS onto a "clean" disk. The FDR will be at sector >4. Edit bytes >1C, >1D, and >1E to be >00, >F2, and >1F (again, this is for MDOS version 2.50 and greater).

WARNING! Do not perform any file operations such as copy, move or delete on the SYSTEM/SYS file after you have made these changes. You can copy other files to or from the disk, but file operations on SYSTEM/SYS will corrupt all sorts of things on the disk. Just don't do it!

The information which enabled me to come up with this procedure can be found on pages 61 through 63 of the HFDC manual, "File Descriptor Records".

Making LOAD/SYS bootable from a 1.44mb floppy

Here's what you need to to LOAD/SYS to make a bootable 1.44mb floppy:
Copy LOAD/SYS onto the floppy after you have copied SYSTEM/SYS.
Make the changes to the SYSTEM/SYS FDR as stated in the main article.
The first six bytes of the data for LOAD/SYS are: >0000, >06DC, >A000. You'll find these at the start of sector >400.

The FDR for LOAD/SYS will be at sector >8. Bytes >1C, >1D, >and >1E will need to be changed from >00, >71, >00 to >00, >74, >00 per the information SYSTEM/SYS.

Now the floppy is bootable. Do not do any file operations on these two files as you will not be reading or writing to the proper sectors and you'll also mess up the bitmap. It's OK to add or remove other files on the disk, just don't do anything to SYSTEM/SYS or LOAD/SYS.

READER TO READER

BY ROGER PRICE

My old TI console used to go into a lockup, just setting there on the front screen. It would not do anything. So I put it up on a shelf for a couple of years and used a second unit that I had purchased used. Later I decided to check it out one more time before junking it and lo and behold it worked ok. The keys had been getting bad on it and it was nothing to see five of a character appear when pressing a key. So, I pulled off all of the keys by lifting with a gentle pull up or pry with a knife on the edge and they pop right up. Then I used a small thin strip of 600 grit sandpaper between the little contacts while pressing down on the side of the key seat. This makes them go together. I moved the paper up and down. After I put the keys back on they all work perfect like a new unit.

FORESTFIRE

Put out the fire

BY WOODY WILSON

The following article, excerpted here, appeared in the newsletter of the Southern California Computer Group and other newsletters.—Ed.

I am including a program that Ray Kazmer gave to me many years ago. It was written by John Behnke and enhanced by Ray. It is called Forestfire.

If you type in the program, do not be confused by the apparent duplication of parts of some of the sentences (such as occurs in lines 120 and 130 of the main program). Just type the line in as given and everything will work as it should.

The loader program below is from Ray and should be placed on the same disk as Forestfire. I recommend you use a newly initialized disk for the programs.

KAZMER LOADER

```
100 DIM A$(127),B$(127):: CA
LL CHARSET :: CALL SCREEN(12
):: CALL CLEAR :: DISPLAY AT
(12,1)BEEP:"       LOADING M
ENU..." :: CALL SOUND(1,4E4,
30)
110 OPEN #1:"DSK1.",INPUT ,R
ELATIVE,INTERNAL :: INPUT #1
:D$,A,A
120 INPUT #1:F$,A,B,B :: C=C
+1 :: IF LEN(F$)=0 THEN 160
130 IF C=36 THEN 160
140 A$(C)=F$ :: A=ABS(A):: I
F A<4 OR(A=4 AND B<>254)OR F
$="LOAD" OR F$="MENU" THEN C
=C-1 ELSE B$(C)=" "&CHR$(C+6
4)&" "&A$(C)
150 GOTO 120
160 CLOSE #1 :: B$(C)=" "&CH
R$(C+64)&" LEAVE MENU"
170 CALL CHAR(96,"0000201008
0000000000038447C4444000000
7824382478000000 3C4040403C")
180 CALL CHAR(100,"000000782
42424780000007C4078407C00000
07C407840400000003C405C4438"
)
190 CALL CHAR(104,"000000444
47C444400000038101010380000
```

Continued on page 28

Continued from page 27

```
008080848300000002428302824"
)
200 CALL CHAR(108,"000000404
040407C000000446C54444400000
04464544C440000007C4444447C"
)
210 CALL CHAR(112,"000000784
4784040000000384454483400000
07844784844000000 3C40380478"
)
220 CALL CHAR(116,"0000007C1
0101010000000444444443800000
044442828100000004444545428"
)
230 CALL CHAR(120,"000000442
810284400000044281010100000
07C0810207C0018202040202018"
)
240 CALL CHAR(124,"001010100
010101000300808040808300000 2
05408000000"):: FOR I=127 TO
 143 :: CALL CHAR(I,"0000000
000000000"):: NEXT I
250 R=5 :: DISPLAY AT(1,10)E
RASE ALL:D$ :: DISPLAY AT(3,
4):"FILENAME"," FILENAME"
:: H=INT(C/2+.5):: FOR I=1 T
O H
260 DISPLAY AT(R,1):B$(I),B$
(I+H):: R=R+1 :: NEXT I :: D
ISPLAY AT(24,8)BEEP:"CHOICE?
 (A-"&CHR$(C+64)&")"
270 CALL KEY(3,K,S):: IF S=0
 OR K<65 OR K>C+64 THEN 270
ELSE K=K-64
280 IF K<1 OR K>C THEN 270 E
LSE IF K=C THEN CALL CLEAR :
: PRINT "TI EXTENDED BASIC"
:: STOP ELSE L=INT(LEN(A$(K)
)):: X=INT(10-(L/2))
290 DISPLAY AT(24,1):"" :: D
ISPLAY AT(24,X)BEEP:"LOADING
 """&A$(K)&""""
300 CALL INIT :: CALL PEEK(-
31952,[,]):: CALL PEEK([*256
+]-65534,[,]):: @=[*256+]-65
534 :: D$="DSK1."&A$(K):: CA
LL LOAD(@,LEN(D$))
310 FOR [=1 TO LEN(D$):: CAL
L LOAD(@+[,ASC(SEG$(D$,[,1))
):: NEXT [ :: CALL LOAD(@+[,
0)
320 RUN "DSK1.FILENAME"
```

Now type in the program below.

FORESTFIRE

```
100 C$="LIGHTENING HAS IGNIT
ED FIRES IN A REDWOOD FOREST
!  YOUR TASK IS TO EXTINGUI
SH THEM AND SAVE AS MANY 200
0 YEAR OLD FIANT TREES AS YO
U POSSIBLY CAN!"
110 D$="TREES AS YOU POSSIBL
Y CAN!   YOUR ""RATING"" IS
 DETERMINED BY:   HOW WELL Y
OU FIGHT EACH FIRE AND HOW M
ANY TREES YOU SAVE."
120 E$="   YOU HAVE TWO WEA
PONS:   YOU CAN DROP CHEMICA
LS TO REDUCE ""BURN TIME"" O
R START A BACKFIRE TO TRY TO
"
130 G$="START A BACKFIRE TO
TRY TO CONTAIN AND CONTROL T
HE FIRE.   (CHEMICALS ARE O
```



```
NLY 50% EFFECTIVE ON AREAS A
DJACENT TO A DROP ZONE.)"
140 CALL CLEAR :: CALL SCREE
N(2):: FOR I=0 TO 14 :: CALL
 COLOR(I,8,1):: NEXT I :: F$
="F9AEEF562BD79A29" :: CALL
CHAR(96,"1038107C10FE1",104,
F$,120,F$):: DIM A(11,14)
150 CALL CHAR(112,"FCE0E0F8E
0E0E0E0FCECECECECECECFCF8CCC
CCCF8D8D8CCFCE0E0F8E0E0E0FC"
,116,"78CCC0780C0CCC78FE3838
38383838387C3838383838387C38
38383838380038")
160 CALL COLOR(9,13,2,10,12,
7,11,14,2,12,2,15):: DISPLAY
 AT(1,1):"      hhhhhhhhhhhh
hhhh            h"""""'
"'h            h'pqrstu'pvr
sw'h"
170 DISPLAY AT(4,1):"     h
""""""""h            h
hhhhhhhhhhhhhhhh":;:;:"  WRI
TTEN BY JOHN BEHNKE":;:;:;:"
   ENHANCED BY RAY KAZMER"
180 DISPLAY AT(14,1):" SAN F
ERNANDO VALLEY 99'ERS":;:;:;
:"      (ALPHA LOCK DOWN)" :
: A$="     INSTRUCTIONS? (Y/
N)    " :: DISPLAY AT(22,1)B
EEP:A$
190 GOSUB 580 :: IF E=78 THE
N 210 ELSE IF E<>89 THEN 190
200 FOR I=1 TO 20 :: CALL SO
UND(30,440,5):: NEXT I :: M$
=A$&C$ :: GOSUB 570 :: M$=D$
```

QuickType!jÖ-†Ö-P´+¯v˘StenciléËv˘Ö-†Ö-P´Textile∏-VWÖ-†Ö-P´AR(7+B,9,65+B):: FOR C=0 TO

```
13
240 IF A(B,C)=10 THEN CALL H
CHAR(B+7,C+11,ASC("`"))ELSE
IF A(B,C)=9 THEN CALL HCHAR(
B+7,C+11,ASC("h"))
250 NEXT C :: NEXT B
260 TS=0 :: WW=7 :: XX=11 ::
 CALL HCHAR(22,1,32,96):: DI
SPLAY AT(6,1):"" :: DISPLAY
AT(22,10)BEEP:"""DROP"" ROW?
":;:" (PRESS ""X"" FOR A BAC
KFIRE)"
270 GOSUB 580 :: IF E=88 THE
N 370 ELSE IF E<65 OR E>76 T
HEN 270 ELSE B=E-65 :: DISPL
AY AT(22,20):"="&CHR$(E):: D
ISPLAY AT(24,1)BEEP:"
  ""DROP"" COL?" :: DR=DR+1
275 FOR C=1 TO 100 :: NEXT C
280 GOSUB 580 :: IF E<65 OR
E>78 THEN 280 ELSE DISPLAY A
T(24,20)BEEP:"="&CHR$(E):: G
OSUB 590
290 C=E-65 :: FOR D=-1 TO 1
:: FOR G=-1 TO 1 :: H=B+D ::
 I=C+G :: IF H<0 OR H>11 OR
I<0 OR I>13 THEN 360
300 CALL GCHAR(H+7,I+11,AAA)
310 IF A(H,I)<1 OR A(H,I)=10
 THEN 350
320 IF RND>1 THEN 350
330 CALL SOUND(80,-3,0,110,0
):: CALL HCHAR(H+7,I+11,120)
:: CALL SOUND(50,-3,0,220,0)
340 A(H,I)=A(H,I)-3 :: IF A(
```

Continued on page 30

FORESTFIRE

Continued from page 29

```
H,I)<1 THEN CALL HCHAR(H+7,I
+11,120):: GOTO 360
350 CALL HCHAR(H+7,I+11,AAA)
360 NEXT G :: NEXT D :: GOTO
 400
370 BF=BF+1 :: DISPLAY AT(22
,9)BEEP:"BACKFIRE ROW?";:;""
380 GOSUB 580 :: IF E<65 OR
E>76 THEN 380 ELSE DISPLAY A
T(22,21):"="&CHR$(E):: B=E-
6
5 :: DISPLAY AT(24,1)BEEP:"
      BACKFIRE COL?"
385 FOR C=1 TO 100 :: NEXT C
390 GOSUB 580 :: IF E<65 OR
E>78 THEN 390 ELSE DISPLAY A
T(24,21)BEEP:"="&CHR$(E):: C
=E-65 :: GOSUB 590 :: IF A(B
,C)=10 THEN A(B,C)=2 :: CALL
 HCHAR(B+7,C+11,104)400 DISP
LAY AT(6,12):"CHECKING" :: F
OR B=0 TO 11 :: FOR C=0 TO 1
3 :: IF A(B,C)<1
 OR A(B,C)>9 THEN 460
410 IF A(B,C)<3 THEN 460
420 D=INT(3*RND-1):: G=INT(3
*RND-1):: H=B+D :: I=C+G ::
IF H<0 OR H>11 OR I<0 OR I>1
3 THEN 460
430 IF A(H,I)<>10 THEN 460
440 IF RND<.3 THEN 460
450 A(H,I)=11 :: CALL HCHAR(
B+7,C+11,104)
460 NEXT C :: NEXT B :: K=0
:: FOR B=0 TO 11 :: FOR C=0
TO 13 :: K=A(B,C):: IF K=11
THEN K=9 :: CALL HCHAR(B+7,C
+11,104)
470 IF K>0 AND K<10 THEN K=K
-1 :: IF K<1 THEN CALL HCHAR
(B+7,C+11,120)
480 A(B,C)=K :: NEXT C :: NE
XT B
490 CALL GCHAR(WW,XX,Z):: IF
 Z=104 THEN 260 ELSE IF Z=96
 THEN TS=TS+1
500 XX=XX+1 :: IF XX=25 THEN
 WW=WW+1 :: XX=11
510 IF WW<19 THEN 490 ELSE L
=TS-(DR+BF)
520 IF L>100 THEN L=100 ELSE
 IF L<1 THEN L=0
530 TT=TT+L :: GOSUB 600 ::
AV=INT(TT/GM):: IF AV>100 TH
EN AV=100 ELSE IF AV<1 THEN
AV=0
540 DISPLAY AT(20,1):"  BACK
FIRES="&STR$(BF):: DISPLAY A
T(20,19):"DROPS="&STR$(DR)::
 DISPLAY AT(22,1):"  '="&STR
$(TS)&"  x="&STR$(168-TS)&"
 RATING="&STR$(L)&"%"
550 DISPLAY AT(24,1):"GAMES=
"&STR$(GM)&"  AVG="&STR$(AV)
&"%" :: DISPLAY AT(24,20)BEE
P:"REDO? Y/N"
560 GOSUB 580 :: IF E=78 THE
N CALL INIT :: CALL LOAD(-31
804,0,36)ELSE IF E=89 THEN 2
10 ELSE 560
570 FOR I=1 TO LEN(M$)-27 ::
 DISPLAY AT(22,1):SEG$(M$,I,
28):: CALL SOUND(-1,-1,1)::
NEXT I :: RETURN
580 CALL KEY(0,E,F):: CALL C
OLOR(10,12,7):: FOR D=1 TO 1
0 :: NEXT D :: CALL COLOR(10
,7,12):: FOR D=1 TO 10 :: NE
XT D :: IF F=0 THEN 580 ELSE
 RETURN
590 DISPLAY AT(6,12):"CHECKI
NG" :: RETURN
600 DISPLAY AT(6,1):"" :: RE
TURN
```


FORESTFIRE

GENEVE

Using wild card character makes file management a breeze

The following is excerpted from David Ormond's column, David's Rants, in the newsletter of the Southwest Ninety Niners User Group.—Ed.

With everything going on, I'll bet you can guess I haven't had a chance to play on my Geneve. You would be right. I was going to look at MyWord and 80-column Multiplan, which is about as much Geneve-specific stuff that I ever used. Maybe next time I will do this, and describe the process of sucking the contents out of cartridges so you can run them under GPL on a Geneve. That would pretty much exhaust my store of contributable wisdom about the Geneve.

> MDOS has one up on peecee DOS in that DOS sees a leading asterisk as matching anything.

Instead, I will just drop a few hints about using MDOS. Specifically, wildcards. If you've used DOS on a peecee, you may be familiar with wildcards, where you can refer to several files with similar names by using the asterisk and question mark characters.

For instance, if you had files DAVE and DAN and DEB on a floppy, you could say, "DEL DAVE" to get rid of one. Getting rid of the others would require re-entering the command for each file which is not too bad under MDOS, since pressing the up-arrow cycles back through the last several previous commands which you can modify. Try it!

But with one command, you can take care of more than one file. Entering "DEL D" would get rid of all the files that started with a D. The

Cont. on page 32

GENEVE

Continued from page 31

asterisk is the wildcard that says "I match any number of characters." DAVE matches D* — the asterisk replaces "AVE." DAN and DEB do, too, with the asterisk matching "AN" and "EB," respectively.

You can narrow it down a bit. "DEL DA*" will get rid of DAVE and DAN, but not DEB. Get it?

The question mark says "I match any single character." So you could say "DEL W??," and this will get rid of DAN and DEB, but not DAVE. The asterisk is generally more powerful than the question mark wildcard, and is used more often, but keep the question mark wildcard in your bag of tricks.

MDOS has one up on peecee DOS in that DOS sees a leading asterisk as matching anything. In MDOS you could say "DEL *A*", and you would get rid of all files with an A in it, such as DAVE and DAN. DOS on a peecee would see the leading asterisk as matching all the files, and get rid of everything!

In light of this, and the powerful and somewhat unpredictable behavior of wildcards, if you are deleting things like this, it is probably a good idea to see what will get deleted. Start off with saying "DIR D*", for example, and look at the list of files that match your wildcard pattern. If that is acceptable, hit the up-arrow, back-arrow over the command, and change "DIR" to "DEL", and you will be safe.

Simple redirection: Many MDOS commands send their output to the screen. Sometimes, you want this to go to a file, or to the printer. You can do this pretty easily. The output redirection symbol is the greater than (>). Say you want a list of all the files that start with a D in a file you can work with under TI-Writer or one of Leonard Taff's text manipulation programs. You can just say "DIR D* > DLIST", and the file DLIST will be created on your drive with a listing of your files in MDOS format. Of course, since DLIST itself starts with D, it will be included in the list!

Sending things to the printer is similar, except you use the predefined filename "PRN." So saying "DIR D* > PRN" would send this list to the printer.

MDOS commands that I use all the time for this are DIR and TYPE. You can see how it could be neat to capture lists of files to disk files or printer output by redirecting DIR output.

Saying "TYPE DAN > PRN" allows you to print out a D/V80 text file without loading up TI-Writer. Really quite useful.

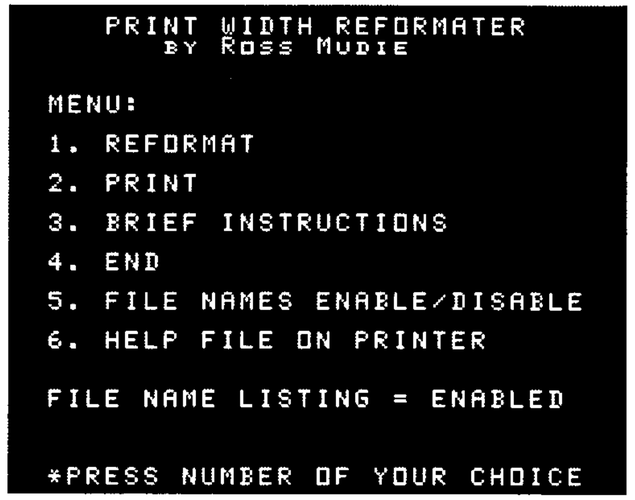
Mid-South, W. Penn change addresses

New mailing address for the West Penn 99ers is c/o Paul R. Brock, P.O. Box 222, North Apollo, PA 15673-0222.

New mailing address for the Mid-South 99 User Group is P.O. Box 27052, Memphis, TN 38127.

REFORMAT

Extended BASIC program reformats column widths



The following Extended BASIC program, REFORMAT, was written by Ross Mudie. The program lets users reformat text file columns and is fully commented. Operating instructions are included as part of the program.

The program allows the user to format columns anywhere from 24 to 80 columns in width. It will take an existing text file and reduce its column length or expand it, depending on user requirements.

The program was developed to allow reformating of files produced on the Editor/Assembler module when TI-Writer is not available.

REFORMAT

```
1 DISPLAY AT(12,1)ERASE ALL:
"For instructions, see the
 file HELPFORMAT"
2 DISPLAY AT(23,5)ERASE ALL:
"PRESS ANY KEY" :: DISPLAY A
T(23,5):"press any key" :: C
ALL KEY(0,K,S):: IF S=0 THEN
 2
100 ! REFORMATER  Version 1.
1, 26/10/84, Ross Mudie.
110 CALL INIT :: CALL LOAD(-
31806,16)! Disables FUNCTION
 QUIT
120 PRINT$="PIO" :: PO=1 ::
PL=60 ! Print Defaults (file
, option, page lines)
130 APN$="Y" ! Auto page num
bering default
140 DIM S$(37),O$(37)! Sourc
e & Object arrays
150 CALL CLEAR :: CALL SCREE
N(6):: FOR S=0 TO 12 :: CALL
 COLOR(S,16,1):: NEXT S :: G
OTO 200 ! Blue screen, White
 characters
160 DISPLAY AT(5,1)ERASE ALL
:"PURGING ARRAYS...."
170 FOR S=1 TO 37 :: O$(S),S
$(S)="" :: NEXT S ! Clear ar
rays
180 EOF_FLAG=0 ! Clear flags
190 !@P-
200 DISPLAY AT(1,4)ERASE ALL
:"PRINT WIDTH REFORMATER":TA
B(7);"by Ross Mudie": : :"ME
```

Continued on page 34

REFORMAT

Continued from page 33

```
NU:": :"1. REFORMAT": :"2. P
RINT"
210 DISPLAY AT(11,1):"3. BRI
EF INSTRUCTIONS"
220 DISPLAY AT(13,1):"4. END
": :"5. FILE NAMES ENABLE/DI
SABLE": :"6. HELP FILE ON PR
INTER": : :"FILE NAME LISTIN
G ="
230 IF NFN=1 THEN DISPLAY AT
(20,21):"DISABLED" ELSE DISP
LAY AT(20,21):"ENABLED"
240 DISPLAY AT(24,1):"*PRESS
 NUMBER OF YOUR CHOICE"
250 CALL SOUND(150,1400,0)!
Beep
260 !@P+
270 CALL KEY(3,K,S):: IF S=0
 THEN 270 :: IF K<49 OR K>54
 THEN CALL WRONG :: GOTO 270
 ELSE IN=K-48
280 CALL CLS(24,24)! Clears
screen (from line,to line) i
n this case line 24 only290
ON IN GOTO 340,1040,1670,300
,310,330
300 END ! If linking to anot
her program substitute e.g.
RUN "DSK1.PROGNAME"
310 IF NFN=0 THEN NFN=1 ELSE
 NFN=0 ! Set/Reset for No Fi
le Names
320 GOTO 230
330 OF$,SF$="HELPFORMAT" ::
CALL CLS(5,15):: CALL CLS(18
,24):: PO=3 :: PL=60 :: SL,P
G_NO=1 :: GOTO 1090 ! To pri
nt HELP file
340 CALL FILENAMES(E,NFN,S$(
),SFREE):: IF E=1 THEN 200 !
 E is Error flag from sub FI
LENAMES
350 !@P-
360 DISPLAY AT(20,1):"SOURCE
 FILE? DSK1.";SF$
370 ACCEPT AT(20,19)BEEP SIZ
E(-10)VALIDATE(UALPHA,DIGIT,
"_-/"):SF$ :: CALL CLS(23,24
)
380 IF SF$="" THEN CALL WRON
G :: GOTO 200 ! Back to menu
 on a null string
390 IF ASC(SEG$(SF$,1,1))<65
 OR ASC(SEG$(SF$,1,1))>90 TH
EN CALL WRONG :: GOTO 370 !F
irst character of source fil
ename to be upper case alpha
400 DISPLAY AT(22,1):"OBJECT
 FILE? DSK1.";OF$ :: ACCEPT
AT(22,19)BEEP SIZE(-10)VALID
ATE(UALPHA,DIGIT,"_-/"):OF$
:: CALL CLS(23,24)
410 IF OF$=SF$ THEN CALL WRO
NG :: DISPLAY AT(23,1):"***
source & object files  must
 have different names **" ::
 GOTO 370
420 IF OF$="" THEN CALL WRON
G :: GOTO 370
430 IF ASC(SEG$(OF$,1,1))<65
 OR ASC(SEG$(OF$,1,1))>90 TH
EN CALL WRONG :: GOTO 400 !
First line of object filenam
e to be upper case alpha
440 DISPLAY AT(24,1):"NEW PA
```


REFORMAT

```
GE WIDTH? ";PAGE$
450 !@P+
460 ACCEPT AT(24,17)BEEP SIZ
E(-2)VALIDATE(DIGIT):PAGE$ :
: IF PAGE$="" THEN CALL WRON
G :: GOTO 200 ELSE PAGE=VAL(
PAGE$)
470 !@P-
480 PAGE=PAGE+1 :: IF PAGE<2
4 OR PAGE>81 THEN CALL WRONG
 :: GOTO 460
490 ON ERROR 1550
500 OPEN #1:"DSK1."&SF$,INPU
T ,SEQUENTIAL,DISPLAY ,VARIA
BLE 80
510 ON ERROR 1570
520 OPEN #2:"DSK1."&OF$,OUTP
UT,SEQUENTIAL,DISPLAY ,VARIA
BLE 80
530 ON ERROR STOP
540 CALL CLS(5,19)! Clears t
he filenames from screen lin
es 5 to 19
550 !@P+
560 S,O,SL,OL=1 :: DISPLAY A
T(11,1):"SOURCE LINE   OBJEC
T LINE"
570 GOSUB 890 ! To load Sour
ce file for first time
580 T$=S$(S)!  Load Tempory
 string with Source file lin
e (S)
590 !@P-
600 DISPLAY AT(7,1):"** FORM
ATING IN PROGRESS **"
610 IF LEN(T$)>=PAGE THEN 73
0 !  Is length tempory file
 > reformat page
620 IF S$(S+1)="EOF!EOF!!" T
HEN EOF_FLAG=1 :: GOTO 730 !
End file in source array630
IF T$="" THEN 650 ! Temp str
ing is a null
640 IF SEG$(S$(S+1),1,1)=" "
 AND LEN(S$(S+1))>1 THEN 730
 ! Is 1st character in next
string a space?
650 S=S+1 :: SL=SL+1 :: DISP
LAY AT(13,5)SIZE(5):SL !Incr
ement Source line counter &
display SL
660 IF EOF(1)<>0 THEN 680 !
To prevent reading disk past
 End Of file
670 IF S=37 THEN GOSUB 890 !
 To get more source file
680 IF T$="" THEN T$=S$(S)::
 GOTO 610 ! If Temp string i
s empty then get next  sou
rce line
690 IF LEN(T$)=1 AND T$=" "
THEN 580 ! This is a string
containing one space
700 IF SEG$(T$,LEN(T$),1)="
" THEN T$=SEG$(T$,1,LEN(T$)-
1):: GOTO 690 ! If last  cha
racter a space then remove i
t
710 T$=T$&" "&S$(S):: GOTO 6
10 ! Concatenate with next s
tring including inter  wor
d space
720 !@P+
730 W=PAGE !  Width=PAGE W
idth selected
```

Continued on page 36

Continued from page 35

```
740 !@P-
750 IF SEG$(T$,W,1)=" " OR S
EG$(T$,W,1)="" THEN 790 ! Lo
oks for inter-word space or
null string
760 W=W-1 !           Decreme
nt Width
770 IF W=0 THEN 780 ELSE 750
 !    Trap for string with n
o spaces
780 W=PAGE !          Default
 full width for no spaces
790 O$(O)=SEG$(T$,1,W)!   St
ore output string
800 O$(O)=SEG$(O$(O),1,80)!
Trap for space at char 81
810 IF LEN(T$)>W THEN T$=SEG
$(T$,W+1,LEN(T$)-W)ELSE T$="
" !  This is the second part
 of the string
820 O=O+1 :: OL=OL+1 :: DISP
LAY AT(13,20):OL ! Increment
 output line counter &   dis
play OL
830 IF O=37 THEN O=36 :: GOS
UB 980 ! Object array full,
save to disk
840 IF LEN(T$)<>0 THEN 600 !
 Possible end of source file
850 IF EOF_FLAG=0 THEN 600 !
 Possible end of source file
860 GOSUB 980 ! Save end of
object file
870 CLOSE #1 :: CLOSE #2 ::
GOTO 160 ! Reformat complete
880 ! ************************
**************************
890 DISPLAY AT(7,1):"*** LOA
DING SOURCE FILE ***" ! Subr
outine gets source file
900 S=1 !           Start poi
nt in Source File
910 LINPUT #1:S$(S)! Input f
rom disk 1
920 IF EOF(1)=1 THEN 950
930 IF S=36 THEN 960 ! Sourc
e array full
940 S=S+1 :: GOTO 910 ! Incr
ement source array pointer
950 S$(S+1)="EOF!EOF!!" ! En
d Of File marker in array
960 S=1 :: RETURN
970 ! ************************
**************************
980 DISPLAY AT(7,1):"SAVING
";OF$;" TO DISK 1" ! Subrout
ine to save object file
990 !@P+
1000 ON ERROR 1630
1010 FOR P=1 TO O :: PRINT #
2:O$(P):: O$(P)="" :: NEXT P
 :: O=1 :: ON ERROR STOP ::
RETURN
1020 ! ***** Print routine *
**************************
**************************
1030 !@P-
1040 DISPLAY AT(1,8)ERASE AL
L:"PRINT ROUTINE" :: CALL FI
LENAMES(E,NFN,S$(),SFREE)::
IF E=1 THEN 200
1050 SL,PG_NO=1 ! Preset val
ues for Source line & Page n
```



```
umber
1060 DISPLAY AT(20,1):"PRINT
 FILE?  DSK1.";OF$ :: ACCEPT
 AT(20,19)BEEP SIZE(-10)VALI
DATE(UALPHA,DIGIT,"_-/"):SF$
 :: CALL CLS(24,24)
1070 OF$=SF$ :: IF SF$="" TH
EN CALL WRONG :: GOTO 200
1080 IF ASC(SEG$(SF$,1,1))<6
5 OR ASC(SEG$(SF$,1,1))>90 T
HEN CALL WRONG :: GOTO 1060
! Checks first character of
filename is upper case alpha
1090 DISPLAY AT(22,1):"WHERE
 IS PRINTER? ";PRINT$ :: ACC
EPT AT(22,19)BEEP SIZE(-7):P
RINT$
1100 IF PRINT$="" THEN CALL
WRONG :: GOTO 1060
1110 IF PRINT$="PIO" OR PRIN
T$="RS232" OR PRINT$="RS232/
1" OR PRINT$="RS232/2" THEN
1120 ELSE CALL WRONG :: GOTO
 1090 ! Validate Printer fil
e
1120 DISPLAY AT(20,1):"PRINT
ER OPTIONS:       ";PO: :"1.
 normal  2. compressed":"3.
line 1 LARGE (40 ch)":"4.com
press with lne 1 normal"
1130 !@P+
1140 ACCEPT AT(20,25)BEEP SI
ZE(-1)VALIDATE("1234"):PO$ :
: CALL CLS(22,24):: IF PO$="
" THEN 1060 ELSE PO=VAL(PO$)
1150 PRINTER$=PRINT$&".CR" !
 Printer file name & softwar
e switch option No CR/LF
1160 DISPLAY AT(20,1):"HOW M
ANY LINES PER PAGE?";PL :: A
CCEPT AT(20,26)BEEP SIZE(-2)
VALIDATE("1234567890"):PL$
1170 !@P-
1180 IF PL$="" THEN CALL WRO
NG :: GOTO 1060 ELSE PL=VAL(
PL$)
1190 DISPLAY AT(20,1):"AUTO
PAGE NUMBERING? Y/N ";APN$ :
: ACCEPT AT(20,26)BEEP SIZE(
-1)VALIDATE("YN"):APN$ ! Sel
ect/deselect auto page no.
1200 IF APN$="" THEN CALL WR
ONG :: GOTO 1060
1210 ON ERROR 1610
1220 OPEN #3:PRINTER$
1230 CALL CLS(20,24):: DISPL
AY AT(24,1):"****  TURN PRIN
TER ON  ****"
1240 ON PO GOTO 1250,1260,12
70,1280 ! Select print optio
n
1250 PRINT #3:CHR$(18):: GOT
O 1290 ! Normal
1260 PRINT #3:CHR$(15):: GOT
O 1290 ! Compressed
1270 PRINT #3:CHR$(18);CHR$(
14):: GOTO 1290 ! Normal wit
h first line enlarged
1280 PRINT #3:CHR$(15);CHR$(
14):: GOTO 1290 !Compressed
with first line enlarged
1290 CALL CLS(24,24):: DISPL
AY AT(20,1):"PRINT FILE   DS
K1.";OF$: :"PRINT OPTION=";P
O;" ON ";PRINT$
```

Continued on page 38

REFORMAT

Continued from page 37

```
1300 ON ERROR 1520
1310 OPEN #1:"DSK1."&SF$,INP
UT ,SEQUENTIAL,DISPLAY ,VARI
ABLE 80
1320 S=1 :: CALL CLS(5,19)
1330 DISPLAY AT(24,1):"****
LOADING PRINT FILE ****"
1340 LINPUT #1:S$(S)! Input
from disk 1
1350 IF S/36=INT(S/36)THEN 1
380 ! If equal array full
1360 IF EOF(1)=0 THEN S=S+1
:: GOTO 1340 ELSE 1380 ! Che
ck for EOF (source file)
1370 !@P+
1380 CALL CLS(24,24):: IF FI
RST_LINE=0 THEN 1400 ELSE 14
40
1390 !@P-
1400 FIRST_LINE=1 ! This fla
g is to permit enlarged prin
t (next 2 program lines)
1410 IF LEN(S$(1))>40 AND PO
=3 THEN 1430 ELSE 1420 !If f
irst line too long for    en
larged characters
1420 IF LEN(S$(1))>40 AND PO
=4 THEN 1430 ELSE 1440 ! If
first line too long for    en
larged compressed print
1430 CALL WRONG :: DISPLAY A
T(14,1):"* FIRST LINE TOO LO
NG FOR": :"LARGE PRINT, > 40
 characters" :: PRINT #3:CHR
$(20)
1440 FOR P=1 TO S :: PRINT #
3:S$(P);CHR$(10);CHR$(13)
1450 IF SL/PL=INT(SL/PL)THEN
 PG_NO=PG_NO+1 :: PRINT #3:C
HR$(12)ELSE 1470 ! Test for
page length & give form feed
1460 IF APN$="Y" THEN PRINT
#3:TAB(38);"-";STR$(PG_NO);"
-";CHR$(10);CHR$(13);CHR$(10
);CHR$(10)! Page numbering
1470 SL=SL+1 :: NEXT P
1480 FOR P=1 TO S+1 :: S$(P)
="" :: NEXT P :: S=1 :: IF E
OF(1)=0 THEN 1330 !Clr    ar
ray
1490 PRINT #3:CHR$(12)!  For
m feed at end of text
1500 PRINT #3:CHR$(18):: CLO
SE #1 :: CLOSE #3 :: FIRST_L
INE=0 :: ON ERROR STOP :: GO
TO 160
1510 !  ******* Error Messag
es ************************
**********************
1520 DISPLAY AT(24,1):"** FI
LE NOT FOUND ON DISK **" ::
CALL WRONG :: ON ERROR 1530
:: CLOSE #1
1530 ON ERROR 1540 :: CLOSE
#3
1540 ON ERROR STOP :: RETURN
 1060
1550 DISPLAY AT(24,1):"** FI
LE NOT FOUND ON DISK **" ::
CALL WRONG :: ON ERROR 1560
:: CLOSE #1
1560 ON ERROR STOP :: RETURN
 370
1570 IF SFREE>0 THEN DISPLAY
 AT(24,1):"*** FILE WRITE PR
OTECTED ***" :: CALL WRONG :
: ON ERROR 1590 :: CLOSE #2
:: GOTO 1590 ELSE 1580
1580 DISPLAY AT(24,1):"****
    DISK FULL       ***" ::
CALL WRONG :: ON ERROR 1590
:: CLOSE #2
1590 ON ERROR 1600 :: CLOSE
#1
1600 ON ERROR STOP :: RETURN
 370
1610 DISPLAY AT(24,1):"****
CAN'T FIND PRINTER  ***" ::
CALL WRONG :: ON ERROR 1620
:: CLOSE #3
1620 ON ERROR STOP :: RETURN
 1090
1630 DISPLAY AT(17,1):"    *
**  DISK FULL ***" :: CALL W
RONG :: ON ERROR 1640 :: CLO
SE #1
1640 ON ERROR 1650 :: CLOSE
#2
1650 ON ERROR STOP :: CALL W
AIT :: RETURN 160
1660 !***** Brief Informatio
n ************************
**********************
1670 CALL CLS(5,24)
1680 DISPLAY AT(5,1):"  This
 Reformater is": :"intended
for DIS/VAR80 files": :"to a
lter the print column": :"wi
dth within the range 24 to"
1690 DISPLAY AT(13,1):"80 co
lumns. The Reformater": :"wi
ll both reduce or expand": :
"a file."
1700 CALL WAIT
1710 DISPLAY AT(5,1):"  Unde
sirable results": :"may be o
btained if a line": :"contai
ns text at either end": :"wi
th many spaces in between."
1720 DISPLAY AT(13,1):"  The
 program was developed": :"t
o allow reformating of": :"f
iles produced on the Editor"
1730 DISPLAY AT(19,1):"Assem
bler module when TI": :"Writ
er is not available."
1740 CALL WAIT
1750 DISPLAY AT(5,1):" By pr
essing ENTER, after": :"dele
ting any default value": :"p
rovided at most prompts,"
1760 DISPLAY AT(11,1):"you m
ay work back to a": :"higher
 prompt or the menu."
1770 DISPLAY AT(15,1):" Menu
 opt 5 is a set/reset": :"fu
nction which allows use of":
 :"options 1 or 2 without th
e": :"available files list."
1780 CALL WAIT
1790 CALL CLS(5,23):: DISPLA
Y AT(13,1):" for full detail
, print out": :" the HELPFOR
MAT file using": :" menu opt
ion 6"
1800 !@P+
1810 CALL WAIT :: GOTO 200
1820 !  ******* Sub Programs
 **************************
**********************
```

Continued on page 40

REFORMAT

REFORMAT

Continued from page 39

```
1830 SUB CLS(S,F):: FOR C=S
TO F :: DISPLAY AT(C,1):: NE
XT C :: SUBEND
1840 SUB FILENAMES(E,NFN,S$(
),SFREE):: L=7 :: C,S=1 :: E
,PP=0 :: CALL CLS(2,20)
1850 IF NFN=1 THEN 2090
1860 ON ERROR 2070
1870 OPEN #1:"DSK1.",INPUT ,
INTERNAL,RELATIVE
1880 INPUT #1:DISKNAME$,TSIZ
E,DSIZE,SFREE ! Gets disknam
e, size & free space etc
1890 DISPLAY AT(3,1):"DSK-";
DISKNAME$;",";SFREE;"SEC FRE
E"
1900 DISPLAY AT(20,1):"LOADI
NG AVAILABLE FILE NAMES"
1910 INPUT #1:FILENAME$,TYPE
,FSIZE,RECLEN ! Gets file ti
tles & Characteristics
1920 IF LEN(FILENAME$)=0 THE
N 1990 ! No more file names
1930 IF ABS(TYPE)=2 AND RECL
EN=80 THEN S$(S)=FILENAME$ E
LSE 1910 ! DIS/VAR only
1940 FSIZE$=STR$(FSIZE)! Fil
e size
1950 IF LEN(S$(S))<10 THEN S
$(S)=S$(S)&" " :: GOTO 1950
! Pad file name to 10 long w
ith trailing spaces
1960 IF LEN(FSIZE$)<2 THEN F
SIZE$=" "&FSIZE$ ! Pad FIZE
to 2 long with leading space
1970 S$(S)=S$(S)&"-"&FSIZE$
:: S=S+1 ! Concatenate file
name & file size
1980 GOTO 1910
1990 DISPLAY AT(5,1):"AVAILA
BLE FILES & -FILE SIZE" :: F
OR P=PP+1 TO S
2000 IF S$(P)="" THEN 2080 E
LSE DISPLAY AT(L,C):S$(P)
2010 IF C=15 THEN C=1 :: L=L
+1 ELSE C=15 ! Where next fi
lename is displayed
2020 IF P/24=INT(P/24)THEN D
ISPLAY AT(20,1):"MORE AVAILA
BLE, press SPACE": :"or pres
s ENTER to continue" :: PP=P
 :: L=7 :: C=1 ELSE 2060
2030 CALL KEY(3,K,ST):: IF S
T=0 THEN 2030
2040 IF K=32 THEN 1990
2050 IF K=13 THEN 2080 ELSE
2030
2060 NEXT P
2070 DISPLAY AT(22,3):"*** D
ISK NOT FOUND ***": :"PRESS
ANY KEY TO CONTINUE" :: CALL
 WRONG :: CALL WAIT :: E=1 :
: RETURN 2080
2080 ON ERROR 2090 :: CLOSE
#1
2090 ON ERROR STOP :: CALL C
LS(20,22):: FOR P=1 TO S ::
S$(P)="" :: NEXT P
2100 SUBEND
2110 SUB DELAY(D):: FOR DELA
Y=1 TO D :: NEXT DELAY :: SU
BEND
2120 SUB WRONG :: CALL SOUND
(200,200,1):: CALL DELAY(50)
:: SUBEND
```


REFORMAT

```
2130 SUB WAIT :: DISPLAY AT(
24,1):"PRESS ANY KEY TO CONT
INUE"
2140 CALL KEY(3,K,ST):: IF S
T=0 THEN 2140
2150 SUBEND
```

MICROREVIEWS

The Complete GPL Package

BY CHARLES GOOD

THE COMPLETE GPL PACKAGE Compiled by Rich Gilbertson

Rich Gilbertson is definitely the best GPL programmer left in the TI community. His enhanced Extended BASIC known as RXB pushes the technological limit of what can be done with the 99/4A. It is programmed in GPL (Graphics Programming Language), which is probably the most efficient programming language there is for our computer. A GPL program usually uses less memory than the same program written in either Extended BASIC or in assembly. In a December 1992 MICROpendium article Rich compared 12 short programs written in the three languages. In each case the GPL version used less memory. Programs written in GPL can be converted to Editor/Assembler-5 format so that can be run without the need of any version of Extended BASIC.

Rich has put together a package of information and software for those who want to try their hand at this uniquely 99/4(A) programming language. This information is all available archived on three DSSD disks. There is also a supplemental hard copy of the above mentioned MICROpendium article and other interesting material. Here is what you get on the disks:

TI Graphics Programming Language Users Guide, a 1979 TI document. This is designed to aid those programming TI command modules. It is comprehensive and even includes chapters on style. You are told how to make good looking screens, what good user interactive prompts should look like, and what colors go together. Special instructions are included for the creation of multilingual command modules.

You also get the following official TI documents, all dated 1980 and all except the speech document updated in March 1983:

"Functional Specifications for the 99/4 disk peripheral"

"GPL Interface Specifications for the 99/4 Disk Peripheral"

"Software Specifications for the

Continued on page 42

MICROREVIEWS

Continued from page 41

99/4 Disk Peripheral"

"Speech Synthesizer Principle of Operation"

The speech document is particularly interesting because TI produced very little documentation about its speech synthesizer.

GPL assembler and linker. This is one of several GPL assemblers that have been made available over the years. It takes GPL source code as created with a text editor such as the Funnelweb editor. You use the TI GPL User Guide as well as the on disk documentation that comes with this assembler to create the source code. Following easy directions you use the assembler to assemble the code and then you use the linker to link this assembled code so that it can be run from an assembly loader. That's right, you can run your assembled GPL program as an EA5 program.

Among the most interesting parts of this GPL package are the two assembler/linker demos. Using two different demo source codes provided you just follow the step by step instructions you are taken through the complete process needed to create demo programs that can be run as EA5. When you run these assembled and linked EA5 GPL programs you are taken back to the "Press 1 for TI BASIC" screen, which displays additional menu items for your GPL programs. One of the demos shows how to take several different GPL programs and display them simultaneously as separate menu items following "Press 1 for TI BASIC".

Using two different demo source codes provided you just follow the step by step instructions you are taken through the complete process needed to create demo programs that can be run as EA5.

For free I can email you the three DSSD disks in PC99 format and the supplemental hard copy as a Microsoft Word file. If you want me to mail real TI disks and the hard copy handout please send me $3 to cover my Xerox media and postage expenses.

ACCESS

Rich Gilbertson (master GPL programmer and the compiler of the complete GPL package)
1901 H Street
Vancouver, WA 98663-3352
Phone: 1-360-737-7963

Charles Good (source of the GPL package by email or regular mail)
P.O. Box 647
Venedocia OH 45894
Phone: 419-667-3131
E-mail: good.6@osu.edu

Booting the Funnelweb 5.01 IBM II character set

This item appeared in the newsletter of the Southwest 99ers.—Ed.

To get Funnelweb into line drawing mode you must hold down the space bar at the proper time.

First, put your FWB program disk into drive 1 and boot with your favorite loader. I use Extended BASIC.

You will get a couple of title screens that you can ignore and at the first menu screen choose:

1. TI-Writer (enter)

At the next menu, choose:

1. Text-Edit (enter)

Immediately after selecting Text-Edit, hold down the space bar. (This is how you get into the appropriate editor to use the IBM II character set. When the next menu comes up, choose:

1. Word Processing (enter)

The next menu offers Select Language Mode. Choose:

3. All Chars Mode (enter)

The next menu offers several languages. Choose:

1. Default (enter)

You are now in the FWB 5.01 40-column editor in the All Chars mode and can use all the characters in the IBM II character set to draw forms, make windows, and type inside them using the CTRL-, command to toggle between drawing and writing modes. If you didn't do it properly, the program will tell you with a message at the bottom of the primary writer screen.

There are two help files on the Funnelweb disk that will help you get used to using this program — FW/GRPH and FW/GRPH1. You can print them out for future reference.

Call/Peek utility

The following utility was written by Charles Good. The program can be used to help understand CALL LOADs in Extended BASIC.

CALL/PEEK

```
1 !Written Aug.28, 1984 by
Charles W. Good, Box 647,
Venedocia OH, 45894, USA.
2 !Program now set up for GE
MINI 10X printer. If needed,
 change CHR$ numbers in line
s 9,18,&20 and OPEN statemen
t in line 8 for your printer
.
3 !RUN this program as is un
der varying conditions such
as protected vs unprotected,
 disk drives on vs off, or m
emory expansion on vs off.
4 !OR save this program in M
ERGE format. Merge and run i
t with another program to se
e how the other program affe
cts memory data.  These
5 !suggestions should help y
ou in experiments with CALL
LOADing (pokeing) data direc
tly into memory.
6 CALL CLEAR :: PRINT "
    CALL PEEK": : : : : : :
: :"Do you want to use print
er? (Y/N)" :: INPUT PR$
7 IF PR$="N" THEN 10 :: PRIN
T "TURN ON THE PRINTER......
": : : :
8 OPEN #1:"PIO"
```

Continued on page 46

USER NOTES

Continued from page 45

```
9 PRINT #1:CHR$(15)&CHR$(27)
&CHR$(68)&CHR$(15)&CHR$(30)&
CHR$(45)&CHR$(60)&CHR$(75)&C
HR$(90)&CHR$(0)!CONDENSED PR
INT; SET TABS EVERY 15 COL
10 PRINT : : : :"Additional
instructions can be read by
LISTing program.": :
11 PRINT : : :"Memory addres
s numbers are  -32768 throug
h 32767 in dec-imal notation
.  Enter the  lowest number
ed address,"
12 PRINT "then enter the hig
hest num- bered address to b
e PEEKED. The most negative
address isthe lowest.  Compu
ter prints"
13 PRINT "address number and
 decimal  byte (0-255) for e
ach mem-  ory location.": :
:
14 INPUT "LOWEST MEMORY ADDR
ESS ":LMA
15 INPUT "HIGHEST MEMORY ADD
RESS ":HMA
16 FOR X=LMA TO HMA STEP 6
17 FOR Z=0 TO 5 :: CALL PEEK
(X+Z,Y):: PRINT X+Z,Y :: IF
PR$="N" THEN 19
18 PRINT #1:X+Z;Y;CHR$(9);!M
ove printer head to next tab
.
19 NEXT Z
20 IF PR$="N" THEN 21 :: PRI
NT #1:CHR$(13)!Printer carri
age return.
21 NEXT X
22 IF PR$="N" THEN END :: CL
OSE #1 :: END
```

Corrections to REM program

This comes from Jacques Groslouis:

The program REM, March/April 1999 MICROpendium, suffers from a number of dropped characters. For reasons that I do not understand a number of minuses (-) and a few blank spaces were dropped when I transferred a LISTing of the program from my TI to my PC using a RS232 cable. My speculation is that I may have to lower the baud rate I use. You may want to make reference to the attachment which contains the affected lines in your next publication.

The same problem has found its way into the program MAILBX where a minus sign should appear in front of 24577 and a space should appear between CALL and LOAD.

Sorry to have caused such a bother. I have a poster near my desk which says "To err is human, To really foul things up requires a computer."

```
180 CALL PEEK(-2,A)
200 CALL LOAD(-2,0)
210 PRINT "  Welcome to my T
I Computer": : : : : : : :
230 CALL SAY("I+UNDERSTAND+T
HE1+Y+TWO+K+PROBLEM. DO+YOU"
)
240 !@P-
```


USER NOTES

```
380 CALL SAY("DID+YOU+HEAR+T
HE1+ONE+ABOUT+THE1 #TEXAS IN
STRUMENTS# HOME+COMPUTER")
450 CALL SPRITE(#2,100,7,R+B
,C-B,SX,-SY)
460 CALL SPRITE(#3,100,6,R-
B
,C,-SX,SY)
470 CALL SPRITE(#4,100,14,R,
C+B,-SX,-SY)
480 CALL SPRITE(#5,100,2,R-
B
,C-B,-SX,0)
490 CALL SPRITE(#6,100,8,R+B
,C,0,-SY)
500 CALL SPRITE(#7,100,4,R,C
-B,0,SY)
510 CALL SPRITE(#8,100,10,R-
B,C+B,SX,0)
```

Auction of TI/Geneve equipment

The following equipment will be auctioned to the highest bidder on July 31, 1999:

- Geneve 9640 (reconditioned by Tim Tesch in fall 1998)
- Western Horizon SCSI card and 255mb hard drive with power supply
- CorComp floppy disk controller
- TI RS232 card
- Four Horizon RAMdisks in undetermined condition (incl. Horizon 4000, HRD+ & 2 others from mid-1980s (all four as a package)
- CorComp Triple Tech with Speech Synthesizer
- P-GRAM card
- Myarc 512K memory expansion

Submit bids by e-mail to: micropendium@yahoo.com; or mail to MICROpendium, P.O. Box 1343, Round Rock, TX 78664.

Half-price on Geneve disks

GENEVE PUBLIC DOMAIN DISKS

These disks consists of hundreds of public domain programs for the Geneve 9640. There are six disk series in all. Prices listed at right are 50% off regular price.

Overseas orders please add 30 cents per disk for shipping.

	SSSD	DSSD	DSDD
❑ Series 1	$4.50	$3.50	$2.50
❑ Series 2	$4.50	$3.50	$2.50
❑ Series 3	$4.50	$3.50	$2.50
❑ Series 4	$4.50	$3.50	$2.50
❑ Series 5	$4.50	$3.50	$2.50
❑ Series 6	$4.50	$3.50	$2.50