Covering the TI99/4A and the Myarc 9640

MICROpendium

Volume 10 Number 12 January 1994 $3.50

Five card draw

See page 18

@#%&*! letter words

See page 14

Protecting your programs

Exploring artificial intelligence

The Art of Assembly

Reviews

- Sliding Block Puzzle Series II
- Japanese Writing System
- TI-Artist Fonts
- Adventure Data Base
- Genealogy Plus

CONTENTS

MICROpendium

MICROpendium (ISSN 10432299) is published monthly for $35 per year by Burns-Koloen Communications Inc., 502 Windsor Rd., Round Rock, TX 78664-7639. Second-class 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. We cannot take responsibility for unsolicited manuscripts but will give consideration to anything sent to the above address. Manuscripts will be returned only if a self-addressed stamped envelope is included.

Foreign subscriptions are $40.25 (Mexico); $42.50 (Canada); $40.00, 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: (512) 255-1512
CompuServe: 75156,3270
Delphi TI NET: MICROpendium
GEnie: J.Koloen
Internet E-mail: jkoloen@holonet.net
John Koloen.....Publisher
Laura Burns.....Editor

*READ THIS

Here are some tips to help you when entering programs from MICROpendium:

1. Most BASIC and Extended BASIC programs are run through Checksum, which places the numbers that follow exclamation points at the end of each program line. Do not enter these numbers or exclamation points. Checksum is available on disk from MICROpendium for $4.

2. Long Extended BASIC lines are entered by inputting until the screen stops accepting characters, pressing Enter, pressing FCTN REDO, cursoring to the end of the line and continuing input.

COMMENTS

Looking forward to the SCSI controller

The new year starts with the debut of the Horizon SCSI controller in a usable form. Bud Mills has announced that it is ready for sale and priced at $170. The card works with SCSI hard drives and some SCSI CD-ROM units. With the addition of an optional card it will also control up to four floppy drives ranging from 360K to 2.88 megabtyes each. This card is priced at $100. Bud will be selling the cards at Fest West in February.

The SCSI controller card is a product with enormous potential. A SCSI interface has the potential to handle all kinds of hardware, ranging from hard drives to scanners.

Of course, the software has to be in place to actually use a scanner or CD-ROM drive with the TI/Geneve, but that's another story. Bud has indicated that the SCSI will immediately give users access to SCSI hard drives and some removable, Macintosh-compatible cartridge drives. I'm excited by cartridge drives because prices have been coming down and the smallest you can buy will give you 44 megabytes of storage on a single $60 Syquest cartridge. A 44-megabyte Syquest-compatible SCSI drive can be had for $200 or so used and $300-$350 new. Larger drives, 88 and 105 megabytes, are considerably more costly. But the average TI/Geneve user a 44-meg. Syquest would be plenty. The advantage to the Syquest is that the mechanism and power supply are built into the Syquest case and all you've got to do is select a SCSI number and plug it in. It's easy to carry around and, if you drop the drive, you won't necessarily lose the data because the data is kept on a removable cartridge.

IT'S BEEN TEN YEARS FOR US

It's hard to believe but this issue of MICROpendium marks the end of our tenth year of publishing. If someone had asked me in 1984 whether we would be around in 1994, I would have laughed. In fact, I didn't expect us to last more than five or six years. But I'm happy to be here, and I hope to stay here for a few more years.

What amuses me, though, is that even after all these years many readers jump to the conclusion that we're out of business if the magazine is a little late arriving in the mail. This happened in November when we were late because of the holiday and a few weather-related problems. Just to reassure everyone, if we ever shut down MICROpendium we will tell you about it before it happens. Many of you have been reading MP for so long that we wouldn't think of treating you as anything less than old friends.

—JK

FEEDBACK

Wishes for columns

I just got my November 1993 issue of MICROpendium to "Goodbye" to Regena. We will all really miss her columns and programs. You mentioned you would like a bi-monthly column on the Geneve. I second that idea. I think the Geneve is as much a mystery now as the TI was back in 1983. Then, finally, we had quite a few hackers who uncovered the secrets, and systems information started to become available. When this happens, all sorts of software becomes available. The Geneve badly needs this.

I was probably one of the first people to buy GenProg Assembler and Linker by Paul Charlton. Unfortunately, my .97H M-DOS couldn't even find the Library so I had to give up the idea of writing M-DOS five mode programs. Then we got TIC. This is terrific. All that I have done in TIC has worked like a charm (and in native M-DOS, too).

I would like to see a column regularly by Paul Charlton or Clint Pulley or somebody of that understanding of the system. With this information, maybe we could write some TSRs (that would load and run to initialize certain things, then return to the system but stay in memory for an interrupt to run it at any time later). I would think this principle could be used, for example, to convert all characters in an M-DOS command line to uppercase, or maybe check and flag a CD command for an invalid directory. Maybe we could patch up some of the bugs.

I really enjoy MICROpendium — keep up the good work.

Norm Sellers
Broomall, Pennsylvania

TI-Writer a helper

I bought my TI99/4A in 1982 and as I am considered over the hill (72 years) I have no reason to get another brand of computer. I have old age shakes so my handwriting is no longer legible and TI-Writer is my helper. I have two RAM cards, a 990 sector and a 1550 sector; six pack widget; and a Honeywell Keyboard, so you may see I have a well complemented library to go with my setup.

On the October MICRO disk the poker game would lock up. Without Disk Editor I would be lost; I saw that I had to change the disk name and then the program operated. I think there are many TIers who received the disk and think there is a glitch in the program because they did not use Disk Editor.

Herb Hogberg
Sunrise, Florida

Comments and info

I would like to offer a couple of comments pertaining to items that appeared in the November 1993 issue of MICROpendium, and a random bit of info I recently discovered about the Personal Record Keeping cartridge.

(See Page 6)

FEEDBACK

(Continued from Page 5)

First, a warning: The Card File program that Charles Good reviewed in the November issue, which I will call Card File November, is not the same Card File that you were kind enough to publish the code for in the October 1993 issue. Card file October was a program that I wrote in 1991 and submitted in 1991 for your consideration, which is why the accompanying article had Brad Snyder's old address (see Newsbytes, p. 15, November 1993). The program that Charles Good reviewed, Card File November, is similar, but uses the standard 10-character file naming methodology that 99ers are familiar with when loading and saving files. Card File October does not use the standard 10-character file naming methodology. Therefore, users who may intend to purchase any of the Card File Libraries mentioned in the Charles Good review for use with the Card File October program that they keyed in or received on their MICROpendium On-Disk will find that the libraries will not load because the file naming schemes are not compatible. Please be warned!

Second, in the Reader to Reader column, Thomas P. Kirkpatrick asked if someone would write a GW BASIC version of the Personal Record Keeping module so he could enjoy its ease of use on a PC. My guess is that no one has done it, and, if it does not get done, a near-perfect solution to Mr. Kirkpatrick's request is to purchase PC 99. As many users today realize, PC 99 enables most 99/4A software to be run on an MS-DOS machine without any modifications to the PC. Simply run PC 99 and you have a TI99/4A computer inside your PC. Not only that, Mr. Kirkpatrick can also buy the Personal Record Keeping Cartridge already converted to run in the PC 99 environment. PC 99 is available from CaDD Electronics, 81 Prescott Rd., Raymond, NH 03077, (603) 895-0119.

Finally, most 99ers are aware of the lack of parallel printer support in the Tax/Investment Record Keeping module, which has stood as a banner to TI's lack of foresight since the program's introduction. However, I recently discovered another example of their shortsightedness in the programming of the Personal Record Keeping cartridge. When PRK is first loaded you are prompted for the date. Try entering a date prior to 1979 or after 1999. Unless your PRK cartridge is different from mine, you'll find the program will not accept either. Neat, huh?

Bill Gaskill
Grand Junction, Colorado

Quadruple Quest

My latest project involves a modification of both the Quest manager program and the PAL chips that control the CRU addresses. The Quest, as you probably know, can be addressed at >1000, >1400 and >1600. That effectively limits the number of cards that can be present in any system to a total of three. I acquired my fourth Quest as a Christmas present from one of my Aussie "mates" and greed quickly set in. I have the commented source code that I got from Ron Kleinschafer years ago and have made some modifications to allow the entry of drive designations 1-9 and A-F which would be enough for six Quest RAMdisks. The CRU addressing is a bit more complicated and I have enlisted the help of Don Walden at Cecure electronics to advise me if it is possible to rewrite the PAL equations to allow for addresses at >1800, >1C00 and >1E00. If it can be done, I'll have him burn a couple of sets of chips and install my fourth Quest in my P-Box.

Bob Carmany
Greensboro, North Carolina

Old information new to many readers

I was just reading the November 1993 issue of MICROpendium which I just received in January 1994. The letter by Jim Uzzell of Key West, Florida, is a disservice to all TI and Geneve users.

Why would you say something like that? you ask. Well, let me try to enlighten all of the programmers and the people who have used all programs and done everything that there is to do on or with the TI now and in the future. I am 67 years young, have owned and used a TI99/4A since 1982 and am the only TI or Geneve user in the Tallahassee, Florida, area. If there are others, they have eluded me.

I have five computers up and running:

- One TI99/4A with a Myarc HFDC and two hard drives, 5¼ and 3½ floppy drives.
- One Geneve with Myarc HFDC and two hard drives, 5¼ and 3½ floppy drives.
- One Atari 800/XL upgraded to ¼ -meg memory.
- One Atari 520ST upgraded to 1 meg. memory running as a BBS — (904) 576-8158.
- One Atari Mega ST4 which has 4 meg. memory, running FAX software, 14,400 baud FAX modem.

As you can see from the equipment that I am running, I know something about computers. I don't know everything that has ever been produced for the TI or Geneve or everything that has been written about them. What I am attempting to get across is this. If you know something about the TI or Geneve, that is great. But because you know or have read somethir doesn't mean that all of us know or hav read the same thing. If that were so, none of us would be subscribing to MICROpendium or searching for new and better programs. I just purchased CFORM from Cecure Electronics. It is a good program for the people who own a Geneve.

Every day I am searching for some information about the TI or Geneve. If you don't believe me, ask Don Walden of Cecure. My phone bill to him is rather large each month. We *do not* need people who criticize others because they ask questions or submit articles that you know the answer to. There are a lot of people who need the same information that you have but you are not giving it to them, just trying to keep it from them, by the comments in the letter.

Richard C. Arthur Jr.
Tallahassee, Florida

Print the ampersand an easier way

Leslie R. Hadley of Levittown, Pennsylvania, wrote a tip in his letter in the September 1993 edition describing how

(See Page 7)

FEEDBACK

(Continued from Page 6)

he uses Transliterate commands for printing the ampersand — it is much simpler than that!

Just type it in *twice* and it will print *one* ampersand through the formatter.

The same applies to the at sign.

E.g. I still have a B&W television (in a cupboard somewhere!)

You have no choice but to use Transliterates for the circumflex (caret).

George J. Clark
Pointe Claire, Quebec, Canada

Reasons, not excuses

One of Mr. Brashear's personal resolves and comments in the November 1993 MICROpendium struck a raw note.

His comment No. 1 states "there is 'no excuse' for the members of the TI community today not to have this capacity (DSDD devices) when you can pick up the devices for as low as $5 apiece quite of-."

Excuse means to make apology. There may be "reasons" why some people don't have DSDD devices. First, some of us are on a fixed income. I began in 1983 with nothing but a console and a TV. I added hardware as I could afford it. It was only in 1992 that I felt able to buy the second disk drive. Second, I have never seen a DSDD disk drive for as little as $5.

I have a good disk library that I have accumulated through user groups, TI99 suppliers, MICROpendium and fairware. I have never failed to pay for a fairware program even though I may not have ever used it. I even have some Asgard disks.

I am not a programmer and no one will ever know how much I appreciate those who have kept the TI99/4A going. I don't always understand the articles in MICROpendium, but I learn from the experiences of others who write User Notes, Feedback and Comments.

I intend to stay with the TI99/4A although I may never have as much hardware as I would like to have. Each person has to use individual judgment as to what is affordable. So it really doesn't come down to "excuses" or "apologies," does it?

Ouida Thompson
Tacoma, Washington

***Feedback is a reader forum. The editor may condense excessively lengthy submissions if necessary. We ask that writers limit themselves to one subject per submission. Our only requirement is that submissions be of interest to those using the TI99/4A, the Geneve 9640 or compatibles. Send items to* MICROpendium Feedback, P.O. Box 1343, Round Rock, TX 78680.**

READER TO READER

❑ Richard C. Arthur Jr., 2626 Peachtree Dr., Tallahassee, FL 32304-1238, (904) 576-4577, GEnie R.C.ARTHUR, writes:

I have a question for the subscribers to MICROpendium. Does anyone know if there has been an upgrade to Telco that will allow you to save to the hard drive. At present I can only save to DSKn. This is normally OK, but when on GEnie and downloading, it gets tiresome waiting for the save to disk function. The floppy disks are so slow and the hard drives are much, much faster.

Reader to Reader is a column to put TI and Geneve users in contact with other users. Address questions to *Reader to Reader, c/o MICROpendium, P.O. Box 1343, Round Rock, TX 78680.* We encourage those who answer the questions to forward us a copy of the reply to share with readers.

Extended BASIC

Protecting your programs

By BARRY TRAVER

Let's suppose that you have a TI Extended BASIC program that you would like to protect. That is, you would like to be able to distribute your program (a) without giving away all your programming secrets and (b) without allowing others to modify your program. What can you do to accomplish this purpose?

One way is to prevent your program from being LISTable. Texas Instruments included such a technique in TI Extended BASIC itself. All you have to do is save your program from memory to disk in this fashion:

```
SAVE DSK1.MYPROG,PROTECTED
```

Now, whenever anyone enters OLD DSK1.MYPROG and tries to LIST the program, the program will not be LISTed on the screen; instead, a honk will sound and the message "* Protection Violation" will appear on the screen. (Even though the program cannot be LISTed in this situation, it will still RUN with no problems as before.)

So far so good. The problem with this technique, however, is that so many people know how to undo that protection! For example, a number of CALL LOADs will undo the protection, including the following sequence of commands:

```
OLD DSK1.MYPROG
CALL INIT
CALL LOAD(-31931,0)
SAVE DSK1.MYPROG
```

The program is now LISTable, and the protection has disappeared. (There are more sophisticated ways of making an XB program unLISTable; I may discuss one in a future article, if there is interest.)

Rather than making the program unLISTable, you might try an entirely different approach: using "reserved words" (ASC, FOR, INPUT, NEXT, POS, PRINT, RPT$, SEG$, TO, etc.) as variable names! I know, you aren't supposed to be able to do this (and it would be foolish to do it, anyway, until your program has first been perfected to your satisfaction), but it is possible, if you use my XB-PROTECT program in this month's issue of MICROpendium!

Using "reserved words" as variable names accomplishes not one, but two things: it makes the code difficult or impossible to understand (since from the LISTing it is not easy, for example, to determine whether "INPUT" is being used as a TI XB statement or as a regular numeric variable), and it makes modifying individual lines of the code difficult or impossible (since TI XB will attempt, for example, to "tokenize" all instances of "INPUT", including those instances where "INPUT" was being used as a variable name rather than as a XB statement!).

You don't have to understand what's happening in order to be able to use my XB-PROTECT program, but let me try to explain it a bit more for those who are interested. Let's suppose you are doing some normal programming and enter the following line:

```
100 INPUT M$
```

The variable "M$" takes up in memory the space of two characters (or bytes), as you might expect: "M" plus "$". Or, to express the same thing using the ASCII table, what is put into memory is CHR$(77) plus CHR$(36). BUT, contrary to what you might expect from looking at the LISTing on the screen, the statement "INPUT" does not take up five characters (or bytes) in memory. Rather TI XB converts that "INPUT" statement into a one-character, single-byte, "shorthand" token, CHR$(146). You can't tell that from looking at the LISTing, but that's what's happening.

The TI XB interpreter does this automatically as you are entering lines, and you can't stop it. That is, whenever it encounters "INPUT" (except the special case, of course, where that sequence of letters is within quotation marks, which means it is intended as a string literal rather than a numeric variable anyway), the XB interpreter will convert that five-character sequence into a one-character token, i.e., it will convert "I" plus "N" plus "P" plus "U" plus "T" to the "shorthand" token CHR$(146).

Here's an example. Let's suppose you enter the following line:

```
100 INPUT=2
```

When you enter that line, it looks for a moment as if you just might have been successful in using "INPUT" as a variable name: there is no honk and no error message after you press enter. Well, appearances are deceiving. Unseen by you, the TI XB interpreter has really converted that "INPUT" into CHR$(146), and if you try to run the program, it will crash at that line, make a honk, and tell you, "* Syntax Error in 100".

How then can you "trick" TI XB into letting you use "INPUT" as a variable name? The answer is to follow XB's rules on variable names for the time being while you are writing the program, save your program to disk in MERGE format when you're satisfied with it, substitute some "not allowable" variable names for the "allowable" variable names (while the TI XB interpreter isn't looking!), and MERGE the program back into memory once these changes have been made! (The "tricky" part is the "substitute" part, and that's where my XB-PROTECT program plays its part.)

You do have to do one thing before you can use XB-PROTECT: you have to add some DATA statements so that the program knows what substitutions to make for your variable names. To illustrate what I mean, I've included with this article some sample DATA statements to add to XB-PROTECT so that you can run XB-PROTECT to protect a version of itself! (With XB-PROTECT in memory, the next thing to do would be to enter SAVE DSK1.OLDPROG,MERGE. See the third paragraph below for the entire procedure.) Try it out! After you've created your new version of XB-PROTECT, then LIST it to the screen. It will look really strange, but it will run just like the old one (but will prob-

(See Page 9)

XB-PROTECT—

(Continued from Page 8)

ably crash if someone tries to modify it).

As you can see from the example, the format for the DATA statements is simple. For each DATA line, just include two items: the first is your original variable name, and the second the variable name to which you want it to be changed. (String variables, of course, must end in a "$".) The final DATA pair should be "STOP,STOP" so that (you guessed it!) the program will know that it has come to the end of the list of desired replacements and can STOP!

You can change as few or as many variable names as you like. The more changes you make, of course, the longer it will take XB-PROTECT to do its work. (Since my sample DATA statements to modify XB-PROTECT include DATA for the replacement of eighteen variable names, for instance, it can take a while to handle each program line. Fortunately, XB-PROTECT is a fairly short program.)

After you've loaded the program that you want to change into memory, here is the series of commands you should enter:

```
SAVE DSK1.OLDPROG,MERGE
OLD DSK1.XB-PROTECT
```

At this point add appropriate DATA statements to the program.

```
RUN
```

When asked, supply DSK1.OLDPROG and DSK1.NEWPROG as names of MERGE files.

```
NEW
MERGE DSK1.NEWPROG
```

That's all there is to it! You can save the new program under whatever name you like (but I strongly recommend NOT using the name of the original program!).

I do have some concluding words of advice. The more multi-statement lines in your program and the more you make use of variables, the more protection that XB-PROTECT will be able to provide. Suppose the original line is this:

```
100 PRINT "Hello!"
```

XB-PROTECT can protect that line if you change it to this:

```
100 M$="Hello!" :: PRINT M$
```

Don't let your program lines be too long, however, if you want to replace your original variable names with "reserved word" names that may be longer than the originals. (Using that FCTN-Redo trick to allow you to have longer program lines than TI XB normally allows is not generally a good idea!)

In the tests I have done, XB-PROTECT seems to perform properly in the especially tricky task of detecting and replacing single-letter variable names (e.g., "I" or "J"). Having slightly longer variable names, however, in your original program might be slightly safer (e.g., using "II" rather than "I" as a loop counter). As you can see from the LISTing for XB-PROTECT, the task of replacement is not as simple a matter as it might at first seem: if you want, for example, to replace "I" with "INPUT" as a variable name, you wouldn't want to replace every "I" in your program, but only those where the "I" is used as a variable name.

My last word of advice is a word of warning: be sure to keep a copy of your unmodified, original program, because without the original, you may have as much trouble as anyone else getting back from the confusing new version to what you had before!

DATAM

```
1000 ! DATA STATEMENTS TO BE
USED WITH XB-PROTECT TO CREA
TE A PROTECTED VERSION OF XB-
PROTECT ITSELF
1010 DATA A,FOR
1020 DATA B,TO
1030 DATA C,NEXT
1040 DATA END$,STR$
1050 DATA I,INPUT
1060 DATA IN$,CHR$
1070 DATA INFILE$,LEFT$
1080 DATA J,POS
1090 DATA LASTP,LEN
1100 DATA LI,DATA
1110 DATA M$,RPT$
1120 DATA M1$,SEG$
1130 DATA OUT$,MID$
1140 DATA OUTFILE$,RIGHT$
1150 DATA P,GOTO
1160 DATA P1,GOSUB
1170 DATA TEST,ASC
1180 DATA VARCTR,PRINT
1190 DATA STOP,STOP
```

XB-PROTECT

```
100 ! XB-PROTECT - Copyright
 1993 by Barry A. Traver, 83
5 Green Valley Drive, Philad
elphia, PA 19128 !156
110 DIM IN$(100),OUT$(100)::
 END$=CHR$(255)&CHR$(255)::
VARCTR=1 !249
120 CALL CLEAR :: CALL SCREE
N(5):: FOR I=0 TO 14 :: CALL
 COLOR(I,16,1):: NEXT I !234
130 PRINT "XB-PROTECT":" (C)
 COPYRIGHT 1993":"  BY BARRY
 A. TRAVER" !245
140 PRINT :"This program pro
tects XB":" programs by allo
wing the":"  use of ""reserv
ed words""":"  as variable
names!":"" !077
150 PRINT :"IMPORTANT:  Appr
opriate DATA":" lines must b
e added to this":"  program
before you can use":"  it."
:"" !197
160 ON ERROR 510 !008
170 READ IN$(VARCTR),OUT$(VA
RCTR)!159
180 IF IN$(VARCTR)="STOP" TH
EN VARCTR=VARCTR-1 ELSE VARC
TR=VARCTR+1 :: GOTO 170 !052
190 ON ERROR STOP !216
200 PRINT "INPUT MERGE FILE?
" :: INPUT "":INFILE$ !126
210 ON ERROR 220 :: OPEN #1:
INFILE$,DISPLAY ,VARIABLE 16
3,INPUT :: ON ERROR STOP ::
PRINT :: GOTO 240 !107
220 ON ERROR 230 :: CLOSE #1
 :: ON ERROR STOP !098
230 PRINT :"FILE ERROR - TRY
 AGAIN!":"" :: RETURN 200 !0
29
240 PRINT "OUTPUT MERGE FILE
?" :: INPUT "":OUTFILE$ !065
250 ON ERROR 260 :: OPEN #2:
OUTFILE$,DISPLAY ,VARIABLE 1
63,OUTPUT :: ON ERROR STOP :
: PRINT :: GOTO 280 !132
260 ON ERROR 270 :: CLOSE #2
 :: ON ERROR STOP !140
270 PRINT "FILE ERROR - TRY
AGAIN!":"" :: RETURN 240 !14
4
280 IF EOF(1)THEN 410 ELSE L
INPUT #1:M$ :: IF M$=END$ TH
```

(See Page 10)

XB-PROTECT—

(Continued from Page 9)

```
EN 400 ELSE M1$=M$ !157
290 A=ASC(SEG$(M$,1,1)):: B=
ASC(SEG$(M$,2,1)):: C=256*A+
B !038
300 TEST=ASC(SEG$(M$,3,1)):: 
 IF TEST=131 OR TEST=147 OR
TEST=154 THEN 380 !239
310 LASTP=POS(M$,CHR$(131),1
):: IF LASTP=0 THEN LASTP=16
1 !146
320 FOR J=1 TO VARCTR !017
330 IF IN$(J)="STOP" THEN 37
0 !177
340 LI=LEN(IN$(J)):: P1=3 !1
96
350 P=POS(M$,IN$(J),P1):: IF
 P>LASTP THEN P=0 !111
360 IF P THEN P1=P :: GOSUB
420 :: GOTO 350 !186
370 NEXT J !224
380 IF M$=M1$ THEN PRINT :"L
INE ";STR$(C);":";RPT$(" ",1
7);M$:"(NO CHANGE)" :: GOTO
400 !240
390 IF M$<>M1$ THEN PRINT :"
OLD LINE ";STR$(C);":";RPT$(
" ",13);M1$:"NEW LINE ";STR$
(C);":";RPT$(" ",13);M$ !000
400 PRINT #2:M$ :: GOTO 280
!164
410 CLOSE #1 :: CLOSE #2 ::
STOP !203
420 TEST=ASC(SEG$(M$,P-2,1))
:: IF P>4 AND(TEST=199 OR TE
ST=200 OR TEST=201)THEN P1=P
1+1 :: RETURN !113
430 TEST=ASC(SEG$(M$,P-1,1))
:: IF P>4 AND(TEST=199 OR TE
ST=200 OR TEST=201)THEN P1=P
1+1 :: RETURN !112
440 TEST=ASC(SEG$(M$,P+LI,1)
)!003
450 IF (TEST>31 AND TEST<127
)THEN P1=P1+1 :: RETURN !115
460 IF P=3 THEN GOSUB 500 ::
 P1=P1+1 :: RETURN !138
470 TEST=ASC(SEG$(M$,P-1,1))
!105
480 IF (TEST>31 AND TEST<127
)THEN P1=P1+1 :: RETURN !115
490 GOSUB 500 :: RETURN !079
500 M$=SEG$(M$,1,P-1)&OUT$(J
)&SEG$(M$,P+LI,LEN(M$)-P-LI+
1):: RETURN !055
510 PRINT "Please do that no
w before":" proceeding.  Tha
nk you!":"" !243
```

THE ART OF ASSEMBLY — PART 31

'This is a Football'

By BRUCE HARRISON
©1994 B. Harrison

This month we'll start with a short story, just to have a laugh before the serious stuff starts. The story goes that the legendary football coach Vince Lombardi once got so disgusted with his team's performance in a game that he called a Monday morning meeting and announced that "We are going back to the fundamentals". He held up a football and spoke very slowly and clearly. "Gentlemen, this is a football." A voice from a player in the back of the room piped up, "Wait a minute, coach, you're goin' too fast." There was a moment of fearful silence from all the players, then coach Lombardi just started to laugh, and all joined in.

This month we're starting at the beginning, for those who have never tried Assembly. In the rest of this column, we will make every effort not to "go too fast" for any of our readers. We all know that Assembly is not an easy language to learn, and that mastering it is a term we simply don't use. Many readers have contacted us to say that, while they enjoy reading these columns, they don't really understand the subject matter. We hope this one will be easy to grasp from beginning to end.

JUST WHAT IS ASSEMBLY, ANYWAY?

Back in the bad old days when digital computers were first being produced, there was no such thing as Assembly Language. Programmers had to write their code in actual machine language, making each instruction in binary notation. This was a very tedious way of making programs, and since binary is a very unfriendly system of numbers for humans to deal with, errors in coding were the rule rather than the exception. Minor improvements came along, with use of either octal or hexadecimal coding, but still the work was tedious and error-prone.

The introduction of Assembly was a real blessing, since it allowed the use of mnemonic symbols that were human-readable to represent quantities and addresses in the source code. Now the programmer could work with a sort of words consisting of letters that meant something. It's a lot easier, after all, to recognize the word FAC in a source file than to pick out the address written as the binary number 10000011 01001010. (That's >834A, incidentally.) Assembly allowed the programmer to work with a kind of quasi-English language, then let the Assembler convert the code into the binary numbers that the computer understands.

As with labels for addresses, Assembly allowed mnemonics to represent the machine instructions as well, so that an instruction in source code could read LI R0,-1, instead of the binary 00000010 00000000 11111111 11111111 or the Hex 0200 FFFF. As mysterious and terse as Assembly source code may be when compared to languages like BASIC, it's still light years ahead of the binary or hex representations of the same addresses or instructions. Assembly, then is a language readable by humans, which can be readily translated by the Assembler into machine code in the binary notation.

SOME TERMINOLOGY

One of the things one encounters when trying to learn any language is the difference between nouns and verbs, adjectives and adverbs, and so on. In human languages these are called "parts of speech". In Assembly, we'll just use the word terminology. Here's

(See Page 11)

THE ART OF ASSEMBLY—

(Continued from Page 10)

a short glossary of terms that are used every day in Assembly programming:

LABEL - a short human-readable quasi-English word that can represent a location (address) in memory or a quantity. The programmer usually invents his own labels. On the TI, these must begin with a capital letter, and may not be longer than six characters.

INSTRUCTION - an abbreviation that represents an operation to be performed by the computer.

EXPRESSION - usually a number or a combination of numbers or labels that the assembler can calculate into an address.

ADDRESS - a specific place in the computer's memory, either a byte location or a word location.

BIT - contraction for Binary Digit. A two-state quantity that may only have the values 1 or 0.

BYTE - a group of eight bits treated as a unit. May take on values from 0 through 255.

WORD - a group of sixteen bits treated as a unit. May take on values from 0 through 65,535. (On some computers, such as the big "mainframe" units, the word length can be 30, 32, 60 or 64 bits.)

DIRECTIVE - an instruction to be executed by the Assembler, not the assembled program.

FIELDS - the different parts of a line in the source code.

LABEL FIELD - the very beginning of a line in the source code. (Most lines do not contain a label.)

OPCODE FIELD - the instruction part of the line.

OPERAND FIELD - the numbers, labels, or expressions upon which the operation is to be performed.

REGISTER - a special purpose word of memory, on which certain operations can be performed that cannot be performed on words in ordinary memory

WORKSPACE - On the TI, this is a set of sixteen registers. The user can select any part of the addressable RAM memory as a workspace for his use. (On most other computers, workspace registers are not assignable, but are considered part of the CPU's private reserve of memory.)

There are additional terms sometimes used, but we'll try to explain those when used, rather than extend the glossary.

USES FOR LABELS

Those who program in BASIC are familiar with the concepts of line numbers and variables. The Labels used in Assembly language can serve as line numbers, variable names, or even as constants, depending how they're used in the source code context. Here are three examples to illustrate these three uses:

EXAMPLE 1 - label used like a line number

```
KEYIN BLWP  @KSCAN            Scan the keyboard
      CB    @STATUS,@ANYKEY   see if a key is struck
      JNE   KEYIN             If not, go back to label
KEYIN
```

EXAMPLE 2 - label used like a variable name

```
      INC   @COUNT            add one to the variable
count
      COUNT DATA 0            One word of memory re-
served as variable
```

EXAMPLE 3 - label used as a constant

```
BUF   EQU   >1050  Make the word BUF represent >1050 ...
      LI    R0,BUF  load register zero with the value BUF
```

Each of these examples is of course just a fragment of source code, and would result in just one tiny operation when assembled. Example 1 would simply cause the computer to loop endlessly between the label KEYIN and the JNE instruction until the user strikes a key on the keyboard. The labels STATUS and ANYKEY are used here as a constant and a "variable", respectively. The label KSCAN is a constant that's usually set by a REF directive.

The small loop in Example 1 is equivalent to an Extended BASIC line like this:

120 CALL KEY(0,K,S):: IF S<>1 THEN 120

Example 2 is one of the simplest operations possible, since it simply adds one to the value of a variable stored at the word location named COUNT. In XB, it would be COUNT = COUNT + 1.

Example 3 has no direct equivalent in XB, because XB doesn't allow the user to mess around with its workspace registers. What the example does is to place the value >1050 into the first word in the workspace, which is identified by the name R0. Each word in the workspace can be identified in the source code by a simple label like R0, R1, ... R15. The Assembler will allow the R to be omitted from these labels, so that R15 could be shown as simply 15. (We find this too confusing for us, even if it's not confusing to the Assembler, so we always use those Rs, and use the R option on the Assembler.)

A LITTLE EXERCISE

Are we "goin' too fast"? We hope not. Here's a little exercise for your fingers, to get you into the "swim" a bit. Try typing in the following source file with the E/A Editor. Save it to some disk as TEST/S, then run the Assembler on it, giving the object file name as DSKx.TEST/O, using the R option, then use the LOAD AND RUN option (3) from E/A to run it. The PROGRAM NAME is START (very imaginative). When typing in, place the label field starting at the very leftmost column.

```
        DEF   START             define our entry point
        REF   VMBW,KSCAN        refer to predefined labels
STATUS  EQU   >837C              the GPL Status byte
WS      EQU   >20BA             the User Workspace
GPLWS   EQU   >83E0             the GPL interpreter's work-
space
START
        LWPI  WS                set the workspace pointer at
>20BA
        LI    R0,11*32+2        set R0 to point at row 12,
column 3
        LI    R1,MSG            set R1 to point at label MSG
        LI    R2,27             set length of MSG in Register
2
        BLWP  @VMBW             write multiple bytes to
screen

KEYIN
        BLWP  @KSCAN            scan keyboard
        CB    @ANYKEY,@STATUS   test for keypress
        JNE   KEYIN             if none, repeat scan
```

(See Page 12)

THE ART OF ASSEMBLY—

(Continued from Page 11)

```
        LWPI   GPLWS              else set workspace for GPL
        B      @>006A             then branch back to E/A

        MSG    TEXT 'Welcome Assembly Programmer'
ANYKEY  BYTE   >20
        END
```

Okay, so that didn't do much, but consider it a "first" program written in Assembly. It's a complete program, and will do something with your computer. More importantly, it will teach in an "object lesson" some of the basic concepts we've been trying to make clear. It illustrates all the uses of labels, for example, plus some directives. We could have made it more complex, and made it do more, but that would defeat our purpose of making an extremely simple Assembly program.

We'll try now to explain what each line of this source code does in some detail, so you'll start to get a "feel" for how this mysterious language works.

The first line says DEF START. This is a directive for the Assembler, and it tells the Assembler to identify the label START as an entry point, so that when the Option 3 object file is loaded, the E/A module will know where to start executing this program.

The next line is a REF directive. The two labels listed after this directive (VMBW and KSCAN) are pre-defined as addresses by the E/A module. When the object file is loaded, each occurrence of VMBW and KSCAN will be replaced by the actual address of the utility "vector" named in the source file.

The next three lines all establish "constants" for the Assembler. After the Assembler encounters these lines, it will look for those three labels in the source file, and will place the number found after the EQU directive into the object file in place of that label. These are placed early in the program, so that their values are established before they need to be used.

You could think of these as non-variable variables. Sometimes in XB, you'll find a variable identified like COLCOUNT=32, which is then never changed during the rest of the program. The difference between this XB example and the EQU directive in Assembly is that, in Assembly, once a label has been made a constant with EQU, it cannot be changed later in the program.

After the three EQU lines, we find the first label at which something is actually to be performed by the computer. Notice that we have placed this label on a line by itself. That's done simply to make it look better. The instruction LWPI WS could be on that same line, and this would make absolutely no difference to the Assembler. The mnemonic LWPI means Load Workspace Pointer Immediate. When the computer encounters this operating instruction, it will take whatever is in the very next word location after the instruction and place that in the workspace pointer, so that from that point on, your set of registers will start at memory location >20BA.

From here on, the computer will interpret the label R0 as the word of memory at >20BA, R1 as the word at >20BC, and so on. In the next three lines, we set values into the registers R0, R1, and R2. These lines use the LI instruction, which means Load Immediate. On the TI, only registers can be loaded with values through this directive. (On PC computers, other memory locations can be loaded with immediate values.) The instruction causes the contents of the next word to be loaded into the register identified in the first part of the operand field. In the first of these three lines, we have used what's called an "expression" as the immediate value, rather than a simple number. What happens here is that the Assembler evaluates the expression, and places just one number into the word following the instruction. In other words, the Assembler does the math for you, taking 11, multiplying that by 32, then adding two to that result. Thus, after Assembly, the single number 354
(11*32)+2] will be the "immediate" value loaded into memory as part of the program. This will result in display of our message Row 12, Column 3 on the 32-character screen.

The next line sets the register R1 to the address of the label MSG, through a LI instruction. Wherever that line of text happens to be in memory, the number representing that location will be contained in R1 after this instruction is performed. The line after this sets R2 to the value 27, which is the length of the message at label MSG. All three of these Load Immediate operations are done to prepare the registers for execution of the next line.

The instruction BLWP means Branch and Load Workspace Pointer. This can be thought of as similar to a subprogram call in XB, but here the registers serve to transfer all needed parameters to the subprogram. When the BLWP is performed, the workspace pointer will be changed, so that a set of registers associated with the "subprogram" will be used, and the workspace we were

(See Page 13)

1994 TI FAIRS

FEBRUARY

Fest-West, Feb. 19-20, Santa Rita Park Inn, Tucson, Arizona. Contact Tom Wills, Fest-West '94 Committee, Southwest 99ers Users Group, P.O. Box 17831, Tucson, AZ 85731 or (602) 886-2460; BJ Mathis, (602) 747-5046; or the Cactus Patch BBS, (602) 290-6277.

MAY

Lima Multi User Group Conference, May 14-15, Ohio State University Lima Campus, Lima, Ohio. Contact Lima Ohio Users Group, P.O. Box 647, Venedocia, OH 45894.

This TI event listing is a permanent feature of MICROpendium. User groups and others planning events for TI/Geneve users may send information for inclusion in this standing column. Send information to MICROpendium Fairs, P.O. Box 1343, Round Rock, TX 78680.

THE ART OF ASSEMBLY—

(Continued from Page 12)

using (at >20BA) will not be altered during the VMBW operation. VMBW means VDP Multi-Byte Write. In all cases, this operation takes the number of bytes indicated by R2, from the address pointed to by R1, and writes that number of bytes into the location in VDP RAM indicated by R0.

This particular case results that the words "Welcome Assembly Programmer" will suddenly appear at row 12, column 3 on the screen. Notice that we did not have to clear the screen here, as the Load And Run option from E/A did that for us.

From here on, the program is just a loop like the one we showed in an example before. The computer will scan the keyboard repeatedly until it finds that a key has been struck, and then will leave our program by reloading the workspace pointer to the Graphics Programming Language (GPL) workspace, then turning control over to the GPL Interpreter at location >006A.

The lines in what we'll call the DATA section contain only labels and directives. At label MSG, the directive TEXT tells the Assembler to insert the characters following between the ' marks into the object file as ASCII codes. Thus the first byte at that point in the memory would be the number 87, the ASCII value for uppe case W. The ASCII value for lower case e would follow, and so on until all 27 bytes of the message were included. The directive at label ANYKEY causes the BYTE at that memory location to be equal to >20. This is normally left alone, since it's there only to compare to the GPL status byte in checking for a keystroke.

The word END in the last line is not related to the same word as used in XB. Here, it's a directive to the Assembler, to tell the assembler it can stop assembling the source file. The actual end of the program in the XB sense of the word END comes at the line containing the instruction B @>006A, since that's where this program relinquishes control of the computer.

EXTENDED BASIC EQUIVALENT

This whole program could be written in two lines of XB program, like this:

```
 100 DISPLAY AT(12,1):"Welc
ome Assembly Programmer"
110 CALL KEY(0,K,S) :: IF S<
>1 THEN 110 ELSE END
```

That seems like an awful lot of Assembly to replace just a little piece of Extended BASIC, doesn't it? Yes, it is, but over time you'll come to appreciate the differences in terms of execution speed and memory use that all fall on the side of Assembly.

For the convenience of those who subscribe to MICROpendium on Disk, we have provided John with the little program above as a separate file named TEST/S, so you can avoid the hassle and time spent typing in the source code.

As always, we're just scratching the surface, and have already written too much for one sitting. Next month we'll continue this lesson for the beginners.

The Simplest, Easiest Kind of Database

Using the computer to search the Lima library disk listings for specific software

By CHARLES GOOD
Lima Ohio User Group

(Reprinted from Bits, Bytes & Pixels)

Several months ago at one of the Lima group's regular meetings, a member asked me where he could find Multiplan templates in the group's software library.

Since I didn't know, exactly, it was necessary to search printed directories and the commented listings of all of the 800-plus disks in the library to find them all. I decided there had to be an easier way. After all, commented listings of all disks in our library, and the disk directories, are all in DV80 text files. Surely the computer could search these text files and find all references to the word Multiplan. Why not use Funnelweb's Disk Review in I(nspect) library listing file and search that file or an entire disk of library listing files for the string "Multiplan"? Unfortunately, it wasn't as easy as that.

Over the years, as I have been adding disks to the group's library I have been adding DSKU comments to the disks in both upper and lowercase. A disk with Multiplan stuff may have a comment such as "All these files are Multiplan templates." Such commented listings look good and are easy to read when printed on paper, but it is hard for the computer to find all Multiplan references in such text. That is because sometimes I spelled it "MULTIPLAN" and sometimes "Multiplan." The 99/4A disk operating system doesn't know that these are both the same.

Another example would be Ms Pacman vs. MS PACMAN. An IBM system using MS-DOS would recognize both of these as the same. MS-DOS is not case sensitive; 99/4A DOS is case sensitive. What I should have done all along is USE ONLY UPPERCASE in our library listing text files. I will do this in the future.

I wrote the XB program below to convert existing Lima library listing files into all uppercase. Put as many of these all uppercase files as you can on a disk or RAMdisk. Using Funnelweb's Disk Review, bring up a disk directory, press I, select Disk Search, Choose ASCII and then type an uppercase key word or text string you want to look for. The computer will search *the entire disk* for this string and display each sector where the string is found. Use CTRL/N or CTRL/B to page sector display forward or backwards until you see the name or number of the disk that contains your software.

Members of the Lima Users Group can request these ALL uppercase DV80 library listings by sending the equivalent 12 DSSD disks and a paid return mailer to the group's mailing address.

This sort of text file data base has many uses. For example, if you have a large collection of music CDs, tapes and phonograph records you may have trouble trying

(See Page 14)

THE SIMPLEST, EASIEST KIND OF DATABASE —

(Continued from Page 13)

to find one particular song on all these media. Just take any TI-Writer-like word processor, such as Funnelweb, and use all uppercase to enter each tape or CD title, artist and all the songs. Use any format you want, such as putting all the information for one CD/tape in a separate paragraph. Save these data files to disk as ordinary text files. When you want to find all the references to a particular song, use F(ind) S(tring) from within the word processor to search a single text file, or I(nspect) from within Funnelweb's Disk Review to search a whole disk of files. It is easy and fast since the searches are at assembly language speed.

UPPERCASE

```
50 REM SAVE DSK6.UPPERCASE ! 247
60 REM Converts all lowercase of D/V80 file to uppercase . !107
70 REM Resulting all upperca se text can easily be !125
80 REM searched for text strings by sector editors. !098
90 ON ERROR 100 !109
100 CALL CLEAR !209
110 DISPLAY AT(3,2):"CONVERT DV80 TO UPPERCASE" !012
120 INPUT "Enter OUTPUT FILE path ":OUTPUTFILE$ !150
130 INPUT "Enter INPUT FILE path ":INPUTFILE$ !211
140 OPEN #1:INPUTFILE$,INPUT !100
150 OPEN #2:OUTPUTFILE$,APPEND !045
160 LINPUT #1:TEXT$ !191
170 PRINT TEXT$ :: PRINT !035
180 FOR T=1 TO LEN(TEXT$)!244
190 A=ASC(SEG$(TEXT$,T,1)):: IF A>96 THEN A=A-32 !203
200 B$=B$&CHR$(A)!198
210 NEXT T !234
220 PRINT #2:B$ :: B$="" !028
240 IF EOF(1)THEN 260 !051
250 GOTO 160 !239
260 CLOSE #1 !151
270 CLOSE #2 !152
280 GOTO 100 !179
```

Four Letter Words (Well, Almost)

By JAMES D. LANMAN

(Taken from Reading-Berks Newsletter of Nov/Dec 1993)

There are several standards for hard drives: SCSI, MFM and IDE are the ones which TI99/4A users are most interested in using. In the TI community, there is room for all of them. However, most of us will be hard pressed to afford a controller card and hard drive for our system. Each standard has its advantages and disadvantages which this article will address. TI99/4A users have a difficult choice in picking a standard. The following information should help.

SCSI (Small Computer System Interface — pronounced "skuzzy") is the latest standard to come to the TI. And there is nothing small about SCSI since it is used in many systems ranging from super computers to TI99/4As. The SCSI card is available from Western Horizon Technologies. It can use any combination of up to eight devices. SCSI isn't limited to just hard and floppy disk drives. The SCSI card with additional software drivers my be able to use CD-ROM drives, scanners, magneto-optical drives and laser printers. Only time, money and interest from TIers will determine if we software drivers for these other devices. This card also has PC transfer capability and RAMBO/4A Memex compatibility. As for SCSI hard drives, they can be found used or refurbished at relatively low cost ($100-$200) in 20, 40 and 80 MB sizes in 5¼-inch or 3½-inch formats. There is still life left in this standard. The HFDC card originally from Myarc will soon be reissued by Cecure Electronics. They also offer repairs and upgrades of Myarc products at a reasonable cost. *(EDITOR'S NOTE: Don Walden of Cecure Electronics says that a number of cards from Myarc's last production run need to be repaired before any production of new cards begins.)*

If you have an older HFDC card, you should check to see if you have a 32K SRAM. M11 Eprom (Note: The "M11" on my copy was smeared and therefore may not be correct.) and 9216B chip. If you only have an 8K SRAM and 9216 chip, get them replaced. Check with Cecure Electronics on upgrading your HFDC. As well as upgrading the HFDC, a new disk manager will soon be available called Uni-Manager. It is from Crystal Software. This new disk manager will support backup tape drives and a host of other features. MDM5 (Myarc's disk manager) will be a thing of the past by comparison. Cecure Electronics may even have a new DSR (Device Service Routine) written.

With an upgrade HFDC, disk manager and new DSR Eprom we may finally get to use 1.44MB floppy drives at full capacity. If you already have an HFDC or are getting one, there is light at the end of the tunnel!

Finally, we come to IDE (Integrated Drive Electronics). This is a very popular standard in the PC community. Drives are easy to find and are relatively inexpensive. If all you want is to add a hard drive to your TI, this would be the best way to go. Unfortunately, an IDE controller only exists as a prototype. And it is unlikely that it will ever go into production. An IDE controller card would use conventional floppy drives just like the HFDC does using DMA (Direct Memory Access) scheme. IDE, for now, is just a dream for TI99/4A users.

Texaments moves

Texaments has relocated to Oklahoma from the state of New York. The new address for the company is 701 S. Wicklow, Ste. 506, Stillwater, OK 74074. New phone number is (405) 372-0819.

Extended BASIC

KINGMOVE helps you explore artificial intelligence

By LUCIE DORAIS

Some food for your thoughts this month — we start to explore AI, Artificial Intelligence (because I got a little book on the subject; it is an antique book by computer standards, but, then, we have an antique computer).

The book is *Experiments in Artificial Intelligence for Small Computers*, by John Krutch (Sams, 1981); the programs were written for the Level II TRS-80, with lots of pokes, so the graphics are entirely mine. KINGMOVE is the first example, and serves to explain some of the basic ideas behind AI, namely the problems of representation, search and range. Here I quote from the book (page 9):

The problem of Representation is "how to represent complex data structures in a computer language." According to Krutch, Search is an acute problem: "frequently, an AI program will try to solve a given task by generating a huge multitude of possible solutions to the task, testing each solution as it is generated to determine if it's the right one .... The problem with the approach ... is that the set of possible solutions can be so enormous that the computer cannot generate and test all of them in a reasonable period." Finally, the problem of Range: to make an AI program workable, it is often necessary to let it "operate within certain narrow, well-defined limits," but then the program's utility is not too great.

KINGMOVE does only one thing, and all you have to do is watch it do its thing. A chess king (which looks more like a pawn, sorry) tries, by using some artificial intelligence, to reach the lower right corner by using the shortest route; of course, since the board is a square, this is the diagonal between the departure (upper left) and arrival (lower right) squares. The Representation is easy: draw the board, get a random move, then eliminate the useless ones (out of the board, left, etc.). The Search here is straightforward: Tex will try anything until it finds the shortest route; at this first stage, it is still stupid enough to try the same path over and over again. And so that you are not too bored watching it, the range is small; the original program used a 3x3 board; for variety, I added a 4x4 and a 5x5 board.

When the program starts (after you choose the size of the board), the king is at the upper left corner. It then can only move one square at a time, and only to the right, whether horizontally or diagonally. The fun of the game is to see how many tries the king will take; sometimes it does it on first try, sometimes it takes ages.

KINGMOVE

```
100 ! *** KINGMOVE *** by L.
 Dorais, Ottawa UG / Oct. 19
93 !135
110 ! Adapted from book "Exp
eriments in Artificial Intel
ligence for Small Computer"
by John Krutch (Sams 1981) !
017
120 OPTION BASE 1 :: DIM R(2
5),C(25):: L$=CHR$(137):: L$
=RPT$(L$,3)&" "&RPT$(L$,9)!0
90
130 GOTO 150 :: COL,CU,D,K,L
SQ,M,N,OC,ROW,S,SQ,ST$,TOT,T
RY,X,Z !104
140 CALL CLEAR :: CALL COLOR
 :: CALL KEY :: CALL CHAR ::
 CALL HCHAR :: CALL VCHAR ::
 !@P- !140
150 FOR X=9 TO 12 :: CALL CO
LOR(X,6,16):: NEXT X ! lower
case !032
160 CALL CHAR(96,"",128,"183
C3C183C7E7E7E",129,"00082A1C
361C2A08",136,RPT$("F",16),1
37,"000000FFFF"):: CALL COLO
R(13,9,16,14,15,1)!238
170 ! ===== start ===== !115
180 DISPLAY AT(1,10)ERASE AL
L:"KING'S MOVE": : :"TI WILL
 TRY TO FIND THE":"SHORTEST
ROUTE FROM THE":"UPPER LEFT
TO THE LOWER" !051
190 DISPLAY AT(7,1):"RIGHT S
QUARE BY TRYING":"ALL POSSIB
ILITIES, MOVING":"ONE SQUARE
 AT A TIME IN ANY DIRECTION.
" !122
200 DISPLAY AT(14,1):"ENTER
THE TOTAL NUMBER OF":"ROWS A
ND COLUMNS YOU WANT": :"(3,
4 OR 5)" !018
210 ACCEPT AT(17,13)BEEP:TOT
 :: IF TOT<3 OR TOT>5 THEN 2
10 !032
220 ! ===== screen init ====
= !153
230 RANDOMIZE :: N=TOT*2+3 :
: LSQ=TOT^2 :: TRY=0 !180
240 CALL CLEAR :: FOR ROW=4
TO N+1 :: CALL HCHAR(ROW,4,1
36,N-2):: NEXT ROW !153
250 SQ=0 :: FOR ROW=5 TO N S
TEP 2 :: FOR COL=5 TO N STEP
 2 :: SQ=SQ+1 !096
260 R(SQ)=ROW :: C(SQ)=COL :
: CALL HCHAR(ROW,COL,SQ+96):
: NEXT COL :: NEXT ROW !042
270 DISPLAY AT(1,16):"TRY PA
TH":TAB(16);L$ !032
280 ! ===== move ===== !252
290 CU=1 :: ST$="a" :: TRY=T
RY+1 :: CALL HCHAR(N,N,LSQ+9
6):: CALL HCHAR(5,5,128)!175
300 Z=INT(TRY/20):: ROW=TRY-
20*(Z+(TRY/20=Z))+2 !227
310 DISPLAY AT(ROW,15):USING
 "####":TRY :: CALL HCHAR(RO
W,22,96,N-4):: CALL HCHAR(RO
W+1,18,32,N)!006
320 M=INT(RND*(TOT+1))+1 ::
IF M>1 AND M<TOT THEN 320 !
only one move, RIGHT OR DIAG
ONALLY !109
330 IF M=1 THEN IF INT(CU/TO
T)=CU/TOT THEN 320 ! cannot
go right at end of row !042
340 IF M=TOT+1 THEN IF INT(C
U/TOT)=CU/TOT THEN 320 ! no
diag from end of row !156
350 IF CU+M>LSQ THEN 320 ! c
annot go out of board !069
360 OC=CU :: CU=CU+M !102
370 ST$=ST$&CHR$(CU+96):: DI
```

(See Page 16)

KINGMOVE —

(Continued from Page 15)

```
SPLAY AT(ROW,20)SIZE(-LEN(ST
$)):ST$ !224
380 CALL HCHAR(R(OC),C(OC),O
C+96):: CALL HCHAR(R(CU),C(C
U),128)! move king !015
390 IF CU<LSQ THEN 320 ELSE
OC=CU !238
400 FOR D=1 TO 300 :: NEXT D
 !243
410 IF LEN(ST$)>TOT THEN 290
 ELSE CALL HCHAR(ROW,N+17,12
9)!168
420 ! ===== end ===== !124
430 DISPLAY AT(24,2)BEEP:"[A
]GAIN  [N]EW  BOARD  [E]ND"
!212
440 CALL KEY(0,K,S):: IF S=0
 THEN 440 ELSE IF K=78 THEN
180 ELSE IF K=69 THEN END !2
43
450 IF K<>65 THEN 440 ELSE T
RY=0 :: DISPLAY AT(23,1):"":
"" ! redo !142
460 FOR COL=17 TO 30 :: CALL
 VCHAR(3,COL,32,20):: NEXT C
OL :: GOTO 290 !061
```

In line 120, the R and C arrays will keep the screen rows and columns of the board, so that a move to square No. 5 (for example, second row, second column of a 3x3 board) can be shown on screen; but since we can have more than nine squares, I used lowercase letters to identify them on screen (colored blue on gray, line 150). The chars defined in line 160 are an empty square (96), the king (128), an asterisk (129), a border square (136) and a nice underline (137). Instructions follow, and you are asked for the size of the board (lines 180-210).

Initialization: N holds the row and column of the lower right square (depending on the board size), LSQ the total number of squares (TOT exponent 2); the number of tries is reset to zero. The board is drawn by lines 240-260.

Each try starts in line 290: the CUrrent starting square is always 1, and the string for the current path is initialized to "a"; TRY is incremented, the lower right square reset to show its letter and the king put in upper left corner. In line 300, the ROW for displaying the current TRY string at the right of the screen is determined; if there are more than 20 tries, the display will start again at the top of the screen, erasing the next line to better show off (line 310).

Tex's artificial intelligence gets in gear in line 320: M is the number of squares to move; it has to be either one (to move right) or equal to the total (to move exactly one row down) or bigger than it (to move diagonally and to the right). Line 330 makes sure the next move is not outside the limits of the board (which happens if the CUrrent square is equal to the TOTal squares on a row), and line 340 does the same for a diagonal move one row below. Finally, of course, the Move cannot go further than the lower right square (line 350). Tex will get to line 360 only if the Move is legitimate; it will then add the Move value to the CUrrent square, keeping the old value in OC so that the king can be erased from there when it moves forward.

Since our squares are actually numbered with lowercase letters, the proper letter for the square is added to the path string and displayed in the Tries table (line 370). The old position is "erased" by placing back its letter in the square and the king is shown in its new location (line 380). Line 390 checks to see if the king has reached the lower right square; if not, go back to line 320 for the next Move. If the king has reached the bottom, a short delay will let you see it, then the path string ST$ is tested for length; since the diagonal is the shortest route, and since it has as many squares as the rows or columns, all we have to do is test for the path length; if it is greater than the TOT rows and columns, wrong path, try again at line 290, otherwise put a red asterisk beside the "winning" path (line 410). At this stage, you can play again, go back to the first screen to change the size of the board, or exit.

Please not that this is just the beginning — the king does not learn from one game to the next, and even in the same game, it can stupidly repeat the same path even if it proved to be wrong. This could, of course, be improved, but I will leave it to you to modify the program once you understand it! It should be fairly easy, but remember that the king would still have to travel a whole path in its brain before deciding it is wrong when compared to a previous wrong path, so the program would not run much faster, at least in XB.

Testing and installing RAMdisk batteries

By COL CHRISTENSEN

(Reprinted from BUG Bytes, the newsletter of the Texas Instruments Brisbane User Group, Australia)

During last week a problem developed with my RAMdisk. The ROS on powerup did not respond to set up the menu program. The ROS on powerup did not respond to set up the menu program.

This has happened from time to time over its five-year lifetime but I remembered that the grandchildren had been using the computer the last time it was on. Maybe the little fellow (he knows how to switch everything on or off just by switching the TV) happened to switch off and on quickly enough to "lock up" the processor so that it wiped some vital part/s of the ROS. So I reloaded the ROS but still could not access any of its "disks"; they simply weren't there. This was no ordinary problem, and I immediately dismissed the little innocents as being the cause of the fault.

The symptoms pointed to a loss of power at some time to all memory chips as well as the ROS chip. On removal of the RAMdisk, an inspection revealed nothing amiss such as a leaking battery cap. This can happen to aging nicads especially when overcharged at the maximum allowable rate. They tend to gas in-

(See Page 17)

RAMDISK BATTERIES—

(Continued from Page 16)

side, the pressure breaking the seal at the top cap and exuding a small amount of chemical at the same time. This eats into the battery holder terminals and forms a telltale whitish powdery mass. The discharge can also soak up inside the insulation of a cable attached to the battery terminal. I have seen wires that have been blackened this way for 300 mm. or more from the battery terminal.

The next test was to measure the voltage produced by the three batteries. With nicads with a terminal voltage of around 1.25 volts each, one would expect well over 3 volts, but the meter read 1.2 volts. No wonder the RAMdisk contents were lost, as the minimum needed to maintain memory, according to the data, is 2.2 volts.

This battery pack is one of a number I had made up for the first RAMdisks in our user group. The nicads were AAA type, smaller than AA pencells, and interconnected with wire soldered to the caps of the cells. This was covered with a length of heat shrink tubing. I couldn't yet tell whether there were any leaking cells or interconnecting wires eaten through until I had cut off the plastic tubing. This done and with the batteries removed from the circuit board, a voltage test on each individual cell now showed less than 0.1 volts on two of them and 0.2 volts on the other. Maybe the value of 1.2 volts across the lot previously when it was still connected to the circuit board really was what was still in a storage capacitor on the circuit board. The cells were clean externally, so I surmised that they had just given up the ghost.

Nicads are not designed for this use in RAMdisks. They are meant to be charged fully, then discharged within a reasonable time almost to the flat state and the cycle repeated up to 1,000 times during their lifetime. Ours in RAMdisks are charged at a low rate to prevent overcharging while you use the computer and discharge only at a minute rate when power is switched off. The cells would never reach the deep cycle state of charge and discharge. Nevertheless they last a long time, as mine have, and that's the main thing to consider.

I have replaced the nicads now with alkaline AA Duracells interconnected with wire and stuck to the board with silastic. Be warned when wiring cells end to end that many of the cheaper cells rely on being held under pressure in a battery holder so that good contact is made from the inner metal case to the outer cap on the bottom of the cell. If you use cells of this type, first slice off the bottom cap with a sharp knife and solder directly to the inner case. I tried this on the Duracells, but found that the bottom cap was already connected internally to the inner case.

Important: When using non-rechargeable batteries in the RAMdisk, the resistor connecting the positive end on the batteries to the PLS (+) point on the circuit board must be removed and replaced with a small signal diode, IN1914, with the anode band marking towards the PLS point on the circuit board. My Duracells are marked "best installed by JAN 1988," so they do have a long shelf life before showing signs of discharge, and I would anticipate that in use with but a few micro-amps of current drain that by January 1998 they will still be operational on the RAMdisk.

Five-card draw

Playing poker against the computer

Poker was written by the late Ron Johnson. Johnson, an Englishman, wrote a number of Extended BASIC programs during his lifetime.

Poker simulates five-card draw. Playing against the computer, you are permitted to draw up to three cards. Other options include folding, placing a bet and raising a bet. You start out with a set amount of money and try to break the house. The program requires Extended BASIC.

Good luck!

POKER

```
10 REM  POKER by R.JOHNSON !
021
20 REM  5/1/85 TI EXT-BASIC
!104
30 ON BREAK NEXT :: CALL CLE
AR :: GOTO 90 :: OPTION BASE
 1 :: DIM P2(5),C2(5),SUIT(4
)!054
40 DIM BINPV(14),BINCV(14),B
INPS(4),BINCS(4),P$(5),PV(5)
,PS(5),C$(5),CV(5),CS(5),S$(
5),PIC$(7),I$(30),B$(52),EX(
5),NUMB(5)!195
50 CALL COLOR :: CALL CLEART
OP :: CALL SCORE :: CALL KEY
 :: CALL CHAR :: CALL SAY ::
 CALL GRAPHICS :: CALL SORT
:: CALL CARDS :: CALL ANALYS
E :: CALL SCORE !12460 CALL
SCREEN :: CALL VCHAR :: CALL
 HCHAR :: CALL CHARPAT :: CA
LL MAGNIFY :: CA
LL SPRITE :: CALL DELSPRITE
:: CALL POKER :: CALL DECK :
: R$,A$=V$ !222
70 R,BET,K,S,V,CF,B,EE,CNP,P
NP,CK3,PK3,CK4,PK4,CF4,PF4,C
F5,PF5,CST,PST,CFH,PFH,CSF,P
SF,CRF,PRF,X,D,N,C,RB,POT,NN
,BP,BC,PM2,CM2=I !184
80 !@P- !064
90 CALL POKER !235
100 CALL DECK(B$(),PIC$(),I$
())!109
110 BP,BC=1000 :: DISPLAY AT
(1,1):"~                POKER
     ~" !050
120 REM!154
130 CALL CLEARTOP :: DISPLAY
 AT(16,1): : : : : : : : : :
 :: CALL SCORE(BP,BC,0):: IF
 BP>0 THEN 150 !227
140 DISPLAY AT(14,1)BEEP:"TH
ANKS FOR THE GAME":"BOUNCER
THROWS YOU OUT!" !097
150 IF BC>0 AND BP>0 THEN 17
0 ELSE IF BC<=0 THEN DISPLAY
 AT(14,1)BEEP:"YOU HAVE BROK
EN THE BANK":"BOUNCER THROWS
 YOU OUT!" !085
160 FOR I=1 TO 30 :: CALL CH
AR(I+96,I$(I)):: NEXT I :: C
ALL SAY("UHOH GOODBYE"):: RU
N "DSK1.LOAD" !043
170 C=1 :: RB=0 :: BP=BP-30
:: BC=BC-30 :: POT=60 :: CAL
L SCORE(BP,BC,POT):: CALL DE
AL(B$(),P$(),C$(),NN)!167
180 CALL GRAPHICS(P$(),NUMB(
),PIC$(),3):: CALL SORT(P$()
):: CALL GRAPHICS(P$(),NUMB(
),PIC$(),3)!093
190 DISPLAY AT(13,1)BEEP:"DO
 YOU WISH TO":"FOLD? N [N/Y]
": :: ACCEPT AT(14,7)VALIDAT
E("YN")SIZE(-1):A$ !076
200 IF A$="Y" THEN CALL CLEA
RTOP :: V$="C" :: BC=BC+POT
:: GOTO 130 !085
210 DISPLAY AT(10,1):"YOUR O
PTION":"TO REPLACE":"CARDS?
Y [Y/N]  [MAX3]": : : : :: A
CCEPT AT(12,8)VALIDATE("YN")
SIZE(-1)BEEP:R$ :: IF R$="N"
 THEN 280 ELSE X,D=1 :: N=0
!146
220 DISPLAY AT(11,1): :"REPL
ACE?" !076
230 DISPLAY AT(14,D)BEEP:CHR
$(NUMB(X));"N    [N/Y]?" ::
DISPLAY AT(15,D):SEG$(P$(X),
1,1)!088
240 ACCEPT AT(14,D+1)VALIDAT
E("YN")SIZE(-1)BEEP:A$ :: IF
 A$="Y" THEN P$(X)=B$(NN)::
N=N+1 :: NN=NN+1 :: IF N>=3
THEN 270 !089
250 IF A$="N" THEN DISPLAY A
T(14,D)SIZE(1):: DISPLAY AT(
15,D)SIZE(1)!222
260 X=X+1 :: D=D+2 :: IF X<6
 THEN 230 !040
270 CALL GRAPHICS(P$(),NUMB(
),PIC$(),3):: CALL SORT(P$()
):: CALL GRAPHICS(P$(),NUMB(
),PIC$(),3)!093
280 DISPLAY AT(10,1)BEEP:"TH
ESE ARE YOUR":"FINAL CARDS":
"----------":"DO YOU WISH T
O":"FOLD? N [N/Y]": :!238
290 ACCEPT AT(14,7)VALIDATE(
"YN")SIZE(-1):A$ :: IF A$="Y
" THEN 200 !119
300 CALL CLEARTOP :: DISPLAY
 AT(9,19)BEEP:"**********":T
AB(19);"*COMPUTER*":TAB(19);
"*THINKING*":TAB(19);"******
****" !229
310 CALL CARDS(P$(),PV(),PS(
),BINPV(),BINPS()):: CALL SO
RT(C$()):: CALL CARDS(C$(),C
V(),CS(),BINCV(),BINCS())!13
5
320 CALL ANALYSE(BINCV(),CV(
),BINCS(),CS(),CNP,C2(),CK3,
CK4,CF4,CF5,CST,CFH,CSF,CRF,
EX())!177
330 CALL EXCHANGE(C$(),EX(),
B$(),BINCV(),CV(),BINCS(),CS
(),NN,EE,CST,CF4,CF5,CFH)!08
2
340 CALL SORT(C$()):: CALL C
ARDS(C$(),CV(),CS(),BINCV(),
BINCS())!028
350 DISPLAY AT(11,19)BEEP:"*
REPLACES*":TAB(19);"* ";STR$
(EE);" CARDS*":TAB(19);"****
******" !124
360 CALL ANALYSE(BINCV(),CV(
),BINCS(),CS(),CNP,C2(),CK3,
CK4,CF4,CF5,CST,CFH,CSF,CRF,
EX())!177
370 B=-1000*(CK4>0)+100*CF5-
700*(CST=5)-400*(CK3>1)-200*
(CNP=4)-100*(CNP>0):: CF=0 :
: GOTO 400 !085
380 DISPLAY AT(9,1)BEEP:TAB(
19);"**********":TAB(19);"*C
OMPUTER*":TAB(19);"*FOLDS***
*":TAB(19);"**********" !077
390 V$="P" :: BP=BP+POT :: P
```

(See Page 19)

POKER —

(Continued from Page 18)

```
OT=0 !148
400 DISPLAY AT(14,1):"PRESS
ANY KEY TO PROCEED****" !158
410 CALL KEY(0,K,S):: IF S=0
 THEN 410 !223
420 IF CF THEN CF=0 :: GOTO
130 !080
430 CALL CLEARTOP !199
440 RB=0 :: IF V$="C" THEN 5
10 ELSE IF V$="P" THEN 460 !
167
450 IF RND<.5 THEN 510 !191
460 DISPLAY AT(12,1)BEEP:"YO
U HAVE FIRST  BET":"HOW MUCH
 DO YOU BET? 0" :: ACCEPT AT
(13,22)VALIDATE(DIGIT)SIZE(-
3):V !136
470 IF V=0 THEN DISPLAY AT(1
4,1)BEEP:"DO YOU QUIT THE GA
ME? N" :: ACCEPT AT(14,23)VA
LIDATE("YN")SIZE(-1):A$ :: I
F A$="N" THEN 460 ELSE BP=0
:: GOTO 130 !215
480 CALL SCORE(BP,BP,POT)!20
8
490 BP=BP-V :: POT=POT+V ::
RB=V !097
500 CALL SCORE(BP,BC,POT)::
DISPLAY AT(5,1):USING " BET#
###.":V :: GOTO 540 !016510
CALL CLEARTOP !199
520 BET,RB=10*INT((B*RND+1)/
10)-10*(B=0):: BET,RB=BET-5*
(BET=0):: CF=B=0 :: DISPLAY
AT(5,21):USING "BET####.":BE
T !137
530 DISPLAY AT(13,1)BEEP:"CO
MPUTER OPENING BET IS ";STR$
(BET):: POT=POT+BET :: BC=BC
-BET :: GOTO 960 !230
540 IF CF THEN 380 !003
550 IF RB<B*.8 THEN 930 !042
560 IF RB>B*2 THEN CF=1 :: G
OTO 380 !023
570 POT=POT+V :: BC=BC-V ::
A$="COMPUTER WILL SEE YOU***
****" !172
580 DISPLAY AT(5,1):USING "
BET####.          BET####."
:RB,RB :: DISPLAY AT(7,1)BEE
P:A$: : : : : : : :!252
590 CALL SCORE(BP,BC,POT)!19
5
600 DISPLAY AT(14,2):"PRESS
ANY KEY TO PROCEED*" !030
610 CALL KEY(0,K,S):: IF S=0
 THEN 610 !168
620 CALL CLEARTOP !199
630 DISPLAY AT(14,1)BEEP:"YO
UR HAND";TAB(16);"COMPUTER H
AND" !088
640 CALL GRAPHICS(C$(),NUMB(
),PIC$(),18)!087
650 CALL ANALYSE(BINPV(),PV(
),BINPS(),PS(),PNP,P2(),PK3,
PK4,PF4,PF5,PST,PFH,PSF,PRF,
EX())!103
660 IF CSF AND PSF THEN 790
!150
670 IF CSF THEN 840 ELSE IF
PSF THEN 890 !008
680 IF CK4>PK4 THEN 840 ELSE
 IF PK4>CK4 THEN 890 !229
690 IF PFH AND CFH THEN 750
!087
700 IF CFH THEN 840 ELSE IF
PFH THEN 890 !242
710 IF CF5 AND PF5 THEN 790
!090
720 IF CF5 THEN 840 ELSE IF
PF5 THEN 890 !204
730 IF CST=5 AND PST=5 THEN
790 !042
740 IF CST=5 THEN 840 ELSE I
F PST=5 THEN 890 !156
750 IF CK3>PK3 THEN 840 ELSE
 IF PK3>CK3 THEN 890 !225
760 IF CNP>PNP THEN 840 ELSE
 IF PNP>CNP THEN 890 !097
770 PM2,CM2=0 :: FOR I=1 TO
5 :: PM2=MAX(PM2,P2(I)):: CM
2=MAX(CM2,C2(I)):: NEXT I ::
 IF CM2>PM2 THEN 840 ELSE IF
 PM2>CM2 THEN 890 !132
780 REM ** ACE HIGH DOWN **
!003
790 FOR I=5 TO 1 STEP -1 !17
0
800 IF CV(I)=PV(I)THEN I=I-1
 !169
810 IF CV(I)>PV(I)THEN V$="C
" :: I=1 ELSE V$="P" :: I=1
!185
820 NEXT I !223
830 IF V$="P" THEN 890 !202
840 V$="C" :: BC=BC+POT :: C
ALL SCORE(BP,BC,0):: DISPLAY
 AT(9,1):TAB(16);"**********
***":TAB(16);"**COMPUTER***"
:TAB(16);"****WINS*****":TAB
(16);"**********" !088
850 CALL SAY("#I WIN#")!045
860 DISPLAY AT(14,1)BEEP:"PR
ESS ANY KEY FOR NEXT DEAL*"
!197
870 CALL KEY(0,K,S):: IF S=0
 THEN 870 !173
880 DISPLAY AT(13,1):: GOTO
130 !045
890 V$="P" :: BP=BP+POT :: C
ALL SCORE(BP,BC,0):: DISPLAY
 AT(13,1):"YOU WIN" :: CALL
SAY("#YOU WIN#")!002
900 DISPLAY AT(14,1)BEEP:"PR
ESS ANY KEY FOR NEXT DEAL*"
!197
910 CALL KEY(0,K,S):: IF S=0
 THEN 910 !213
920 DISPLAY AT(13,1):: GOTO
130 !045
930 BET=10*INT((RND*(B-RB)+1
)/10):: BET=BET-(B-RB)*(BET<
=0)!183
940 BC=BC-(BET+V):: POT=POT+
BET+V :: RB=RB+BET :: DISPLA
Y AT(5,21):USING "BET####.":
RB !116
950 DISPLAY AT(12,1):: DISPL
AY AT(13,1)BEEP:USING "YOUR
BET STANDS AT     #### ":RB-
BET :: DISPLAY AT(14,1):USIN
G "COMPUTER RAISES YOU BY ##
## ":BET !050
960 CALL SCORE(BP,BC,POT)!19
5
970 DISPLAY AT(16,15)BEEP:"Y
OUR OPTIONS" :: DISPLAY AT(1
8,15):"F.FOLD" :: DISPLAY AT
(19,15):"S.SEE ME" :: DISPLA
Y AT(20,15):"R.RAISE ME" !19
0
980 DISPLAY AT(22,15):"? [F.
S.R.]" :: DISPLAY AT(24,15):
"? YOUR CHOICE" !184
990 ACCEPT AT(22,15)VALIDATE
("FSR?")SIZE(-1):A$ :: IF A$
="?" THEN CALL SAY("PLEASE+E
NTER+F S OR+R"):: GOTO 990 !
085
1000 R=-1*(A$="F")-2*(A$="S"
)-3*(A$="R")!131
1010 DISPLAY AT(16,15):: DIS
PLAY AT(18,15):: DISPLAY AT(
```

(See Page 20)

POKER —

(Continued from Page 19)

```
19,15):: DISPLAY AT(20,15)::
 DISPLAY AT(22,15):: DISPLAY
 AT(24,15)!248
1020 ON R GOTO 1030,1040,107
0 !132
1030 BC=BC+POT :: V$="C" ::
CALL CLEARTOP :: GOTO 130 !2
21
1040 A$="********YOU HAVE SE
EN ME" !144
1050 POT=POT+BET :: BP=BP-BE
T !065
1060 GOTO 580 !149
1070 CALL SCORE(BP,BC,POT)!1
95
1080 DISPLAY AT(13,1)BEEP:"C
OMPUTER BET STANDS AT ";STR$
(RB):"HOW MUCH DO YOU RAISE?
 1" !218
1090 ACCEPT AT(14,24)VALIDAT
E(DIGIT)SIZE(-3):V !136
1100 IF V=0 THEN 1090 !080
1110 POT=POT+V+BET :: BP=BP-
V-BET !112
1120 RB=RB+V :: DISPLAY AT(5
,1)SIZE(10):USING " BET####.
":RB :: GOTO 540 !011
1130 !@P+ !062
1140 SUB EXCHANGE(S$(),EX(),
B$(),BINV(),PA(),BINS(),SA()
,NN,EE,ST,F4,F5,FH)!202
1150 IF FH OR ST=5 OR F5=5 T
HEN EE=0 :: SUBEXIT !058
1160 IF F4<4 THEN 1210 !241
1170 REM  OK TRY FOR FLUSH !
060
1180 FOR I=1 TO 5 :: S=SA(I)
:: EX(I)=-(BINS(S)=1):: NEXT
 I !224
1190 GOTO 1250 !053
1200 REM  ALMOST STRAIGHT ?
!143
1210 IF ST<4 THEN 1250 !070
1220 !  YES TRY FOR STRAIGHT
 !096
1230 H=PA(5):: L=PA(1):: FOR
 I=1 TO 5 :: EX(I)=0 :: NEXT
 I :: IF FH OR ST=5 OR F5=5
THEN SUBEXIT !042
1240 IF BINV(H)AND BINV(H-1)
THEN EX(1)=1 ELSE IF BINV(L)
AND BINV(L+1)THEN EX(5)=1 !2
34
1250 EE=0 !065
1260 FOR I=1 TO 5 !060
1270 IF EX(I)THEN S$(I)=B$(N
N):: NN=NN+1 :: EE=EE+1 :: I
F EE>=3 THEN I=5 !226
1280 NEXT I !223
1290 SUBEND !168
1300 SUB ANALYSE(BINV(),PA()
,BINS(),SA(),NP,K2(),K3,K4,F
4,F5,ST,FH,SF,RF,EX())!210
1310 FOR I=1 TO 5 :: K2(I),E
X(I)=0 :: NEXT I !015
1320 K3,K4,F4,F5,NP,FH,SF,ST
,RF=0 :: BINV(PA(1)-1)=1 !20
7
1330 FOR I=1 TO 5 !060
1340 V=PA(I):: Q=BINV(V):: E
X(I)=(Q=1):: S=SA(I):: T=BIN
S(S):: F5=F5-(T=5):: F4=F4-(
T=4)!043
1350 K4=K4-V*(Q=4):: K3=K3-V
*(Q=3):: K2(I)=-V*(Q=2):: NP
=NP-(K2(I)>0):: ST=ST-(BINV(
V-1)*Q=1)!050
1360 NEXT I !223
1370 SF=(F5=5 AND ST=5):: RF
=SF AND PA(5)=14 :: FH=K3 AN
D NP>0 !140
1380 SUBEND !168
1390 SUB DECK(B$(),PIC$(),I$
())!113
1400 GOTO 1420 :: N,A,R,G,X=
0 :: E$="" !213
1410 !@P- !064
1420 CALL CLEAR !209
1430 CALL MAGNIFY(2):: CALL
SPRITE(#1,80,16,10,24,#2,79,
16,26,60,#3,75,16,42,112,#4,
69,16,58,164,#5,82,16,74,216
)!059
1440 DISPLAY AT(18,1)ERASE A
LL:"   ~ ONE MOMENT PLEASE ~
   " :: FOR N=1 TO 30 :: CAL
L CHARPAT(N+96,I$(N)):: NEXT
 N !018
1450 E$="003C42423E02423C002
6692929292976000E04040444443
80038444444544C3E" :: CALL C
HAR(120,E$,138,E$):: E$="" !
003
1460 CALL CHAR(40,"007061420
50A14297E99245AC324A5A5000E8
642A050289452E44241201C0201"
)!066
1470 CALL CHAR(44,"A5A5A524A
57E1800",96,"4A2742820438408
0000000081C1C2A7F81422424242
42424")!165
1480 CALL CHAR(99,"000000367
F7F7F3E7F2A010E0E0100081800E
71818E718181C08107171100081"
)!098
1490 CALL CHAR(103,"1C3E7F7F
3E08030E1818181818E781181C3E
7F3E1C08C0703841475C70000070
")!222
1500 CALL CHAR(107,"7EC35ABD
5A183C421C86E23ADE00000E88A9
AAB4A8906000C35A5A4224181800
",111,"1195552D25090600")!01
6
1510 E$="00182442427E4242003
C4202023C407E003C42021C02423
C00040C1424447E04" :: CALL C
HAR(112,E$,130,E$):: E$="" !
240
1520 E$="007E407C0202423C001
C22203C22221C003E04081020202
0001C22221C22221C" :: CALL C
HAR(116,E$,134,E$):: E$="" !
226
1530 E$="0044485070484442"
: CALL CHAR(124,E$,142,E$)::
 E$="" !205
1540 A=97 :: FOR R=3 TO 7 ::
 PIC$(R)=CHR$(A)&CHR$(A+1)&C
HR$(A+2):: A=A+3 :: NEXT R :
: PIC$(1)="()*" :: PIC$(2)="
+,`" !116
1550 CALL COLOR(2,11,14,9,11
,14,10,11,14,11,2,16,12,2,16
,13,9,16,14,9,16)!255
1560 G=1 :: FOR X=126 TO 129
 :: FOR N=2 TO 14 :: B$(G)=C
HR$(X)&STR$(N):: G=G+1 :: NE
XT N :: NEXT X !054
1570 CALL SAY("#READY TO STA
RT#"):: CALL DELSPRITE(ALL)!
186
1580 !@P+ !062
1590 SUBEND !168
1600 SUB CARDS(S$(),PA(),SA(
),BINV(),BINS())!044
1610 FOR I=2 TO 14 :: BINV(I
)=0 :: NEXT I !237
1620 FOR I=1 TO 4 :: BINS(I)
=0 :: NEXT I !183
1630 FOR I=1 TO 5 !060
1640 PA(I),V=VAL(SEG$(S$(I),
2,2)):: SA(I),S=ASC(S$(I))-1
```

(See Page 21)

POKER —

(Continued from Page 20)

```
25 :: BINV(V)=BINV(V)+1 :: B
INS(S)=BINS(S)+1 !000
1650 NEXT I !223
1660 SUBEND !168
1670 SUB SORT(S$())!006
1680 F=3 :: L=5 !138
1690 IF F>1 THEN F=F-1 :: W$
=S$(F)ELSE W$=S$(L):: S$(L)=
S$(1):: S$(1)=W$ :: L=L-1 ::
 IF L=1 THEN SUBEXIT !229
1700 J=F !078
1710 I=J :: J=J+J :: IF J<L
THEN J=J-(VAL(SEG$(S$(J),2,2
))<VAL(SEG$(S$(J+1),2,2)))!0
98
1720 IF J>L THEN S$(I)=W$ ::
 GOTO 1690 !097
1730 IF VAL(SEG$(W$,2,2))>=V
AL(SEG$(S$(J),2,2))THEN S$(I
)=W$ :: GOTO 1690 ELSE S$(I)
=S$(J):: GOTO 1710 !033
1740 SUBEND !168
1750 SUB DEAL(B$(),P$(),C$()
,NN)!045
1760 DISPLAY AT(20,1)BEEP:"~
* * * SHUFFLING PACK * * *~"
 !059
1770 RANDOMIZE :: FOR I=1 TO
 26 !135
1780 X=INT(RND*52)+1 !216
1790 T$=B$(X):: B$(X)=B$(I):
: B$(I)=T$ :: T$="" !059
1800 NEXT I !223
1810 NN=1 !084
1820 FOR I=1 TO 5 !060
1830 P$(I)=B$(NN):: NN=NN+1
:: C$(I)=B$(NN):: NN=NN+1 !1
37
1840 NEXT I !223
1850 DISPLAY AT(13,1)BEEP: :
"YOUR CARDS": : : : : : :!21
0
1860 SUBEND !168
1870 SUB CLEARTOP !203
1880 DISPLAY AT(5,1): : : :
: : : : : : :!113
1890 SUBEND !168
1900 SUB SCORE(BP,BC,POT)!19
9
1910 DISPLAY AT(3,1):USING "
YOUR       ########COMPUTER
":"POT"&STR$(POT):: DISPLAY
AT(4,1):USING "BANK####.
      BANK####.":BP,BC !131
1920 SUBEND !168
1930 SUB GRAPHICS(S$(),NUMB(
),PIC$(),X)!144
1940 GOTO 1960 :: Z,V,SUIT(1
),PTS,DOT,PTS,R=1 !237
1950 !@P- !064
1960 FOR Z=1 TO 5 :: SUIT(Z)
=ASC(S$(Z)):: PTS=VAL(SEG$(S
$(Z),2,2)):: NUMB(Z)=PTS+129
+18*(SUIT(Z)<128)+13*(PTS=14
)!045
1970 FOR V=1 TO 5 :: CALL VC
HAR(16,X+V-1,143,9):: NEXT V
 !094
1980 CALL HCHAR(16,X,NUMB(Z)
):: CALL HCHAR(24,X+4,NUMB(Z
)):: CALL HCHAR(17,X,SUIT(Z)
):: CALL HCHAR(23,X+4,SUIT(Z
))!186
1990 ON PTS GOSUB 2020,2040,
2070,2090,2110,2130,2150,217
0,2190,2210,2230,2230,2230,2
020 !093
2000 X=X+2 :: NEXT Z :: SUBE
XIT !197
2010 REM ACE !131
2020 CALL HCHAR(20,X+2,SUIT(
Z)):: RETURN !249
2030 REM 2 !236
2040 CALL HCHAR(18,X+2,SUIT(
Z)):: CALL HCHAR(22,X+2,SUIT
(Z)):: RETURN !115
2050 CALL HCHAR(20,X+1,SUIT(
Z)):: CALL HCHAR(20,X+3,SUIT
(Z)):: RETURN !106
2060 REM 3 !237
2070 FOR DOT=1 TO 3 :: CALL
HCHAR(16+2*DOT,X+2,SUIT(Z)):
: NEXT DOT :: RETURN !189
2080 REM 4 !238
2090 FOR DOT=1 TO 2 :: CALL
HCHAR(18,X+2*DOT-1,SUIT(Z)):
: CALL HCHAR(22,X+2*DOT-1,SU
IT(Z)):: NEXT DOT :: RETURN
!151
2100 REM 5 !239
2110 GOSUB 2020 :: GOSUB 209
0 :: RETURN !073
2120 REM 6 !240
2130 GOSUB 2090 :: GOSUB 205
0 :: RETURN !104
2140 REM 7 !241
2150 GOSUB 2130 :: GOSUB 205
0 :: RETURN !144
2160 REM 8 !242
2170 FOR DOT=1 TO 4 :: CALL
HCHAR(15+2*DOT,X+1,SUIT(Z)):
: CALL HCHAR(15+2*DOT,X+3,SU
IT(Z)):: NEXT DOT :: RETURN
!152
2180 REM 9 !243
2190 GOSUB 2170 :: GOSUB 202
0 :: RETURN !153
2200 REM 10 !027
2210 GOSUB 2170 :: GOSUB 204
0 :: RETURN !173
2220 REM PICTURE !214
2230 FOR R=1 TO 7 :: DISPLAY
 AT(16+R,X-1)SIZE(3):PIC$(R)
:: NEXT R :: RETURN !0152240
 !@P+ !062
2250 SUBEND !168
2260 SUB POKER :: N,S=K !206
2270 !@P- !064
2280 CALL COLOR(3,14,16,4,14
,16,13,9,1,14,9,1):: FOR N=5
 TO 8 :: CALL COLOR(N,16,1):
: NEXT N :: CALL SCREEN(5)!0
20
2290 CALL CHAR(126,"081C1C2A
7F6B0808081C3E7F7F6B081C0008
1C3E7F3E1C0800367F7F7F3E1C08
",143,RPT$("0",16),59,"001C2
2207C20227E")!019
2300 CALL COLOR(9,2,1,10,2,1
,11,2,1,12,2,1):: N=49 !038
2310 DISPLAY AT(1,3)ERASE AL
L BEEP:"WINNING ORDER OF POK
ER HANDS":RPT$("-",28)!130
2320 DISPLAY AT(5,1):"1)ROYA
L FLUSH- A,K,Q,J,10":,"all s
ame suit" !103
2330 DISPLAY AT(7,1):"2)STRA
IGHT FLUSH- run of 5":,"all
same suit":"3)FOUR CARDS of
same value": :"4)FULL HOUSE-
 1 PAIR and 3":,"of a kind"
!167
2340 DISPLAY AT(13,1):"5)FLU
SH- 5 cards, same suit" !215
2350 DISPLAY AT(15,1):"6)STR
AIGHT- run of 5 cards":,"of
any suit": :"7)THREE CARDS o
f same value" !085
2360 DISPLAY AT(19,1):"8)TWO
 PAIRS.":"9)ONE PAIR.":"10AC
E HIGH downwards." !247
2370 DISPLAY AT(24,1):"   PR
ESS ANY KEY TO PLAY ~ " !041
```

(See Page 22)

BUGS & BYTES

SCSI update

Here's another update on the state of the SCSI project. The information is from November 1993, and Bud Mills has indicated that the SCSI will be shown at Fest West '94 Feb. 19-20 in Tucson, Arizona. He says the TI version should be ready at that point.

Brad Snyder released "SCUZZY," a formatter and sector editor for the Western Horizon Technology SCSI interface that can be used on an SCSI hard drive. The program allows users to format and test hard drives. It also polls and identifies other devices on the SCSI bus. SCSI devices are daisy chained. No EPROM is required on the SCSI card. This is freeware to SCSI owners.

Bud Mills is redesigning the SCSI cable to allow connection to Macintosh-compatible external SCSI devices, including CD-ROM drives. Using the SCSI, you can play audio CDs on a CD-ROM drive using the headphone jack on the ROM drive. It works independently of the computer, so you can listen to music on the headphones while using your computer. No EPROM is required for this capability either.

Mills says programmer Mike Maksimik has the Geneve talking to floppies and the hard drive, and that Geneve users will be able to use the SCSI as soon as Beery Miller releases the latest MDOS which contains SCSI patches.

He also says Jim Schroeder has written routines which will allow Geneve to address continuous RAMdisk up to 3.5 meg.

Mills notes that early purchasers of SCSI cards still require a hardware modification to be fully functional, and need to contact him.

"We're ooking forward to being at Fest-West and wowing everybody's socks off," Mills says.

TI programs on CD-ROM?

Fred Moore, a member of the Los Angeles 99ers user group, is looking for some help in collecting enough TI programs to produce a CD-ROM. Writing in the group's newsletter, he says, "I'm convinced that all TI99/4A programs from all of the TI user groups can be put on one CD-ROM, and for about $50.

"Let me tell how. I have put all of the LA user's group library, over 1,000 disks, on a tape that took only 40 megabytes. Remember, one CD-ROM will hold more than 600 megabytes. That's about 20,000 TI disks. The 1,000 disks from our club is too small to put on a CD. What I need are the TI program disks from other user groups.

"Let me tell you how this is done:

1. All of the TI disks must have a number assigned to them.

2. An index or category disk must be made telling what each TI disk contains.

3. Using Barry Boone's Archiver program, all disks were archived using the number assigned to each.

4. Using PC-Transfer, each TI archived disk was transferred to an IBM formatted disk. This can also be done by transferring the files using modems or a connecting cable between the two computers.

5. Once in a PC, the files can be saved to a data tape cartridge, which can hold upwards of 200 megabytes.

6. These tapes can be uploaded to a CD-ROM for about $50.00

7. To get a program from the CD where all of the disks are displayed by a number, tab the number and save it to an IBM floppy. Transfer this file back to the TI format. Then extract the files so you can use them.

8. This may sound like a lot of work but when you have a system set up it takes about four minutes to download a program from a CD to a TI disk.

"Just think about it? About 20,000 TI disks at your fingertips.

"To do this, I need the TI disks from other user groups. If you are interested, send me your archived disks (I will return if you wish) and I will do the rest.

Readers may contact Moore at 7730 Emerson Ave., Los Angeles, CA 90045; 310-670-4293.

Readers might also keep in mind that when the Horizon SCSI card comes out, users will be able to connect CD-ROM drives TI or Geneve directly.

Watch out for static

Don Walden of Cecure Electronics warns users that in winter when static electricity is high, putting your hand on the ground when touching your computer cards is often not enough to keep from zapping them. He notes that sometimes you don't even have to touch the cards under high-static conditions to zap them — just being near them is enough. Use a wrist grounding strap when doing things with cards, he recommends.

POKER—

(Continued from Page 21)

```
2380 CALL KEY(3,K,S):: IF S=
0 THEN 2380 !156
2390 !@P+ !062
2400 SUBEND !168
```

Keep track of TI fairs around the world in 1994

(See TI fair listings on page 12)

Sliding Block Puzzles Series II

Amusement for the puzzled

By JOHN KOLOEN

Sliding Block Puzzles Series II, written by Norman Rokke, is one of those deceptively simple games that, once you start playing it, is difficult to stop. Oh, you can walk away from it for awhile, but it's hard to just stop.

No one knows the origin of slide block puzzles, but I've played several versions on various computer systems. And no matter how many bells and whistles programmers try to add, the basic puzzle itself is intriguing and universal.

Performance: Sliding Block Puzzles Series II runs out of Extended BASIC. And runs well.

The object of the puzzle is to move a certain block from one position in the block to another position, say from the left upper corner to the right corner. Pieces within the block are various shaped and of different colors. For example, in the first puzzle, the sliding blocks range from a large square block that takes up about 20 percent of the puzzle area to two small blocks that are one-fourth the size of the large block. Remaining blocks are rectangular, two of them set horizontally in the puzzle area and the remaining four set vertically.

Each of the sliding blocks is a different color and you designate which one to move by first selecting its color and then selecting a direction in which to move it. Sliding blocks may be moved in any of eight directions using the WERSDZXC. Unlike similar puzzles I've played, this version lets you move the blocks horizontally.

After selecting the block's color and direction, the movement occurs immediately, with the block appearing in its newly designated location. This goes on until you've finally solved the puzzle, which I think you'll find much easier said than done. The first puzzle is supposed to be solved within 45 moves, but don't be surprised if you use several times that many the first time you play.

Each of the colored blocks is displayed along the top of the screen to make selection easy. Directional keys are likewise displayed at the bottom of the screen for easy reference.

Another feature allows taking back a move prior to executing, which is very useful in a puzzle. Suppose you select the red block to move and then change your mind. Simply press the "R" key and you'll be able to redo the move.

Other nice features include the ability to save your progress to a disk and and a function called Object that is used as a reminder of what the object is of the puzzle you are playing. This comes in handy if you happen to forget whether the object is to move the large red block to the top right or bottom right of the puzzle. By selecting "O" the target block will move to the position it is supposed to end up in, and then return to the position from which it came. Unfortunately, I was not able to get the "O" command to work, though I never actually needed it. The puzzles and objects are also described in the manual.

Quitting is a matter of pressing the "Q" key. You then have the option of resuming, restarting, returning to the main menu or exiting the program. The main menu is used to select one of the three puzzles. Puzzle No. 1 consists of nine blocks and has two objectives, so it's as if you had two puzzles to solve. Puzzle No. 2 consists of ten blocks and has one objective. Puzzle No. 3 consists of 11 blocks and has two objectives.

Ease of Use: Sliding Block Puzzles is easy to use. This is due to the fact that the directional keys are permanently listed on the game screen so that you don't have to memorize them or refer to the manual. Solving the puzzles is another question. I'm still working on them. And I probably will for quite some time. (You might consider ordered Sliding Block Puzzles Series I for starters.)

Documentation: The 12-page booklet-size manual briefly discusses sliding block puzzles and then outlines every function of the program, as well as thoroughly describing the various puzzles. Included in the descriptions are the minimum number of moves that the each puzzle can be solved in. It's something I don't need to be reminded of. There are also on-screen instructions that may be selected from the main menu, providing information about the puzzle and other information.

Value: My general guideline in determining the value of my entertainment dollar is to compare how much amusement I get for my dollar. Sliding Block Puzzles scores highly because it is not only inexpensive to buy but lasts for hours and hours, which reduces the per hour cost of entertainment to a pittance. For the price of Sliding Block Puzzles I could rent three videos and be entertained for upwards of 5-6 hours. Sliding Block Puzzles promises much more than 5-6 hours of entertainment, making it a very good value.

Final Grade: Aside from not being able to get the Object command to work — which I don't consider necessary — I'd like to have seen a move counter that tabulated the total number of moves as they occurred. As it is, the move total may be recorded at the end of the puzzle, but since I haven't finished one yet I can't say for sure. Otherwise, Sliding Block Puzzles are well worth the price for those who enjoy being puzzled.

REVIEW

REPORT CARD

Performance	**A**
Ease of Use	**A**
Documentation	**A**
Value	**A**
Final Grade	**A**

Cost: $10.00
Publisher: MS Express Software, P.O. Box 498, Richmond, OH 43944
Requirements: TI-99/4A console, color monitor or TV, disk drive system, memory expansion and Extended BASIC

MICRO-REVIEWS

Japanese Writing System, TI Artist Fonts, Adventure Data Base and Genealogy Plus

By CHARLES GOOD

JAPANESE WRITING SYSTEM and TI ARTIST FONTS by Don Shorock.

Don Shorock, known in the TI community for his spoken Japanese and other language teaching software, is releasing some new fairware products at Fest West '94 for use in conjunction with TI Artist. I hope those attending Fest West will see this review in advance of the show and know what to expect.

The most amazing of these products is his Japanese Writing System. As you may know, Japanese is not written in "left to right, top to bottom of page" and is not written using standard keyboard letters. Don has created a way of creating "what you see is what you get" Japanese text for on screen display and printing. Jim Peterson (Mister Tigercub) agreed to Microreview Don's Japanese Writing System for me. Jim is fluent in Japanese. His cover letter to me that accompanied his review says, "This is really quite an achievement. I just can't imagine going to the work of creating 881 'kanji' instances, but Don doesn't think it will be such a job. He seems to have become a master of TI-Artist and Page Pro." Below is Jim Peterson's review:

"Many centuries ago, the Chinese developed a system of writing using a different stylized picture, or ideograph, for each word in their language. Since China was the center of Asian culture, this impractical system was later adopted in the ancient kingdoms of Annam (now Vietnam), Korea and Japan. In modern times the Vietnamese and Koreans have abandoned this system and use their own phonetic alphabets, and even the Chinese have simplified many of the more complex ideographs. The Japanese, however, stubbornly cling to their version of this impractical system.

"Since Japanese, unlike Chinese, is a conjugated language, it was necessary to develop a supplemental phonetic alphabet for verb and adjective endings. For some reason, the Japanese developed not one but two such alphabets. Japanese newspapers are printed with thousands of different 'kanji' ideographs combined with characters of the 'hiragana' alphabet for conjugation, characters of the 'katakana' alphabet to spell foreign words which have come into use in Japanese, and sometimes the Roman alphabet for foreign names.

"Japanese typewriters are therefore very complex and extremely slow, and typesetting is very difficult. I have no idea how they have managed to adapt their writing system to the computer age, but it has certainly not seemed possible to do it with the TI99/4A. Even their phonetic alphabets cannot be properly downloaded as printer fonts because they have optional diacritical marks which will not fit within the standard dot matrix.

"Don Shorock had previously included Japanese in his series of foreign language tutorial disks, and had managed to display the Japanese phonetic characters on screen. He even used the Speech Synthesizer to actually speak the language with a fairly high degree of accuracy. He has now developed a remarkable method of using TI-Artist to create text which can be output to a printer.

"Requirements are the TI99/4A console with Extended Basic, 32k memory expansion, at least one SS/SD disk drive, RS232 card and 9-pin dot matrix printer, and TI-Artist (TIA-Plus preferred). To fully use the system, Page Pro is also required.

"Don's Extended BASIC program enables you to lay out text in blocks of up to 12x14 or 14x12 characters, although a maximum of 12x12 is recommended. After these blocks have been converted to Japanese by TI-Artist and saved as pictures or instances, Page Pro can be used to combine eight or more blocks into a full page of text.

"It is not even necessary to memorize the Japanese alphabets in order to use this program, because text is entered in normal keyboard letters. Japanese is a language built on syllables rather than individual sounds, and each syllable is represented by a phonetic character. For instance, to print the name of that famous mountain you would enter 'fu' and 'ji' and 'ya' and 'ma.' As these syllables are entered, they appear within a hollow cursor which automatically advances to the next space; it can also be moved with the arrow keys. Text will normally be entered for conversion to the 'hiragana' alphabet but can optionally be wholly or partly in 'katakana.' A screen prompt, as well as the size of the letters, informs you which mode you are in. Standard Japanese punctuation and special characters can also be entered, and blank spaces can be left for 'kanji' ideographs to be added later.

"When this text has been completed, the "print" command will output to the printer the codes which must next be typed manually into TI-Artist. The printout gives two code layouts, one for the usual vertical left to right Japanese printing and one for standard horizontal printing. Two layouts are also given for any 'katakana' text, and instructions are provided.

"TI-Artist is then loaded, the 'hiragana' font file is loaded into it, and the coding is carefully typed in. The 'what you see is what you get' Japanese text can then be viewed on screen.

"If the text was entirely in 'hiragana,' it is now ready to be saved and/or printed. However, if 'katakana' characters were also used, the 'katakana' font file must be loaded in, and the 'katakana' part of the code must be typed in. This will overlay the previously created text, making a complete block.

"The test disk that Don sent me contained 50 of the 'kanji' ideographs, saved as instances. He plans to create all 881 of the 'essential' ideographs. These can be loaded and moved into the blank spaces left in the text.

"All this is really far easier than it may sound. Although I am totally unfamiliar

(See Page 25)

MICRO-REVIEWS —

(Continued from Page 24)

with TI-Artist, I was able to create a block of mixed 'hiragana' and 'katakana,' convert and overlay them, and print out the result with little trouble. Students of Japanese will probably prefer to do most of their printing in 'hiragana' alone, and this will be even easier."

Don is also releasing some really cute TI Artist fonts. These are regular fonts all with normal uppercase keyboard letters and characters and some with lowercase letters as well. My favorite are the fonts called HEARTS_F, LINESHOW_F, and JUSTDOTS_F. These three consists of large block letters (uppercase only) each of which has a shadow offset a bit to the right and down from the actual letter. HEARTS_F letters have little hearts within the letters and would be suitable for creating valentines or writing notes to spouses and other loved ones. LINESHOW_F and JUSTDOTS_F have dot patterns in the letters and are perhaps more generally useful. There is also a font (IMAGINE_F) that produces normal sized shadowed letters, both upper and lowercase.

Some of the other fonts in Don's package include olde English calligraphy (OLDENGL_F), handwritten "cursive" calligraphy that allows letters to appear to touch each other (WRITEMED_F and WRITETHN_F), a reduced size version of the RODEO_F font that comes with TI Artist. They all seem quite creative.

All these products require TI Artist and can be run from SSSD disks. Don is asking a fairware donation of $10-$20 for the Japanese Writing System and an unspecified fairware donation for the fonts. For more information phone after 6 p.m. central time most evenings or write: Don Shorock, P.O. Box 501, Great Bend KS 67530. Phone is (316) 792-3097.

GENEALOGY PLUS!
By Mickey Cendrowski

Till now, if you wanted to keep genealogy records using the TI you either had to use dedicated software such as the public domain "Genealogy Workshop" by Les and Cindy Cattin, or you had to use a generalized commercial or fairware data base program such as "PR Base" or "TI Base". The problem with dedicated software is that it is often inflexible. If you don't like the kind of data it accepts and the format of its printed reports you are usually out of luck. The problem with generalized data base software is that although it is very flexible, it is difficult for some to use. You have to create your own data base outline, deciding what data to store and in what form and deciding how exactly to format printed reports. Ordinary mere mortal 99/4A users often feel uncomfortable trying to do this. Genealogy Plus! deals effectively with both these problems. It is both flexible and easy to use.

The software package includes a TI BASE template, a Personal Record Keeping module template, and a bunch of TI-Writer templates. The TIB and PRK templates allow you to enter data for each individual under the categories of first and last name, date and location of birth, chart number, lineage number, forms, and documents. These data can then be sorted, displayed on screen, and printed according to the capabilities of TIB and PRK. For both types of data bases, but particularly for TIB, excellent documentation shows you step by step how to load the data base, then how to load the Genealogy Plus! template into the data base, and then how to enter and display data.

I find the blank TI-Writer templates particularly handy and well designed. Each is the size of a single 8.5x11 inch page. You load a template into any TI-Writer type of text editor, turn off word wrap, and type your data onto the appropriate sections of the page, using the text editor to modify the data or sheet format as needed easily. You can then save these pages back to disk and print them. Use a hole punch on the printed pages and insert them into a three-ring binder. I think genealogical data is best displayed and appreciated by you and others as hard copy, such as some of the pages in the large 200-year-old family Bible I have in my living room.

The following TI-Writer templates are available. Many are of the "one for each individual" type: Personal Biography Sheet; Research Calendar (used for library research — date, call number, source, period of time covered by source, results, doc that refers to source); Correspondence Record (dates mailed and received, address, purpose, results); Research Extract (notes taken from library documents, date extract taken, call number, source, description and condition of source, indexed by, purpose of search, period of time searched, document number); Family Group Sheet; Lineage Chart (a family tree); Additional Marriage Sheet (for multiple marriages); Source Checklist (very useful and comprehensive); Sibling Lineage Chart; Extended Family Sheet (for adopted children).

Although the software documentation is not a tutorial on genealogy record keeping, a complete set of sample data is included on a separate disk. These data are entered onto the TI-Writer templates to show how these templates are supposed to be used.

I know of no better genealogy software for the 99/4A. You need an SSSD disk system and TI-Writer or one of its clones for text editing. A printer and either TI BASE or the Personal Record Keeping cartridge are recommended but not required. Genealogy Plus! costs $19.95 + $1 postage from MS Express Software, P.O. Box 498, Richmond OH 43944.

ADVENTURE DATA BASE
by Mickey Cendrowski

Probably nobody knows more about adventure games for the 99/4A than Mickey. In fact she wrote the book on the subject, "The Adventure Reference Guide", published by Asgard a few years ago. Her current offering is a TI BASE data base that includes all the information from her book plus additional information gleaned in recent years. First you load TI BASE, and then you load in Mickey's Adventure Data Base from a SSSD disk. Almost 250 adventures are covered. These include public domain and fairware adventures, those published from commercial sources such as TI Infocom and MS Express, and type in adventure programs published in magazines and books. With the data base you can, for example, produce lists of all the Adventure Module adventures, all the Infocom adventures, or all the adventures that run out of the XB module. Most users

(See Page 26)

MICRO-REVIEWS —

(Contined from Page 25)

of this data base will begin, as I did, by printing out the entire data base. This yields eight printed pages of condensed print. Since this is a TI BASE template, you have all the features of a good database available to you. You can, for example, add your own comments to the "comment" field that is part of each adventure's listing. You can also delete entries or expand the database by adding information about additional adventures.

For each adventure title the database provides the following information: Official TI software (Y/N); copyrighted (Y/N); fairware or public domain (Y/N); single or multiple file program; total disk sectors; source or publisher; module needed to run game; is software a role playing game an all text game or an adventure utility; is speech used; graphics (Y/N); runs from cassette or disk or either; level of difficulty (3 levels). Also included is Mickey's overall quality rating of 1-5. A difficult game is not necessarily a high quality game. If you disagree with Mickey's quality rating you can substitute your own complete with your own personal comments.

Adventure Data Base is available for $19.95 plus $1 shipping from MS Express, P.O. Box 498, Richmond Ohio 43944. TI BASE is available for about $25 from most TI dealers including TM Direct Marketing, 800-336-9966.

Send your software for me to MicroReview and your requests for my various $1 software mailings to Charles Good, P.O. Box 647, Venedocia, OH 45894.

NEWSBYTES

Fest West to host vendors, user groups

A number of vendors and users groups have expressed interest in attending Fest West '94 Feb. 19-20 in Tucson, Arizona, according to Tom Wills of the SouthWest Ninety-Niners, the host group for the fair.

Those expressing interest include Competition Computer of Milwaukee, Wisconsin; the Chicago Users Group; the Salt Lake and Ogden users groups of Utah, hosts of last year's Fest West; the Southern California User Group; programmers Don and Tom Shorock; Don O'Neil of Western Horizon Techonogies; Ken Gilliland of Notung Software; Bud Mills of Bud Mills Services; and the VAST Users Group.

For further information about Fest West, write the SouthWest Ninety-Niners, P.O. Box 17831, Tucson, AZ 85731; or call Wills at (602) 886-2460, BJ Mathis at (602) 747-5046 or the Cactus Patch BBS at (602) 290-6277.

4A programs stored on CD-ROM disk

Fred Moore of the LA Users Group has undertaken a project to place TI99/4A programs from all the TI users groups on CD-ROM.

He believes that programs from all the groups will fit on one CD disk, which holds about 800 megabytes, or approximately 20,000 disks. Moore says he has placed the entire LA Users Group library of more than 1,000 disks on a tape, which took 44 megabytes.

For the procedure, each disk must have a number assigned to it and a category disk must be made telling what each disk contains. TI disks are archived using Barry Boone's Archiver program and each TI archived disk transferred to an IBM formatted disk using PC Transfer or another converting program. Once in an IBM computer, files can be saved to a disk or tape cartridge. A tape cartridge can hold about 200 megabytes. These tapes can be uploaded to a CD disk for about $50, according to Moore.

To get a TI disk from the CD disk, the user needs to tag the disk number and save to an IBM floppy. Transferring back to the TI format and extracting the files to be used takes about four minutes, Moore says.

Moore is asking for TI disks from other users groups, in archived format if possible. For information, write Moore at the LA 99ers User Group, 7730 Emerson Ave., Los Angeles, CA 90045-1117.

PUNN changes BBS number, sysop

The Portland (Oregon) Users of Ninety-Nines have a new bulletin board phone number, (503) 232-5954. Ron Mayer is the new sysop for the board, which operates 24 hours at 300, 1200 and 2400 baud, E-0-1.

Paris group disbands

The Paris 99/4A User Group in Paris, Texas, has disbanded as of December 1993 and published the final issue of its monthly newsletter.

Club president Jerry Keisler says that quarterly meetings will be held on an informal basis. The users group formed in 1986.

Getting on First Floor

Two digits were reversed for the listing of Brad Snyder's BBS "The First Floor" in the November 1993 MICROpendium, and the area code has changed.

Correct number for the BBS is (610) 760-0527.

9T9 User Group gets new mailing address

New address for the 9T9 User Group of Toronto is c/o Neil Allen, 52 Graystone Gardens, Islington, Ontario, Canada M8Z 3C4.

Send information about your products and services for the TI/Geneve community to MICROpendium Newsbytes, P.O. Box 1343, Round Rock, TX 78680.

USER NOTES

TML Windows

Jim Lesher continues his series for use with The Missing Link. He writes:

For those who may have missed the first in the window series, let's reiterate.

As we mentioned before, the circles are the easiest to make. The windows are just a bit more interesting. It helps to have graph paper laid out to represent your screen, which is 193 dots (or pixels) high and 241 wide. Graph paper with 1/4-inch squares with each square representing six dots works very well for me.

Most of us who are not mathematically or geometrically inclined will look at the sample on page 7 of the TML manual and say, *Huh?* The first time I typed that line in 10 CALL LINK("WINDOW"[,row1,column1,row2,column2,1]), I did everything wrong. In the first place, the [is not to be typed in, only the ( )s, or open and close parentheses. Also, somehow I missed the ,1 at the end, which resulted in a blank screen.

Now, thinking of your screen as a grid of 193x241 dots = 46,513 dots, let's begin to try to make this as simple as possible. To make a window takes only two dots on your graph or screen and the program will do the rest. So we want to make a window in the upper left-hand corner, say 20 dots high and 40 dots wide. To make it easy, let's use 1 for the row and 1 for the column, (abbreviated COL). OK, so that is our first dot, that's simple enough. Now the second dot. Row 2 will determine the height, so we punch in 20 and for Col2 we punch in 40 and that's all there is to it. So your program line will look like this:

10 CALL LINK ("WINDOW",1,1,20,40

Then run it, but remember, you must first load "The Missing Link" program. Also, add a second line 20 GOTO 20. This will keep the image on the screen for your viewing pleasure. And if you wish to send it to your printer, press FCTN and CTRL at the same time.

This time we will do two things to the window. In the first program we are going fill the window with a violet color and the window will be drawn in white. When putting anything in a window, the window becomes the frame of reference. In other words, row 1 becomes the upper left-hand corner of the window and not the whole screen. As you will see when the window comes up on your screen, there will be a three-dot margin all around the filled part relative to the frame and it all works out perfectly, mathematically. Since we want to have a three-dot border, our first point will be three rows down and three rows over; to find the next point of reference, look at 60 in line 110, subtract 10 from it, which is 50, minus 3 = 47, our next number. Now for the last number, 177 is derived from subtracting 20 from 200 = 180 and the width of our border is 3 minus 180 = 177.

```
100 CALL LINK("COLOR",16,14)
110 CALL LINK("WINDOW",10,20
,60,200,1)
120 CALL LINK("FILL",3,3,47,
177)
130 GOTO 130
```

In this program we are going to print something in the window, keeping in mind the rule about windows becoming the reference for entering anything within its frame. To center the text within the window, we should know the size of the text. The default text uses a 5x8 dot matrix. The actual character is four dots wide with one column of dots as the spacing between the characters, so when counting characters count them as 5x6. Also, when figuring the centering, the row number you enter will not be the center of the character, but the top of the character. For example, if you had a window 50 dots high as in the problem below and want to center it vertically, half of 50 is 25, but the character is six dots high, so half of that is 3 from 25 = 22, as in the example below. If all this is confusing, remember, you are not alone.

```
100 CALL LINK("COLOR", 16,14)
110 CALL LINK("WINDOW",10,20
,60,200,1)
120 CALL LINK("PRINT",22,5,"
**HELLO YA'ALL, FROM TEXAS**
')
130 GOTO 130
```

If you would like to have a sheet of graph paper with the squares numbered (then, of course, you can make as many copies you want), send $2 to cover the shipping and handling to Jim Lesher, 722 Huntley, Dallas, TX 75214.

Watch those letters

This tip comes from James F. Murta of Glendale, California. He writes:

When typing a DATA program, if your data file has a subfile, the first character of the filename cannot be the same as the first character of the subfile name of the same file.

Example:

JCAD: ATI

JCAD: JTP — wrong

JCAD: QTZ

If it is, the data from the preceding file will move to the file in trouble.

Let Telco do the calling

This tip comes from Glenn Bernasek of TI-CHIPS, Cleveland Ohio, and was published in the newsletter of the Cleveland Area TI99/4A Users Groups:

It goes without saying that Telco is one super piece of software for the TI99/4A. This program enables the TIer to communicate with the outside world. And not only can we talk to fellow TIers, but we are able, thanks to TELCO, to communicate with any and all systems with the greatest of ease!

Recently, I decided to have Telco perform another service for me. This time I use Telco to call relatives, friends and businesses for me. This allows me to dial frequently used phone numbers that I can't remember and don't want to look up every time.

The method is simple. I just load Telco and choose (A)uto dialer from the main menu. I then press M and select the phone line number I want to add or edit. I type in the name and phone number, and leave the modem defaults alone. (Remember to press FCTN-9 to return to the Main Menu and choose (S)etup options and to save (C)hanges.)

From then on, all I have to do is load Telco, choose (A)uto dialer, *pick up the phone* and enter the phone line number of the person I want to call. All I have to remember is to press FCTN-4 when the *first* ring is sounded. This will isolate me

(See Page 28)

USER NOTES

(Continued from Page 27)

from Telco and the modem, and I will be able to continue my phone call as I usually would. The only difference between manually dialing the phone and letting Telco do the calling is that I don't have to look up phone numbers anymore!

(You might want to stick "1170," in front of the phone numbers that you don't want "Call Waiting" to interrupt.)

Hexadecimal converter

The following program comes from W.L. Shepard, of Liverpool, New York. He writes:

This program is one that I have found to be very useful when dealing with numbers in assembly programming. It will convert hex numbers to decimal, two's complement, and binary code. It will convert both positive and negative decimal numbers to their hex and binary code. In addition, it will add and subtract hex numbers. The program uses several formulae that have been used before and some new ones, as well as modifications of some of the old formulae.

It is menu driven and easy to use.

```
100 !HEXADECIMAL CONVERTER !
120
110 !!131
120 !   W.L. Shepard !201
130 !!131
140 GOTO 150:;A$,B$,BIN$,HEX
$,S$ :: A,B,D,D2,DEC,DEC2,DE
C3,DEC4,DEC5,FLG,I,K,M,N,S,Y
,Z :: CALL CLEAR :: CALL COL
OR :: CALL KEY :: CALL SCREE
N :: DIM C$(20):: DIM H$(10)
:: !@P- !069
150 FLG=0 :: CALL CLEAR :: C
ALL SCREEN(5):: FOR I=1 TO 8
 :: CALL COLOR(I,16,5):: NEX
T I !211
160 DISPLAY AT(1,3)BEEP:"HEX
IDECIMAL CONVERSIONS" :: DIS
PLAY AT(4,6):"1.HEX TO DEC":
 :"     2.DEC TO HEX": :"
  3.ADDITION OF HEX#" !178
170 DISPLAY AT(10,6):"4.SUBT
RACTION OF HEX#" :: DISPLAY
AT(24,1)SIZE(11):"SELECT ONE
." !083
180 CALL KEY(0,K,S):: IF K=4
9 THEN 190 ELSE IF K=50 THEN
 450 ELSE IF K=51 THEN 660 E
LSE IF K=52 THEN 730 ELSE 18
0 !144
190 DISPLAY AT(24,12):"NO "&
CHR$(K):: DISPLAY AT(12,1):"
HEX NUMBER.   >" !017
200 ACCEPT AT(12,15)BEEP SIZ
E(4)VALIDATE(DIGIT,"ABCDEF")
:HEX$ !123
210 GOSUB 220 :: GOSUB 330 :
: GOTO 420 !191
220 FOR N=1 TO LEN(HEX$)!142
230 A=ASC(SEG$(HEX$,N,1))!06
8
240 IF A<58 THEN 260 !057
250 A=A-55 :: GOTO 270 !022
260 A=A-48 !056
270 DEC=DEC*16+A :: NEXT N !
178
280 IF FLG=1 THEN 310 !197
290 DISPLAY AT(14,1):"DECIMA
L";DEC !205
300 IF DEC>32767 THEN 630 !2
15
310 RETURN !136
320 DEC=0 :: A=0 :: Y=Y+1 ::
 GOTO 190 !057
330 !!131
340 BIN$="" !130
350 B=(DEC/2-INT(DEC/2))*2 !
063
360 DEC=INT(DEC/2)!081
370 BIN$=STR$(B)&BIN$ !250
380 IF DEC>0 THEN 350 !225
390 A$="" :: FOR I=1 TO 16-L
EN(BIN$):: A$="0"&A$ :: NEXT
 I !247
400 B$=A$&BIN$ :: FOR I=1 TO
 16 STEP 4 :: C$(I)=SEG$(B$,
I,4):: NEXT I !243
410 DISPLAY AT(18,1):"BINARY
" :: Z=1 :: FOR I=1 TO 16 ST
EP 4 :: DISPLAY AT(20,Z):C$(
I):: Z=Z+5 :: NEXT I !250
420 DISPLAY AT(24,1)BEEP:"AN
OTHER?(Y/N)" !202
430 CALL KEY(0,K,S):: IF K=8
9 THEN 440 ELSE IF K=78 THEN
 640 ELSE 430 !114
440 FLG=0 :: D=0 :: BIN$=""
:: H$(1)="" :: DEC=0 :: CALL
 CLEAR :: GOTO 160 !180
450 DISPLAY AT(24,12):CHR$(K
):: DISPLAY AT(12,1):"DECIMA
L NUMBER?" :: ACCEPT AT(12,1
8)BEEP SIZE(6)VALIDATE(DIGIT
,"-"):D !031
460 D2=D :: IF D>65535 THEN
450 ELSE IF D<0 THEN 650 !15
3
470 GOSUB 480 :: GOSUB 330 :
: GOTO 420 !196
480 D2=D :: IF D<16 THEN 490
 ELSE IF D<256 THEN 500 ELSE
 IF D<=4095 THEN 510 ELSE 52
0 !036
490 S$=">000" :: GOTO 530 !1
79
500 S$=">00" :: GOTO 530 !13
0
510 S$=">0" :: GOTO 530 !081
520 S$=">" !059
530 M=D/16-INT(D/16)!123
540 D=INT(D/16)!119
550 IF M>.5625 THEN 560 ELSE
 570 !143
560 H$(1)=CHR$(M*16+55)&H$(1
):: GOTO 580 !172
570 H$(1)=CHR$(M*16+48)&H$(1
)!151
580 IF D>.6254 THEN 530 !225
590 H$(1)=S$&H$(1)!147
600 IF FLG=1 THEN 620 !252
610 DISPLAY AT(14,1):"HEX NO
." :: DISPLAY AT(14,8):H$(1)
!026
620 DEC=D2 :: RETURN !010
630 DEC2=DEC-65536 :: DISPLA
Y AT(16,1):"TWO'S COMPLEMENT
 NO.";DEC2 :: GOTO 310 !060
640 END !139
650 D=D+65536 :: GOTO 470 !1
33
660 FLG=1 :: CALL CLEAR :: D
ISPLAY AT(1,5):"HEX ADDITION
." :: DISPLAY AT(5,1):"1ST H
EX#     >" :: ACCEPT AT(5,1
6)BEEP SIZE(4)VALIDATE(DIGIT
,"ABCDEF"):HEX$ !156
670 GOSUB 220 :: DEC3=DEC !0
55
680 DEC=0 :: DISPLAY AT(7,1)
:"2ND HEX#     >" :: ACCEPT
 AT(7,16)BEEP SIZE(4)VALIDAT
E(DIGIT,"ABCDEF"):HEX$ !247
690 GOSUB 220 :: DEC4=DEC ::
```

(See Page 29)

USER NOTES

(Continued from Page 28)

```
 DEC5=DEC3+DEC4 !057
700 IF DEC5>65535 THEN 720 :
 : D=DEC5 :: GOSUB 480 :: DIS
PLAY AT(9,1):"HEX#1+HEX#2="
:: DISPLAY AT(9,15):H$(1)!03
0
710 GOTO 420 !244
720 DISPLAY AT(22,1):"NUMBER
 TOO LARGE!TRY AGAIN." :: FO
R I=1 TO 300 :: NEXT I :: DE
C5=0 :: DEC=0 :: GOTO 660 !2
02
730 FLG=1 :: CALL CLEAR :: D
ISPLAY AT(1,5):"HEX SUBTRACT
ION." :: DISPLAY AT(5,1):"1S
T HEX#       >" !210
740 ACCEPT AT(5,16)BEEP SIZE
(4)VALIDATE(DIGIT,"ABCDEF"):
HEX$ :: GOSUB 220 :: DEC3=DE
C :: DEC=0 :: DISPLAY AT(7,1
):"2ND HEX#      >" !174
750 ACCEPT AT(7,16)BEEP SIZE
(4)VALIDATE(DIGIT,"ABCDEF"):
EX$ :: GOSUB 220 !253
760 DEC4=DEC :: IF DEC4>DEC3
 THEN 790 :: DEC5=DEC3-DEC4
:: D=DEC5 :: GOSUB 480 !027
770 !@P+ !062
780 DISPLAY AT(9,1):"HEX+1-H
EX#2=  >" :: DISPLAY AT(9,15
):H$(1):: GOTO 420 !000
790 DISPLAY AT(24,1):"ENTER
THE LARGEST # AT HEX#1" :: D
ISPLAY AT(7,16):"     " :: D
EC=0 :: GOTO 740 !184
```

Improving the flexibility of catalog programs

This item was written by W. Leonard Taffs and appeared in his Feedforth column in the newsletters of the Southwest 99ers of Tucson, Arizona.

With disk cataloging programs, one frequently runs into programs that will catalog disks with specified choice of drives, such as 1-4 or 1-9, or sometimes only for drive number 1. In the majority of cases, the variable supplied for this INPUT statement is a *numeric* variable. Nothing wrong with this, of course, but it does limit your access to which drives can be cataloged.

How does one modify a catalog program to access other drives? The answer is to use a STR$ing variable instead of a numeric variable. Substitute a STR$ variable such as A$ or DSC$. Here are the lines to implement this:

```
100 INPUT "Enter Disk Drive:
":DSC$
110 DSC$="DSK"&DSC$&"."$FN$
```

FN$ equals the filename entered in your program *prior* to the request for the drive number. Add the following to be assured you have entered what you meant:

```
120 PRINT
130 PRINT DSC$,"O.K.? "
140 PRINT
150 INPUT "":K$
```

If you are modifying an existing catalog program, you need to check how the original INPUT variable was treated and follow up on any pertinent program lines that process this variable. Sometimes that variable will have been set, such as A=1. Delete it unless you intend to make use of it in some way. Chances are that later reference to that numeric variable is in the form of STR$(A) — if "A" was the variable used. Substitute your STR$ variable in place of it. Where it is STR$(A), replace with DSC$. Do not use STR$(DSC$).

(See Page 30)

USER NOTES

(Continued from Page 29)

DSC$ is already a string and you will get an error message.

This is a simple solution to the problem and I am curious why this was not made a universal practice. It makes accessing any numbered or lettered drive, such as DSKA, possible. If you want to avoid specific drive number errors, it is simple to following the DSC$ INPUT line with a limiting line or lines that specify which INPUT numbers or letters are acceptable. For example:

```
110 INPUT "Disk Drive? ":DSC$
115 IF DSC$<>"99" THEN 110
```

Line 115 would refuse any INPUT entry that was not 99, etc. If you understand how to use ACCEPT AT statements, you can limit your drive numbers to whatever is included in the VALIDATE part of your ACCEPT AT statement. Any other entries are then refused.

Saving time in Funnelweb

This item has appeared in several user group newsletters. The author is unknown.

This is a time saver for people who use the TI-Writer formatter of Funnelweb. You can do a disk directory while in the editor and mark a file so that you do not have to retype in the DSKx.FILENAME. This is a big help if you can't remember the file name.

However, if you do a disk directory while in the formatter, apparently you can't mark the file. So, if you want to mark the file, you have to enter the editor, do a disk directory, mark the file, exit the editor, and re-enter the formatter. This is clumsy and slow if you are not using a RAMdisk.

The trick to make this tip work is, while in the formatter, do a disk directory (FCTN-7). Arrow down to the file you want and press the space bar. This places an invisible mark on the file. Press CTRL-= to return to the formatter, then press FCTN-D (right arrow) to the place for the new file name in the formatter mail box. The name of the file you marked will automatically appear.

This saves the time of repeatedly loading the editor and formatter just to mark files. This is also useful for users who are intimidated by long file names or who cannot remember whether the file name was DOCS or *DOCS or READ-ME or README. You get the idea.

COMPARE finds differences between programs

The following program was written by Mike Dodd, a former MICROpendium columnist, a number of years ago. It is used to compare one program file with another and then lists all non-matching lines. This is useful when working on several similar versions of the same BASIC or Extended BASIC program.

It requires a disk drive. Instructions are in the form of REM statements at the beginning of the program.

```
100 !************************
 !073
110 !*  COMPARE PROGRAM    *
 !214
120 !*   by Mike Dodd      *
 !019
130 !************************
 !073
131 ! In K-Town 99'er V.2 *1
     April 1985 !042
140 !Version 85.0406.1XB
    Requires disk drive.
    Compares two programs,
    gives list of all diff-
    erences. !150
150 !Save old program in
 MERGE format (SAVE DSK1.(ol
 dfilename),MERGE). SAVE up-
 dated program in MERGE for-
 mat(SAVE DSK1.(newfilename)
 ,MERGE) !166
160 !RUN this program, answe
r prompts for OLD FILE name,
 NEW FILE name, and a differ
ent OUTPUT FILE name. !046
170 !When finished, type NEW
, then MERGE DSK1.(outputfil
ename) and ENTER !183
180 ! Can be MERGED into oth
er copies of OLD program to
update them !175
190 DEF @(@$)=ASC(SEG$(@$,1,
1))*256+ASC(SEG$(@$,2,1))!22
1
200 A$=CHR$(255)&CHR$(255)::
 DISPLAY AT(1,1)ERASE ALL:"O
LD    FILE:": :"NEW    FILE:
": :"OUTPUT FILE:" !028
210 ACCEPT AT(1,13)BEEP:B$ :
: ACCEPT AT(3,13)BEEP:C$ ::
ACCEPT AT(5,13)BEEP:D$ :: OP
EN #1:B$,INPUT ,VARIABLE 163
 !063
220 OPEN #2:C$,INPUT ,VARIAB
LE 163 :: OPEN #3:D$,OUTPUT,
VARIABLE 163 !239
230 LINPUT #1:@$ :: LINPUT #
2:E$ :: F$=SEG$(@$,1,2):: G$
=SEG$(E$,1,2):: A=@(F$):: B=
@(G$)!144
240 IF F$=A$ AND G$=A$ THEN
CLOSE #1 :: CLOSE #2 :: PRIN
T #3:A$ :: CLOSE #3 :: STOP
!033
250 IF B>A THEN PRINT #3:F$&
CHR$(131)&" **DELETED LINE *
*"&CHR$(0):: LINPUT #1:@$ ::
 F$=SEG$(@$,1,2):: A=@(F$)::
 GOTO 240 !117
260 IF A>B THEN PRINT #3:E$
:: LINPUT #2:E$ :: G$=SEG$(E
$,1,2):: B=@(G$):: GOTO 240
!123
270 IF @$<>E$ THEN PRINT #3:
E$ !051
280 GOTO 230 !053
```

Fast dialing with FAST-TERM

This item appeared in The Computer Voice, the newsletter of the Southern California TI99/4A Computer Group.

If you are using FAST-TERM, and have an auto-dial modem, having it get you online can be super fast. All you need to do is create a D/V80 file without control codes.

Load TI-Writer or its equivalent and, while in edit mode, press CTRL 0 to disable word-wrap. Then enter the phone number of the service you want to dial as follows:

ATDP2788155 for pulse dialing, or;
ATDT2788155 for tone dialing. Of

(See Page 31)

USER NOTES CLASSIFIEDS

(Continued from Page 30)

course, substitute the phone number you want to dial for the phone numbers listed here.

Save the file with the phone number under the name "DSK1.SENDFILE." Place this file on your disk with FAST-TERM. Now, at the point you would normally dial the service, or type in the phone number, just hit FCTN. FAST-TERM will retrieve the number and do the dialing for you.

Another way to use a RAMdisk with a single floppy system

Don Walden of Cecure Electronics follows up the User Note on using a RAMdisk with a single floppy system in the December 1993 MICROpendium with a solution requiring no hardware modifications.

Set the DIP switches for Drive 1 and rive 2. Then the floppy drive will respond to either, so when the RAMdisk overrides DSK1, you have access to DSK2 automatically without any external switches. You can even do this for Drive 3, he notes, and, if you have a Myarc or CorComp controller, for Drive 4.

Walden also observes that users can always access anything on DSK1 by using the disk name.

MICROpendium pays $10 for items submitted by readers that appear in this column. Send your routines, ideas and tips to MICROpendium User Notes, P.O. Box 1343, Round Rock, TX 78680.

Policy

The cost of classified advertising is 25 cents per word. Classified display (i.e., special formatting or graphics) is $9 per column inch. Classified advertisements must be paid in advance. Classified advertisers may request a category under which they would like their advertisements to appear, but the final placement decision is the responsibility of the publisher.

Classified deadlines will be kept open for as long as practical. For the purpose of classified advertising deadlines, any classified ad received later than the first day of any month cannot be assured of placement in the next edition. We will do our best to include every advertisement that is submitted in the earliest possible edition.

The publisher offers no guarantee that any advertisement will be published in any particular issue. Any damages that result either from errors in copy or for failure to be included in any particular edition will be limited to the amount of the cost of the advertisement itself. The publisher reserves the right to reject any advertisement.

The advertiser may elect to publish the advertisement in subsequent editions at the same charge, payable prior to publication. The deadline for carryover classifieds is the same as for new advertising.

In submitting an ad, please indicate whether you would like a refund if it is not published in the requested edition or whether you would like us to hold it for the next edition. Cancellations and refunds cannot be made after the second day of the month.

Send classified advertising to: MICROpendium, P.O. Box 1343, Round Rock, TX 78680.

FOR SALE

TI99/4A

PHP1100	TI, 99/4A DUAL JOYSTICKS	14.95
PHP1260	32K MEMORY CARD	99.95
PHP1500	SPEECH SYNTHESIZER	29.95
PHM3109	TI LOGO II (32K Mem. Req.)	1.95
PHA4100	TI VIDEO MONITOR (Used)	99.95
PHM3058	MINI-MEM (W/Assembler)	39.95
UM1381-1	RF MODULATOR	19.95
4530	JOYSTICK "Y" ADAPTER	7.95
AC9500	TI99/4A POWER SUPPLY	9.95

JOY ELECTRONICS INC.—P.O. BOX 542526 DALLAS, TX 75354-2526
(800) 527-7438, OUTSIDE DALLAS AREA
(214) 243-5371, DALLAS AREA
ADD $5.00 S&H—TEX RES ADD 8 ¼% TAX
10/12 **CALL OR WRITE FOR FREE CATALOG**

SEVERAL TI SYSTEMS & parts for sale. For price list, write Jim Ferguson, 22 Welland Crescent, St. Albert, Alberta, Canada T8N 3W4. 11/1

FOR SALE

ENORMOUS TI99/4A INVENTORY. CATALOGS $2.00. **BRAATZS COMPUTER SERVICES**, 719 E. BYRD ST., APPLETON, WI 54911. **1-414-731-3478.** 10/12

WANTED

AMIGA MONITOR WANTED
Looking for a used RGB monitor for use with the Geneve. Call 800-777-6632.

Use MICROpendium classifieds to buy and sell your unused equipment

Buy/Sell Used Hardware and Software

National Used Software/Hardware Club has buyers looking for TI products, as well as sellers. More than 170 TI items are listed for sale. The membership fee is only $15/year and entitles you to buy or sell computer items via the NUS/HC database. You will also receive FREE our newsletter (mailed every 8-12 weeks). If you don't buy or sell anything during the term of your membership we will refund your annual fee. For a free, no-obligation infor-mation packet, write to NUS/HC, P.O. Box 1343, Round Rock, TX 78680; or call 512-255-1512. Here are some of the items offered for sale:

Advanced Diagnostics $25.00
MG Explorer $25.00
Editor/Assembler.....$10.00
Horizon RAMdisk (192K) $125.00

Call our BBS for periodic price and product updates—512-255-1557. Hours are from 5:30 p.m. to 8 a.m. weekdays and from noon Saturday through 8 a.m. Monday weekends. Supports 300-2400 baud.

The ONLY monthly devoted to the TI99/4A

Subscription Fees

❑ 12 issues, USA, **$35** ❑ 12 issues, Mexico, **$40.25**

❑ 12 issues, Canada **$42.50** ❑ 12 issues, other countries surface mail, **$40.00**

❑ 12 issues, other countries, air mail, **$52.00**

Outside U.S., pay via postal or international money order or credit card; personal checks from non-U.S. banks will be returned.

Address Changes

Subscribers who move may have the delivery of their most recent issue(s) delayed unless MICROpendium is notified six weeks in advance of address changes. Please include your old address as it appears on your mailing label when making an address change.

Check each item ordered (or list on separate page) and enter total amount here: _____

Check/MO ❑ VISA ❑ MasterCard ❑ (check one)

Card No. _____

Expiration Date _____

(Minimum credit card order is $9)

Signature _____

(Required on credit card orders.)

No sales tax on magazine subscriptions. Texas residents add 7.75% sales tax on other items, including back issues and disk subscriptions.

Mail to: MICROpendium, P.O. Box 1343, Round Rock, TX 78680

Name _____

Address _____

City _____

State _____ **ZIP** _____

The set of numbers at the top of your mailing label indicates the cover date of the last issue of your subscription.

Disks, Etc.

❑ **Back Issues**, $3.50 each. List issues: _____

_____

No price breaks on sets of back issues. Free shipping USA. Add 30 cents, single issues to Canada/Mexico. Other foreign shipping 50 cents single issue surface, $1.50 airmail. Write for foreign shipping on multiple copies.

OUT OF STOCK: Vols. 1, No. [illegible]; Vol. 2, No. 1; Vol. 4, No. 9

❑ **MICROpendium Index** (2 SSSD disks, 1984-1992), Extended BASIC required $6.00

❑ **MICROpendium Index II** (9 SSSD disks — 1 for each year — 1984-1992), XB required $30.00

❑ **MICROpendium Index II with MICROdex 99** (11 SSSD disks), XB required $35.00

❑ **MICROdex 99** (for use with MP Index II, 2 SSSD disks), XB required $10.00

❑ **MICROpendium Index II** annual disks ordered separately (1 disk per year, 1984-1992); each $6.00

MICROdex 99, by Bill Gaskill, is a collection of programs that allow users of MP Index II to modify their index entries, as well as add entries. MICROdex 99 supports many other functions, including file merging, deletion of purged records, record counting and file browsing.

GENEVE DISKS (SSSD unless specified)

❑ **MDOS 1.23F** (req. SSDD or larger, for MBASIC) $4.00

❑ **MDOS 1.50H** (req. for MBASIC) $4.00

❑ **GPL 1.5** $4.00

❑ **Myarc Disk Manager 1.50** $4.00

❑ **Myarc BASIC 3.0** $4.00

❑ **MY-Word V1.21** $4.00

❑ **Menu 80** (specify floppy or hard disk versions(s); includes SETCOLR, SHOWCOLOR, FIND, XUTILS, REMIND $4.00

GENEVE PUBLIC DOMAIN DISKS

These disks consists of public domain programs available from bulletin boards. If ordering DSDD, specify whether Myarc or CorComp.

	SSSD	DSSD	DSDD
❑ **Series 1**	$9.00	$7.00	$5.00
❑ **Series 2**	$9.00	$7.00	$5.00
❑ **Series 3**	$9.00	$7.00	$5.00
❑ **Series 4**	$9.00	$7.00	$5.00
❑ **Series 5**	$9.00	$7.00	$5.00
❑ **Series 6**	$9.00	$7.00	$5.00

SECOND CLASS