Covering the TI99/4A and the Myarc 9640

MICROpendium

Volume 14 Number 3 May/June 1997 $6

Fest West
RXB module, new SCSI DSR debut

Extended BASIC
Energy Efficiency

Hardware
Using a 3.5-inch drive with a CorComp disk controller
The 16-slot PEBMaking the change from jumper to switch
Doing your own repairs

The Art of Assembly
Thank you notes and some more tricks

Geneve 9640
Crunching time and date
What SCSI means to Geneve users

Beginning c99
Flow control statements

Reviews
Scanned Images, TI Module User Guides and Juno

User Notes
Controlling a cassette recorder, Tiny Lotto,
and that'll be the day, Fest West in Lubbock

CONTENTS

MICROpendium

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

All correspondence should be mailed to MICROpendium at P.O. Box 1343, Round Rock, TX 78680. 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 & FAX: (512) 255-1512
Delphi TI NET: MICROpendium
Internet E-mail: jkoloen@io.com
Home page: http://www.io.com/~jkoloen/
John Koloen .....Publisher
Larua Burns .....Editor

Fest West

RXB module, SCSI DSR debut .....Page 5
Lectures provide information .....Page 6

Extended BASIC

Energy Efficiency .....Page 6

Hardware

Using a 3.5-inch drive with a CorComp disk controller .....Page 10
The 16-slot PEB .....Page 17
Making the change from jumper to switch .....Page 25
Doing your own repairs .....Page 26

The Art of Assembly

Thank you notes and some more tricks .....Page 11

Geneve 9640

Crunching time and date .....Page 18
What SCSI means to Geneve users .....Page 30

Beginning c99

Flow control statements .....Page 20

Reviews

MICRO-Reviews: Scanned Images, and TI Module User Guides Page33
Juno .....Page 35

User Notes

Controlling a cassette recorder, Tiny Lotto, and that'll be the day, a TI Fest West in Lubbock .....Page 36

Classified .....Page 39

*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

List server gets lots of action

Subscribers to the TI list server, which is moderated by Tom Wills, have fairly broad interests but their messages generally center around the TI. Topical subjects do come up. For example, before the Super Bowl, a number of subscribers weighed in on the game and declared the Packers champs well before the game got under way.

And there's a good bit of what a non-user might call "bickering." But it's usually of the type where one person says the SCSI controller is better than the Myarc HFDC. And it goes from there. It's a lot like the age-old issue of whether the Mac OS is better than Windows. There's a lot of give and take and, remarkably, most of these issues are resolved in an amicable way, which is often not the case on other list servers I'm familiar with. I think that may be because TIers have been through a lot. After all, we predate the PC crowd and the Mac crowd. Of course, TIers are a helpful lot, and this trait shows up frequently on the list server where all it takes is one e-mail to get a handful of responses. It's a good place to go if you've got a software or hardware problem. There are a lot of experts who subscribe and they're very quick to respond. It's a great place for newbies, yes, there are TI newbies. Those of us who have been TIers for more than a decade forget what it was like in 1983 when everything was new.

TI CHAT SITE

Brad Snyder is hosting a TI chat page on his Web site. You can reach it at the following address: http://www.enter.net/~bsnyder/tichat.html. However, in order to participate, you'll need a Java-enabled browser.

BETA TESTING PC99 VERSION 4

Mike Wright says that he's getting ready to run V.4 of PC99 through beta testing. PC99 is the very fine TI emulator that has been getting better with every release. Version 4.0 will have support for the Myarc floppy disk controller, which means users will be able to create double-sided, double-density Myarc "disks" within PC99 on their PCs. Users are currently limited to DSSD TI format disks. I'm looking forward to it.

TI GETS INTO THE ACT AT FEST WEST '98

Tom Wills says Texas Instruments is going to help put on Fest West '98 in, of all places, Lubbock, Texas, birthplace of the TI. The event is planned for February. For more information, see page 38.

—JK

FEEDBACK

What programs do readers want?

I am wondering if you are getting any type or survey type answers as to what readers want? Perhaps some of us spend too much time on the type of programming that nobody needs or wants. Maybe there is so much out there already that everybody is satisfied with what they already can get easily. Anyway, food for thought.

Was appreciative of the fact you published my "MAKE1FROM2"Extended BASIC program (Jan/Feb 1997). I should point out that the italics/editor note insertion concludes with an erroneous sentence. the tilde needs to be used as the first character of the *last* record in the file.

The tilde was chosen as a file-end marker for the last file record as it is the highest visible ASCII character (126). In terms of sorting, no average user input items in a file will exceed this value. So, if you tell them to put this character at the beginning of their file, the file will be *ignored* as the program uses this marker to *close* the file being read.

While MAKE1FROM2 uses the tilde as EOF marker, if a file happened to make use of multiple tildes in records the program could be edited to use another marker or use customary EOF(n) procedures.

W. Leonard Taffs
Tucson, Arizona

(Big) Blue farewell

Your notice of renewal of my subscription to MICROpendium was received today and I wish to inform you that I will not be renewing the magazine or disk of the month. This is not because of anything you have said or done or left unsaid or undone. I still think your magazine is largely responsible for the extended life of the TI99/4A, and I wish you well in continuing publication.

No, my nonrenewal is caused by my converting to a Big Blue clone and your magazine will no longer be of direct interest to me. Poor business practices on the part of some distributors and/or writers of software accelerated my move.

Another factor which entered into my decision is the inability of the TI to operate at full throttle on the Internet and World Wide Web. I want to get onto the Net and the only way I know of is to get a clone which operates at many times the speed of the TI and has practically no limit to size of memory.

I would like you to know that I have enjoyed your magazine and I offer you my

(See Page 5)

FEEDBACK

(Continued from Page 4)

best wishes for continuing success in your endeavors.

David C. Swartz
Aurora, Colorado

Bashing away

Here at DORTIG we are still following the TI99/4A and doing a small amount of programming, nothing really fantastic, just a hobby to keep us out of trouble.

I have sent for Vern Jensen's starter kit for c99 and with a bit of luck may even write some programs in this language, have tried assembly but cannot seem to get really going, still we will see whether c99 can be mastered!

I correspond with Bruce Harrison and John Bull in the USA and they have kindly sent quite a few of their very good programs, and articles by Bruce. From reading MICRO it appears quite a few people are leaving the TI clubs and going over to PCs; still, we still bash away and enjoy this old but reliable machine (mine gets covered with cig. ash and lager, but except for a cough and hiccups it still keeps going) after a few thumps with a hammer!

Having read the reviews by Charles Good, in the MICROpendium January/February 1997 issue I would like to comment on the following programs.

Contract Bridge V4.0 by John Bull: Over the past years John has sent me various versions of this program, and I have spent many happy hours playing this game.

I never was a Bridge Player,Whist, Solo Whist,Pontoon(Blackjack!), were my games, so I was really starting from scratch. Whether my play is according to the rules leaves much to be desired but up to date the score is about 60/40 in favor of the computer.

With all the options given in this game I think it's worth getting, even if it means you can curse and kick the computer as much as you like without getting thumped back.

Assembly Poker from Marcel Software: I also play this game,very fast; if I had as much money as I lose then a visit to Las Vegas would be a disaster. More a game of *bluff* than Bridge but very enjoyable to play. *very fast*

Years ago I made a small program to play Pontoon (21s) but nowhere as good or as fast as the above games though for me it *worked* and sometimes I won!!.

Chainlink Master Solitaire V1.0 1989 by Walter Howe and Wayne Stith: Another card game I play (from MICRO). Good game to keep brainbox active and pass a quiet cold evening away.

When the grandchildren come I play card games with them, both young, and I'm a great believer that it helps them to count and remember cards played. From above you can see I like playing cards and must control myself or playing will take over.

Plenty of games to play but I am not a SHOOT BANG FIRE addict though those games do develop the reflexes and reactions.

Just knocked me bottle of lager over, but this time it missed TI so there should be no hiccups from 4A tonight?!.

As in the USA the TI users are getting fewer, DORTIG has two active members and three others who, through distance, keep in touch and visit now and then. For myself using the TI and corresponding with other users is all I need. Fairware/Shareware for programs is not that expensive,considering the time and effort by programmers is taken into consideration. For U.K. the cost of airmail, averaging $1.50 U.S to the pound, for MICRO works out about £37 per year,so if one puts say £2 (pounds U.K.) $3 U.S. a week towards costs it allows one to at least help the programmers out and keep the TI alive.

The TI99/4A U.K. User Group, to which I belong, produce a good magazine, TI*MES, quarterly and that keeps one in touch with both the U.K.,USA and other countries' groups. Mr. Ross Bennet repairs members consoles, etc., for just the cost of bits and postage thanks to him I hope to get some decent disk drives at the AGM in May.

Maybe in a year's time I might work myself up to another letter so until then many thanks to all TI users ,programmers, the staff who produce MICROpendium, hardware producers and commercial firms that keep the TI alive and kicking.

John Murphy
Creekmoore, Poole, Dorset
United Kingdom

Send your letters and comments to *MICROpendium Feedback*, P.O. Box 1343, Round Rock, TX 78680.

Fest West report

RXB module announced, SCSI DSR fully functional

Although attendance at Fest West '97 was less than expected, several announcements were made of interest to TIers.

Upwards of 50 TIers attended the fair, which was held in San Jose, California. There were seven exhibitors.

Announced at Fest West were:

• RXB in module form. Rich Extended BASIC was developed by Rich Gilbertson and has been available for use with GRAM devices. The product is already shipping. It includes all disk manager functions for all devices, including SCSI, HFDC, floppy drives, and RAMdisks. It also supports the AMS card.

• Don O'Neil announced that V1.1 of the SCSI DSR is fully functional. It includes new utilities and is available for $5, or can be exchanged without charge for the old DSR. The DSR can be downloaded from O'Neil's web page (http://www.sonyx.com/wht/).

• Availability of new SCSI cards.

• New CD-ROM with more files.

• New features added to CaDD Electronics' PC99 TI emulator.

NEXT YEAR

The Southwest 99ers are going to host Fest West '98. The group will be

(See Page 6)

FEEDBACK —

(Continued from Page 5)
putting the event on in Lubbock, Texas, where the TI was produced.

Report on PC99, new RXB module, and WHT SCSI workshops

By DAVID ORMOND

The following article is excerpted from the newsletter of the Southwest 99ers User Group.—Ed.

There were four lecture/demos at Fest West '97. I'll start with the presentation by Mike Wright of CaDD Electronics. It went off the best.

CaDD's offering is PC99, a TI emulator for the PC. For $47, you get the basic emulator, which runs the 99/4A with 32K of memory, the TI RS232 card, the TI disk controller, and three DS/SD drives. For $94 you also get the "professional" packages which includes emulation of the P-Code card and a debugger that completely displays the content of the 99/4A as it runs or steps through instructions.

PC99 comes with programs that, along with a serial cable (null modem), allows you to transfer disk images between the PC and a TI. File conversion programs on the PC allow you to convent PC files to and from TI files in all the PC-Transfer formats — text, D/F128, SYLK, and TI-Files-encoded. If you PC has a Soundblaster card, you get full 99/4A sound capability. Without it you get only one channel the built-in speaker. It makes games sound very strange.

Wright ran several programs to demonstrate the quality of the emulation, which is indistinguishable from the 99/4A for even the most hardware-dependent and tricky code-infested programs. Wright said that CaDD is far along in adding emulation of the Myarc floppy controller.

CaDD also has manuals for just about everything in the TI world are available in Adobe Acrobat format for $2 each.

PC99 would be useful for extracting files from the Competition Computer's TI CD-ROM. By pulling them off on the PC, putting them into a disk image file for PC99, and then using the utilities for transferring disks to the TI, you can easily tap the vast resources of the CD-ROM.

RXB has dozens of useful new commands, including some that give full use of AMS memory.

RXB MODULE

Rich Gilbertson is the author of RXB, a new dialect of TI Extended BASIC. There is a shareware version circulating, but you need a GRAM device to use it. Gilbertson has been working with Don O'Neil and Kyle Crichton to create a new RXB cartridge, which would be better than the disk-based, shareware version.

This new RXB module comes with a menu that allows you to run a "batch" program (similar to PC DOS "BAT" files or Unix scripts, which more or less input keystrokes from a file), enter RXB itself, run Editor/Assembler, and run a disk manager which already fully supports SCSI, as well as all other storage technologies.

The disk manager seems to be integrated into the other packages. In E/A, you can call up a catalog to select a file to run or view. The ASSM and EDIT program files can be on any disk, not just DSK1. It will scan all drives for a UTIL1 when you hit Enter from the E/A5 prompt, not just DSK1.

RXB has dozens of useful new commands, including some that give full use of AMS memory. Gilbertson says that you can load multiple programs into AMS pages and run them from a single XB program.

Although plagued with breakdowns — his PGRAM and Horizon cards died — Gilbertson attempted to demonstrate a program similar to BOOT but with no practical limit on menu pages, and would execute BASIC, XB, E/A3, and E/A5 programs. This loader program itself is constructed by running a batch file!

Crichton is going to produce the cartridge, which will cost $59.95, and will be available soon. It would been available at Fest West but there were some problems with a programmable logic device on the cartridge.

Crichton says this cartridge is a very good "general purpose" cartridge. RXB is going to be offered on it but there are plans to use the same design to offer several unreleased GROM programs, such as Germ Patrol and several Disney titles, at a lower cost. He showed me the cartridge and opened it up. This is for real.

COMPETITION CD-ROM

Crichton gave a short discussion on his CD-ROM. TIers may not that Beery Miller has offered a CD-ROM with public domain and shareware files on it. Crichton says that his CD has more, including the libraries of several user groups.

SCSI UPDATE

Don O'Neil of Western Horizon Technologies attempted to demonstrate the new directory manager program the SCSI controller card. Unfortunately, the monitor died and we couldn't see anything. The directory manager allows copying files from floppies to the root directory of the SCSI drive. It doesn't support copying to or from subdirectories, yet. Dave Nieters said that since the DSR is pretty far along, he will take some time to fix the program so that he will be in a truly usable state. O'Neil reminded us that the directory manager was originally intended for debugging purposes and that the RXB cartridge includes a fully-featured disk manager at your fingertips.

Anyway, O'Neil has a new SCSI design, rev 3. He had the SCSI card at the event but he wasn't selling. It uses the same kind of programmable logic device as the RXB cartridge, and he hasn't quite got the hang of it yet. However, he has new software from Lattice Semiconductor and has high hopes of finishing soon.

He also said that DSR V1.1 is thoroughly tested and available but that DSK1 emulation for SCSI is not yet implemented This means some programs that look at DSK1 for files will not be able to find them on the SCSI drive.

Use your computer to improve household energy efficiency

The following program, Energy Efficiency, was released by Program Innovators of Milwaukee, Wisconsin.

The program requires Extended BASIC. It is menu driven and can be used to measure household energy efficiency.

ENERGY EFFICIENCY

```
100 DIM N$(7),M$(20),L(20),P
B(20),O(20)!055
110 N$(0)="THERMS" !054
120 N$(1)="CUBIC FEET" !018
130 N$(2)="GALLONS" !118
140 N$(3)="POUNDS" !063
150 N$(4)="THERMOSTAT SET-BA
CK" !201
160 N$(5)="FURNACE HEAT-EXTR
ACTER" !157
170 N$(6)="FURNACE EXHAUST D
AMPER" !152
180 N$(7)="NEW FURNACE" !125
190 CALL CLEAR !209
200 PRINT " *** ENERGY EFFI
CIENCY ***": : : :TAB(10);"C
OPYRIGHT 1984":TAB(10);"PROG
RAM INNOVATORS": :!228
210 FOR X=1 TO 300 !171
220 NEXT X !238
230 PRINT : :" 1. R VALUES":
:" 2. ENERGY COSTS": :" 3.
HOME EFFICIENCY" !020
240 PRINT :" 4. COMPARING VA
RIOUS ENERGY MEASU
RES": : : :!085
250 INPUT "CHOICE ":J !042
260 IF (J>4)+(J<1)THEN 250 !
156
270 CALL CLEAR !209
280 ON J GOTO 2400,1670,290,
1720 !126
290 PRINT " < HOME ENERGY EF
FICIENCY >" !096
300 PRINT :" FUEL USED:": :
:" 1. ELECTRICITY": :" 2. NA
TURAL GAS": :" 3. FUEL OIL":
:" 4. WOOD": :" 5. KEROSENE
": :" 6. COAL": :" 7. PROPAN
E(LPG)": : :!105
310 INPUT "":X !102
320 IF (X>7)+(X<1)THEN 310 !
248
330 CALL CLEAR !209
340 PRINT "INPUT TOTAL ANNUA
L FUEL USE": :!084
350 ON X GOTO 360,400,460,50
0,550,590,650 !234
360 PRINT " ELECTRICITY": :"
KILOWAT HOURS ";!208
370 INPUT X !234
380 B=B+3413*X !181
390 GOTO 750 !063
400 PRINT " NATURAL GAS": :"
MEASURED IN":" 1. THERMS":"
2. CUBIC FEET": :!065
410 INPUT X !234
420 PRINT :" NUMBER OF ";N$(
X-1);!060
430 INPUT Y !235
440 B=B-100000*Y*(X=1)-1025*
Y*(X=2)!002
450 GOTO 750 !063
460 PRINT " FUEL OIL": :" GA
LLONS";!217
470 INPUT X !234
480 B=B+138700*X !031
490 GOTO 750 !063
500 PRINT " WOOD": :" TYPE U
SED": :" 1. HARDWOOD":" 2. M
IXED":" 3. SOFTWOOD": :!135
510 INPUT "":X !102
520 INPUT "NUMBER OF CORDS "
:Y !176
530 B=B+(21-2*X)*Y*1000000 !
116
540 GOTO 750 !063
550 PRINT "KEROSENE": :!049
560 INPUT "GALLONS ":X !158
570 B=B+135000*X !021
580 GOTO 750 !063
590 PRINT "COAL": :"1. ANTHR
ACITE":"2. HI-VOL BITUMINOUS
":"3. LO-VOL BITUMINOUS":"4.
LIGNITE": :!139
600 INPUT X !234
610 INPUT "TONS ":Y !208
620 A=(X=1)*25400000+(X=2)*2
2000000+(X=3)*28600000+(X=4)
*13800000 !179
630 B=B-Y*A !097
640 GOTO 750 !063
650 PRINT "PROPANE GAS ": :"
MEASURED IN": : :!007
660 FOR X=1 TO 4 !074
670 PRINT X;". ";N$(X-1)!102
680 NEXT X !238
690 PRINT : : :!187
700 INPUT X !234
710 PRINT : :"NUMBER OF ";N$
(X-1)!028
720 INPUT Y !235
730 A=(X=1)*100000+(X=2)*250
0+(X=3)*91000+(X=4)*21500 !0
86
740 B=B-Y*A !097
750 PRINT !156
760 INPUT "ANY OTHER FUEL US
ED (Y/N) ? ":X$ !147
770 CALL CLEAR !209
780 IF X$="Y" THEN 300 !133
790 PRINT !156
800 INPUT " TOTAL SQUARE FOO
TAGE OF      LIVING SPACE ":X
!020
810 PRINT !156
820 INPUT " TOTAL ANNUAL DEG
REE DAYS IN YOUR AREA
(HEATING & COOLI
NG IF USED) ":D !232
830 PRINT : :"RATINGS:":" >
4.....TOP EFFICIENCY":" 4-1
0...GOOD":" 10-20..AVERAGE"
:" >20....BAD": : :" YOUR E
FFICIENCY RATING:": :INT(100
*B/X/D)*.01: : :!090
840 INPUT "":X$ !138
850 GOTO 190 !013
860 PRINT " TYPE OF FUEL": :
" 1. OIL":" 2. NATURAL GAS":
" 3. PROPANE GAS":" 4. ELECT
RICITY": : :!148
870 INPUT " ":A !112
880 IF A<>3 THEN 930 !094
890 PRINT " PURCHASED BY THE
":"1.";N$(0),"2.";N$(1):"3."
;N$(2),"4.";N$(3)!155
900 INPUT " ":X !135
910 F=100000*(X=1)+2500*(X=2
)+91000*(X=3)+21500*(X=4)!09
1
920 GOTO 980 !038
930 IF A<>2 THEN 970 !133
940 INPUT " PURCHASED BY THE
```

(See Page 8)

ENERGY EFFICIENCY —

(Continued from Page 7)

```
1. THERM
2. CUBIC FOOT
":X !083
950 F=100000*(X=1)+1025*(X=2
)!201
960 GOTO 980 !038
970 F=138700*(A=1)+3413*(A=4
)!178
980 PRINT !156
990 INPUT " COST PER FUEL UN
IT USED      (THERM, KWH etc)
$":X !057
1000 PRINT !156
1010 INPUT " ANNUAL DEGREE D
AYS ":D !130
1020 C=X/-F !037
1030 PRINT !156
1040 INPUT " FURNACE EFFICIE
NCY   0 TO 99.9 % ":E
!038
1050 IF (E<0)+(E>99.9)THEN 1
040 !083
1060 RETURN !136
1070 PRINT :" < CONDUCTION H
EAT LOSSES >": :!168
1080 INPUT " IS THIS ABOVE G
ROUND AREA ??? (Y/N) ":A$
!231
1090 IF A$="Y" THEN 1160 !20
5
1100 PRINT :"BELOW GROUND LE
VEL LOSSES": : :!073
1110 A$="N" !057
1120 INPUT "DEGREE DIFFERENT
IAL   INSIDE-OUTSIDE
":P !041
1130 PRINT !156
1140 INPUT "NUMBER OF DAYS "
:Q !093
1150 PRINT !156
1160 INPUT " AREA TO BE COMP
UTED  (SQUARE FEET) "
:A !068
1170 PRINT !156
1180 IF J=4 THEN 1660 !132
1190 INPUT " R VALUE OF AREA
PRESS 'R' TO SE
E LIST ":R$ !221
1200 IF R$<>"R" THEN 1240 !2
32
1210 GOSUB 2430 !215
1220 INPUT "R=":R$ !021
1230 CALL CLEAR !209
1240 R=VAL(R$)!205
1250 Z=(A$="Y")*D+(A$="N")*P
*Q !094
1260 H=-C/R*A*24*Z !053
1270 PRINT :" HEAT LOSS = $"
;INT(10000*H/E)*.01 !073
1280 IF J=4 THEN 1660 !132
1290 T=T+H !111
1300 PRINT :" ANOTHER ";!038
1310 IF A$="N" THEN 1370 !14
9
1320 PRINT "ABOVE GROUND" !2
03
1330 INPUT " AREA (Y/N) ? ":
A$ !128
1340 IF A$="Y" THEN 1160 !20
5
1350 IF A$<>"N" THEN 1300 !0
16
1360 PRINT :" ANY ";!249
1370 PRINT "BELOW GROUND LEV
EL" !117
1380 INPUT " AREA (Y/N) ? ":
B$ !129
1390 IF B$="Y" THEN 1100 !14
6
1400 PRINT : :"TOTAL CONDUCT
ION HEAT LOSS":INT(100*T)*.0
1: :" < INFILTRATION HEAT LO
SS >": :!107
1410 INPUT "AVERAGE SIZE WIN
DOW   (LENGTH,WIDTH) "
:[,] !241
1420 PRINT :"NUMBER OF WINDO
WS ";!199
1430 INPUT "":\ !106
1440 _=\*(2*[+3*])!159
1450 PRINT :"NUMBER OF OUTSI
DE DOORS ";!102
1460 INPUT "":\ !106
1470 PRINT :"DOOR SIZE (L,W)
";!203
1480 INPUT "":[,] !121
1490 _=_+\*6*([+])!004
1500 PRINT :"PERCENT DOORS &
WINDOWS     WITH STORMS ( 0
-100 )";!201
1510 INPUT [ !237
1520 IF ([>100)+([<0)THEN 15
00 !003
1530 PRINT :"PERCENT DOORS &
WINDOWS     WEATHERSTRIPPED
";!033
1540 INPUT ] !239
1550 IF (]>100)+(]<0)THEN 15
30 !037
1560 PRINT :"TIGHT-FIT RATI
G (10 TO 30):": :" POOREST
(10)":" TIGHTEST (30)" !16
4
1570 INPUT \ !238
1580 IF (\>30)+(\<10)THEN 1
60 !069
1590 @=1600/(\+[/5+]/5+10)!
00
1600 H=.108*_*@*C*D !203
1610 PRINT :"AIR INFILTRATI
N HEAT LOSS $";H !232
1620 IF J=4 THEN 1650 !122
1630 T=T+H !111
1640 PRINT :" TOTAL COST OF
HEAT LOSS $";INT(T*10000/
E)*.01: : :!056
1650 INPUT "":X$ !138
1660 RETURN !136
1670 PRINT :" < TOTAL HEAT
OSS EXPENCE >": : :!124
1680 T=0 !011
1690 GOSUB 860 !175
1700 GOSUB 1070 !130
1710 GOTO 190 !013
1720 PRINT "< COMPARING SAV
NGS FOR > < VARIOUS ENERGY
MEASURES >": :!082
1730 GOSUB 860 !175
1740 N=N+1 !021
1750 CALL CLEAR !209
1760 PRINT :" METHOD NUMBER
";N:" TO BE TRIED,": :"1. I
SULATION":"2. INFILTRATION"
"3. ";N$(4):"4. ";N$(5):"5.
";N$(6):"6. ";N$(7): :"7. F
NISHED": : :!161
1770 INPUT "":W !101
1780 IF (W>7)+(W<0)THEN 177
!174
1790 ON W GOTO 1800,1800,22
0,2030,2030,2520,2330 !243
1800 INPUT "NAME OF METHOD
:M$(N)!032
1810 IF W=2 THEN 1830 !058
1820 GOSUB 1080 !140
1830 PRINT :"COMPUTE PRESEN
HEAT LOSS": :!156
1840 IF W=1 THEN 1870 !097
1850 GOSUB 1410 !215
1860 GOTO 1880 !174
1870 GOSUB 1190 !250
1880 G=INT(10000*H/E)*.01 !
48
```

(See Page 9)

ENERGY EFFICIENCY —

(Continued from Page 8)

```
1890 PRINT :"COMPUTE NEW HEA
T LOSS": :!097
1900 IF W=1 THEN 1930 !157
1910 GOSUB 1410 !215
1920 GOTO 1940 !234
1930 GOSUB 1190 !250
1940 H=INT(10000*H/E)*.01 !0
49
1950 PRINT : :"COST OF ";M$(
N);!119
1960 INPUT "$":O(N)!061
1970 L(N)=G-H !022
1980 PB(N)=INT(10*O(N)/L(N))
*.1 !246
1990 PRINT :"ENERGY SAVINGS
= $";L(N): :"PAYBACK IN ";PB
(N);" YEARS": :" TRY ANOTHER
METHOD ? (Y/N)" !226
2000 INPUT X$ !014
2010 IF X$="Y" THEN 1740 ELS
E 2340 !161
2020 IF X=4 THEN 2340 !061
2030 IF E<87 THEN 2070 !088
2040 PRINT :"YOUR HIGH EFFIC
IENCY FURNACEIS ALREADY EQUI
PPED WITH A":N$(W+1)!148
2050 INPUT "":X$ !138
2060 GOTO 1760 !053
2070 IF W=5 THEN 2170 !146
2080 GOSUB 2490 !019
2090 PRINT :"A ";N$(5):"SHOU
LD SAVE YOU ";97-E;"%":"OF Y
OUR HEATING BILL, OR $";I
NT(T*(97-E))*.01: :!086
2100 INPUT "COST OF HEAT EXT
RACTER$":O(N)!027
2110 L(N)=INT(T*(97-E))*.01
!227
2120 PB(N)=INT(O(N)/T/(97-E)
)*.01 !248
2130 PRINT :"PAYBACK IN ";PB
(N);" YEARS": :!133
2140 M$(N)=N$(W+1)!219
2150 INPUT "":X$ !138
2160 GOTO 1740 !033
2170 GOSUB 2490 !019
2180 PRINT :"A ";N$(6):"SHOU
LD SAVE YOU 25%":"OF YOUR HE
ATING BILL, OR $";INT(T*2
5)*.01: :!211
2190 INPUT "COST OF EXHAUST
DAMPER$":O(N)!034
2200 L(N)=INT(T*25)*.01 !102
2210 PB(N)=INT(O(N)/T/25)*.0
1 !123
2220 GOTO 2130 !169
2230 INPUT "HOW MANY DEGREES
SET-BACK ":X !142
2240 [=INT(1200000*X/(D+250*
X))*.01 !098
2250 GOSUB 2490 !019
2260 L(N)=INT(T*[)*.01 !144
2270 PRINT :"A TEMPERATURE S
ET-BACK OF":X;" DEGREES":"BE
TWEEN 10:00 PM AND 6:00 AM":
"WOULD SAVE ";[;"%":"OF YOUR
HEATING BILL, OR $";L(N)
: :!128
2280 INPUT "WOULD YOU INSTAL
L AN AUTO- MATIC THERMOSTAT
SET-BACK? ":X$ !087
2290 IF X$<>"Y" THEN 2130 !1
15
2300 INPUT "COST OF SET-BACK
$":O(N)!166
2310 PB(N)=INT(O(N)/T/[)*.01
!165
2320 GOTO 2130 !169
2330 N=N-1 !022
2340 PRINT :" COST SAVING
S PAYBACK": :!202
2350 FOR X=1 TO N !155
2360 PRINT M$(X):"$";O(X);TA
B(9);"$";L(X);TAB(19);PB(X);
"yrs" !247
2370 NEXT X !238
2380 INPUT "":X$ !138
2390 GOTO 190 !013
2400 GOSUB 2430 !215
2410 INPUT "":X$ !138
2420 GOTO 190 !013
2430 PRINT : :" < INSULATI
ON VALUES >":"CELLULOSE
R 4.0/in":"MINERAL WOOL
R 3.2/in":"FIBERGLASS BA
TTS R 3.5/in":" "" ""
BOARD R 4.0/in" !212
2440 PRINT " "" "" LOO
SE R 3.0/in":"PERLITE
R 2.8/in":"VERMICULITE
R 2.2/in":"POLYSTYRENE
R 5.0/in":"FIBERBOARD
R 2.0/in" !072
2450 PRINT "HARDWOOD
R 0.9/in":"SOFTWOOD
R 1.2/in":"PLASTERBOARD
R 0.9/in":"ASPHALT
R 2.8/in":"CONCRETE/STONE
R 0.1/in" !246
2460 PRINT "BRICK COMMON
R 0.2/in":"PLASTER
R 0.4/in":"ASBESTOS
R 1.0/in":"CORKBOARD
R 3.7/in":"CONCRETE BLOCK
R 1.0" !159
2470 PRINT "GLASS
R 0.9/pane":"AIRSPACE 3/4in
+ R 0.9":"AIRFILM (INSIDE)
R 0.7":" "" "" (OUTSIDE) R
0.2 ";!021
2480 RETURN !136
2490 IF T>0 THEN 2510 !225
2500 INPUT "AVERAGE ANNUAL H
EATING BILL $":T !000
2510 RETURN !136
2520 PRINT TAB(10);N$(7): :"
NEW EFFICIENCY RATING" !205
2530 INPUT "":U !099
2540 GOSUB 2490 !019
2550 _=INT(100*T*E/U)*.01 !0
10
2560 INPUT "COST OF NEW FURN
ACE   $":O(N)!149
2570 L(N)=T-_ !058
2580 PB(N)=INT(O(N)/L(N)*100
)*.01 !088
2590 PRINT "ANNUAL SAVINGS =
$";L(N): :"PAYBACK IN ";PB(
N);" YEARS": :!105
2600 GOTO 2140 !179
```

Austrian user offers program for PCX-TI graphics

Robert Kacsich of the Vienna University of Technology in Austria has made his program PIC2TI (V0.1) available for download. It is an MS-DOS program that reads a part of a monochrome PCX picture on a PC, and creates TI-BASIC source code that a user can type into a TI99 to display the picture on the TI. The TI source code creates the appropriate characters using CALL CHAR() and displays them using CALL HCHAR().

Users can download the program from: http://stud1.tuwien.ac.at/~e9125767/pub/ti99/PIC2TI01.ZIP (45K)

Using a 3.5-inch drive with a CorComp disk controller

By JOHN PARKEN

The following article appeared in the newsletter of the Cleveland Area TI99/4A User Groups. Parken is a member of the TI-CHIPS group.—Ed.

The 5.25-inch format of the TI has been a plus because 5.25-inch disks have been more or less replaced by the more popular 3.5-inch format on the PC. The people who get rid of 5.25-inch disks because they don't use them anymore give me a good source of disks to reformat for my TI.

For people who are putting no more money into their TI, you can buy 3.5-inch PC drives for use with your TI until you decide to retailer It. Then recycle them on your PC system. If you have a CorComp disk controller, you can use 3.5-inch disk drives with low density disks for them. I have thought about putting two 3.5-inch drives on my system because I have seen so many 3.5-inch disks on sale with a rebate for the whole purchase price. I just bought 100 Sony disks for $39 and got a $30 rebate check, which means that I paid just nine cents for each disk. It's hard to pass up a deal like that. I can't understand how the manufacturers can make any money off a deal like that but I love them.

There shouldn't be a problem hooking up two 3.5-inch drives to a CorComp disk controller card, even if there are already two 5.25-inch drives connected. Not quite. Drives have changed over the years. All PC 3.5-inch drives are set up for drive two. This is OK for PCs. The PC cable takes care of selecting drive one or two, but what about the TI? Gone are drive select jumpers. There is no select for high or low density. Fortunately, I can use the same approach on the TI that is used on a PC. I will do my switching in the cable. Pin 12, or wire 12, is the drive select for drive number 2. If I want drive 3, all I have to do is move wire number 14 coming out of the disk controller to wire number 12 into the drive. The disk controller will then consider that drive as drive number 3. The drive select lines on CorComp card are:

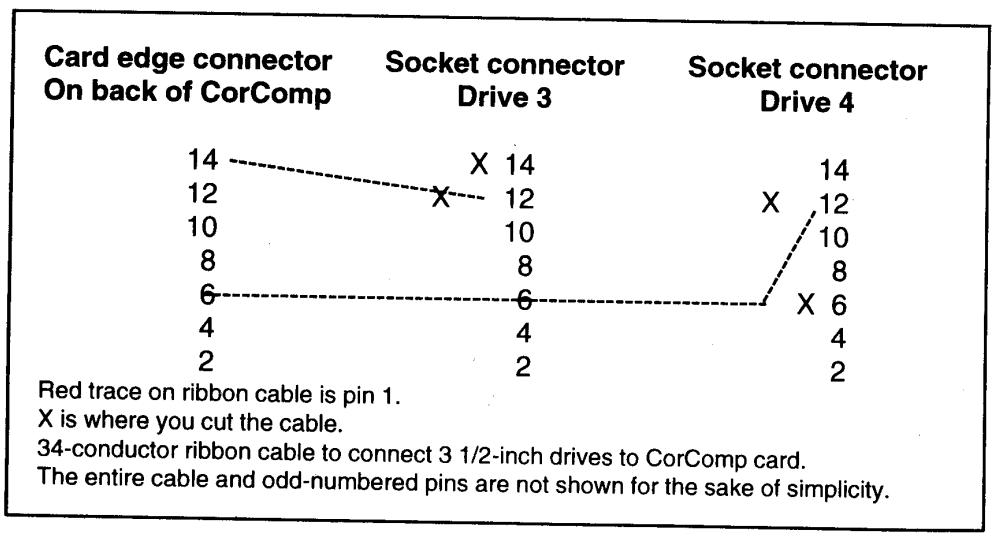
Select line	Drive
6	drive #4
10	drive #1
12	drive #2
14	drive #3.

The drives I used I bought for $19.50 each at a computer show. They are Neutronics (Mitsumi) 3.5-inch, 1.44mb units. I also bought splitter cables to power the drives. I had to go to two tables to get the right combination I needed. I used a Y-cable for a 5.25-inch drive (larger connector) to get my power outside of my Peripheral Expansion Box. I experienced some problems with the 3.5-inch drives by using just the PEB power supply, so I added an external 5-volt power supply. The power supply I use provides one amp @5VDC. You can buy a wall plug in a 5-volt power supply from Jameco or Radio Shack. When you hook it up, the black and red wires are GND and +5VDC, respectively.

The second connector is a double 3.5-inch (Small Connector). These 3.5-inch drives need only 5 VDC. Use a voltmeter to see which wire from the wall-mounted 5-volt power supply is GND and +5VDC. A 24-inch length of 34-conductor ribbon cable with one 34-position card edge connector and two 34-pin socket connectors make up the cable. All odd pins are at ground potential. In my cable diagram I show only part of the cable and only the even-numbered wires.

Please remember this is for a CorComp disk controller only. Even when you use a 3.5-inch 1.44mb drive, it will format 3.5-inch disks to 360 kilobytes only, and only if you cover the density hole with tape (black) or use a write protect tab.

As I wrote this article I built the cable. Then I tried it. It did not work. The drive light came on (all four drives) all the time. Upon further inspection, when the card edge connector was crimped on it was off just enough side-to-side to short all the pins together. I cut off that connector and re-crimped a new one on to fix the problem. These connectors should be available at Radio Shack. Having the 3.5-inch drives is nice. Again, the TI user is a winner only because he has some compatibility with PC equipment.



THE ART OF ASSEMBLY — PART 64

Thank you notes and some more tricks

By BRUCE HARRISON

Once again we're passing out thanks. Today's first person for thanks is Tony McGovern. Once again Tony's talents have made our lives easier than we have a right to expect. The main reason for our thanks this time is, of course, the Funnelweb system of programs.

This story starts with the previous column's problem about using the c99 compiler while our RAMdisks were turned on. It turned out that there's an exception. In our setup, there are three Horizon RAMdisk cards. The first one is at CRU address >1000, and that has only DSK3. The second is at >1200, and has drives DSK4, 5, and 8. The third one is at >1400, and has the drives DSK6 and DSK7 on it. Through a series of carefully controlled experiments, (accidentally left 6 and 7 card turned on) we found that the c99 compiler would work with that RAMdisk turned on. Apparently the problem with c99 is related to CRU addresses. What was important, though, was that we keep Funnelweb on drive 7 at all times. Thus we could gain access to editing capability without having to have E/A's EDIT1 on a disk in drive 1, and could also use drive 6 for our c99 source files and such. Funnelweb had once more saved our bacon, but that's not all.

This story starts with the previous column's problem about using the c99 compiler while our RAMdisks were turned on. It turned out that there's an exception.

WHAT IS THIS SCRIPT STUFF?

There are a lot of features packed into Funnelweb, many of which we've never tried. Mainly that's because we're too busy creating new stuff to properly examine the old stuff. We'd used some of Tony's loaders to check out our programs, but there was one on that list called Script Loader that was a mystery. In our recent work, we've been using pre-assembled object modules with little "demo" programs, both from c99 and pure Assembly. This means that for each test run we've got several object files to load before we can run anything. On countless occasions, we've got partway through the loading and couldn't remember whether the one just loaded was the demo program or the support files. Editor/Assembler has that nasty habit of blanking the input field after a file loads, so if you're distracted by something, it's easy to lose track, and then you'll surely get some kind of stupid error report from E/A. When we get lost like that, we just FCTN-9 back to the main menu of E/A and start over.

Tony's option 4 loader leaves the previous file name on the screen (scrolled up some) so you can much more easily track which files have already been loaded. But there was still that option 5 on Tony's Loader menu, called Script Loader. Finally, we decided to try that out. After some fumbling about, we found that our old floppy copy of Funnelweb had a sample file on it called SCRIPT. By using that as a starting point, we were able to put together a couple of script files for our projects. Boy did that help! By having a couple of those on drive 6, right along with the object files, we could get into Funnelweb, then use that script loader option, and very quickly load just the right files for our test. Tony even thoughtfully provided for an auto-start capability so that we can enter at any defined label after loading all necessary files. WOW! Never again will we forget what we're doing part way through a load sequence, at least not when we can quickly make up a little Script file. Tony even designed his sample so that sufficient instructions for using it are built right in!

ART'S ARTFUL ASSEMBLER

We've said thanks to Art Green before for his wonderful RAGASM assembler. That was for the fact that his error reporting approach is so much better than the TI Assembler gives. Now we're thanking Art again for the same product, but this time for how smart the assembler itself is. We keep that assembler handy at all times on RAMdisk, including a copy on our drive 7. The copies of RAGASM that we use have all been through an install process so that the "R" option is there by default, so we don't have to answer the Options prompt at all. That in itself is a neat feature! You can install RAGASM tailored to your particular needs

Recently, as our readers know, we've been conducting some experiments using Clint Pulley's c99 compiler. That program produces Assembly source code that's without the "R" for the registers. Again by carefully controlled experiment (namely just forgetting about it) we discovered that Art Green's RAGASM can handle non-"R" files just the same as "R" files, even when the "R" option is set. In at least one case, we embedded some original assembly code of our own within the c99 source, so the compiled file contained a mixture of "R" and non-"R" register references. RAGASM handled all that without a whimper! The TI Assembler can work this way too, but having this feature available in our favorite version of an assembler is great. Thank you, Art!

STILL MORE THANKS

To Clint Pulley, for making a truly marvelous C compiler on our little machine. In the process of trying to make little test programs for our c99 support libraries, we got a number of books on C from the local library. Reading those provided a whole load of ideas that we wanted to try out, but we couldn't be sure whether

(See Page 12)

THE ART OF ASSEMBLY —

(Continued from Page 11)

these slightly advanced tricks would work, so we just tried them out. They worked!

For example, in the C version of a FOR-NEXT loop, one can use more than one argument in the FOR part, thus setting initial values to other variables besides the one controlling the loop. In fact, the TO part can even use a variable that's not part of the FOR. Though the TO part can only have one test variable, the STEP part can include more than one variable as well. That looked like something that would be handy in one of our test programs, so we tried it this way:

```
for(row=8,col=12,chr='0';chr<':';++col,++chr)
hbchr(row,col,chr);
```

That worked! Notice that there's no NEXT required. The semi colon at the end of the second line marks the end of the loop, which keeps repeating so long as the variable chr is less than ':' (58). Notice also, as is the case in assembly, that we can load chr with the number 48 by putting a zero between single quote marks, and could make the limit 58 in a similar fashion using a colon between single quotes. In the last part of the FOR statement, (STEP) we have made both col and chr increment by placing both variables there preceded by a double plus sign. In case you're curious about such things, this loop puts the numbers from 0 through 9 on the screen at row 12, starting at column 8. The same thing could be done in Extended BASIC like this:

```
ROW=8:: COL=12:: FOR CHR=48 TO 57 :: CALL
HCHAR(ROW,COL,CHR)
:: COL=COL+1 :: NEXT CHR
```

Clint has built lots of nice features into his c99, including the ability to directly embed Assembly source, which is a great feature for us "Assembly guys". That's a feature that the PC versions of C lack. And so we thank Clint Pulley for incorporating such nifty tricks into c99.

FOR MY NEXT TRICK

No, I won't be leaping higher than Michael Jordan. In today's sidebar, however, is the source for an object module that takes string, numeric, and single-key inputs from the screen while in Half Bit Map mode. This has been designed so that the string and numeric input parts roughly equal an ACCEPT AT as used in Extended BASIC. Thus the function keys have the same actions as you'd expect.

Function-1 deletes the character under the cursor, moving the rest of the field to the left by one, and putting a space at the last character.

Function-2 puts you into Insert mode, so that each key typed moves all text following it rightward by one character, then appears.

Function-3 clears the input field.

Function-S moves the cursor left, and cancels insert.

Function-D moves the cursor to the right and cancels insert. Both Function-S and Function-D have a pause-then-repeat action, so you can quickly move many places. Enter or Function-E, Function-X, Function-8 or Function-9 will exit the routine.

There are two numeric input routines, which share much of the same source code, but take numbers instead of strings. One of these takes an integer in the range -32768 through 32767. The other takes a floating point number, which can span the whole range of TI floating point numbers. The integer case gives you an input field of only six spaces, so it has room for -32768. The floating point input gives a 15-space field, which should be large enough. In either case, the result is left for you at >834A, the floating point accumulator. If it's an integer, it will be just the word at that location. If it's floating point, it will be eight bytes starting at >834A. In both cases, scientific notation is accepted, and in both cases leaving the field blank will report 0. In the integer case, if you enter something with a decimal, that will be rounded to the nearest integer. (e.g. 4.4 will become 4, 4.5 will become 5) No, that wasn't our magic, but TI's, in the Convert Floating Point to Integer routine that we use.

For string inputs, you specify the number of spaces in the field up to a maximum of 80. Input in all cases is taken at the row and column you specify, where row runs from 1 through 24 and column from 1 through 32. All three routines check the validity of row and column, and the validity of the endpoint after the field length is added. If either the start or end point is out of range, the routine will exit back to your code without doing anything. In other words, be careful what you specify for row, column, and max length.

In addition to these three multi-character input routines, there are two single-key inputs, one called HBGK, and the other HBGKF. Each requires that row and column get specified. HBGK just awaits a keystroke, then echoes that key at row, col. HBGKF is a bit fancier, in that it puts a flashing cursor at row, column, and can accept a default response if you've put a character on the screen at row, col.

Let's say for example that you've put a Y at 12,16, and then used HBGKF at that spot. Pressing Enter will accept the Y as the input. Pressing Y will do the same, but any other key will be accepted "as is" and echoed to the screen at 12, 16. The key pressed or the default response if Enter was pressed, will be placed in R1 of your workspace as a word. The makes it easier for you to check what the response was. For example you could CI R1,'N' to see whether N was pressed.

THE PUBLIC DOMAIN DISK

As usual with things like this, we've put together a public domain disk with the stuff in today's sidebar, plus a collection of other routines for dealing with Half Bit Map mode. These include capabilities for using sprites, for displaying strings or numbers on the Half Bit Map screen, and so forth. The disk is called ADVHB and is available through the Lima Users Group. All the source code is included, with full annotation, so you can see how it works. Instructions are also provided, and several "demo" programs that let you see these routines at work without too much effort.

THE LITTLE DETAILS

You may notice something peculiar in the sidebar, in that we've included instructions that save the User Interrupt from >83C4 before our cursor starts and restored it afterward. This was done so that these routines would be compatible with the routines that pro

(See Page 13)

THE ART OF ASSEMBLY —

(Continued from Page 12)

vide sprites on the Half Bit Map screen. If there are sprites in motion when you enter one of these flashing cursor input routines, they will simply stop in their tracks while input is accepted, but will resume their motion when you exit the input routine. We thought this would be more desirable than just making you re-set all the sprites after the input routines. The simple key input routine HBGK does not do this, so your sprites will remain in motion while that routine is awaiting a keystroke.

Since all the source code is provided, you can modify some of this to suit your own needs, or just take excerpts for use in your own program. Note that none of this is useful unless you're in the Half Bit Map mode already. The Public Domain disk ADVHB includes the routines to get you smoothly into and out of that mode.

We hope that some of you will find this useful. Next time we'll nose around some other dark corner of the TI, and write about what we find.

SIDEBAR64

```
0001  * SIDEBAR64
0002  * AKA HBINP/S
0003  *
0004  *   FOR INPUTS TO ASSEMBLY PROGRAM
0005  *   USING HALF BIT MAP SCREEN
0006  *   Code by Bruce Harrison
0007  *   26 MAY 1995
0008  *   PUBLIC DOMAIN
0009  *
0010        REF  VSBW,VSBR,VMBW,VMBR,KSCAN  REF UTILS
0011        REF  XMLLNK
0012        DEF  ACCNUM,ACCSTR,HBGK,HBGKF,ACCFP
0013  *
0014  * REQUIRED EQUATES
0015  *
0016  STATUS EQU  >837C        GPL STATUS BYTE
0017  KEYADR EQU  >8374        KEY-UNIT
0018  KEYVAL EQU  >8375        KEY VALUE
0019  FAC    EQU  >834A        FLOATING POINT ACCUMULATOR
0020  FAC12  EQU  >8356        FAC + 12
0021  CSN    EQU  >1000        CONVERT STRING TO NUMBER
0022  CFI    EQU  >1200        CONVERT FLOATING TO INTEGER
0023  *
0024  * ACCNUM accepts an integer input
0025  * INVOKE ACCUM by:
0026  *      BLWP @ACCNUM
0027  *      BYTE ROW,COL
0028  *      DATA CLRSIG
0029  * ON EXIT, INTEGER RESULT IS AT >834A
0030  *
0031  *
0032  ACCNUM DATA WS,ACNCOD
0033
0034  ACNCOD LI   R2,6         LENGTH 6
0035         CLR  R5           R5=0
0036         CLR  R12          R12=0
0037         BL   @GRC         R0 HAS SCREEN ADDRESS
0038         MOV  @>83C4,R9    SAVE EXISTING USER INTERRUPT
0039         MOV  *R14+,R3     R3 HAS CLEAR FIELD FLAG
0040         JMP  ACCST2       JUMP AHEAD
0041  *
0042  * ACCFP  accepts a floating point number input
0043  * INVOKE ACCFP by:
```

You may notice something peculiar in the sidebar, in that we've included instructions that save the User Interrupt from >83C4 before our cursor starts and restored it afterward. This was done so that these routines would be compatible with the routines that provide sprites on the Half Bit Map screen.

```
0044  *      BLWP @ACCFP
0045  *      BYTE ROW,COL
0046  *      DATA CLRSIG
0047  * ON EXIT, FLOATING POINT NUMBER IS AT >834A
0048  * (EIGHT BYTES IN RADIX 100 FORMAT)
0049  *
0050  ACCFP  DATA WS,ACFCOD
0051
0052  ACFCOD LI   R2,15        LENGTH 15
0053         CLR  R5           R5=0
0054         LI   R12,1        R12=1
0055         BL   @GRC         R0 HAS SCREEN ADDRESS
0056         MOV  @>83C4,R9    SAVE EXISTING USER INTERRUPT
0057         MOV  *R14+,R3     R3 HAS CLEAR FIELD FLAG
0058         JMP  ACCST2       JUMP AHEAD
0059  *
0060  * ACCSTR ACCEPTS A STRING INPUT
0061  * INVOKE ACCSTR BY:
0062  *      BLWP @ACCSTR
0063  *      BYTE ROW,COL
0064  *      BYTE MAXLEN,CLRSIG
0065  *      DATA BUFFER ADDRESS
0066  *
0067  * ON EXIT, STRING IS AT BUFFER ADDRESS AS LENGTH,CONTENT
0068  *
0069  ACCSTR DATA WS,ACSCOD
0070
0071
0072  ACSCOD BL   @GRC         R0 HAS SCREEN ADDRESS
0073         MOV  @>83C4,R9    SAVE INTERRUPT STATE
0074         MOVB *R14+,R2     R2 HAS MAX LEN
0075         SRL  R2,8         RT. JUST LEN
0076         MOVB *R14+,R3     R3 HAS CLEAR FLAG
0077         SRL  R3,8         RT. JUST CLRSIG
0078         MOV  *R14+,R5     R5 HAS ADDRESS OF BUFFER
0079         MOV  R2,R2        CHECK FOR ZERO LENGTH
0080         JEQ  EMEX         EXIT IF SO
0081         CI   R2,80        CHECK MAX ALLOWED
0082         JGT  EMEX         IF GREATER, EXIT
0083  ACCST2 CLR  @INSFLG      NOT IN INSERT
0084         CLR  @KEYADR      KEY-UNIT 0
0085         MOV  R10,R10      CHECK POSITION ERROR
0086         JNE  EMEX         IF ERROR, EXIT
```

(See Page 14)

THE ART OF ASSEMBLY —

(Continued from Page 13)

```
0087          MOV  R0,R7          SAVE START POSITION
0088          MOV  R2,R4          SAVE LENGTH
0089          A    R2,R0          ADD LENGTH
0090          CI   R0,>1B00       CHECK END OF SCREEN
0091          JLT  SAVEND         IF LESS, OKAY
0092  EMEX    CLR  R8             EMERGENCY EXIT
0093          MOV  R5,R5          CHECK R5 FOR 0
0094          JEQ  EMEXX          IF SO, SKIP
0095          MOVB R8,*R5         ELSE NULL THE STRING
0096  EMEXX   MOV  R9,@>83C4      PUT BACK OLD USER INTERRUPT
0097          RTWP                RETURN TO CALLER
0098  SAVEND  MOV  R0,R6          SAVE THAT POSITION
0099          DEC  R6             LAST ALLOWED
0100  CLRSNS  MOV  R7,R0          BACK TO START
0101          MOV  R3,R3          CHECK SIGNAL
0102          JEQ  KEYFRC         IF ZERO, JUMP
0103          MOV  R4,R2          GET LENGTH BACK IN R2
0104          MOVB @ANYKEY,R1     SPACE CHAR
0105  CLRFLD  BLWP @VSBW          WRITE ONE SPACE
0106          INC  R0             MOVE AHEAD ONE
0107          DEC  R2             DEC COUNT
0108          JNE  CLRFLD         IF NOT 0, RPT
0109          MOV  R7,R0          GET START BACK
0110  *
0111  * KEYFRC GETS THE CURRENT CHARACTER
0112  * FROM THE SCREEN, FORCES THE CURSOR
0113  * TO THAT POSITION, THEN ACTIVATES THE
0114  * USER INTERRUPT TO BLINK CURSOR
0115  *
0116  KEYFRC BLWP @VSBR           READ BYTE AT R0 POSITION
0117         MOVB R1,@ALTKEY      PLACE AT ALTKEY
0118         MOVB @CURSOR,R1      PUT CURSOR IN R1
0119         BLWP @VSBW           WRITE CURSOR
0120         CLR  @>8378          CLEAR TIME COUNTER
0121         MOV  @INTLOC,@>83C4 ENABLE USER INTERRUPT
0122  *
0123  * KEYIN IS THE PART THAT GETS KEYSTROKES
0124  *
0125  KEYIN  BLWP @KSCAN          SCAN KEYBOARD
0126         LIMI 2               ALLOW INTERRUPTS
0127         LIMI 0               STOP THEM
0128         CB   @STATUS,@ANYKEY KEY STRUCK?
0129         JNE  KEYIN           IF NOT, REPEAT
0130  *
0131  * FOLLOWING CODE USES THE KEYSTROKE
0132  *
0133         MOV  @KEYADR,R8      KEY AS WORD IN R8
0134         MOVB @ALTKEY,R1      OLD CHAR IN R1
0135         BLWP @VSBW           WRITE TO SCREEN
0136         CB   @KEYVAL,@ENTERV "ENTER" STRUCK?
0137         JEQ  KEYEX           IF YES, EXIT
0138         CB   @KEYVAL,@BACKUP FUNCTION-S?
0139         JNE  KEY0            IF NOT, JUMP
0140  *
0141  * FOLLOWING IS CODE THAT HANDLES FUNCTION-S
0142  * IT MOVES CURSOR ONE SPOT, THEN GOES TO
0143  * RPTKEY, WHICH DELAYS BEFORE ALLOWING REPEAT
0144  *
0145         DEC  R0              BACK ONE
0146         C    R0,R7           CHECK FOR FIRST POS
0147         JGT  BCKX            IF GREATER, JUMP
0148         MOV  R7,R0           R7 TO R0
0149         B    @KEYFRC         BACK TO MAIN ENTRY
0150  BCKX   B    @RPTKEY         AHEAD FOR REPEAT ACTION
0151
0152  KEY0   CB   @KEYVAL,@FWARD FUNCTION-D?
0153         JNE  KEY1            IF NOT, JUMP AHEAD
0154  *
0155  * FOLLOWING IS CODE THAT HANDLES FUNCTION-D
0156  * IT MOVES CURSOR ONE SPOT, THEN GOES TO
0157  * RPTKEY, WHICH DELAYS BEFORE ALLOWING REPEAT
0158  *
0159         INC  R0              POINT AHEAD
0160         C    R0,R6           LAST SPOT?
0161         JLT  FWKX            IF LESS, JUMP
0162         MOV  R6,R0           ELSE POINT BACK
0163         B    @KEYFRC         THEN BRANCH BACK
0164  FWKX   B    @RPTKEY         AHEAD FOR REPEAT ACTION
0165  KEY1   CI   R8,32            COMPARE TO SPACE BAR
0166         JLT  FUNCT           IF LESS, CHECK FOR FUNCT
0167  *
0168  * FOLLOWING HANDLES KEY VALUES 32 AND ABOVE
0169  *
0170         MOV  @INSFLG,R1      INSERT MODE?
0171         JEQ  KEY1A           IF NOT, JUMP AHEAD
0172  *
0173  * FOLLOWING HANDLES INSERT IF IN INSERT MODE
0174  *
0175         C    R0,R6           AT END OF FIELD?
0176         JEQ  KEY1A           IF SO, NO INSERT
0177         MOV  R6,R2           GET LAST POSITION
0178         S    R0,R2           SUBTRACT CURRENT POSITION
0179         LI   R1,TEMSTR       TEMP STORAGE SPACE
0180         BLWP @VMBR           PUT BYTES THERE
0181         INC  R0              POINT AHEAD ONE
0182         BLWP @VMBW           WRITE THERE
0183  DEC0   DEC  R0              BACK TO OLD POSITION
0184         JMP  KEY1A           PUT IN THE KEYSTROKE
0185  *
0186  * FOLLOWING HANDLES FUNCTION KEYS WITH VALUES BELOW 32
0187  *
0188  FUNCT  CB   @KEYVAL,@DELKEY DELETE KEY?
0189         JNE  FUNCT2          IF NOT, JUMP
0190  *
0191  * FOLLOWING HANDLES DELETE WITH FUNCTION-1
0192  *
0193         MOV  R0,R3           STASH AWAY R0
0194         MOV  R6,R2           GET END OF FIELD
0195         S    R0,R2           SUBTRACT CURRENT POSITION
0196         JEQ  NULDEL          IF ZERO, JUMP AHEAD
0197         INC  R0              ELSE POINT AHEAD ONE
0198         LI   R1,TEMSTR       TEMPORARY STORAGE
0199         BLWP @VMBR           READ TO THERE
0200         DEC  R0              POINT BACK ONE
0201         BLWP @VMBW           WRITE TO THERE
0202  NULDEL MOV  R6,R0           GET END OF FIELD
0203         MOVB @ANYKEY,R1      SPACE CHAR
0204         BLWP @VSBW           WRITE A SPACE
0205         MOV  R3,R0           GET OLD POSITION BACK
0206         JMP  KEYFRC          JUMP TO GET NEXT KEY
0207  FUNCT2 CB   @KEYVAL,@INSKEY FUNCT-2 PRESSED?
0208         JNE  FUNCT3          IF NOT, JUMP
0209  *
0210  * FOLLOWING SETS INSERT MODE ON FUNCTION-2
0211  *
0212         INC  @INSFLG         SET INSERT FLAG
0213         JMP  KEYFRC          THEN BACK
0214  FUNCT3 CB   @KEYVAL,@ERSKEY FUNCT-3 PRESSED?
0215         JNE  FUNCTX          IF NOT, JUMP
0216  *
```

(See Page 15)

THE ART OF ASSEMBLY —

(Continued from Page 14)

```
0217  * FOLLOWING ERASES FIELD IF FUNCTION-3 STRUCK
0218  *
0219  ERSFLD MOVB @ANYKEY,R3   SET R3 NON-ZERO
0220         B    @CLRSNS      BRANCH TO CLEAR FIELD
0221  *
0222  * FUNCTION-X OR FUNCTION-E
0223  * OR FUNCTION-8 OR FUNCTION-9
0224  * ALL EXIT PROGRAM
0225  *
0226  FUNCTX CI   R8,10         FUNCTION-X?
0227         JEQ  KEYEX         IF SO EXIT
0228         CI   R8,11         FUNCTION-E?
0229         JEQ  KEYEX         IF SO, EXIT
0230         CI   R8,15         FUNCTION-9?
0231         JEQ  KEYEX         IF SO, EXIT ROUTINE
0232         CI   R8,6          FUNCTION-8?
0233         JEQ  KEYEX         IF SO, EXIT
0234         B    @KEYFRC       ELSE IGNORE KEYSTROKE
0235  *
0236  * FOLLOWING PUTS CURRENT KEYSTROKE ON SCREEN
0237  * THEN MOVES CURSOR TO NEXT SPOT
0238  *
0239  KEY1A  MOVB @KEYVAL,R1    GET KEY VALUE IN R1
0240         BLWP @VSBW         WRITE THAT
0241         INC  R0            POINT AHEAD
0242         C    R0,R6
0243         JLT  KEY1X
0244         MOV  R6,R0
0245  KEY1X  B    @KEYFRC       THEN BRANCH BACK
0246  *
0247  * KEYEX IS THE EXIT FROM THIS ROUTINE
0248  *
0249  KEYEX  MOV  R9,@>83C4     PUT PREVIOUS USER INTERRUPT BACK
0250         MOV  R5,R5         NUMBER INPUT?
0251         JEQ  NUMOUT        IF SO, JUMP
0252         MOV  R4,R2         GET LENGTH
0253         MOV  R6,R0         AND LAST POSITION
0254  RDBYT  BLWP @VSBR         READ A BYTE
0255         CB   R1,@ANYKEY    SPACE?
0256         JNE  RDSTR         IF NOT, JUMP
0257         DEC  R0            ELSE DEC POSITION
0258         DEC  R2            AND CHAR COUNT
0259         JNE  RDBYT         IF NOT ZERO, GO BACK
0260  RDSTR  MOV  R5,R1         GET STRING LOCATION
0261         MOV  R7,R0         AND START POSITION
0262         MOVB @WS+5,*R1+    PUT LENGTH AT STRING ADDRESS
0263         JEQ  KEYXX         IF NULL, EXIT
0264         BLWP @VMBR         READ STRING FROM SCREEN
0265  KEYXX  RTWP               RETURN TO CALLER
0266  NUMOUT MOV  R7,@FAC12     START POSITION TO >8356
0267         BLWP @XMLLNK       USE XML VECTOR
0268         DATA CSN           CONVERT STRING TO NUMBER
0269         MOV  R12,R12       FLOATING POINT?
0270         JNE  FPOUT         JUMP IF SO
0271         BLWP @XMLLNK       USE XML VECTOR
0272         DATA CFI           CONVERT FLOAT TO INT
0273  FPOUT  MOV  @FAC,R1       GET NUMBER
0274         JNE  NUMOX         IF NOT ZERO, OKAY
0275         MOV  R7,R0         GET START POINT
0276         MOV  R4,R2         AND LENGTH
0277         LI   R1,SIXBLN     SIX SPACES
0278         BLWP @VMBW         WRITE THOSE
0279         LI   R1,>3000      '0' IN LEFT BYTE R1
0280         BLWP @VSBW         ECHO 0 TO SCREEN
0281  NUMOX  RTWP               RETURN TO CALLER
0282  *
0283  *
0284  * FOLLOWING IS THE REPEAT-KEY FUNCTION FOR LEFT AND RIGHT
0285  * MOVEMENT OF THE CURSOR
0286  *
0287  RPTKEY BLWP @VSBR         READ CURRENT CHAR
0288         MOVB R1,@ALTKEY    PLACE AT ALTKEY
0289         MOVB @CURSOR,R1    GET CURSOR
0290         BLWP @VSBW         WRITE THAT
0291         CLR  @INSFLG       CLEAR INSERT MODE
0292         CLR  @>8378        CLEAR TIMER
0293         CLR  @>83C4        DISABLE USRINT
0294  *
0295  * THE LOOP STARTING AT RPT1 DELAYS REPEAT MOTION FOR
0296  * 32/60THS OF A SECOND UNLESS KEY IS RELEASED
0297  *
0298  RPT1   BLWP @KSCAN        SCAN KEYBOARD
0299         LIMI 2             ALLOW INTS
0300         LIMI 0             STOP INTS
0301         CB   @KEYVAL,@NOKEY NO KEY?
0302         JEQ  RPTEX         IF SO, EXIT
0303         CB   @>8379,@ANYKEY COMPARE TO 32
0304         JLT  RPT1          IF LESS, JUMP
0305  RPT1A  CLR  @>8378        CLEAR TIMER
0306         MOVB @ALTKEY,R1    GET ALTKEY BACK
0307         BLWP @VSBW         WRITE
0308         CB   @KEYVAL,@BACKUP BACKWARD?
0309         JNE  RPTF          IF NOT, JUMP
0310         DEC  R0            ELSE BACK ONE
0311         C    R0,R7         AT START OF FIELD?
0312         JGT  RPTFA         IF GREATER, OKAY
0313         JEQ  RPTFA         IF EQUAL, OKAY
0314         INC  R0            PUT POSITION BACK
0315         JMP  RPTEX         THEN EXIT
0316  RPTF   INC  R0            AHEAD ONE
0317  RPTF1  C    R0,R6         AT END OF FIELD?
0318         JLT  RPTFA         IF LESS, OKAY
0319         JEQ  RPTFA         IF EQUAL, OKAY
0320         DEC  R0            BACK ONE
0321         JMP  RPTEX         THEN EXIT
0322  RPTFA  BLWP @VSBR         READ BYTE AT CURRENT POSITION
0323         MOVB R1,@ALTKEY    STASH CURRENT CHAR
0324         MOVB @CURSOR,R1    CURSOR IN R1
0325         BLWP @VSBW         WRITE CURSOR
0326  *
0327  * THE LOOP AT RPT2 DELAYS 8/60THS UNLESS KEY IS RELEASED
0328  *
0329  RPT2   BLWP @KSCAN        SCAN KEYBOARD
0330         LIMI 2             INTS ON
0331         LIMI 0             THEN OFF
0332         CB   @KEYVAL,@NOKEY NO KEY?
0333         JEQ  RPTEX         IF SO, EXIT
0334         CB   @>8379,@BACKUP COMPARE TO 8
0335         JLT  RPT2          IF LESS, REPEAT
0336  *
0337  * AFTER 8/60THS, CURSOR ADVANCES ANOTHER STEP
0338  *
0339         JMP  RPT1A         ELSE JUMP BACK
0340  RPTEX  MOVB @ALTKEY,R1    OLD CHAR
0341         BLWP @VSBW         WRITE THAT
0342         B    @KEYFRC       THEN BRANCH BACK
0343  *
0344  * FOLLOWING IS THE "BLINK", DONE WITH USER INTERRUPT
0345  * EVERY 20 60THS, THIS WILL BLWP @CHVECT TO CHANGE
0346  * FROM CURSOR TO CHARACTER OR VICE VERSA
```

(See Page 16)

THE ART OF ASSEMBLY —

(Continued from Page 15)

```
0347  *
0348  USRINT CB   @>8379,@TWENTY TIMER=20?
0349         JLT  INTEX          IF LESS, EXIT
0350         BLWP @CHVECT        ELSE CHANGE CHAR
0351  INTEX  RT                  RETURN TO INTERRUPT HANDLER
0352  *
0353  * CHVECT CHANGES FROM CURSOR TO CHAR AND VICE VERSA
0354  * EVERY 20/60THS OF A SECOND.  (THAT'S 1/3 SECOND)
0355  *
0356  CHVECT DATA INTWS,CHG0
0357  CHG0   MOV  @WS,R0         GET MAIN R0 INTO THIS R0
0358  CHG1   BLWP @VSBR          READ CURRENT BYTE FROM SCREEN
0359         CB   R1,@CURSOR     IS THAT CURSOR?
0360         JEQ  CHG2           IF YES, JUMP
0361         MOVB @CURSOR,R1     ELSE GET CURSOR
0362         BLWP @VSBW          AND WRITE THAT
0363         JMP  CHGX           THEN EXIT
0364  CHG2   MOVB @ALTKEY,R1     PUT OLD CHAR IN R1
0365         BLWP @VSBW          WRITE THAT
0366  CHGX   CLR  @>8378         CLEAR TIMER
0367         RTWP                THEN RETURN
0368  *
0369  * HBGKF takes a single keystroke
0370  * AT ROW,COL
0371  * WITH FLASHING CURSOR
0372
0373  * and flashes the cursor
0374  * INVOKE HBGKF by:
0375  *      BLWP @HBGKF
0376  *      BYTE ROW,COL
0377  * ON EXIT, KEYSTROKE IS IN USERS R1
0378  * AND IS ECHOED AT ROW, COL
0379  HBGKF  DATA WS,HBF
0380  HBGK   DATA WS,HBG
0381  HBF    BL   @GRC            SET R0 TO SCREEN ADDRESS
0382         MOV  R10,R10         NO ERROR?
0383         JNE  EMERX2          IF ERROR, JUMP
0384         BLWP @VSBR           READ PRESENT CHAR
0385         MOVB R1,@ALTKEY      PLACE AT ALTKEY
0386         MOVB @CURSOR,R1      CURSOR
0387         BLWP @VSBW           WRITE THAT
0388         MOV  @>83C4,R9       SAVE USER INTERRUPT
0389         MOV  @INTLOC,@>83C4  ENABLE USER INT
0390         CLR  @>8378          CLEAR VDP TIMER
0391         BL   @KEYLOO         WAIT FOR KEYSTROKE
0392         CB   @KEYVAL,@ENTERV ENTER PRESSED?
0393         JNE  HBFX            IF NOT, JUMP
0394         MOVB @ALTKEY,@KEYVAL PUT PREVIOUS CHAR IN KEYVAL
0395         MOV  @KEYADR,R8      KEY AS WORD IN R8
0396  HBFX   MOV  R9,@>83C4       PUT BACK OLD USER INT
0397         JMP  HBGKX           JUMP AHEAD
0398  *
0399  * HBGK accepts a single keystroke
0400  * AND ECHOES AT ROW, COL
0401  * WITHOUT CURSOR
0402  * INVOKE HBGK by:
0403  *      BLWP @HBGK
0404  *      BYTE ROW,COL
0405  * ON EXIT, KEY VALUE IS IN CALLER'S R1
0406  *
0407  HBG    BL   @GRC            SCREEN LOCATION TO R0
0408         MOV  R10,R10         NO ERROR?
0409         JNE  EMERX2          IF ERROR, JUMP
0410         BL   @KEYLOO         GET KEYSTROKE
0411  HBGKX  MOVB @KEYVAL,R1      KEY VALUE IN R1
0412         BLWP @VSBW           ECHO TO SCREEN
0413         MOV  @KEYADR,@2(R13)  PUT IN CALLER'S R1
0414         RTWP                 RETURN TO CALLER
0415
0416  EMERX2 CLR  @2(R13)         CLEAR CALLER'S R1
0417         RTWP                 RETURN
0418  *
0419  * KEYLOO WAITS FOR A KEYSTROKE, THEN RETURNS
0420  * THE VALUE OF THE KEY STRUCK GOES INTO REG 8
0421  *
0422  KEYLOO CLR  @KEYADR         KEY-UNIT 0
0423  KEYLO1 BLWP @KSCAN          SCAN KEYBOARD
0424         LIMI 2               ALLOW INTS
0425         LIMI 0               THEN STOP
0426         CB   @STATUS,@ANYKEY ANY KEY?
0427         JNE  KEYLO1          IF NOT, REPEAT
0428         MOV  @KEYADR,R8      KEY AS WORD INTO R8
0429         RT                   THEN RETURN
0430  * GRC IS INTERNAL SUBROUTINE THAT GETS
0431  * THE ROW, COL FROM CALLER'S DATA
0432  * AND TRANSLATES THAT TO SCREEN ADDRESS IN R0
0433  *
0434  GRC    CLR  R10             CLEAR ERROR FLAG
0435         MOVB *R14+,R0        ROW IN R0
0436         MOVB *R14+,R1        COL IN R1
0437         SRL  R0,8            RT. JUST ROW
0438         SRL  R1,8            RT. JUST COL
0439         DEC  R0              ZERO BASE ROW
0440         DEC  R1              ZERO BASE COL
0441         SLA  R0,5            ROW * 32 IN R0
0442         A    R1,R0           ADD COL TO R0
0443         JLT  GRCERR          IF < 0, ERR
0444         CI   R0,>2FF         CHECK SCREEN LIMIT
0445         JGT  GRCERR          IF >, ERR
0446         AI   R0,>1800        ADD SCREEN OFFSET
0447         RT                   RETURN
0448  GRCERR INV  R10             SET ERROR FLAG
0449         RT                   RETURN
0450  *
0451  * DATA SECTION
0452  *
0453  WS     BSS  32              OUR WORKSPACE
0454  INTWS  BSS  32
0455  INTLOC DATA USRINT          USER INTERRUPT ADDRESS
0456  NUMFLG DATA 0               NUMBER INPUT FLAG
0457  INSFLG DATA 0               INSERT FLAG
0458  DELKEY BYTE 3               FUNCTION-1 VALUE
0459  INSKEY BYTE 4               FUNCTION-2 VALUE
0460  ERSKEY BYTE 7               FUNCTION-3 VALUE
0461  TEMSTR BSS  80              TEMPORARY STRING
0462  ALTKEY BYTE 0               CURRENT CHARACTER FROM SCREEN
0463  ENTERV BYTE 13              ENTER KEY VALUE
0464  CURSOR BYTE 30              CURSOR CHAR
0465  BACKUP BYTE 8               FUNCTION-S
0466  FWARD  BYTE 9               FUNCTION-D
0467  ANYKEY BYTE 32              SPACE OR COMPARISON BYTE
0468  TWENTY BYTE 20              CURSOR BLINK NUMBER
0469  NOKEY  BYTE >FF             NO KEY INDICATION
0470  EDGE   BYTE 31              EDGE CHAR
0471  SIXBLN TEXT
0472         END
```

The 16-slot PEB

Plenty of room for every card you can think of

By BOB CARMANY

Remember, this project follows the usual precautions inherent with any hardware project. Undertake this project at your own risk. If you blow something up, tough! This author assumes no liability for this project nor warrants that it will meet the requirements of your computer system.

Several months ago, I was faced with a perplexing dilemma. I had just finished what was to become the definitive version of the upgraded Quest manager program. The rewrite allowed for up to seven Quest cards to be installed in a TI system. I don't have seven of them but it did lead me to increase the number of Quests in my arsenal. I added a fourth Quest RAMdisk permanently to my system.

Ordinarily, this wouldn't have created a difficult problem. The Flex interface card occupied slot No. 1, the RS232 slot No. 2, the four Quest cards Nos. 3-6, the AMS card slot No. 7, and the disk controller slot No. 8. Suddenly, I was out of space in my Peripheral Expansion Box with other options that I wanted to add. What was I going to do with my MBP clock card and where would I put a SCSI controller? Something had to be done!

There are a couple of alternatives to "daisy-chaining" a couple of PEBs. The first is a not-so-elegant approach that involves a whole lot of soldering and likely colorful language. If you can find a couple of defunct PEB cards with the full complement of 60 separate traces, a bit of ribbon cable with at least 54 strands, and have a lot of patience, you can probably finish the job.

This approach will use one of the slots in PEB No. 1 to link with a vacant slot No. 1 in PEB No. 2. The first task is to cut off the portion of each of the defunct PEB cards that goes into the 60-pin edge card connector in the PEB leaving enough of each trace protruding to solder a strand of the ribbon cable to it. Now, arrange the cards so that pin 1 of each one is at the top away from you and mark it with a bit of nail polish or model paint to avoid confusion. You now have 54 solder connections to make at each end of the cable. Everything is wired straight through except for the six power lines. **These are not connected.** The power lines are pins 1, 2, 57, 58, 59, and 60. Pins 1 and 2 are +5V, pins 57 and 58 are -12V, and pins 59 and 60 are +12V.

After the other 54 pins are soldered at each end, check each of them for continuity with a voltmeter to make sure that there are no short circuits or cross connections. The cable length should be as short as possible. A cable length of 18 inches is approaching the limit. Simply plug one end of the cable into an unused slot (ie. slot No. 7) of PEB No. 1 and the other end into slot No. 1 of PEB No. 2. You now have two PEBs chained together at the cost of one slot and a lot of effort.

ANOTHER METHOD

The second method is the more elegant of the two. For this project you will need a 44-pin female edge card connector (wire mount) and two 44-pin male wire mount connectors. All should be 0.100-inch spacing and gold contacts are recommended. You will also need about 8-10 inches of 44-strand ribbon cable. The connectors are very difficult to find (particularly the male ones) and you might have to check several suppliers to find them.

Again, mark pin No. 1 on each of the male connectors and place them in the same relative position. Crimp one on each end of the cable being careful not to sever the wires. Check each pin with a voltmeter for continuity and look for short circuits and cross-connections. Find the middle of the cable and very carefully crimp the female connector in place. Check everything again.

You have just created the equivalent of a Y-cable for the console that will allow the use of 2 PEBs at one time. The only drawback is that you have to have a Flex card in each box. The female connector plugs into the expansion port on the console or speech synthesizer and each of the male connectors plugs into each of the PEB Flex cables.

I am using a cable constructed by the second method. Besides the gear that I mentioned at the start of the article, I have a MPB clock card, second RS232 card, Speech Synthesizer card, and by the time you read this, hopefully, a SCSI card in PEB No. 2. Everything so far is working very nicely.

The only restriction in having more than one PEB is to make certain that only one device occupies a single CRU address. That is the same restriction that you have with a single PEB, though, so it's nothing new.

There is an alternative to putting in all of the time and effort to make your own cables. That is to buy one ready-made. Ron Markus (RAMCHARGED Computers) was selling the second variety at one time but his cable supplier quit making them because of low demand. Perhaps if a minimum order can be put together, he might be persuaded to make up a batch of them. It's worth a try, anyway.

DDI Software offers new MY-BASIC

DDI Software announces the release of its version of MY-BASIC. According to Jim Uzzell of the company, Version 4.0 has 99 percent of the known bugs of Version 3.0 fixed.

A personally registered copy is available for a $15 registration fee. Registered users will receive a hard copy of 88 replacement pages, which include commands not included in the original manual, new commands, deleted commands and changed commands; five new appendixes: Call Key ASCII Chart for Modes 0-5, Assembly Support Information, Color Palette and Hexadecimal Charts, RS-232 DSR Intro and Disklayout-Floppy; and 17 DDI Software MY-BASIC utility programs. Also, Uzzell says, it is the intent of DDI Software to support registered users.

To order, or for information, in the United States, write James Uzzell, HQ Airsouth, PSC 813 Box 105, FPO, AE 09620; in other countries write him at Airsouth (NATO), Bldg. Q Box 105, Viale Delca Liberazione, 80124 Banoli, Napoli, Italy.

Geneve 9640

Crunching time and date

By JIM UZZELL
DDI SOFTWARE

With the introduction of the Geneve and HFDC cards, time and date was added to each file created. The last time and date it was updated was also added.

To do this efficiently, the time/date information was "crunched" so it would fit into two words (hex values) beginning at hex location 14 of the File Descriptor record.

Crunch means stripping out all bits that have no value, i.e. the year 97 in hex is >61 and in binary format is 0110 0001. We know that the left bit will never be used. That leaves us with seven bits to use in creating the crunched value. (See the October 1994 MICROpendium, page 12, for the format of crunched values.)

The following program is designed (with the help of a small object code file) to read the system time and date then convert it to a crunched value. The program will display the binary value of these crunched values, which will then be separated into components that represent either the time or date.

The program will take the components and convert them into a readable time and date.

The source code of the call load portion of program is included.

TIME&DATE

```
100 !Converted by ASM2MYB fr
om DDI SOFTWARE
110 CALL INIT
120 CALL LOAD(-8352,84,73,77
,68,65,84,37,24)
130 CALL LOAD(8194,38,2,223,
96)
140 CALL LOAD(9460,0,0,0,0,1
65,90,165,90,165,90,165,90,1
65,90,0,0,165,90,165,90,165,
90)
150 CALL LOAD(9482,165,90,16
5,90,0,0,165,90,165,90,0,9,0
,10,200,11,36,244,2,224,240,
0)
160 CALL LOAD(9504,2,0,0,4,2
,1,37,4,44,32,37,20,2,0,0,1,
2,1,36,248,44,32)
170 CALL LOAD(9526,37,20,4,3
2,37,110,37,10,37,18,4,32,37
,134,37,4,37,19,4,32,37,158)
180 CALL LOAD(9548,37,7,37,1
8,4,32,37,178,36,248,37,16,4
,32,37,202,36,251,37,17,4,32
)
190 CALL LOAD(9570,37,236,36
,254,37,16,194,224,36,244,4,
91,254,48,37,114,192,190,192
,254,2,0)
200 CALL LOAD(9592,0,16,44,3
2,37,22,10,17,6,193,212,193,
3,128,254,48,37,138,192,190,
192,254)
210 CALL LOAD(9614,2,0,0,16,
44,32,37,22,10,81,6,193,212,
193,3,128,254,48,37,162,192,
190)
220 CALL LOAD(9636,192,254,2
,0,0,16,44,32,37,22,164,193,
3,128,254,48,37,182,192,190,
192,254)
230 CALL LOAD(9658,2,0,0,16,
44,32,37,22,10,49,6,193,212,
193,3,128,254,48,37,206,192,
190)
240 CALL LOAD(9680,192,254,2
,0,0,16,44,32,37,22,193,1,9,
49,6,193,184,1,37,16,10,84)
250 CALL LOAD(9702,6,196,212
,196,3,128,254,48,37,240,192
,190,192,254,2,0,0,16,44,32,
37,22)
260 CALL LOAD(9724,9,17,164,
193,3,128,165,90)
270 CALL GRAPHICS(3,1)
280 !DDI SOFTWARE
290 !1997
300 CALL LINK("TIMDAT")
310 DIM D$(20),D(21)
320 FOR X=0 TO 7 :: CALL PEE
K(VALHEX("24F8")+X,V(X)) ::
PRINT CHR$(V(X)); :: NEXT X
330 PRINT " TIME" :: PRINT
340 FOR X=0 TO 7 :: CALL PEE
K(VALHEX("2504")+X,V(X)) ::
PRINT CHR$(V(X)); :: NEXT X
350 PRINT " DATE" :: PRINT
360 FOR X=0 TO 5 :: CALL PEE
K(VALHEX("2510")+X,V(X)) ::
NEXT X
370 A$=SEG$(HEX$ (V(0)),3,2)
&SEG$(HEX$ (V(1)),3,2) :: A=
VALHEX(A$)
380 B$=SEG$(HEX$ (V(2)),3,2)
&SEG$(HEX$ (V(3)),3,2) :: B=
VALHEX(B$)
390 PRINT "FILE DESCRIPTOR R
ECORD TIME & DATE" :: PRINT
:: PRINT ">"&SEG$(HEX$ (V(0)
),3,2)&SEG$(HEX$ (V(1)),3,2)
;" TIME "
400 M=1 :: GOSUB 530 :: PRIN
T "    ";SEG$(E$(1),1,5);" "
;SEG$(E$(1),6,6);" ";SEG$(E$
(1),12,5);
410 PRINT "  HHHH HMMM MMMS
SSSS"
420 N$="000"&SEG$(E$(1),1,5)
 :: PRINT " ";N$; :: GOSUB 6
10 :: PRINT
430 N$="00"&SEG$(E$(1),6,6)
:: PRINT " ";N$; :: GOSUB 61
0 :: PRINT
440 N$="000"&SEG$(E$(1),12,
) :: PRINT " ";N$; :: ZM=1 :
: GOSUB 610 :: ZM=0
450 PRINT :: PRINT
460 PRINT :: PRINT ">"&SEG$(
HEX$ (V(2)),3,2)&SEG$(HEX$ (
V(3)),3,2);" DATE "
470 A=B :: M=2 :: GOSUB 530
:: PRINT "    ";SEG$(E$(2),1
,7);" ";SEG$(E$(2),8,4);" ";
SEG$(E$(2),12,5);
480 PRINT "  ";"YYYY YYYM MM
MD DDDD"
490 N$="0"&SEG$(E$(2),1,7) :
: PRINT " ";N$; :: GOSUB 610
 :: PRINT
500 N$="0000"&SEG$(E$(2),8,
) :: PRINT " ";N$; :: GOSUB
610 :: PRINT
510 N$="000"&SEG$(E$(2),12,
) :: PRINT " ";N$; :: ZM=0 :
: GOSUB 610 :: ZM=0
520 END
530 C=2 :: D(1)=A :: IF SGN(
D(1))=-1 THEN D(1)=65536+D(
)
540 FOR X=1 TO 16
550 D0=INT(C*((D(X)/C)-INT(
(X)/C))+.5) :: D$(X)=STR$(D
```

(See Page 19)

TIME AND DATE—

(Continued from Page 18)

```
)
560 D(X+1)=(D(X)-D0)/C :: NE
XT X
570 Y=0 :: FOR X=16 TO 1 STE
P -1
580 PRINT D$(X); :: Y=Y+1
590 E$(M)=E$(M)&D$(X) :: NEX
T X
600 RETURN
610 Q=0 :: FOR X=8 TO 1 STEP
 -1
620 FOR Z=1 TO 2 :: READ Z$
630 IF SEG$(N$,X,1)=Z$ THEN
D0=Z-1
640 NEXT Z
650 P=8-X :: D0=INT(D0*(C^P)
+.5) :: Q=Q+D0 :: RESTORE ::
 NEXT X
660 IF ZM=1 THEN PRINT Q*2;
ELSE PRINT Q;
670 RETURN
680 DATA 0,1
```

SOURCE CODE OF CALL LOAD PORTION OF PROGRAM

```
*Routine to create time and date in File Descripter
Record format
*Can be used for BWRITE create new file routine
*DDI SOFTWARE 1997
       DEF  TIMDAT
       DXOP DDI,0
WS1    EQU  >F000          workspace
WS2    EQU  >FE30             ""
SAVRTN DATA 0              save rtn to mybasic
       DATA 0
TIMBUF BSS  10       <----- /hh:mm:ss buffer must be at
least 10 bytes
       DATA 0        FORMAT   (string length is at buf -
1)
DATBUF BSS  10       <----- \mm-dd-yy buffer must be at
least 10 bytes
       DATA 0              separators are created by xop
routine
*
BUFTIM BSS  2         crunched hex val for FDR  hhhhh mm-
mmmm sssss
BUFDAT BSS  2            "      "  "   "   "   YYYYYYY
mmmm ddddd
*
UTL    DATA >0009          util xop
MTH    DATA >000A          math xop
TIMDAT MOV  R11,@SAVRTN
       LWPI WS1
       LI   R0,4
       LI   R1,DATBUF      get date  i.e. 03-01-97
       DDI  @UTL
       LI   R0,1
       LI   R1,TIMBUF      get time  i.e. 18:50:10
       DDI  @UTL
*start crunch routine
       BLWP @YEAR                YY
       DATA DATBUF+6,BUFDAT
       BLWP @MONTH               MO
       DATA DATBUF,BUFDAT+1
       BLWP @DAY                 DD
       DATA DATBUF+3,BUFDAT
       BLWP @HOUR                HH
       DATA TIMBUF,BUFTIM
       BLWP @MINUTE              MM
       DATA TIMBUF+3,BUFTIM+1
       BLWP @SECOND              SS
       DATA TIMBUF+6,BUFTIM
OUT    MOV  @SAVRTN,R11
       RT                  back to mybasic
YEAR   DATA WS2,GO
GO     MOV  *R14+,R2
       MOV  *R14+,R3
       LI   R0,16
       DDI  @MTH
       SLA  R1,1
       SWPB R1
       MOVB R1,*R3
       RTWP
MONTH  DATA WS2,GO1
GO1    MOV  *R14+,R2
       MOV  *R14+,R3
       LI   R0,16
       DDI  @MTH
       SLA  R1,5
       SWPB R1
       MOVB R1,*R3
       RTWP
DAY    DATA WS2,GO2
GO2    MOV  *R14+,R2
       MOV  *R14+,R3
       LI   R0,16
       DDI  @MTH
       A    R1,*R3
       RTWP
HOUR   DATA WS2,GO3
GO3    MOV  *R14+,R2
       MOV  *R14+,R3
       LI   R0,16
       DDI  @MTH
       SLA  R1,3
       SWPB R1
       MOVB R1,*R3
       RTWP
MINUTE DATA WS2,GO4
GO4    MOV  *R14+,R2
       MOV  *R14+,R3
       LI   R0,16
       DDI  @MTH
       MOV  R1,R4
       SRL  R1,3
       SWPB R1
       AB   R1,@BUFTIM
       SLA  R4,5
       SWPB R4
       MOVB R4,*R3
       RTWP
SECOND DATA WS2,GO5
GO5    MOV  *R14+,R2
       MOV  *R14+,R3
       LI   R0,16
       DDI  @MTH
       SRL  R1,1           DIV by 2
       A    R1,*R3
       RTWP
       END
```

Begining c99 — Part 4

Flow control statements

By VERN JENSEN

One of the nice things about C is that it has many flow control statements that Extended BASIC doesn't have. These statements give you much more flexibility than XB's statements, and often help to make the code much clearer. Last issue we took a look at the **if-else** flow control statement. Now that you have that under your belt, we can take a look at the rest of them: the **for**, **while**, **do-while**, and **switch** statements. We'll also cover the **break**, **continue**, and **goto** statements.

THE WHILE AND DO WHILE STATEMENTS

One of the most basic flow control statements in c99 is the **while** statement. The format of a while statement is:

```
while (expression)
      statement;
```

The first expression is evaluated, and if it is true (non-zero), the statement is executed. The expression is then re-evaluated, and this cycle continues until the expression becomes false (zero). Here's a very basic example of a while statement:

```
while (status != 1)
      keyCode = Key(0,&status);
```

This continually calls the **Key** function (part of the GRF1 library) until the status returned by the function is 1, indicating that a new key was pressed. However, one must remember that **status** is evaluated before the call to **Key** is made in the first place, which means that if **status** was already equal to 1 before this code was encountered, this code would be skipped, since the evaluation would fail before **Key** could even be called. There is a way around this problem, however, and it's by using the **do-while** statement. This statement allows you to execute the **statement** before doing the evaluation. The format of a **do-while** statement is

```
do
      statement;
while (expression);
```

So our code above could be written like this:

```
do
      keyCode = Key(0,&status);
while (status != 1);
```

This would correctly make the call to **Key** before **status** is evaluated, so that it won't matter what value is contained in the **status** variable before this code is executed. Because the statement is executed before the evaluation is made, the statement part of a **do-while** loop will always be executed at least once, even if the expression is false. It is important to note the semicolon that is placed at the very end of the **do-while** statement; you'll get compile errors if you leave it out.

And as you'd expect, you can group multiple statements together for use in the various flow control statements by using braces. For instance, we could write

```
do
{
      keyCode = Key(0,&status);
      HChar(1,1,keyCode,768);
} while (status != 1);
```

Because the statement is executed before the evaluation is made, the statement part of a do-while loop will always be executed at least once, even if the expression is false.

The semicolon at the very end of the **do-while** statement is still necessary even when you use braces to group multiple statements together. And in case you're wondering, let me also say that the **expression** in a **do-while** statement (or just about any other flow control statement, for that matter) is just the same as the **expression** in an **if** statement. That is, anything we talked about last article dealing with the **expression** part of an **if** statement applies here as well. So another example of the while statement would be this:

```
gameOver = 0;
while (!gameOver)
      PlayGame();
```

This code repeatedly calls the function **PlayGame** until the **gameOver** variable is set to true (1).

THE FOR STATEMENT

Now we come to a flow control statement you're most likely already familiar with: the **for** statement. The format of the **for** statement is

```
for (expr1; expr2; expr3)
      statement;
```

which is functionally equivalent to

```
expr1;
while (expr2)
{
      statement;
      expr3;
}
```

An example of a simple **for** statement would be

```
for (chr=65; chr <= 128; chr++)
      HChar(1,1,chr,768);
```

This would fill the screen with all the ASCII characters from 65 to 128. I should mention that the "++" is an "increment operator"; the statement "chr++" is the same as "chr = chr+1", but is more efficient. The equivalent of the example above in Extended Basic would be:

(See Page 21)

BEGINNING C99 —

(Continued from Page 20)

```
FOR CHR=65 TO 128 :: CALL HCHAR(1,1,CHR,768)
:: NEXT CHR
```

In practice, the first expression in a **for** statement is an initializer (such as "**n=1**"), the second expression is an evaluation (such as "**n < 100**"), and the third expression increments the variable used in the loop by a value (such as "**n = n+5**"). However, these expressions don't have to be this way. Each expression can be any kind of statement you want to put there, although you must remember that the first one is only executed once, the second one should be some sort of evaluation, and the third is executed after each pass of the loop. You also have the option of leaving out any or all of the expressions if you so desire; for instance, here's a **for** loop that omits the first expression (although you must leave the semicolon there):

```
for (; n < 100; n=n+5)
      statement;
```

This would be equivalent to:

```
while (n < 100)

      statement;
      n=n+5;
```

You can also add more than one statement to each expression by separating the statements with commas. For instance:

```
for (a=low,b=high; a < b; a++,b-);
```

In this example we have two statements in the first expression, and two statements in the third expression, both separated by commas. We also omit the statement that usually follows the **for** loop, since it isn't necessary in this example. (Notice the semicolon at the end of the **for** statement signifying its end.) This **for** statement assigns the variables **low** and **high** to **a** and **b** and then increments a and decrements b until the two values meet or pass each other.

Whether it is better to use the **for** or **while** statement for a particular situation largely depends on which better suits your needs. For example, if you don't need the first expression in the **for** statement, it would probably be better to use the **while** statement instead. But when you need all three expressions, the **for** will most likely be the best choice, since it places all elements of the loop in a common location making it easier to read.

THE SWITCH STATEMENT

Next we come to the switch statement, a multi-way decision that compares an expression with any number of integer constants. (A constant is a value that doesn't change, such as a number; a variable is not a constant, since its value can change. An integer constant is a constant value that must follow the same rules applied to integer variables; that is, the constant must be a whole number between -32768 and 32767.) The format of the **switch** statement is

```
switch (expression)
{
      case constant:
            statements
```

(See Page 22)

BEGINNING C99 —

(Continued from Page 21)

```
        case constant:
              statements
        default:
              statements
}co
```

The **expression** is the same as what we've already discussed for the **if** statement; it can be a variable or a relational expression such as "**a > b**". The result of the expression (**true** or **false**) will be compared with each case. There can be as many or as few cases as you want, and the default case is optional. Execution starts with the case that matches the expression, or with the default case (if there is one) if no matching cases can be found. Execution continues until the **break** statement or the end of the **switch** statement is encountered.

Below is an example of the **switch** statement that compares a variable with eight constants. In this case, the constants are "character constants", such as 'A', which, as I've mentioned in the past, are translated by the compiler into the corresponding ASCII value when the program is compiled ('A' is translated into 65). Character constants are still valid integer constants, so they can be used in the **switch** statement.

```
switch (myKey)
{
      case 'E': case 'e':
            MoveUp();
            break;
      case 'S': case 's':
            MoveLeft();
            break;
      case 'D': case 'd':
            MoveRight();
            break;
      case 'X': case 'x':
            MoveDown();
            break;
}
```

We opted not to use a **default** case for this example, since it isn't needed. We use two cases for each section of code, which is perfectly legitimate; you don't need to have code after each case. (Remember that execution starts at the code following the matching case and continues until the break statement is encountered.) Cases are simply "labels", which means that when a matching case is found, the code beneath the case is executed without delay, regardless of how many cases, or labels, are between the matching case and the actual code.

Each of the cases, including the **default** case, can be in any order; you could put the **default** case at the top if you wanted to, or in the middle. However, it's generally common practice to put it at the end. (Just like the **else** part of the **if-else** statement is always at the end.)

It is important to remember to put in **break** statements after each section of code when you want it to stop. If you don't, the execution will "fall through" to the code intended for the next case. Consider the example above; if no break statements were used, and myKey were equal to 'E', the execution would start at that case and continue through all the code, calling each of the four functions, since a **break** statement would never stop it. This means you don't have to use braces to group together more than one statement, but do have to remember to put the **break** statement at the end of each group.

Every now and then you may run across a situation where you find it desirable to write code that "falls through" to the code intended for the next case as well, but usually you should avoid this practice, since you can easily introduce bugs into your code when adding new cases, forgetting that your other cases will fall through into it.

Every now and then you may run across a situation where you find it desirable to write code that "falls through" to the code intended for the next case as well, but usually you should avoid this practice, since you can easily introduce bugs into your code when adding new cases, forgetting that your other cases will fall through into it. If you ever do write code that falls through, make sure to comment it well, so that if you need to modify it at a later date, you aren't as likely to introduce problems. The example above where I use more than one label for each statement isn't considered "falling through", since there are no statements between the extra cases.

As you may have guessed, most anything you could write with a **switch** statement could also be written using the **if-else** statement. Each have their advantages and limitations. With **if** statements, you can compare an expression or variable with anything; you aren't limited to constant integers, and **if** statements are often more compact. However, if you find that you need to compare a single variable or expression with a number of constants, you may find that the switch statement is more desirable. In case you're curious, this is what the example above would look like if written using the **if-else** statement:

```
if (myKey == 'E' || myKey == 'e')
      MoveUp();
else if (myKey == 'S' || myKey == 's')
      MoveLeft();
else if (myKey == 'D' || myKey == 'd')
      MoveRight();
```

(See Page 23)

BEGINNING C99 —

(Continued from Page 22)

```
else if (myKey == 'X' || myKey == 'x')
     MoveDown();
```

This may appear to be more compact than the **switch** statement, but I'll bet it's less efficient. My C manual doesn't say, but my guess is that since the expression in a **switch** statement is compared with constant values that are known at compile time, the compiler can take advantage of the situation and write code that is "smarter", and finds the matching case faster, without comparing each case one-by-one. If this is true, then the **switch** statement would be somewhat similar to XB's ON GOTO statement, where the code

```
ON X GOTO 100,200,300
```

would be more efficient than

```
IF X=1 THEN GOTO 100 ELSE IF X=2 THEN GOTO
200 ELSE IF X=3 THEN GOTO 300
```

THE BREAK AND CONTINUE STATEMENTS

The **break** and **continue** statements don't provide new ways of doing loops or making decisions, but rather, these statements are to be used with the flow control statements we've already covered, such as the **while** or **for** loops.

The **break** statement lets you immediately exit a **for**, **while**, or **do** loop, and can also be used to exit a **switch** statement, as you've already seen. This allows you to exit a loop at any time, without having to reach the test at the top or bottom of the loop. Here's an example:

```
char n, myArray[100], foundNum = 0;

for (n=0; n < 100; n++)
{
      if (myArray[n] == 150)
      {
            foundNum = 1;
            break;
      }
}
```

This example cycles through all elements of an array searching for the number 150. If the number is found, the variable **foundNum** is set to true, and the **break** statement is used to exit the **for** loop, since we don't need to cycle through the rest of the array now that we know the result of our search. Here's the same thing in XB:

```
100 DIM MYARRAY(100) :: FOUNDNUM=0
110 FOR N=0 TO 99 :: IF MYARRAY(N)=150 THEN
FOUNDNUM=1 :: GOTO 130
120 NEXT N
130 ! PUT REST OF PROGRAM HERE
```

I should mention that the **break** statement only exits the innermost enclosing loop; if you have two nested loops and use the **break** statement inside the innermost loop, that will cause an exit from that loop only, but the outer loop will continue, such as in the example below, which cycles through a two-dimensional array, again searching for the value 150. This time, when the number is found, the number 1 (true) is stored in **myArray[a][0]**; otherwise, this element is set to 0 (false). Keep in mind that this is just a code "snippet"; I haven't included the part that actually fills the array with values in the first place, since that part isn't necessary for you to understand the point I'm making.

```
char a, b, myArray[10][100];

for (a=0; a < 10; a++)
{
      myArray[a][0] = 0;
      for (b=1; b < 100; b++)
      {
            if (myArray[a][b] == 150)
            {
                  myArray[a][0] = 1;
                  break;
            }
      }
}
```

This may be a little confusing, so let me also give you an XB version:

```
90 DIM MYARRAY(9,99)
100 FOR A=0 TO 9
110 MYARRAY(A,0)=0
120 FOR B=0 TO 99
130 IF MYARRAY(A,B)=150 THEN MYARRAY(A,0)=1
:: GOTO 150
140 NEXT B
150 NEXT A
```

Of course, there are times when you'll want to be able to exit all enclosing loops; not just the innermost loop. This can be accomplished with c99's goto statement, which is covered later on in this article. I should also mention that even though the **break** statement in the example above is "enclosed" in an **if** statement, it will still exit properly from the enclosing **for** loop. This is because the **break** statement only affects the **for**, **while**, **do**, and **switch** statements, but has no effect on **if** statements; it just acts as if the **if** statement isn't even there, and exits the **for** loop as desired.

The **continue** statement is similar to the **break** statement, but instead of causing an exit from the innermost enclosing loop, it causes the next iteration of that loop. Therefore the **continue** statement can be used in the **for**, **while**, and **do** loops, but not in the **switch** statement. (Although if you use it in a **switch** statement that is enclosed in a loop, it will cause the next iteration of the loop.) Here's an example of the **continue** statement where it is used to skip the 50th element of an array, but process all other elements:

```
for (a=0; a < 100; a++)
{
      if (a==50)
            continue;

      for (b=1; b < 100; b++)
      {
            if (myArray[a][b] > 99)
                  myArray[a][b] = 99;
```

(See Page 24)

BEGINNING C99 —

(Continued from Page 23)

```
        }

        myArray[a][0] = a;
}
```

Of course, this could've been written like this instead, using an **if** instead of a **continue**:

```
for (a=0; a < 100; a++)
{
        if (a != 50)
        {
                for (b=1; b < 100; b++)
                {
                        if (myArray[a][b] > 99)
                                myArray[a][b] = 99;
                }

                myArray[a][0] = a;
        }
}
```

However, we then have to indent everything a level and add another set of braces, which could be undesirable in some situations, especially if the code being indented is quite long and complex. Therefore, the **continue** statement can be used at times during loops to keep them cleaner than they would otherwise be, and is often simply more convenient than adding another **if**. It can also be used anywhere in the loop; you don't have to put it at the beginning of the loop like I do in the example above. This means the **continue** statement can be used to start the next iteration of the loop in places where it would be difficult to do the same thing using other methods.

THE GOTO STATEMENT

The goto statement is used to branch to a label that is in the same function as the goto statement. You should avoid using the goto statement, since it often makes the code harder to read. (Basic programmers are all too familiar with this fact.) However, there are times when the statement can help. One example would be if you want to jump out of a nested loop. Here's an example:

```
for (a=1; a<100; a++)
        for (b=1; b<100; b++)
                if (myArray[a][b] == 150)
                        goto exit;
exit:
        DoOtherStuff();
```

This example may have been a little clearer if I had used braces, but I decided not to, since there was only one statement for each "group", so this is a perfectly legitimate way to write the code. In any case, this code cycles through all elements of myArray, searching for the value 150. If it is found, the statement "goto exit;" transfers control to the label "exit:". Labels are familiar to those of you who are assembly programmers. For you XB programmers, a label is similar to a line number: going to a label starts executing the statements immediately following the label.

Label names follow the same rules applied to variable names: they must start with a letter, only the first six characters are used, and you can't rely on capitalization to distinguish between two labels of the same name. Keep in mind that your label names shouldn't conflict with any other names, such as function or variable names. Put a colon after the label name, so the C compiler knows that it is a label.

Any code that is written using the goto statement can be written without it, although sometimes at the expense of extra comparisons. For instance, the example above could have been written as:

```
exitLoop = 0;
for (a=1; a<100 && !exitLoop; a++)
        for (b=1; b<100 && !exitLoop; b++)
                if (myArray[a][b] == 150)
                        exitLoop = 1;
DoOtherStuff();
```

Obviously, the goto method would be more efficient in this case, since it is then not necessary to check the **exitLoop** variable each pass through the loop. However, there are few times when the goto statement provides a more efficient way of doing things than other methods. As a rule of thumb, avoid the goto whenever possible, using it only in situations where it is more convenient or helps the code run faster, and doesn't make the code much harder to read.

MORE ON THE IF STATEMENT

Last installment, when discussing how in c99 you can assign the result of a relational expression to a variable (such as "result = a<b;"), I expressed doubt about whether one could do the same thing in Extended Basic. Bruce Harrison, our local Assembly Language guru (who also knows a thing or two about Extended Basic) was quick to write in and point out that Extended Basic can do the same thing.

There are only two differences. The first is that you must enclose the relational expression in parentheses. The second is that in XB, the "true" value is -1, rather than 1. Here are some examples Bruce sent that you can try in XB's Command mode:

```
A=1 :: N=(A=1) :: PRINT N <ENTER>  (prints -1)
```

Now hit FCTN-8 and modify this command to:

```
A=1 :: N=-(A=1) :: PRINT N <ENTER>  (prints 1)
```

Hit FCTN-8 again and change it to this:

```
A=5 :: N=(A=1) :: PRINT N <ENTER>  (prints 0)
```

You can also use the operators AND and OR within the expression, but more parentheses are needed. [e.g. **N=-((A=1) AND (B=0))**] This also works using the other relational operators, such as >, <, >=, <=, etc.

CONTACT INFORMATION

If you have any questions or comments, feel free to email me at Jensen@lafn.org, or write to me at Vern L. Jensen, 910 Linda Vista Ave., Pasadena, CA 91103. The email address I gave out in the first two installments will also still work; I'm just giving out the Jensen@lafn.org address instead because I "lost" a few messages at the other address. The U.S. Postal Service isn't the only place that loses mail!

Making the change from jumper to switch

By RICHARD LINDWAY

This article first appeared in the CPUG Newsletter.—Ed.

This article should be especially useful to hard drive users who use a Myarc hard and floppy disk controller, however, it may be of use to anyone using a disk drive as well. Since this is a hardware modification, all the necessary precautions should be taken against static damage. If you are not sure what you are doing, don't do it or ask someone for advice.

For readers without a hard drive, here is some background as to why I made this change. If there is a diskette in drive No. 1 with a program called LOAD, when you select Extended BASIC this program will load and run. This also will happen with a hard drive except that the program name must be WDS1.DSK1.LOAD. This is a nice feature but it presents a small problem if you must run software from floppy DSK1, like Advanced Diagnostics by Miller Graphics, or many of the adventure games. If you know the disk name you can get around this problem by using the disk name in the program name — OLD DSK.DISKNAME.LOAD. But the disk name must be known, and how many users know the disknames of all their disks? I know I don't.

Another way around it is to put the diskette in another drive, OLD DSKx.LOAD to load the program, put the diskette into drive 1 and run the program. This, of course, assumes you have more than one diskette drive. Well, I don't like all this disk swapping. It is the major reason I bought a hard drive in the first place.

One other solution would be to make the diskette drive itself switchable from one drive to another; DSK1 to DSK3 or DSK4. By changing the jumper block, of course the drive designation is changed, but it requires the drive be removed from the Peripheral Expansion Box to move the jumper block. Finally, this brings me to the reason for this article. Why not replace the jumper block with a switch?

Even though this is a rather simple project it does require some technical knowledge and expertise. You will need a single-pole, double-throw switch and about two feet of small gauge, insulated wire. It's advisable not to use enamel-insulated wire as it could be scratched off and cause a short. A suitable switch from Radio Shack is part number 275-624.

First locate the jumper block where drive selection is made. This is usually two rows of four pins located near the rear of the drive. This isn't true on the original TI drives (Shugart SA-100), and some other drives but it will be true on many half-height drives. One of the rows of four pins will be "tied" together, or "commoned" together through the traces on the circuit board. You can determine which row it is using with ohmmeter. Do not use the battery light bulb type continuity tester — the current needed to illuminate the light bulb may be excessive and damage the drive. It may be possible to determine visually which row is common. If you are not sure, do not guess, enlist the help of a friend. This is the most important part and must be right.

Once you have the correct pins, determine where the switch is to be located, and the route the wires will follow from the switch to the pins, watch for moving parts such as the drive motor, drive belt and the door linkage. Drill the proper size hole for the switch and mount it. The center terminal of the switch will be connected to the commoned row of pins. I suggest you not solder the wires to the pins themselves but to the PC board. This way you can go back to using the jumper block later if you wish.

The other two switch terminals will go to the pins for the drive designation you wish to select. First, determine which two drive numbers you want the drive to be. Any two drives, 1-4, except for any existing drive(s) already attached to the system. Next determine which pin is used for each drive designation and solder a wire to each pin. There are usually numbers near the pins to help. (See Fig. 1.)

I used DSK1 and DSK4 for this example but you will use the drive designation you have chosen. Once everything has been assembled, triple-check it a few times and reinstall it in the PEB. It's not a good idea to change drive designation while the drive is running.

Now that you have finished, here's a way to use it. Insert a disk with a LOAD program and switch the drive to the alternate number. OLD DSKx.LOAD and switch to drive 1 and run the program.

Lucie Dorais writes PC version of Chainlink Solitaire

This comes from Lucie Dorais:

For all lovers of Walt Howe's Chainlink Solitaire game: If you have a PC with Windows (3.1 up), you can now play your favorite game on it. This is a total rewrite (with Walt's permission: he was the beta-tester!) but the game is the same, with lots of new features allowed by the speed of the PC. You can get the 30-day trial version from Compuserve (go WINFUN, lib 2, Card Games, look for CHAINLNK.ZIP) or from Lucie Dorais's home page on the Web (http://www.cyberus.ca/~ldorais/mycat.htm). After 30 days of playing you must register ($15 US).

If you don't have access to the Web, you can send $15 to the author: Lucie Dorais, 603-222 Guigues, Ottawa, Ont., Canada K1N 5J2. You will get a registration key. Unfortunately, I cannot send the demo version through the mail, as I am disabled and cannot make too many trips to the post office.

Sydney BBS shuts down

TEXPAC, the bulletin Board system of TIsHUG, the TI users group in Sydney, Australia, closed down March 2 after 12 years and 7 months of operation. In January and February of 1997, it received eight minutes worth of calls from two users, according to the *TIsHUG News Digest*, but in its history it handled 34,000 calls.

Sending true ASCII codes using Telco

*We are not certain who wrote this article, but it appeared in TI*MES, the newsletter of the TI User Group United Kingdom.—Ed.*

The ASCII codes sent by TI-Writer, unless you do some really clever file editing, send out its files in DIS/F IX 80 with a header and a footer file which is confusing to other computers. Also, there are TI commands which are in text. So, when you send the file as a direct D/V80, the file goes but when read by a text editor from another computer looks like this:
TIFILES88uuuuuuuu\uuuuuu\uu\u\u

this a test0file
uuuuffufufu (.these are control codes)

The file has bad format and words sometimes get lost. We get over this by sending the file as an ASCII-coded file from Telco. This allows the reader to see your file as you intended. However, if you do not set up your file transfer properly, then it will still be set for 80 columns with CR (carriage return) markers still in the file. This can be read but it starts the new line too early and is not padded out to 80 columns. Remember, TI-Writer uses DISplay/VARiable as its format. This means the lines of text are saved to the exact length of your text. How do we get around this?

We use Telco to strip the CR/LF commands from the send file and force it to send only CR at the end of each line.

First start Telco. Choose Set Up Options from the main menu. Choose File Transfer setup. From this menu set the following defaults:

A)	Abort Downloads	DISCARD
B)	Default Error Check	CRS
C)	Echo Locally	Off
D)	Blank Lines	Off
E)	Character Pacing	000
F)	Line Pacing	010
G)	Pace Character	000
H)	Strip Leading Space	Off
I)	Line by Line Feed	Off
J)	Send At End Of Line	CR
K)	CR Translation	Strip
L)	LF Translation	Strip

Save these changes before continuing. Now you can send your text file to any BBS and it will be able to read your work. You do this by using the following five steps:

1. When cued by your BBS to send the file, press FCTN-6.
2. Choose the ASCII option and press Enter.
3. Type the full name of your text file, i.e. DSKn.FILENAME, then press Enter.
4. Your file will then upload.
5. If you watch the bottom line of the file transfer you will see the line counter. The process ends when you see "Transfer Complete Press Any Key." The last line will be shown by Telco and the cursor will appear on the next blank line. To signal the End of File (EOF), type ### and press Enter.

That is all there is to uploading an ASCII file.

When you use Telco, this form of upload must be used if you want to put files now onto a PC-based BBS.

TI99/4A computer repair

Users can fix many common problems on their own

By GARY W. COX

This article appeared in Bug Bytes, the newsletter of the TI user group in Brisbane, Queensland.—Ed.

With some TI99/4As more than 15 years old they are bound to start having problems. In this article I will discuss options on repairing your TI and I will discuss many ways to do repairs yourself.

The best and easiest way for repairs is to send the equipment to one of the following repair centers:

TI and Myarc products:
Cecure Electronics Inc.
c/o Don Walden
P.O. Box 132
Muskego, WI 53150
1-800-959-9640 (orders only)
(414) 679-4343
BBS (414) 422-9669

Cecure Electronics is a Texas Instruments and Myarc authorized repair center.

Corcomp Products:
99 Computer Repair
c/o David Lynch
2101 W. Cresent Ave., Unit A
Anaheim, CA 92801
(714) 539-4834

Before you read on any further let me say that any of the following procedures that I discuss you do at your own risk. While many repairs are easy, if you are not electronically oriented it may be best to send your equipment to a repair facility rather than attempt a repair yourself.

COMMON PROBLEMS

Many problems associated with the TI99/4A are caused by dirty contacts on the module port. Some of the symptoms of dirty contacts include having trouble going into extended BASIC or having frequent lockups while in Extended BASIC or another cartridge.

The problem with the console module port is that the contacts become corroded and need cleaning. One way to clean the module port is to use a pipe cleaner and bend the end into a circle. Then dip the end with the circle into an alcohol-based cleaning solution such as audio and video head cleaner or TV tuner cleaner. Make sure the solution does not leave any

(See Page 27)

REPAIRS —

(Continued from Page 26)

residue behind and that the cleaner does not harm plastic. Then insert the rounded end into the module port connector and pull it straight out. **Do not** move it side to side or it will bend the contacts in the module port. After removing the rounded end, move it over a little and insert it again until you have done the entire module port.

You can also clean the module port by removing the connector from the console, but this entails disassembling the console. However, you may wish to just purchase a new module port connector and replace your old one. L.L. Conner Enterprises sells the 90 degree module port assembly for just $8.

Purchasing a module expander device that fits permanently in the module port will help prevent wear and tear on the module port connector itself as cartridges can plug into the module expander rather than in and out of the module port. Many such devices are available, some of which handle multiple modules.

DISCOVERING THE PROCESS OF ELIMINATION

In repairing your own equipment the repairer (that's you) must follow a procedure to determine what part of the system is defective. This is done by a process of elimination. For example, if you turn on your system and you do not get a display check if the monitor is on. Or, if your monitor is a TV, check if the TV is tuned to the correct station. Check the modulator switch and make sure it is in the correct position. This may sound silly but, as a computer repair technician, I find many problems are simple.

A good rule of thumb is to check the obvious first. Then, if your 4A still doesn't power up, try the system at the basic level by disconnecting the console from the PEB and remove any cartridge that may be inserted. If the console is still dead, the problem is is either the computer, modulator, external power supply, or TV/monitor. If you are using a TV check if the TV works on other stations.

It the TV/monitor works then the problem is either the computer, modulator, or power supply. If you have a black and silver 4A you can see a red light on the front of the console which usually indicates that power is fine. So now you can have either an internal computer problem or the video modulator is bad. Since video modulators go out quite often, that would be a good item to replace.

In repairing your own equipment the repairer (that's you) must follow a procedure to determine what part of the system is defective. This is done by a process of elimination.

If replacing the modulator does not solve the problem, then the problem is in the console, which could be the internal power supply or system board. The console internal power supply is replaceable but the system board will require that the computer be sent to Texas Instruments for repair. It is always good to confirm your diagnosis by having another computer that you can hook up in place of your console to confirm that the problem is actually in the console itself. If your system works after testing a second console, then the problem is definitely in the console.

This process of elimination works on the rest of the equipment, too. If something isn't working in the PEB, try stripping your system down to the basic unit and start testing from there. For example, will your PEB work correctly with another console? A problem which appears to be in the PEB can be in the console. You could take out all cards in the PEB except for the flex cable interface card and see if the flex cable card light will come on when the console is turned on. If the flex cable light comes on, the cable is probably good. Insert an additional PEB card card and see if it works. Be advised that you should always have power off to the PEB when changing a card. Texas Instruments recommends waiting two minutes after turning power off before changing a card to drain latent electrical charges. Keep inserting cards until the problem occurs again and then you have your guess at what is bad and you now need to borrow a like part to confirm your diagnosis.

Note that at the end of this article I will give a list of where to purchase some of the parts that you need for simple repairs.

MORE PROBLEMS

Getting back to actual problems, a very common problem with the TI is the video modulator. If you turn on your TI and do not get a picture or do not get sound it could be the modulator. The easiest thing to do is just try another modulator. This part costs from $5 to $20.

Another problem is with the keys. When the contacts inside of them get dirty they will double-strike or not work at all. The easiest thing to do here is to purchase the keys on a keypad and replace the keyboard unit itself. This unit can be replaced by removing the screws on the back of the console, which allows access to the keypad itself. The keypad is held by several screws and then a ribbon cable connects it to the system board, which can be unplugged. Just unplug the ribbon cable (this is a little tricky) and pull out the keypad and replace the keypad with a new one. Note that it does not matter if you replace a black keyboard with a grey one as the only difference is the color of the keys.

If you do not have access to a keyboard unit you could try cleaning the one that you have. On some of the keyboards the keys can be removed and two contacts can be seen inside of each key which tap together to make contact. Spray some nonconductive cleaner (such as Blue Shower cleaner) inside or TV tuner cleaner (allow to dry before placing the key back on) and be sure the cleaner does not harm plastic. You may need to remove only the key or keys that you are having trouble with. Be sure to turn the console over to allow any excess to drip out and be sure it is cleaner which will air dry (alcohol-based cleaner).

DISK READ ERRORS

If you are having trouble reading disks, such as having disk read errors, the heads on your drive may be dirty. However, be

(See Page 28)

REPAIRS —

(Continued from Page 27)

sure you are trying to read a compatible disk first. For example, a single-sided/single-density drive will only read the first side of a disk formatted as double-sided/single-density. A disk formatted as double-density will not read at all in a single-density drive. Note when I speak of formatting on the TI it is usually referred to as "initializing" a disk.

If you insert a disk and you get disk read errors you could have a dirty head. Just purchase a disk head cleaning kit and try cleaning it. Be sure you purchase a head cleaning kit that uses liquid (do not use dry cleaning kits) and be sure not to run the cleaning disk in your drive dry. Also only use your cleaning disk for the recommended amount of times, as overuse can cause the head to stick to the cleaning media and damage the head. Also, only use a cleaning disk when you are having problems — if it ain't broke don't fix it. Excessive use of cleaning disks can wear out a disk drive's head. If this doesn't fix your problem the disk could be bad, so try other disks. The disk could have been used on a system with it's drive head is out of alignment. Or your drive head could be out of alignment. Both of these problems are best solved by a computer technician. (It may be that repairing a floppy disk drive is more expensive than replacing it. Always get an estimate before authorizing repairs.—Ed.)

WHERE TO FIND PARTS

Where do you find the parts that you need? At one time Radio Shack carried video modulators, keyboards (keys on the keypad) and internal console power supplies. Each of these items said somewhere on the package that it was once for the TI99/4A. Most Radio Shacks are out of these items but you may find one somewhere that still has one in stock.

One place to get replacement keyboards is All Electronics. Their catalog number for the item is KP-48S and it sells for $2 each. Their phone number is 800-826-5432 or (818) 904-0524. At one time they had TI video modulators but are currently out of stock of those items and are not expecting any more.

Another good place for parts is L.L Conner Enterprise at 1521 Ferry St. Lafayette, IN 47901, (317) 742-8146. They carry the cartridge port L connector (this is what the cartridges plug into) for $8. They have video modulators for $15 each, external power supplies for $10 and even various chips necessary for component level repair of TI boards.

Another place to try for various parts is Joy Electronics, 800-527-7438 (outside Texas) or 800-442-3892 (inside Texas).

I highly recommend that everyone have at least an extra video modulator on hand. Most TI owners have at least one backup system as well, since consoles can be purchased used for well under $20. Hamfests are also very good places to pick up parts and used TI equipment at very low prices, not to mention TI fairs.

Printing 9-pin TIPS files to a 24-pin printer

By JOHN PARKEN

This article first appeared in the newsletter of the Cleveland Area TI99/4A User Groups. Parken is a member of TI-CHIPS.—Ed.

If you want to use a 24-pin printer with Page Pro, you'll need to do some sector editing. The same holds true for Certificate 99. Graphic Label Maker gives you options for setup. The code for line spacing is one of these. You can then save this to your working disk. For TIPS 1.8 you can modify a few lines of Extended BASIC code and it will print just fine.

Disk Labeler 99 is also written in XBASIC and can be modified to print to a 24-pin printer. It is not the graphics commands that need changing, but the line spacing.

One of the reasons I mention Disk Labeler is that I have disks with labels that are falling off. Some of the labels have been affixed since 1983, which translates into 14 years. I guess the adhesive was never designed to last forever, though 14 years is a long time. The labels I bought for my inkjet printer do not seem to be of the highest quality in terms of adhesiveness or thickness. If there is writing underneath the label, you can read it through the label. I did shop around for the least expensive labels and I guess that is what I got. Before I do some serious disk relabeling, I will buy free and, I hope, thicker labels. This relabeling will be a good time to weed out some old copies of duplicate or old programs.

Back to what I started to talk about. Any Extended BASIC program like TIPS or DL99 can be listed to a disk file and then searched with TI-Writer for occurrences of CHR$(27);CHR$(65);CHR$(n), which is the line spacing of N/72 of an inch. In most 9-pin printers there are five vertical line spacing commands:

Line space	Command
1/8	CHR$(27);CHR$(48)
7/72	CHR$(27);CHR$(49)
1/6	CHR$(27);CHR$(50
N/216	CHR$(27);CHR$(51);CHR$(N)
N/72	CHR$(27);CHR$(65);CHR$(N)

The program I am looking at is Ron Wolcott's TIPSSHOW, line 260, which includes the following printer command: CHR$(27);CHR$(65);CHR(8). This equates to 1/72 of an inch equals 0.01388 times 8, which equals 0.1111104 of an inch linefeed. In the 24-pin printer, 5/360 of an inch equals 1/72 of an inch. There eight times that would equal 40/360 of an inch which, when divided out, is accurate to five places. This rewrites the line to

(See Page 29)

PRINTER —

(Continued from Page 28)

CHR$(27);CHR$(43);CHR$(40). After making this change, save the program to DSKx.TIPSSHOW. Make sure you use a working copy of the program and not the original. This program does a great job of cataloging the TIPS disk on my Canon inkjet.

Let's do the same thing with TIPS 1.8. We'll look at the main program, which is called TIPSX. I found CHR$(27);CHR$(65);CHR$(08) in the program by listing it to a disk file. I had to save the program to two D/V80 files because it was too large to fit in the TI-Writer text buffer. I did it by saving the first half with LIST "DSK1.TIP1":-1990 and the second half with LIST "DSK1.TIP2":2000-. The XBASIC manual explains how to do this.

Using TI-Writer and the Find String command, I searched for CHR$(27) to find all the printer line spacing commands. Changes needed to be made in only three places: lines 1830, 1910, and 2760. Line 2760 may not be related to label printing and the commands are concatenated. I left 2760 unchanged and my labels printed correctly. But, if you are printing a card, not changing this line could be a problem. I am not sure what function this line goes with.

After making the changes and saving the program, it printed the 1x4-inch labels flawlessly. All inkjet labels come on full 8 1/2 x 11 sheets so, if you run the sheets through one more time and set the tab to 41, you will have all your labels facing the same direction.

One of the reasons I wrote this article with exact line numbers I had changed is that anybody with the TIPS 1.8 program can change it easily and have a 24-pin TIPS program to use. But by knowing how I found what needed to be changed and how to change it, you can change other programs written for 9-pin printers. Just to make it easier, here are what the lines should be:

TIPSHOW

```
260 PRINT #3:CHR$(27);CHR$(4
3);CHR$(40)
TIPS
1830 PRINT #1:CHR$(27);CHR$(
43);CHR$(40)
1910 PRINT #1:CHR$(27);CHR$(
43);CHR$(40);C$
2760 GOSUB 3060 :: FQ$(1)=RR
$&CHR$(27)&"L"&CHR$(88)&Z$ :
: FQ$(2)=CHR$(27)&CHR$(65)&C
HR$(45)&C$&C$&C$&C$&CHR$(27)
&CHR$(43)&CHR$(4)
```

1997 TI FAIRS

APRIL

Fest West '97, April 5, San Jose Civic Auditorium, San Jose, California. Contact Fest West '97 c/o Don O'Neil, 3297 Woody Lane, San Jose, CA 95132, or call (408) 934-0352.

TI-Faire, April 26. Hotel Starkenburger-Hof, Heppenheim, Germany. Contact Volker Papst, Heppenheim, Phone: +49-6252-2903 or Michael Becker, Mannheim, Phone: +49-621-722735

MAY

TI Users Group U.K Annual Group Meeting, May 10, Princess Anne St. Johns Ambulance Training Centre, Trinity Street, Derby, England. Contact Trevor Stevens, 249 Southwell Rd. East, Rainworth, Notts., NG21 0BN, UK, telephone 01623 793077, or BBS 01623 491282 (Friday 7 p.m. to Sunday 10 p.m. only, times are United Kingdom times).

Multi Users Group Conference, May 23-24, Ohio State University, Lima Campus. Contact Charles Good, P.O. Box 447, Venedocia, OH 45894. Phone (419) 667-3131. Preferred e-mail address good.6@osu.edu.

OCTOBER

12th International TI-Tref, Oct. 24-26, Ibis Hotel, Utrecht, the Netherlands. Contact Berry Harmsen, 1e Oosterparkstraat 141 E, 1091 GZ Amsterdam, the Netherlands. Phone: 00 31 20 6941047.

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.

Dutch TIers set October fair

The Dutch TI Users Group (TI-Gebruikersgroep Nederland) will host the 12th International TI-Tref Oct. 24-26 at the Ibis Hotel in Utrecht, the Netherlands.

According to Berry Harmsen of the group, the hotel has special room rates of $40 for a double room for attendees. Telephone for the hotel is 00 31 30 2942066.

For further information write Harmsen at 1e Oosterparkstraat 141 E, 1091 GZ Amsterdam, the Netherlands or telephone 00 31 20 6941047.

Tesch has new email address

Tim Tesch, author of Port and other programs, can now be reached at the following email addresses:
ttesch@juno.com
ttesch@execpc.com

SCSI — MFM — RLL — WHAT? — Part 2

An in-depth look at what SCSI means to TI/Geneve users

By MICHAEL J. MAKSIMIK

The following article first appeared in the newsletter of the Chicago TI User Group.—Ed.

The first article in this series dealt with an overview of the different standards of magnetic storage (see MICROpendium January/February 1997). This time we will concern ourselves with the SCSI standard, because it more closely affects our future as an operable, viable storage option.

SCSI is an interface standard which consists of an electrical level standard, and a command level standard. To understand why the combination of both ofthese options is important, we take a quick look at the ST506 interface standard.

We use this particular standard when communicating with an MFM hard disk or RLL hard disk. (Note that you can use an RLL drive with an MFM controller, but you will use only about 50 percent of the listed capacity of the drive.)

The ST506 consists of control signals (Step, Direction, Drive Select, etc.) and sense signals (Write Protect, Index, Drive Ready, etc.) and data signals (+/- Data In, +/- Data Out). Note the data signals are both positive and negative. Two lines are used each for reading and writing using a current loop interface. This provides a more accurate data transmission interface than a TTL logic interface.

We can use current loop in other communications applications as well — MIDI is one well known current loop interface. This provides a signal which is immune to RF interference, such as that generated in an internal computing environment from motors, circuits, etc. So to read this data, the controller circuitry, that which moves the drive heads, selects which head to read, samples the input lines and places signals on the output lines, is offboard, and is a separate device.

Since the connecting lines, cables, etc. between the controller and the drive are not on the same circuit board, you can introduce erroneous data into the control signals — which are not protected from interference because they are TTL logic — and therefore can generate read errors/write errors and general drive failure.

We can send a message to the drive, which communicates our intentions to the drive, such as the request for drive status. The drive will, in time, respond to our request with a message, and perhaps a request of its own for more information from us.

When SCSI was in the planning stage, it was decided to try a different approach. Instead of having an offboard disk controller controlling several drives, why not place a more intelligent controller on the drive itself, and allow the controller to work like a microcomputer? Give it a command set to control the drive, and allow a message-based system to indicate problems and drive status. Also, allow more than one of these controllers in the same system, and allow these controller/drive combos to control more than one drive. And, at the same time, abstract the concepts of heads, sectors per track, cylinders, write precomp, etc. from the host. All the host need specify when requesting a sector on the drive is to request the actual sector number.

For example, if a drive contains 30,000 sectors, you can specify sector number 20,000 and the drive will do the calculations for you, deciding which head to use, which cylinder to step to, etc. The host should be able to acquire information about the drive: What company made it, what version is the firmware on the drive, what standards it supports, if is ready to operate, how fast it can go, and how many sectors it has available. We should be able to also tell the drive to start or stop its motor, reset itself, and format its media. We let the drive worry about how to accomplish these things. It is not our concern just how the SCSI drive does these things, we just understand that if we stick to the standards the drive will perform properly.

The SCSI standard (1986) outlined the ability of eight total devices on the SCSI bus, the 50-pin (TTL) or 68-pin(current loop) interface. There are eight lines on the data bus, hence it is an 8-bit bus. The newer, wide SCSI standard is a 16-bit bus. Our SCSI implementation using the 53C80 chip is an 8-bit SCSI bus. When we want to select a drive, we activate certain select lines and place one of the data lines active, indicating that the number of the data line is the drive we want to talk to. That drive will eventually respond to our request. For example, it we activate data line 5, that will select device number 5. Once a drive is selected, we open a conversation to the drive.

We can send a message to the drive, which communicates our intentions to the drive, such as the request for drive status. The drive will, in time, respond to our request with a message, and perhaps a request of its own for more information from us. Since there are eight data lines, there can be eight devices on an 8-bit SCSI bus.

We can even assign priorities to the devices by causing the higher-numbered device to take control of the bus if two devices are requested simultaneously.

MULTITASKING THE BUS

On the SCSI bus, each device has a role to play. In fact, disk drives are intelligent enough to initiate a conversation themselves. Frequently, tape backup drives are used to backup a hard drive across a SCSI bus, without intervention by the host computer. We can even start a conversation with a device, ask it to fetch some data, and leave it to do its job without waiting. That way, the bus is free to do other things, so we can start conversations on other devices.

This is multitasking the bus, and allows a host computer to access devices nearly

(See Page 31)

SCSI —

(Continued from Page 30)

simultaneously, allowing different programs running on the computer to share the SCSI bus. In fact, you can have two or more hosts on the same SCSI bus (such as a TI and a Geneve) each with its own SCSI host adaptors, sharing peripherals. If you had a single CD ROM drive you can share the drive between both the TI and Geneve. With a little more cooperation and perhaps a more robust file system which handled record and file locking, we could even share hard drives simultaneously.

A SMALL LAN

I am not saying that the TI couldn't read and use files at the same time the Geneve does. But if both computers try to create data on the drive using the same copy of the bitmap, the files will be cross-linked and will be unusable. Rather a system should finish its I/O operations to the bitmap and directory areas before releasing the the bus to allow another computer to use the drive. But enough of this, lets get back to the issue of sharing the bus.

When we view the SCSI bus as a small LAN, we see that it more closely resembles just that, a local area network. Each device on the bus has enough processing power to run independently. Disk drives are actually DASD, Direct Access Storage Devices. Host computers are actually Processor Devices, as are scanners and other special devices. Tape drives are Sequential Access Storage Devices. Since each device on the SCSI bus is independent, can they control devices of their own? Yes — up to eight devices. These are known as LUs (Logical Units). Each SCSI device on an 8-bit bus can itself control up to eight devices on its own. That way we can have up to 64 devices cooperating on the SCSI bus together.

One device I have worked with, the Teac FC-1, is a floppy controller card. It controls up to three floppy drives. It has a 50-pin SCSI connector, and a 34-pin floppy interface. The floppy interface is identical as a connector to the TI floppy controller card. The Teac FC-1, is about the size of an index card. It can be mounted on a drive or the SCSI card (or even the inside of the power compartment of a Peripheral Expansion Box.

In any case, it allows selection of various drives and density defaults. Using SCSI commands, we can detect whether a drive is present, what type of drive it is, and if there is a disk inside the drive. There are jumpers on the FC-1 which we can use to select the individual drives by using the LU value corresponding to the drive. So, while FC-1 occupies a single SCSI bus address (I assign it address 3 on my system), it can control up to three LUs. CD-ROM jukeboxes work in a similar way. We can have a stack of eight CD-ROM drives with a single controller talking to all of them. The controller uses on SCSI address, and there are eight valid LUs.

When inquiring about a controller's status, we can also determine if any LUs are present, or whether the controller supports any LUs other than its base unit. Most hard drives do not use more than one LU, that is, the controller and the drive respond as LU #0. All devices must have at least one LU, #0. Having multiple CD-ROM drives, or multiple tapes on the system is useful for moving volumes of data between programs, processors, and computers.

LIMITATIONS OF SCSI

You can understand that the SCSI interface is very powerful and offers features that other storage/peripheral interfaces cannot. But what are the limitations of SCSI?

Well, we have only eight data bits, so we can send only eight bits at a time. There is a certain physical speed to data transfers. On the Geneve I have been able to transfer about 550 kilobytes per second. Single speed CD-ROM drives will transfer about 150K per second, which is the minimum speed to do full-motion video. One of the hard drives I used (Seagate ST157N) transferred data at about 350K per second. I have faster drives (Quantums and most drives on the market are very fast) and they meet or exceed the bus limitation of the 55K per second transfer rate.

If the transfer rates are due to SCSI bus overhead, we have an option, defined as SCSI-2, to speed things up. This is known as "Fast SCSI," or synchronous mode. During large data volume requests, we can cut the time needed to transfer the data in half, and we can eliminate the full handshake of the data transfer with the understanding that the devices will return back to normal once the data transfer phase of the conversation is complete. This could save significant time in data transfer speeds, and is wise to do if implementing multimedia video or audio playback to get the best performance.

FUTURE OPTIONS

Currently, we don't implement synchronous transfers on the SCSI card, but that is an option to use in the future. Our transfer rates are slower than the maximum due to TI/Geneve I/O overhead operations. We must buffer our sectors in video memory or wherever, and that takes time to do. However, we can take advantage of synchronous transfers when writing multimedia or data backup programs which would directly interface with the SCSI bus. We can also take advantage of disconnection when doing tape backups so that, during a backup, we can still use the computer to do something useful. One plan is to add an online backup to my BBS, but you would not really know that the backup is occurring because it would allow the SCSI bus (and the BBS itself) to be used while a backup task also runs.

Have you heard of "Ultra-Wide" SCSI? In that adaptation of the SCSI bus, the physical characteristics have changed, but the protocol remains the same. With 16 data bus lines, it can move twice the amount of data in the same bus cycle. It also allows fast synchronous transfers at one-quarter the rate of the SCSI-1 timing standard. Under the SCSI-1 standard we could transfer 5 megabits per second, for example. By doubling the bus width, we can now transfer 10. By using synchronous transfer we can double that to 20 megabits per second, then by using "ultra" fast SCSI transfers, we double that, to 40 megabits per second.

Transfers that fast require very fine wiring, and perfect data connections. So that required a higher density connection. The physical connection itself is an issue. I have seen six types of SCSI connections. First, the 50-pin header connection, similar to thc internal floppy connector on the TI floppy controller. Second, the 50-pin Centronics-type. This looks like a parallel port connector on a printer. Some tele-

(See Page 32)

SCSI —

(Continued from Page 31)

phone linkup systems use this connector and call it a TELCO connector. Third, a 25-pin DB-25 female connector. This is popular but you can confuse this with a PC parallel port and can actually damage either your printer or your SCSI card by accidentally plugging your printer into the SCSI card, or your SCSI device into your PCs parallel port. Some of you have made both types of connectors on your TI/Geneve, especially if you have mounted your PEB in a PC tower case. MARK THEM! Don't get them confused. On the TI RS232 card, the ports are also female DB-25, so label all your ports if you decide to use the DB-25 for your printer, modem and SCSI connections.

The forth type of SCSI connector I have seen is the Apple Powerbook connector, a high density 1 centimeter square connector, which is the best way Apple decided to connect external SCSI hard drives. Go figure. Anyway, there is a fifth, a new high density 50-pin connector, similar to the Centronics as it is impossible to plug in backwards or upside down, and it is shielded and has strain relief/secure clips to hold the connection tight. This is the SCSI-2 connector. You can get adaptors to convert this type of interface to the other types.

There is a sixth type. The SCSI-3 connector is a 68-pin connector, very similar to tlle SCSI-2 connector, but on a SCSI-3 bus. It is the only type of connector used in SCSI-3, and it is used internal and external. This ensures that there is one standard connector, and it also ensures that you don't accidentally plug in the drive and cables backwards. The wiring for SCSI-3 is finer and better protected from interference, making it capable of carrying faster and more precise signal transitions. The SCSI-3 command set is more standard, making device functionality and response much more independent of manufacturer features.

Can we implement SCSI-3 on the SCSI card at this time? No. The Western Horizon Technology SCSI adapter would have to be redesigned to accommodate a different SCSI processor chip.

Can we implement SCSI-3 on the SCSI card at this time? No. The Western Horizon Technology SCSI adapter would have to be redesigned to accommodate a different SCSI processor chip. The 53C80 is designed to support SCSI-1 and SCSI-2, both 8-bit non-differential (single-ended) bus styles. It isn't really important, because there are plenty of SCSI-1 and SCSI-2 peripherals to choose from for the time being.

Flash! The ZIP drive by Iomega has been released in a compact, 3 1/2-inch form factor. That is, it comes with a 5 1/4-inch mounting bracket and faceplate but it can be installed in a 3 1/2-inch slot if you have such a slot in your TI/Geneve tower or desktop Rave expansion box. It also allows full selection of SCSI ID. Older versions of the drive allow only two addresses, 5 and 6, which aren't used by MDOS or the TI EPROM for hard disk drives. With this new drive, however, you can format and use the disks as bootable hard disks on your TI or Geneve as SCS1, SCS2 or SCS3.

ZIP disks are being manufactured by many companies now, allowing the price to drop due to competition. It seems that the 3 1/2-inch floppy drive, like the 5 1/4-inch floppy drive and 8-inch floppy drive, will become the next dinosaur. Disks with 1.44 megabytes running at floppy disk speeds do not compare to 100 megabytes at near-hard disk speeds.

MICRO-REVIEWS

Scanned Images, and TI Module User Guides

By CHARLES GOOD

I remember reading in many TI newsletters 8 or 10 years ago statements along the lines of this: "I'm *never* gonna get one of *those* computers. No, not me. The good old 99/4A is all the computer I will ever need."

Such statements, of course, referred to IBM-compatible computers, which at the time were very expensive and ran very expensive software.

Well, folks, as we all know, times have changed. IBM-compatible PCs have become much more powerful and much, much cheaper. These days, a used 386 system with 4 megs of memory, a 100-meg hard drive and a VGA monitor can be purchased from many sources for about $300, and a state-of-the-art PC can be purchased new for well under $2,000.

Almost all dedicated 99/4A and Geneve users now also have an IBM-compatible at home. All except one local member of my user group also owns an IBM-compatible machine because there are important computing tasks that just can't currently be done with our TI equipment. For example, if you want to use the World Wide Web on the Internet you can't, for now, do this with the good old 99/4A or Geneve. You need a more modern PC. Because of the fact that most 99/4A users also own IBM-compatibles, it is appropriate that I should review some of the 99/4A-related products that run on IBM compatibles. You probably all know about the emulators that can turn an IBM into a 99/4A, but you might be surprised what additional IBM-99/4A software is available. It is this additional software I am reviewing this month. The software I am reviewing comes on IBM formatted disks and requires at least a 386 computer (a 486 or Pentium is much better) running Windows 3.1 or Windows 95.

SCANNED IMAGES
created by Bill Gaskill

Bill has taken some gems from his extensive collection of 99/4 and 99/4A promotional hard copy materials and scanned them using a Paper Point black-and-white whole page scanner. These scanned images can be viewed on an IBM using the Paper Point viewer program that is available in IBM format as freeware. The Paper Point viewer software works with both Windows 3.1 and Windows 95. With this free viewer software you don't need the Paper Point hardware in order to view the scanned images. Just run the viewer software on your IBM and from within the viewer program click on "file" and then click on "open." You can then select from a list of scanned images, each with a file name ending in "max," to view. Once the scanned document page is on screen you can scroll left/right and up/down if necessary to view the whole page. You can also enlarge or shrink the page on screen and you can print the page using any Windows-compatible printer.

The following scanned 99/4 (without the "a") documentation is available. TI distributed these to the public in 1979 and 1980. Each brochure describes one peripheral and includes a picture of a complete 99/4 system with 13-inch monitor and stand-alone peripherals. There are separate brochures for the Speech Synthesizer, thermal printer, sidecar RS-232, telephone coupler modem, stand-alone floppy controller and stand-alone floppy drive. The most interesting thing to me about these brochures is that they all illustrate the never-released and never-discussed Speech Synthesizer cartridge. The little flip-up door on the Speech Synthesizer is designed to accept a cartridge that adds to or replaces the built-in 300-word speech vocabulary. No mention is made of these cartridges in the Speech Synthesizer brochure, which mentions only the Speech Editor command module as the only mechanism available at the time to program speech in BASIC. However, a Speech Synthesizer cartridge is definitely illustrated as part of the 99/4 system shown on each brochure.

Other TI promotional brochures scanned by Bill Gaskill include the power transformer safety notice asking users to obtain a free modification to the original unsafe power supply. Also available are brochures describing the video modulator, the 10-inch color monitor, the 99/4A console, and the Peripheral Expansion Box system.

Also included is a complete list of TI exchange centers. I always thought the exchange centers were great, well above and beyond the normal level of consumer support. I well remember accidentally spilling Pepsi into the top vent slots of my first 99/4A console. This taught me that it is not a good idea to set your can of soda on top of the computer. My console was still under warranty, so I took it to the TI exchange center in Dayton, Ohio, and said, "This doesn't seem to be working any more." They gave me another working console and another new product warranty without asking me *why* the old console ceased to work.

Another item in the Gaskill scanned collection is a newspaper advertisement about the Munch Man offer. This was sort of a rip-off by TI. You got a "free" Munch Man cartridge, "a $39.95 value," if you purchased six other command modules. For many months this was the *only* way to obtain this cartridge. You couldn't find it in any store. It was only available "free."

One of the strangest documents Bill has scanned for us is the May 1985 press release from Tex-Comp announcing their 99/9 computer upgrade. This was in part also published in MICROpendium and shows a photograph of nothing more than a console, monitor and multiple-outlet power center. This was model TC1 and cost $400. The TC2 had a built in CorComp microexpansion system and cost $800. The TC3 had 128K of RAM, for which "no software has yet been written" and cost $1,000. The TC4 was sup-

(See Page 34)

MICROREVIEWS—

(Continued from Page 33)

posed to be a stand-alone 48K coprocessor so that one could run 99/4A software on an IBM computer, and the model 99/9 would have a 40/80-column display and a combination keyboard/trackball/keypad. At the time these sounded great, but I don't think any of these various upgrades were ever sold. At least Tex-Comp never took people's money and then failed to deliver product.

Bill has also scanned in a 1992 Asgard catalog. This is the last of the great Asgard catalogs and is just chock full of fantastic software. The catalog is more than 40 pages long. Reading through the software descriptions, I see lots of products I wish I had. Some of this software has never been reviewed in MICROpendium, and Chris Bobbitt's descriptions make Asgard's software products sound eminently useful and fun. The vast offerings of this catalog remind us of the go-go days of the 99/4 when new great software was coming out all the time.

Last, and certainly not least, Bill Gaskill has scanned most of the old official TI newsletters. This publication, called *Home Computer Newsletter*, was sent out at irregular intervals to registered owners of 99/4 and 99/4A computers. Many of us purchased our 99/4As in late 1982 or 1983 and remember the newsletters TI published near the end of the 99/4A's commercial life. Bill has scanned newsletters from the very first 99/4 (without the "A") newsletter in August 1980 through the March 1982 edition, a total of 11 newsletters. They provide a great trip down memory lane, a look at the early history of user groups and the 99/4, as well as some potentially new practical information even to long-time users of the 99/4A.

Available newsletters are dated August 1980, February 1981, April 1981, May 1981, June 1981, August 1981, September 1981, October 1981, November 1981, January 1982 and March 1982. Bits of information that caught my eye include the following: Active user groups and groups in the process of organizing are listed, as are contact people for these groups. Some very well-known user groups are included in these lists along with a few well-known TIers and many people I have never heard of who were group officers or organizers. Gary Kaplan organized a user group. This is the guy who later published *99er Magazine*, and his magazine is mentioned in the TI newsletters.

The International User Group of Charles LaFara is first mentioned in the February 1981 TI newsletter where it is called a "national" group and claims 3,000 members and a 300-program library. At the time this group did not charge dues. In a later newsletter this group changes itself to "international." The Extended BASIC module is announced in the February 1981 issue and a later issue quotes a suggested price of $99.95 plus tax. Texas Instruments participated in a February 1982 computer expo in Orlando, Florida, hosted by Scott and Alexes Adams. A TI Learning Center opened in January 1982 in the Chicago Merchandise Mart. You could go there to take computer courses for adults and children in various aspects of programming the 99/4A.

The International Home Computer User's Association of Rancho Santa Fe, California (not LaFara's group. Who were they?), according to the January 1982 newsletter, was making a videotape for learning how to program in TI BASIC. Munch Man is actually available to purchase according to the March 1982 newsletter. In the January 1982 newsletter you can only get Munch Man "free" if you purchase four command modules. And on and on and on, a real trip down memory lane.

All this scanned-in documentation and the Paper Port viewing software comes on seven high-density 3.5-inch IBM disks. Bill Gaskill wrote to TI asking their permission to distribute this material and they failed to reply. Bill sees no reason why TI would object to distributing this promotional material so he has given his blessing to its dissemination To get the whole package, send me $7 which includes the cost of media and return postage. Alternatively, you can send me a zip disk and a self-addressed paid return mailer. I'll put the files on your zip disk and mail it back to you.

TI MODULE USER GUIDES
by CADD Electronics

CADD Electronics has scanned in the user guides of all the command modules ever published by TI and is making them available to the TI community as IBM-compatible files. These can be viewed on screen and printed using any computer capable of running Windows 3.1 or Windows 95. These are legal copies of copyrighted TI documentation. TI is paid a royalty for each user guide sold.

Have you ever purchased a command module used with no documentation? How about your collection of disk versions of what was once command module software? I'll bet you don't have documentation for some of these command modules on a disk. Maybe you just lost the book or can't find it. For example, I have piles of TI documentation in my attic, not very well organized. If had to dig up the instructions for the game Hopper I am not sure I could do so easily.

Wouldn't it be great to have all your important TI documentation, including the XB user guide and the Editor/Assembler manual, all neatly organized in one easily accessible place, such as your IBM's hard drive or a small pile of 3.5-inch disks. I recently purchased CADD's entire collection of command module documentation and really appreciate the convenience of being able to easily find and display on my IBM any of this information. For me this was an excellent investment even though I already own hard copies of most of these user guides.

The user guide files are viewed on an IBM compatible using the Acrobat reader software from the Adobe company. This software is free and can be downloaded from the Internet (www.adobe.com) or obtained from CADD on a three-disk set for a $5 copying media and postage fee.

You only need one copy of the Adobe reader to view all the user guide files. There are different versions of Acrobat reader for Windows 3.1 and Windows 95. The Windows 3.1 version will not work with Windows 95 or with Windows 3.0.

(See Page 35)

MICROREVIEWS —

(Continued from Page 34)

Functionally and visually the two versions are almost identical. You select "file" and then "open" to load a document file. Such files have a name ending with ".pdf"; once the file is loaded you can view a single page at a time scrolling up/down and left/right. You can greatly enlarge or reduce the page, making the whole page fit perfectly on screen if desired. You can also display two pages side by side. When moving from page to page you can move to the next or last page by scrolling up/down or you can go to a particular page. Document page numbers that are displayed at the bottom of the screen usually don't correspond to the TI pagination because the document cover is called page 1. A very useful feature of the Adobe reader is the ability to do a string search comparable to "find string" in TI-Writer. This lets you find needed information in a scanned document quickly and is much better than using a printed index. You can also print pages displayed by Adobe reader, so you can, if you want, make printed copies of your most important TI user guides.

The user guides are almost, but not quite, facsimiles of the originals. They have been improved a bit. Unlike in the originals, all major headings have been numbered and these numbered headings have been used to make a table of contents, something lacking in most of the originals. Spelling errors in the originals (yes, there were some) have been corrected and the pages renumbered so that the first text page is page one. The first text page is usually the inside front cover where the quick reference 99/4 and 99/4A keyboard chart was found. In TI's originals this page was usually unnumbered. Printed indexes in some of the originals have been left out because the find string feature is so much better than an index. All of the original graphics have been maintained, and the color covers have been scanned in all their glorious original colors. TI published many of the user guides with several different cover designs. All of the different known TI cover designs are included with each CADD user manual file.

My only complaint about the command module user guide scans is that the pagination listed in the newly created table of contents does not match the pagination displayed by the software. The Acrobat reader lists a "page number" at the bottom of its screen display and it considers the cover to be page one. If there are two covers, thEn the table of contents' page one is considered page three by the Acrobat software. This is kind of confusing.

The IBM file sizes of these module user guide files might surprise you. File size is much more dependent on how many graphics are in the user guide than it is on the number of pages in the guide. Thus, the XB manual and EA manual are relatively small files which can both fit on the same 3.5-inch high density disk with room for a third user guide on the same disk. However, the Video Chess user guide with all its chessboard graphics completely fills a 3.5-inch disk. The Microsoft Multiplan guide is the largest file and requires two disks. The next largest file is the TI Logo II user guide, which will fit on a single 3.5-inch disk but only if the file is compressed.

To date, 123 different command module user guides are available from CADD. Except for the larger files, you can put 4-6 user guides on one IBM 3.5-inch disk. The price for most user guides is $2. A few of the larger user guides cost $3 each. The entire set will set you back almost $260, but of course you don't have to purchase them all. These prices are plus your media and your return postage. You should contact CADD in advance for a price list and an estimate of how many disks will be required. My entire set of user guides fits on 32 3.5-inch IBM disks. If you are purchasing a lot of user guides it is probably easiest to purchase one or two 25-disk packages and send them all to CADD along with adequate return postage and the fee for your user guides. CADD can also put the complete collection of user guides and the Acrobat reader software on one zip disk. If you obtain the free Adobe reader software from CADD instead of from a friend or the Internet then be sure to include CADD's $5 copying fee and specify whether you want the Windows 3.1 or Windows 95 version.

ACCESS

Charles Good (source of Bill Gaskill's scanned TI promotional documentation). P.O. Box 647, Venedocia, OH 45894. Phone 419-667-3131. Preferred e-mail address good.6@osu.edu

CADD Electronics (source of command module user guides on IBM disks). 45 Centerville Dr., Salem, NH 03079. Phone 603-895-0119 or 603-893-1450

Juno may be free, but is it worth it?

By TOM HALL

This article first appeared in Dallas 99 Interface, *the newsletter of the Dallas TI user group.—Ed.*

Motivated by talk of the Internet and e-mail, plus a couple of free copies of Juno, I decided to get brave and try the free e-mail service for PCs called Juno. (Juno software does not work with the TI99/4A or Geneve.—Ed.) Notice I am talking free

REVIEW

e-mail. That is, if you want to send a message to your cousin in New York City, it is free. And the company's programs are free, too. In fact, a user is encouraged to copy programs and pass them along. Incredible! Does the US Postal Service know about this? Will the price of first class postage go down? Will junk mail go away? Shades of Prodigy!

There are several sites offering free Internet e-mail. The North Texas PC User Group BBS has it. There are two companies offering it, that I know of. They are Juno and FreeMark.

How can they do that, you may ask? Here is how — Juno and FreeMark give

(See Page 36)

JUNO —

(Continued from Page 35)

you free e-mail for the opportunity of dumping advertising onto your screen. Both companies have a lot of local access phone numbers to make it easy to dial into their computers. Juno even has toll-free numbers. FreeMark also may have these features, but I haven't tried.

Messages are read and written off-line. You click an icon and your computer will dial Juno/FreeMark and upload your messages — if you have any messages the program will grab those and download them. The program then disconnects. You read and compose messages at your leisure. Juno even has a spell checker.

Sounds great! And it works most of the time.

I received a copy of Juno from Juno On Line Services in New York City. Also, and this was the turning point, my friend Jim Stewart gave me a copy of Juno. The program(s) loaded from a "setup.exe" utility. The install program also interrogates your system to determine what kind of hardware you are using. Then you are given the option of signing up for the free service. Of course, they want to know who you are and your demographics. It is evident they plan to use this information to market their service (advertising).

The instructions are clear and you can't proceed until you answer every single question. Then, the install program dials the Juno computer for the sign up procedure. On a Sunday night, it took me a long time. Juno runs at 9600 baud. Finally, I left my modem in redial mode and left the room. When I returned a message on the screen informed me my password was too easy to guess and would I like to change it. I did, and again tried to get on Juno's computer. Again I left the room. And again when I returned a message awaited me that I had been signed up.

I immediately clicked the "write" icon and started composing messages. This was a waste of time because after a lot of frustration I couldn't get on the Juno computer. So I tried again the next day. The first attempt worked like a charm. I composed a few more messages and again tried to get back on the Juno computer. This time not only did I fail to send my messages, Juno grabbed my phone line and wouldn't release it. After an hour and a half I went to a neighbor's phone and called Southwestern Bell, which released my phone.

Subsequent attempts to connect with Juno have proven difficult, but there have been some messages that got through. One message to a CompuServe user on the East Coast was returned to me. It appeared to me the e-mail address had been garbled by Juno.

In summary, the idea and concept of free e-mail are just great. The Juno motto is "E-Mail Was Meant to be Free." The free software seems to be well thought out. The downside is connecting with Juno. I am hoping that this connection problem is temporary and just a matter of growing pains or something else that can be resolved. Of course, 9600 baud is not state of the art. And, at this moment, I am locked out of my modem because of some signal Juno sent to me.

USER NOTES

Controlling a cassette recorder

Here's a little program with big potential. It was originally published by Jim Peterson in his Tips from the Tigercub column.

This is an oldie, but well worth repeating. You can use it to turn your cassette recorder on and off, to add speech or music from tape to a running program. With the proper hardware, you could write a program to control almost anything from the cassette port. If it doesn't work, reverse the polarity of the remote. The program was written by Ed Hall.

```
100 CALL INIT
110 CALL LOAD(16368,79,70,70
,32,32,32,36,252)
120 CALL LOAD(16376,79,78,32
,32,32,32,36,244)
130 CALL LOAD(8194,37,4,63,2
40)
140 CALL LOAD(9640,2,12,0,45
,29,0,4,91,2,12,0,45,30,0,4,
91,203,78)
150 PRINT "PRESS":" P Play":
"S Stop"
160 CALL KEY(3,A,B)
170 IF B<1 THEN 160
180 ON POS("PS",CHR$(A),1)+1
GOTO 160,190,200)
190 CALL LINK("ON"):: GOTO 1
60
200 CALL LINK("OFF"):: GOTO
160
```

Another fix

This comes from Oliver D. Hébert of Brewton, Alabama. He writes:

Martin Zeddies of Germany (MICROpendium, March/April, p.39) has documented what seems to be a real TI XBASIC bug. Changing the last part of statement 150 from Y=5 :: P=3 to P=3 :: Y=5 will give the expected value for variable P, but the variable Y is now in error.

Martin suggested one fix. Another fix is to rewrite the original statement 150 IF D=0 TEN GOSUB 170 ELSE Y=5 :: P=3 as statement 150 *should* produce the expected values, but we know that the original statement 150 doesn't. Perhaps someone can explain why this error occurs.

Tiny Lotto

The following is a TIny Gram by Ed Machonis.

This screenful of code does a lot of work and once again demonstrates the power of the TI-99/4A. It will generate random numbers for any of the popular lottery games, WIN 3, WIN 4, Pick 6 LOTTO, and WIN 10 or Keno. The low number can be a zero or a one. The high number can be

whatever is being used, 40, 48, 54, 80, 999 or 9999. It should work in any state.

The same RND statement, in Line 4, is

(See Page 37)

USER NOTES

(Continued from Page 36)

used to generate the random numbers for all games, whether the low number be zero or one and whatever high number has been selected.

Where multiple numbers are generated for a game, as in Lotto or Keno, duplicate numbers are discarded and the numbers are sorted in ascending order to make it easier to fill out your bet slip. Output can be directed to screen or printer. When several games are played, the hard copy is easier to check for winners than the individual tickets.

Leading zeroes are inserted where required to keep the columns neatly aligned and to reduce the possibility of transcription errors. A total of 10 Lotto games (the bet slip capacity) can be displayed on the screen without any scrolling off.

TI's User's Reference Guide states on page II-96, "The random number function gives you the NEXT PSEUDO-RANDOM number in the current SEQUENCE of pseudo-random numbers." Page II-95 states: "When the RANDOMIZE statement is used .... a different and unpredictable SEQUENCE of random numbers is generated ..... each time the program is run." RND generates numbers in accordance with a built-in sequence. The RANDOMIZE statement merely insures that a program does not ALWAYS start with the same sequence. But it can, HAS and will.

The RANDOMIZE statement in Line 3 can be placed in three different positions. Placing it before the start of the G loop will cause an unpredictable sequence to be selected each time the program is RUN. Placed before the start of the K loop, a new sequence is used for each game. Placing it after the start of the K loop, as it is, causes an unpredictable sequence to be selected for each number that is generated. As only one number is used from each sequence, we are no longer governed by the built-in sequence and the program generates truly random numbers.

WIN 3 numbers can be selected with TIny LOTTO in one of two ways. We can use a Low Number of 0, a High Number of 999, and one number per game. Or one can use a Low Number of 0, a High Number of 9, and three numbers per game. The same two methods are available for four digit numbers, using 9999 and 1, or 9 and 4, as required. In the first case, a three digit number is selected, in the second case each digit of the three digit number is separately selected. Just use the method that conforms to the way you think of the numbers.

Cassette-only owners can delete the last input statement in Line 2 and the IF THEN statement following it to tailor the program for their system and save answering a useless query. Cassette users will appreciate how quickly the program loads.

TINY LOTTO

```
1 !$$$$ TIny LOTTO $$$$
```

(See Page 38)

USER NOTES

(Continued from Page 37)

```
$      by Ed Machonis $
$ QB-99'ers, Bayside NY $
$$$$$$$$$$$$$$$$$$$$$$$$
!186
2 CALL CLEAR :: INPUT "LOW N
UMBER? ":L :: INPUT "HIGH NU
MBER? ":H :: INPUT "NUMBERS
PER GAME? ":T :: INPUT "HOW
MANY GAMES? ":Q :: INPUT "SC
REEN=0 - PRINTER=1 (0/1)?":P
:: IF P THEN OPEN #P:"PIO"
!210
3 H$=STR$(H):: FOR G=1 TO Q
:: FOR K=1 TO T :: RANDOMIZE
!246
4 N(K)=INT(RND*(H+ABS(L=0)))
+L :: FOR D=1 TO K-1 :: IF N
(K)=N(D)AND H>9 THEN 4 !037
5 NEXT D :: NEXT K :: U=T-1
:: IF H<10 THEN 9 !156
6 F=0 :: FOR K=1 TO U :: IF
N(K)<N(K+1)THEN 8 !033
7 M=N(K):: N(K)=N(K+1):: N(K
+1)=M :: F=1 :: U=K !198
8 NEXT K :: IF F=1 THEN 6 !1
00
9 FOR K=1 TO T :: N$(K)=STR$
(N(K)):: PRINT #P:RPT$("0",L
EN(H$)-LEN(N$(K)))&N$(K)&" "
;:: NEXT K :: PRINT #P:;:;:;
:: NEXT G ! GOOD LUCK! !027
```

SW99ers receive support for Fest West in Lubbock

Texas Instruments has agreed to co-sponsor TI Fest West '98 to be held in Lubbock, Texas. Tom Wills, president of the SouthWest Ninety-Niner User Group in Tucson, Arizona, which will serve as host group for the Fest, received confirmation from Rodney Cates of Texas Instruments that TI had agreed to the proposed co-sponsorship.

TI has agreed to allow the Southwest 99ers to hold the Fest on the grounds of TI in Lubbock, the actual "birthplace" of the TI99/4A.

Wills has asked that if possible some of the original developers be in attendance at the event, tentatively set for Feb. 14. Wills says he will be checking in regard to hotels in Lubbock with the help of Zachary Douglas (zachd@hub.ofthe.net), "a TIer in Lubbock who has helped us bring this dream to reality."

For further information, contact the SouthWest 99ers, P.O. Box 17831, Tucson, AZ 85731-7831, or twills@TheRiver.com.

Permanently set TI-Writer tabs and margins

This item was taken from MUNCH, the newsletter of the Massachusetts Users of the Ninety-nine and Computer Hobbyists. We're not sure who wrote it.—Ed.

If you decide to follow these instructions, make sure you are working with a copy of the software and not your original disk.

TI-Writer can be an insufferable pain sometimes because of the windowing of the 80-column screen. Most of us have learned to use it with 40 columns and let the formatter do the work of making the correct left and right margins and indent. However, being a lazy person, I also hate to set the margins to 40 columns when I go into the editor for the first time. Both the editor in the Editor/Assembler cartridge and the one in TI-Writer default to 80 columns, but you can change this with a sector editor.

The following instructions should be considered general in nature. They may not hold true for your particular version of TI-Writer, though you should be able to gather enough information here to handle your needs.

You will be manipulating the EDIT1 file. It's best to start by copying this file to a newly formatted disk. If your editor files use different names than EDIT, select the first of the two files that make up the editor.

Somewhere around the last sector (No. 32 in my case), you will find a sector that reads in ASCII something similar to what you see in Fig. 1. Place an "I" (indent) and and "R" (right margin) where indicated by the shaded letters. Write it back to disk.

To test, load TI-Writer (or whichever writer program you use), and select Tab. You should see the changes reflected here.

Fig. 1

```
0 2 4 6 8 A C E 0 2 4 6 8 A C E
nt er s ta rt l in e, a nd s to p li ne :&
PU RG E FI LE , ar e yo u su re ( Ye s, o
r No )? . TA BS 12 34 56 78 9 12 34 56 78
9 12 34 56 78 9 12 34 56 78 9 12 34 56 78 9
12 34 56 78 9 12 34 56 78 9 12 34 56 78 9
    0 .. .I .. .. .1 .. .. .. .. .2 .. .. .. ..
.3 .. .. .. ..R4 .. .. .. .. .5 .. .. .. .. .6
.. .. .. .. .7 .. .. .. .. .8 .. .. .. .. .. ..
```

Changing file lengths

The following was written by Andy Frueh of the Lima TI User Group.

If you have a D/V40, 132, or whatever text file that you need to edit with TI-Writer, you can change it to a D/V80 files by using a sector editor. TI-Writer will load only D/V80 files.

Find the header sector of the D/V file you are trying to change. Edit that sector in hex mode. Go to byte >11. If the file is D/V40, you should see the number 28, which is an open parenthesis in ASCII. To change it to 80 (or "P" in ASCII), replace the 28 with 50.

Adapter lets TI use VGA monitors

The following item comes from the TI list server. It's about an adapter that Lew King (kingt@rams1.rasd.k12.pa.us) used that allows the TI to work with a VGA monitor.—Ed.

Well, I just connected my TI to a VGA monitor. Term-80 in a full 80-column screen is very easy to read. The configuration screens which were previously impossible to read are now decent. There is almost no color bleeding. And, yes, FunnelWeb 8-bit is far superior to Win 95. Aside from the SAMS card, this is the best thing that ever happened to the TI.

The little adaptor box that does this is about the size of a small external modem. Color, con-

(See Page 39)

USER NOTES

(Continued from Page 38)

trast, brightness, etc. are all controlled by a controller that's similar to a TV remote.

The adapter was obtained from Tiger Direct Power Up, 1-800-335-4055, item No. P129-1000. The cost was $119.99, plus 6.99 shipping.

This adapter will accept any NTSC composite video input. So it will work with the Geneve or anything else. Output is a 15-pin standard connector to VGA. The only other thing you need is a VGA monitor.

I loaded a bunch of _P and color files into Bruce Harrison's AMS Slideshow. The results were spectacular.

Also inside the adapter is a TV tuner which will connect to a cable or outside aerial to watch TV on a VGA monitor. It has a little speaker inside and comes with audio and video cables.

XB screen saver

The following program was written by Tom Jakabfy of the OSHTI user group.—Ed.

This Extended BASIC screen saver program can be modified to suit your taste. Here is what the variables do:

B — Bias or offset of sprite starting position.

N — Number of sprites.

R — Starting screen row.

C — Starting screen column.

SX — Speed in X direction.

SY — Speed in Y direction.

A$ — Large sprite (64).

B$ — Small sprite.

Both of the shapes for the sprites were developed with Easysprite, an Extended BASIC program by Mike Freeman.

SCREEN SAVER

```
1 !*************** !035
2 !* SCREEN SAVER * !120
3 !*     SS-B     * !044
4 !* TOM JAKABFY  * !063
5 !*    OSHTI     * !126
6 !*  MARCH 1997  * !188
7 !*************** !035
100 CALL CLEAR :: CALL SCREE
N(5)!233
105 B=15 :: N=8 :: R=96 :: C
=104 :: SX,SY=10 !226
110 READ A$ :: READ B$ :: CA
LL CHAR(96,A$):: CALL CHAR(1
00,B$)!045
120 Q=0 :: CALL MAGNIFY(1)::
 CALL SPRITE(#1,100,16,R,C,S
X,SY)!059
121 CALL SPRITE(#2,100,7,R+B
,C-B,SX,-SY)!234
122 CALL SPRITE(#3,100,6,R-B
,C,-SX,SY)!231
123 CALL SPRITE(#4,100,14,R,
C+B,-SX,-SY)!217
124 CALL SPRITE(#5,100,2,R-B
,C-B,-SX,0)!054
125 CALL SPRITE(#6,100,8,R+B
,C,0,-SY)!057
126 CALL SPRITE(#7,100,4,R,C
+B,0,SY)!116
127 CALL SPRITE(#8,100,10,R-
B,C+B,SX,0)!166
130 GOTO 500 !068
140 GOSUB 9999 !134
150 GOTO 130 !209
500 CALL POSITION(#1,Y,X)!09
3
515 IF (Q=0)AND(Y>100)THEN G
OSUB 700 :: GOSUB 600 :: CAL
L MAGNIFY(3):: Q=1 :: GOTO 1
40 !202
520 IF Y>130 THEN CALL MAGNI
FY(4)!141
530 IF Y>210 THEN CALL DELSP
RITE(ALL):: GOTO 120 !103
540 GOTO 140 !219
600 FOR I=1 TO N :: CALL PAT
TERN(#I,96):: NEXT I !152
610 RETURN !136
700 FOR X=1 TO 20 :: NEXT X
:: RETURN !243
1000 ! WINDOW DATA !181
1010 DATA 803606803606803606
803606803606003EFFCB09C9C909
FFFF09C9C91DFFE300 !179
1020 DATA 8E5F955F95559F5B !
026
1030 RETURN !136
9999 CALL KEY(0,K,S):: IF K=
47 THEN STOP ELSE RETURN !08
0
```

CLASSIFIED

FOR SALE

TI99/4A COMPUTER SPECIALS

PHM3055	Editor Assembler	4.95
PHM3109	TI LOGO II (32K Req)	1.95
UM13181-1	RF Modulator	19.95
AC9500	TI Power Supply	8.95
PHP1100	TI Dual Joysticks	14.95
06059	TI99/4A Users Handbook	4.95
	GAMES	
RX8506	Defender (Joysticks Req.)	.49
PHM3031	The Attack	.49
PHM3158	MASH	.99
PHM3153	Super Fly	2.95
	EDUCATIONAL	
PHM3051	Numeration II (Ages 10-12)	1.95
PHM3094	Integers (Ages 11-14)	1.95
PHM30120	Reading Rally (Ages 10-12)	3.95
HOME MANAGEMENT		
PHM3007	Household Budget Mgt.	4.95
PHM3022	Personal Real Estate	3.95

JOY ELECTRONICS INC., 1-800-527-7438
You may call in your orders on credit card or COD. Call for free catalogue of TI products.

FOR SALE

Signalman Mark XII (1200 baud) modem, $7; Commodore 1702 composite color monitor (works with TI), $35. Call 512-255-1512, email jkoloen@io.com.

WANTED TO BUY

VIDEO GAME MAGAZINES WANTED

"Electronic Games," "Joystik," "Electronic Fun," "Video Games," "Video Games Player," "Atari Age," & others. "Video Games" Box 9542 Pgh., Pa. 15223 v14n3

Devoted to the TI99/4A since 1984

Subscription Fees

❑ 6 issues, USA, **$35**

❑ 6 issues, Mexico, **$40.25**

❑ 6 issues, Canada **$42.50**

❑ 6 issues, other countries surface mail, **$40.00**

❑ 6 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 ❑ MasterCard ❑ VISA ❑ (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. Credit card orders add 5%.

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 to March 1996, later $6 each. List issues: _____

No price breaks on sets of back issues. Free shipping USA. Add $1. single issues to Canada/Mexico. Other foreign shipping75 cents single issue surface. $2.80 airmail. Write for foreign shipping on multiple copies.

OUT OF STOCK: Vols. 1, No. 1-2; Vol. 2, No. 1

❑ **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 2.21** (req. DSSD or larger (for floppy & hard drive systems) $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

PERIODICAL