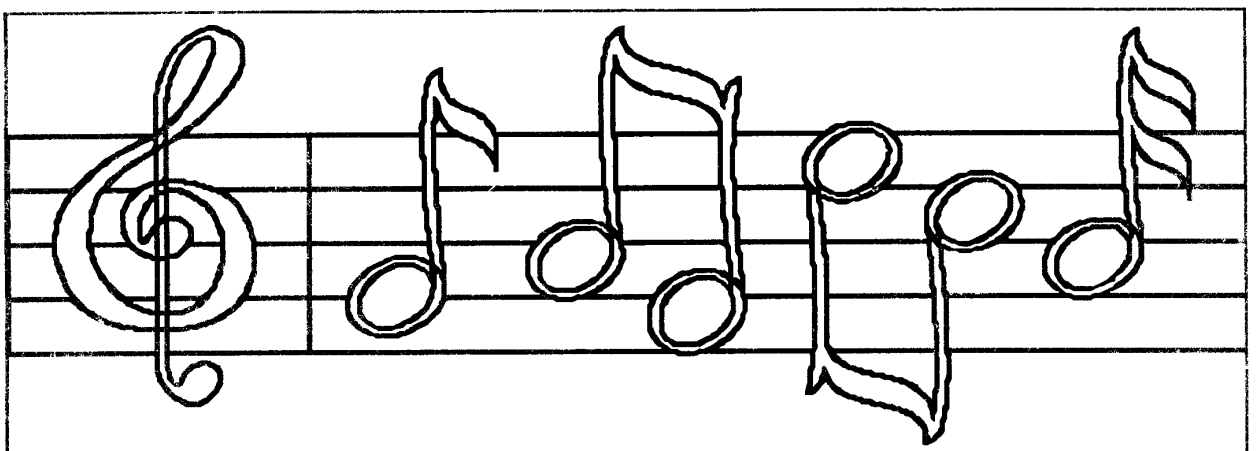
Covering the TI99/4A and the Myarc 9640

MICROpendium

Volume 11 Number 8 September 1994 $3.50



Experimenting with sound

page 6

PLUS:

Databases with Cardfile
Assembly mysteries unraveled
Reviews of XB Compiler, Drawing Program, Video Titler, Font Converter, Turnfont, and CALL LINKable XB Enhancements

CONTENTS

MICROpendium

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

Display advertising deadlines and rates are available upon request.

All correspondence should be mailed to MICROpendium at P.O. Box 1343, Round Rock, TX 78680. We cannot take responsibility for unsolicited manuscripts but will give consideration to anything sent to the above address. Manuscripts will be returned only if a self-addressed stamped envelope is included.

Foreign subscriptions are $40.25 (Mexico); $42.50 (Canada); $40.00, surface mail to other countries; $52 airmail to other countries.

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

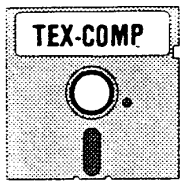
Mailing address: P.O. Box 1343, Round Rock, TX 78680.
Telephone: (512) 255-1512
Fax: (512) 255-1557
CompuServe: 75156,3270
Delphi TI NET: MICROpendium
GEnie: J.Koloen
Internet E-mail: jkoloen@io.com
John Koloen.....Publisher
Laura Burns.....Editor

*READ THIS

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

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

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



ANNOUNCING NEW LOW PRICES ON FREEWARE DISKS FROM $2.95 TEX COMP $2.95

TEX-COMP HAS SLASHED PRICES ON THE BIGGEST & BEST COLLECTION OF FREEWARE FOR THE 99/4A. NOW ONLY $2.95 PER DISK WITH A 5 DISK MINIMUM. CHOOSE FROM HUNDREDS OF GREAT PROGRAMS. NO EXTRA CHARGE ON PROGRAMS THAT REQUIRE MORE THAN ONE DISK SIDE. ALL PROGRAMS HAVE BEEN TESTED AND ALMOST ALL HAVE BEEN PROVIDED WITH EXBASIC AUTOLOAD. CHOOSE FROM THE BEST IN GAMES, UTILITIES, GRAPHICS, HOME & BUSINESS AND DISK BACKUPS OF DOZENS OF YOUR FAVORITE MODULES THAT ARE NOW OUT OF PRODUCTION.

GAMES • BUSINESS • GRAPHICS • WORD PROCESSING • UTILITIES • DATABASE • MUSIC • COMMUNICATIONS • HOME

GRAPHICS, MUSIC & ANIMATION
#1 THE SINGING TI VOL. 1 (S)(2)
#4 PRINTART (2)(P)
#5A MUSIC/GRAPHICS
#6 EXBASIC MUSIC (2)
#7 SPACE SHUTTLE MUSIC/GRAPHICS
#9 MONA LISA PRINTOUT (P)
#11 ANIMATED XMAS (WOODSTOCK)
#14 FIGURE STUDIES (P)(PG)
#32 EXBASIC XMAS MUSIC (2)
#41 VIDEO GRAPHS (M)
#52 ANIMATION 99' (2)
#69 PLAYER PIANO/KEYBOARD ANALYSIS
#93 KBGB GIRLIE CALENDAR (4)(P)
#103 SORGAN THE TI ORGAN
#107 STARTREK MUSIC ALBUM
#111 POP MUSIC & GRAPHICS
#114 PANORAMA
#115 GRAPHICS DESIGN SYSTEM
#120 BITMAC (2)(P)
#230 THE SINGING TI VOL. 2 (S)(2)
#231 THE SINGING TI VOL. 3 (S)(2)
#246 THE SINGING TI VOL. 4 (S)(2)
#313 3-D WORLD (CAD for the TI)
COMPUTER UTILITIES, PRINTER UTILITIES & PROGRAMMING LANGUAGES
#3 DUMPIT (E/A)
#15 STAR/EPSON PRINTER DEMO (P)(2)
#16 SIDEWAYS PRINTOUT (P)(2)
#8 TI DIAGNOSTIC (MM)(2)
#8 LOADERS & CATALOGERS
#30 HOUSEHOLD BUDGET PRINTOUT (P)
#35 PROGRAMMING AIDS & UTILITIES I
#42 FUNNELWEB FARM (SHELL UTILITY)
#53 HACKER CRACKER
#55 SCREEN DUMP (P)
#62 DISK MANAGER II (M)
#75 DISK CATALOGER
#76 PROGRAMMING AIDS & UTILITIES II
#78 ARICON GRAPHIC CONVERSION
#79 DISK MANAGER 1000
#80 BIRDWELL DISK UTILITY (2)
#85 AUTOBOOT UTILITY
#86 COLUMN TEXT TII (P)
#87 ARCHIVER III
#89 PROCALC DECIMAL/HEX CONVERTER
#96 STATISC & SORT ROUTINES
#97 MEMORY MANIPULATOR
#101 ENHANCED DISPLAY PACKAGE
#108 FUNLPLUS (PLUS!)
#110 DISK+ AID
#117 UNIVERSAL DISASSEMBLER
#119 RAG LINKER CONVERSION
#127 PIX GRAPHICS UTILITY
#243 OS/99 (GD)
#251 PC TRANSFER TI/IBM (DD)
#253 THE EXPLORER/DM1000
#254 NIBBLER/TURBO
#260 TI FORTH (DISK ONLY- add $8 for manual)
#17 TI FORTH DEMO
#116 TI FORTH TUTORIAL
#5079 FORTH SOURCE CODE
#104 C99 COMPLIER & LIBRARY
#5007 TEACH YOURSELF TI BASIC
#5019 TEACH YOURSELF EX-BASIC
#5067 BEGINNING BASIC TUTOR
#307 GRAPHICS CODE GENERATOR
#3912 SUPER BUGGER
#3913 BETTER BANNERS
#3914 CERTIFICATE 99
#3915 HOROSCOPE MAKER
#3916 GRAPHX+ PRINT SHOPPE
#3917-3720
GRAPHX COMPANIONS 1-4
#3721 MAC FLICK (G)
#3722 PRINTING TO GO (G)
#3723 GRAPHX DINOSAURS (G)

BUSINESS, ACCOUNTING, WORD PROCESSING, DATA BASE
HOME OFFICE & HOME
#10 GOTHIC PRINTOUT (P)
#19 TI WRITER/MULTIPLAN UPGRADE
#20 ACCOUNTS RECEIVABLE (2)
#21 DATA BASE DEMO
#23 MILL WRITER
#29 LABEL MAKER I (P)
#36 STRICTLY BUSINESS (2)
#56 SPREAD SHEET
#58 PR BASE (database)
#59 GRAPH MAKER
#74 LABEL MAKER II (P)
#77 MICRodex 99 (database)
#81 HOME ACCOUNTING SYSTEM
#83 HOME APPLICATION PROGRAMS (2)
#90 JET CHECKBOOK MANAGER
#92 HOUSEHOLD INVENTORY
#109 TI WRITER MINI MANUAL
#112 INVOICE PACK
#113 LABEL MAKER III
#129 CASH DRAWER (point of sale)
#130 THE ORGANIZER
#147 CALENDAR-NOTEPAD
#177 HOUSEHOLD BUDGET MANAGEMENT (M)
#30 HBM DATA PRINTOUT (P)
#221 PERSONAL REAL ESTATE (M)
#249 MAPMAKER
#252 99 WRITER II (TI WRITER)(P)
#253 AMA MAILING LIST
#255 TRAK-A-CHECK
#257 DAILY DIARY
#257A EXER-LOG
#23 MILL WRITER
#310 SELF HELP TAX CUT
TELECOMMUNICATIONS (MODEM)
#57 IELCO
#57D IELCO (FOR SYSTEMS WITH DS DRIVES)
#118 FAST TERM
EDUCATION & PERSONAL DEVELOPMENT
#22 ASTROLOGY
#24 ENGINEERING CALCULATIONS (2)
#25 MEDICAL ALERT
#27 KIDS LEARNING I (2)
#31 MORSE CODE TRAINER
#37 LAPD COOKBOOK (2)
#40 ARTIFICIAL INTELLIGENCE (ELIZA)
#54 ASTRONOMY
#66 HEBREW TYPEWRITER
#67 GENEALOGY (2)
#71 KIDS LEARNING II (2)
#95 WEATHER FORECASTER +
#138 FIREHOUSE COOKBOOK
#142 TOUCH TYPING TUTOR (M)
#184 FACE MAKER
#194 ST. NICK/GHOSTLY SPELLING (M)
#195 TINY LOGO
#199 MILLIKEN ADDITION (M)
#200 MILLIKEN DECIMALS (M)
#203 MILLIKEN FRACTIONS (M)
#204 MILLIKEN INTEGERS (M)
#205 MILLIKEN LAWS OF MATH (M)
#211 MIND CHALLENGERS (M)
#212 MINUS MISSION (M)
#213 MILLIKEN PERCENTS (M)
#214 STORY MACHINE (M)
#215 BEGINNING GRAMMAR (M)
#216 METEOR MULTIPLICATION (M)
#217 HANGMAN (M)
#222 MUSICMAKER (M)
#223 PHYSICAL FITNESS (M)
#225 ALIEN ADDITION (M)
#226 ALLIGATOR MIX (M)
#227 DEMOLITION DIVISION (M)
#228 DRAGON MIX (M)
#270 TRIGONOMETRY
#271 CALCULATORS & CONVERSIONS (2)
#272 HIGHER MATH (2)
#273 ASTRONOMY 2
#306 SPEAK & SPELL II (S)(EX)

GAMES
#2 WH. OF FORTUNE, BLKJACK, POKER
#3 TI TREK
#8 LOTTO PICKER
#13 STRIP POKER (PG)
#26 R-RATED NOVELTY GAME
#33 CHECKERS & BACKGAMMON
#34 SOLITAIRE & SCRABBLE
#38 GREAT TI GAMES VOL 1
#39 GREAT TI GAMES VOL 2
#43 BEST OF BRITAIN GAMES VOL 1
#45 BEST OF BRITIAN GAMES VOL 2 (LEGEND OF CARFAX ABBY GRAPHIC-INTERACTIVE ADVENTURE)
#46 SUPER TRIVIA 99
#47 INFOCOM RAPID LOADER
#48 GHOSTMAN (from U.K.)
#49 DEMON DESTROYER (from FRANCE)
#50 OH MUMMY! (HIT from GERMANY)
#51 BERLIN WALL (from CANADA)
#60 FREDDY (HIT from GERMANY)
#61 THE MINE (from GERMANY)
#63 ASTROBLITZ & MAZOG
#64 MAJOR TOM & SPACE STATION FRETA
#65 PERFECT PUSH (HIT)
#69 CHESS (SARGON)
#70 TI RUNNER II (HIT)
#72 CEBERUS (HIT SPACE GAME)
#73 CRYTO (GRAM)
#82 CROSSWORD (PUZZLES)
#84 GALACTIC BATTLE & SPY ADVEN.
#88 AUSSIE GAME COLLECTION VOL 1
#91 THE MAZE OF GROG (WOODSTOCK II)
#24 GREAT TI GAMES VOL 3
#98 DAYS/DOORS OF EDEN (BIBLE ADV)(AM)
#99 GREAT TI GAMES VOL 4
#100 ASSAULT THE CITY (TD)
#102 COLOSSAL CAVES (ADVENTURE)
#105 KINGS CASTLE (M)
#106 QUEST (D&D)
#121 SUPER YAHIZEE & WHEEL II
#122 ADULT ADVENT & GAMES (PG)
#123 GREAT TI GAMES VOL 5 (2)
#124 GREAT TI GAMES VOL 6 (2)
#125 BLACKJACK & POKER (M)
#126 VIDEO CHESS (M)
#128 TETRIS (HIT from RUSSIA)
#131 COMPUTER CRAPS
#132 AMBULANCE (M)
#133 DRIVING DEMON (M)
#134 ROTO-RAIDER (M)
#135 ARTURUS (HIT-ZARGON)
#136 ANT-EATER (M)
#137 CROSS-FIRE (M)
#139 MOONMINE (M)
#140 MASH (M)
#141 MOONSWEEPER (M)
#143 CONGO BONGO (M)
#144 STAR TREK (M)
#145 BUCK ROGERS (M)
#148 KENO & SLOTS
#149 GREAT TI GAMES VOL 7 (with HIT BLOCKBUSTER)
#150 ULTIMATE TRIVIA
#151 JUNGLE HUNT (M)
#152 POLE POSITION (M)
#153 DONKEY KONG (M)
#154 PROTECTOR II (M)
#155 PAC MAN (M)
#156 CENTIPEDE (M)
#157 DEFENDER (M)
#158 SHAMUS (M)
#159 MS. PAC MAN (M)
#160 DIG-DUG (M)
#161 PICNIC PARANOIA (M)
#162 MOON PATROL (M)
#3711 ARCADE SPECIAL (4 GAMES)
#3712 THREE GREAT GAMES
#3713 PRO TENNIS+

INFOCOM ADVENTURE BACKUPS
#163 ZORK I
#164 ZORK II
#165 ZORK III
#166 HITCHIKER'S GUIDE
#167 WITNESS
#168 ENCHANTER
#169 INFIDEL
#170 PLANETFALL
#171 SORCERER
#172 DEADLINE
#173 CUTTHROATS
#174 SUSPENDED
#175 STARCROSS
****
#176 AMAZING (M)
#178 DEMON DESTROYER (M)
#179 POPEYE (M)
#180 QUEBERT (M)
#181 METEOR BELT (M)
#182 BLASTO (M)
#183 CAR WARS (M)
#184 FACE MAKER (M)
#185 SUPER FLY (M)
#186 SPACE BANDITS (M)
#188 KILLER CATERPILLER (M)
#190 BLACK HOLE & SPACE AGRESS (M)
#191 GREAT TI GAMES VOL 8
#192 GREAT TI GAMES VOL 9
#193 SPY'S DEMISE (HIT) (M)
#194 ST. NICK+ (M)
#196 JOTTO
#197 PRO TENNIS+ (HIT)(M)
#198 TI INVADERS/TOMBSTONE CITY (M)
#202 CONNECT FOUR (M)
#205 HOPPER (M)
#206 TREASURE ISLAND (M)
#206B SLYMOIDS (M)
#207 OTHELLO (M)
#208 PARSEC (M)
#209 SOCCER (M)
#210 SEMERMANIA (M)
#218 HUSTLE/FOOTBALL (M)
#219 CHISOLM TRAIL (M)
#220 ZERO ZAP (M)
#224 ATTACK (M)
#229 4A FLYER (FLIGHT SIM.)(M)
#232 TUNNELS OF DOOM (MOD BACKUP PLUS 2 NEW ADVENTURES)(M)
#233 MS ADVENTURES (3 ADV-EXBASIC)
#248 STRIKE THREE BASEBALL (M)
#248 GREAT TI GAMES VOL 10
#250 BARRAGE/SPOTSHOT
#318 THREE GREAT GAMES
#319 ARCADE SPECIAL
#317 BEANSTALK ADVENTURE
****
SCOTT ADAMS ADVENTURES (USE WITH ADVENTURE MODULE)
#349 II ADVENTURES 1-13+
#350 TI ADVENTURES 14-16+

P = Printer required
G = Graphx required
S = Speech required
M = Module Backup
MM = Mini Memory req.
E/A = Editor Assem.
Exbasic and 32k mem req. for most programs.

ORDER BY PHONE
(818) 366-6631 24 HOURS A DAY

Authorized Dealer
TEX COMP
America's Number One TI computer retailer

P.O. Box 33084, Granada Hills, CA 91344

TERMS: MINIMUM ORDER FOR $2.95 PRICE IS 5 DISKS (REG. 4.95) ADD $4.00 PER ORDER FOR SHIPPING (U.S.)
ALL PRICES ARE FOR CASH/CHECK, ADD 3% FOR CREDIT CARD ORDERS. INCLUDE STREET ADDRESS FOR U.P.S.

COMMENTS

Reunion set for Dallas

The Dallas TI Home Computer Group has a great idea.

The group, which has about 60 members and meets monthly at Dallas' Infomart, has scheduled a reunion of all former and current members Oct. 29.

"I'm sure the majority of MICROpendium readers are users of the 4/A, and therefore might be interested in coming," writes Dan Lowe, the group's president.

He adds, "There will be lots to do, lots to see, lots of chit-chat and visiting, and lots of refreshments. There will be displays and demonstrations of TI99/4A equipment from the early days to the present; some of the early, innovative pieces of hardware illustrating 'look how far we've come'; and, if we can swing it, authors/programmers of some of the most popular TI programs. And while we plan to have one or two speakers, this event will not be highly structured or tightly organized, but rather a loose, mix-and-mingle relaxed atmosphere."

For further information, contact Lowe at 1514 Greentree Lane, Garland, TX 75042-4648, or (214) 276-0240.

MICROPENDIUM DISK SALE

Our annual sale of MICROpendium program disks starts this month. This is an opportunity for readers to obtain the disks at a 40 percent discount. For more information, see the advertisement elsewhere in this edition.

—JK

READER TO READER

❑ Alfred Slovak, Fugbachgasse 18/17, A-1020, Vienna, Austria, writes:

I copied MDOS 2.00 to my hard drive. My hard drive is a 10-megabyte Seagate ST 412. The Geneve booted without any problems from the hard drive (drive letter E>), but then failed to run the AUTOEXEC file that is also stored on the root directory of the hard drive and responded with drive letter A>. Changing the drive letter to E? and entering AUTOEXEC by hand from the MDOS prompt ressulted in execution of AUTOEXEC (what else?)

Did I make anything wrong? Do I need a special loader-program (LOAD/SYS) stored in subdirectory DSK1?

❑ Bruce Harrison of Hyattsville, Maryland, has shared with us his reply to Jerry Keisler, who wrote in concerning his problems with error control in his programming in the August 1994 issue. He writes:

You've run across the old "file ajar" problem. That's a case in which the file hasn't really opened, but can't be read or closed without generating another error.

I, too, ran into this problem, when making a test program for my Compiler, but I have an easy solution for you. The trick is to set up a dummy error trap within your primary error trap, then CLOSE #3. The computer will close the "ajar" file, but doing so will generate an error in itself. The dummy error trap takes you to the next program line (where you'd go anyway) within your error routine. This looks utterly stupid, but is the only way I know of to get the TI to forget about the error of the nonexistent drive's catalog not opening.

The listing shows my solution to this problem, using line 350 as the initial error trap, setting the new trap at 355, and then closing the "ajar" file.

```
190 DISPLAY AT(10,1)ERASE ALL:"CATALOG TI-W
RITERS FILES ON": :"DISK DRIVE";DR :: ACCEP
T AT(12,12)BEEP SIZE(-1_VALIDATE("12345678
9"):DR
200 DK$="DSK"&STR$(DR)&"."
210 ON ERROR 350 :: OPEN #3:DK$,INPUT RELAT
IVE,INTERNAL
220 INPUT #3:A$(0),J,N,K ! YOUR PROGRAM WOU
LD CONTINUE HERE
230 DISPLAY AT(18,1):A$(0),J,N,K :: CLOSE #
3
240 CALL KEY(0,K,S):: IF S<1 THEN 240 ELSE 1
90
    ...
350 ON ERROR 355 :: CLOSE #3
355 DISPLAY AT(24,1):"DISK ERROR  PRESS ENT
ER"
360 CALL KEY(0,K,S):: IF S<1 THEN 360 ELSE R
ETURN 190
```

❑ Ted Stringfellow of Ocean Springs, Mississippi, has shared with us his reply to Robert Schulz, who wrote in concerning his problems with TI-Base in the August 1994 issue:

Seems like your best bet to reroute your print functions using a menu is to create a couple of small command files. Try the following:

1. Type MODIFY COMMAND MODULE
2. When the edit screen comes up, type:

```
SET PRINTER=DSK1.TIBLIST
* PRINTER OUTPUT HAS NOW BEEN REDIRECTED
*  TO FILENAME "YOURFILE"
* BE SURE THERE IS A FORMATTED DISK
*  IN DSK1 PRIOR TO PRINTING
```

3. F8

To reset output to the printer, create another command file that contains:

```
SET PRINTER=PIO.CR.LF
```

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

FEEDBACK

AMS card may have future production

I'd like to thank you for your reviews in the last two issues of our AMS card. I'd like to applaud you and Bruce Harrison for being fair and thorough.

The primary reason that Asgard Peripherals was forced to go out of business was the TI community's slow acceptance of this device. We started this project almost five years ago. When we completed it about three years ago, it was promptly blitzed with FUD ("Fear, Uncertainty and Doubt") generated by our competitors. These same competitors have yet to ship their product.

I have to emphasize that no one works in the TI community for the money; they put in the time in order to be rewarded with recognition by their peers, the technical and intellectual challenge, and the sense that the TI community is still a place where a few can make a meaningful difference. Robbed of recognition and a sense we were making a difference, the team of programmers and hardware designers I put together drifted apart.

Some of us have moved on to other 99/4A projects. I was so disgusted to see how my 12 years in the community culminated that I decided to quit — for the last few years I've found a new niche in designing networks and Internet consulting. My partner Jim Krych and I are also still working on hardware together, but more oriented towards the general market.

However, I don't want to belabor the issue further. After some discussion, Jim and I have agreed that it is in the best interest of the community to put our designs — functional and tested versions of both 512K and 1Mb AMS designs — into the public domain.

Right now we have been working with a user group to set up facilities to produce the device. We are willing to work with any other user group or company that would like to do the same. Because of the complexity of the design, we don't feel it's possible to just publish the specs and be done with it. Instead, anyone who is interested should contact either of us, and we'll provide them the materials and some technical support.

Further, I have on hand about 80 74LS612 chips (the memory management chip required by the device) which have become somewhat hard to find. Anyone who would like them can have them for $5 each (plus $5/order shipping and handling). I have a variety of other parts also available (AMS chips, Asgard Mouse cables, etc.) and will be happy to send a price list to anyone who wants one.

Please send all orders or inquiries to 1423 Flagship Dr., Woodbridge, VA 22192. Inquiries about producing the AMS can also be e-mailed to C_BOBBITT@DELPHI.COM. Thank you.

Chris Bobbitt
Woodbridge, Virginia

Fireside chat: The beginning

It is altogether fitting and proper, this month, September 1994, that the four-year anniversary of the AEMS Project be celebrated.

Let it be said here, that although Asgard Peripherals is no longer making TI99/4A Home Computer hardware, we have been working very hard in the background, to ensure that the hard work and effort that went into the AEMS Project shall not die but live on.

The AMS schematics, the SRAM designs both AMS and SUPERams, have been released to the public domain. A third party, who shall remain unmentioned for now, has been given the technology for the Pseudo SRAM based SUPERams. And finally, the possibility exists for emulator support.

None of this has been easy. But I think that what we have done is an incredible achievement, given the following:

1. None of us on the AEMS Project Team were in the same room, let alone same state.

2. Limited access to digital debugging tools.

3. All that was done was part time, not full-time design work.

All of this, a lot of hard work and perseverance, and the grace of God, led us to develop this unique expanded memory system for the 99/4A.

Not only does hardware exist, but an extensive line of tools. Some may say that we failed to get more software, considering one time Asgard Peripherals was part of Asgard Software. The goal of the AEMS Project was always "ease-of-use" expanded memory. This, we achieved.

I admit that the lack of more software is a problem. The best hardware is useless without it. But slowly, very slowly, more programmers and more people are getting the cards and the developmental software. With support by the third party, the public domain releases and possible emulator support, more users will beget more software.

Much has been done even though we at Asgard Peripherals announced that the AEMS Project was completed six months ago. With this four year anniversary, I am looking back at what has happened. This is the last memory Fireside Chat and I have a few things to say.

Our choice of printed circuit board designer was flawed, and I alone am responsible. The choice resulted in a changed design, without our knowledge, and a loss of money when more boards were ordered but never delivered.

The memory debates were absolutely useless. We offered a new idea and many resisted – to the point where what we had done and offered was pushed aside. These people I call the first generation. Another point, although those involved will deny it, the rumors and outright lies during the first year proved to us the character of our competitors. I believed then, as I still do, that my team members had legitimate complaints, and I trust them to this day.

But let it be said that even during the first year, we started to gain supporters of the card, some of which are the third party. Let me say their names now – they are the ones who fought a bias and examined a real product, for what it did and achieved.

I thank: Brad Snyder, Bruce Harrison, Mike Doane, David Ormand, Jon Dyer, Tom Willsand others; forgive me if I did not mention your names. And I thank the AEMS Project Team: Chris Bobbitt, Art Green, Tony Lewis, Joe Delekto, and Chuck Abdouch.

The AEMS Project, one way or another, will not die off. At this timeI am providing schematics to two people who are also programmers.

(See Page 6)

Extended BASIC

Experimenting with sound

By W. LEONARD TAFFS

The following program and table were written by Al Armstrong, a member of the Southwest 99ers user group of Tucson, Ariz. Taffs, also a SW99er member, wrote the text.—Ed.

XCPSNDBA and XSNDTBL came about when Al saw my SOUND/EXP in the SW99ers newsletter of July 1994. SOUND/EXP also appeared as "SOUND" in the July 1994 MICROpendium.

XCPSNDBA will load from Extended BASIC or from TI-BASIC using either Mini-Memory or the Editor/Assembler module. When module menu appears, select TI-BASIC. Once in TI-BASIC, enter "OLD DSK1.XCPSNDBA" to load the program. There are minimal instructions in the program itself — Al did not know I was interested in passing this program on — so a few notes may help:

1. As the program screen will remind you, if you do not make this change before running program, you will need to un-REMark one of two lines. For Extended BASIC, un-REMark line 204. To run in TI-BASIC, un-REMark line 202.

2. When program is running, you will be prompted three times for DATA frequencies. After entering these frequencies the program advances to a menu with the following six keypress options:

T — to hear the tones selected sounded simultaneously.

U — to hear the individual tones linked with attack/decay/level modification.

C — to hear a chimes sound (not using input frequencies).

V — to hear a vibraphone-like sound or vibrato effect (not using input frequencies).

D — to input more DATA frequencies (prev. entry still on screen)

Sp Br — Space Bar clears the screen and returns to the beginning screen.

3. To repeat a key selection, press the key again. To terminate a key selection, hold the key down.

4. The difference between selection D or pressing Space Bar — both allow you to resume entering frequencies — is that selecting D will leave the previous entry scrolling up the screen. The Space Bar will return you to the opening screen.

5. The C and V key options do not employ the frequencies you enter but are fixed. If you wish to try other frequencies, change the appropriate program lines for these.

6. You can enter "1" as the frequency data at any DATA prompt and "1" will act as a default to move to the next program line. Thus you can hear just one or two frequencies at a time if you wish. The "1" behaves like a null string.

With the accompanying frequency table and sound chip information, XSNDTBL should prove to be a valuable resource to programmers who are interested the sound reproduction capabilities of the TI-99/4A. It is obvious these two files by Al Armstrong represent a great deal of work and it is to be hoped that they will be useful references to anyone interested in programming or learning about sound with the TI.

XCPSNDBA

```
1 DIM A(12)!101
2 CALL CLEAR !209
3 CALL INIT !157
4 CALL SCREEN(3)!148
5 REM XCPSNDBA program BY AL
 ARMSTRONG of SW99ERS TUCSON
, AZ was in response to TAFF
S' FEEDFORTH COLUMN's !086
6 REM program "SOUND/EXP" fi
rst published IN JULY '94 SW
99ERS Newsletter P.4. !037
7 PRINT :"INPUT SOUND DATA
          Freq 110 to 10,00
0 hz.":!195
8 I=0 !000
9 PRINT : :!006
10 INPUT " DATA= ":T(I)!182
12 F=T(I)!014
14 I=I+1 !011
16 IF F*(I<3)THEN 10 !129
18 IF (I<>3)+(T(3)<>0)+(T(0)
<110)+(T(1)<110)+(T(2)<110)T
HEN 46 !206
20 I=0 !000
22 PRINT : :"Tones or UFO":
:!087
24 PRINT ;STR$(T(I));",";!25
4
25 I=I+1 !011
26 IF I<2 THEN 24 !024
27 PRINT ;STR$(T(I)): :"GT1,
GT2,GT3":"Frequencies";!082
28 REM!154
30 GOSUB 200 !024
32 CALL KEY(0,K,S)!187
```

(See Page 7)

FEEDBACK

(Continued from Page 5)

But I am personally glad that the project for us is over. It was a tremendous effort. The community has been given a real choice for expanded memory. And the options exist for people to get the cards. Not many orphan computers can boast of an expanded memory system with extensive tools, and "ease-of-use!"

I thank you, the TI99/4A community, for the support over these years. To you, take care and God Bless!

James W. Krych
Director, Research and Development,
Asgard Peripherals
North Olmsted, Ohio

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

XCPSNDBA—

(Continued from Page 6)

```
34 IF S<1 THEN 32 !041
36 ON 1+POS("DTUCV",CHR$(K),
1)GOTO 2,8,60,70,120,90 !245
38 T(I)=20000 !135
40 IF I=2 THEN 20 !019
41 PRINT " DATA " !195
42 I=I+1 !011
44 IF T(I)>109 THEN 40 ELSE
38 !196
46 T(3)=0 !116
48 I=0 !000
50 GOTO 44 !123
52 DATA 144,178,211,192,145,
177,193,146,176,212,192,147,
177,193,148,178,213,192,149,
179,193,150,180,214,192,151
!113
53 DATA 181,193,152,182,215,
192,153,183,193,154,184,216,
192,155,185,193,156,186,217,
192,157,187,193,158,188,218
!131
54 DATA 192,159,189,193,219,
190,159,192,220,191,193,221,
  1,222,193,223,0 !029
55 DATA 0 !140
56 DATA "KEY OPTION"," T   T
ones"," U   UFO  "," C   Chi
me"," V   Vibro"," D   DATA
","SpBr REDO " !154
57 REM !186
58 REM !186
60 CALL SOUND(750,T(0),0,T(1
),8,T(2),4)!033
62 GOTO 32 !111
70 RESTORE 85 !178
72 I=0 !000
74 CALL SOUND(1,T(0),30,T(1)
,30,T(2),30)!068
76 READ I !224
77 IF I=0 THEN 82 !079
78 CALL LOAD(-31744,I)!132
80 IF I THEN 76 !146
82 CALL KEY(0,K,S)!187
84 IF S=0 THEN 70 ELSE 32 !1
83
85 DATA 144,145,146,147,148,
149,150,151,152,153,154,155,
156,157,158,159 !110
86 DATA 223,222,221,220,219,
218,217,216,191,190,189,188,
187,186,185,184,183,182,181,
180,179,178,177,176,215,214
!117
87 DATA 213,212,211,210,209,
208,152,150,148,146,144 !208
88 DATA 191,223,159,0 !215
90 RESTORE 95 !188
91 READ I !224
92 IF I=0 THEN 100 !097
93 CALL LOAD(-31744,I)!132
94 GOTO 91 !170
95 DATA 143,9,144,141,145,14
3,146,141,147,143,140,141,14
9,143,150,141,151,143,152,14
1,153,143,154,141,155,143 !1
91
96 DATA 156,141,157,143,158,
142,159,0,191,223,255,0 !045
97 DATA 144 !247
98 DATA 0 !140
99 DATA 144 !247
100 CALL KEY(0,K,S)!187
101 IF S=0 THEN 100 !107
102 IF K=32 THEN 90 !143
103 READ I !224
104 IF I=0 THEN 32 !029
105 CALL LOAD(-31744,I)!132
106 GOTO 103 !182
120 RESTORE 52 !145
122 CALL SOUND(1,550,30,782,
30,980,30)!080
124 READ I !224
126 IF I=0 THEN 32 !029
128 CALL LOAD(-31744,I)!132
130 GOTO 124 !203
150 I=0 !000
152 AD=404+32*RD !239
154 READ I$ !004
```

SOUND TABLES

The following table was created by Al Armstrong of the Southwest 99ers, Tucson, Ariz.—Ed.

*** TONE-GENERATOR: (2 bytes) *****

Binary	Hex	Dec	Fcode
GT1; (Byte 1)			(Lb)
10000000	>80	128	0
1000xxxx	>81-F	129-142	1-14
10001111	>8F	143	15
GT1,2,3	(Byte 2)		(Hb)
00000000	>00	0	0
00xxxxxx	>1-3E	1-62	1-62
00111111	>3F	63	63
GT2;	(Byte 1)		(Lb)
10100000	>A0	160	same
1010xxxx	>A1-F	161-174	as
10101111	>AF	175	GT1
		(Byte 2)	(Hb)
GT3;	(Byte 1)		(Lb)
11000000	>C0	192	same
1100xxxx	>C1-F	193-206	as
11001111	>CF	207	GT1
		(Byte 2)	(Hb)

Computation of Fcode, Hb and Lb segments.

Hb= INT(0.0625*Fc)= INT(Fb)

Lb= INT(Fc-16*Hb)= INT(16*(Fb-Hb))

Frequency Code vs Frequency Number

Fc= 111860.8/Fn Fb= 6991.3*Ti

Fn= 111860.8/Fc Ti= 1/Fn

SOUND-BYTE VALUES

TABLE Ia

*** NOISE-GENERATOR: (1 byte) *****

Binary	Hex	Dec	TYPE
GPN;			
11100000	>E0	224	1 *\
11100001	>E1	225	2 *-Periodic

(See Page 8)

SOUND TABLES—

(Continued from Page 7)

```
 11100010    >E2      226      3 */
 11100011    >E3      227      4 PNmod*GT3
GWN;
 11100100    >E4      228      1 *\
 11100101    >E5      229      2 *- White
 11100110    >E6      230      3 */
 11100111    >E7      231      4 WNmod*GT3
```

ATTENUATION (1 byte) ******

Binary	Hex	Dec	DB*
GT1;----------			
10010000	>90	144	0
1001xxxx	>91-E	145-158	-2\-28
10011111	>9F	159	Off
GT2;----------			
10110000	>B0	176	0
1011xxxx	>B1-E	177-190	-2\-28
10111111	>BF	191	Off
GT3;----------			
11010000	>D0	208	0
1101xxxx	>D1-E	209-222	-2\-28
11011111	>DF	223	Off
GPN,GWN;----------			
11110000	>F0	240	0
1111xxxx	>F1-E	241-254	-2\-28
11111111	>FF	255	Off

SOUND-BYTE VALUES TABLE Ib

SOUND-BYTE VALUES Tables Ia,Ib and Hb-Fcode VALUES Tables IIa,IIb,IIc,IId assist in deriving a decimal or hex number that will produce sound signals of specific quality and loudness when stored in the TI-99/4A System Console sound-chip. Numbers may be used individually or grouped into a sound list which is written to the sound-chip address. All numbers are byte values of 0 to 255 decimal, (>0 to >FF Hex).

**** GT1,GT2,GT3 **************

Hb Fcode Hex	Dec	FREQ @ Lb >F hz	[Fi]
>3F	63	109.3	0.2
>3E	62	111.1	0.1
>3D	61	112.9	0.1
>3C	60	114.7	0.1
>3B	59	116.6	inc
>3A	58	118.6	0.1
>39	57	120.7	0.1
>38	56	122.8	freq
>37	55	125.0	0.1
>36	54	127.3	0.1
>35	53	129.6	hz
>34	52	132.1	0.1
>33	51	134.6	0.2
>32	50	137.3	0.1
>31	49	140.0	0.2
>30	48	142.9	0.1

Hb-Fcode VALUES TABLE IIa

Hb Fcode Hex	Dec	FREQ @ Lb >F hz	[Fi]
>2F	47	145.8	0.2
>2E	46	148.9	0.2
>2D	45	152.2	0.2
>2C	44	155.6	inc
>2B	43	159.1	0.2
>2A	42	162.8	0.3
>29	41	166.7	0.3
>28	40	170.8	freq
>27	39	175.1	0.2
>26	38	179.6	0.2
>25	37	184.3	hz
>24	36	189.3	0.3
>23	35	194.5	0.4
>22	34	200.1	0.4
>21	33	206.0	0.4
>20	32	212.3	0.4

Hb-Fcode VALUES TABLE IIb

Hb Fcode Hex	Dec	FREQ @ Lb >F hz	[Fi]
>1F	31	218.9	0.4
>1E	30	226.0	0.4
>1D	29	233.5	0.5
>1C	28	241.6	0.5
>1B	27	250.2	0.6
>1A	26	259.5	0.6
>19	25	269.5	inc
>18	24	280.4	freq
>17	23	292.1	0.7
>16	22	304.8	0.8
>15	21	318.7	hz
>14	20	333.9	1.0
>13	19	350.7	1.1
>12	18	369.2	1.2
>11	17	389.8	1.3
>10	16	412.8	1.5

Hb-Fcode VALUES TABLE IIc

Hb Fcode Hex	Dec	FREQ @ Lb >F hz	[Fi]
>0F	15	438.7	1.8
>0E	14	468.0	2
>0D	13	501.6	2

(See Page 9)

SOUND TABLES—

(Continued from Page 8)

>0C	12	540.4	3
>0B	11	585.7	3
>0A	10	639.2	4
>09	09	703.5	5
>08	08	782.2	6
>07	07	880.8	7
>06	06	1007.8	9
>05	05	1177.5	12
>04	04	1416.0	?
>03	03	1775.6	?
>02	02	2380.0	?
>01	01	3608.4	?
>00	00	7459.4	?

Hb-Fcode VALUES
TABLE IId

There are four ways to invoke sound from the sound-chip.

1. A routine in BASIC grom is accessed by the CALL SOUND() statement. Parameters in that CALL are translated into the binary form shown in TABLE I.

2. A routine in Console ROM is activated by assembly language instructions and code as shown in the Hex column. Again data are translated into the binary form of TABLE I.

3. The sound-chip has a direct access address port. Hex data bytes can be moved to this address by assembly language routines. They are interpreted in the binary form.

4. The CALL LOAD() routine in grom of the XBASIC, MINI MEMORY and ED/AS SS Modules will send data bytes from a Basic program directly to sound-chip. Decimal data is converted to binary form.

All four sound producing methods can be accessed from a BASIC program if an appropriate SS Module is installed in the console.

Because the sound-access methods differ in ease of use, code complexity, functional speed and dynamic flexibility some combined use of them in a program can be attractive. Such a scheme is entirely practical. The sound-chip has no time-dynamic function of its own other than tonality and loudness. Once the registers are set, the tone pitches, noise quality and loudness remain fixed until another set of data is written in. No auto-sound-sprites here! So, the burden of making neat, informative and distinctive sound signals rests right on the writers shoulders. And don't burn up all the time and memory overhead doing it.

Reference Material

Editor/Assembler Manual, Texas Instruments, Sound pgs 312-317.
Users Reference Guide, Texas Instruments, CALL SOUND Subprogram pgs II84-85.
TI 99/4A Intern, Heiner Martin, CALL SOUND GPL code, pgs 164-165. Interrupt Routine, sound processing segment, ROM code >09EC->0A64, pgs 31-32.
Beginner's Guide to Assembly Language on the TI-99/4A, Peter Lottrup, Chapter 10 Generating Sounds, pgs 167-180.

Cecure takes over CC-40 service

Cecure Electronics is scheduled to take over as the only TI-authorized repair and service center for the CC-40 as of Oct. 1, according to Don Walden of Cecure.

Cecure became the authorized repair and service center for the TI99/4A Sept. 1.

Walden says TI has agreed to his reducing prices on repairs and services. The company has been reviewing to ascertain for which items reductions will be appropriate.

Walden notes that the CC-40 is popular in Europe. Some users use a German hexbus interfacing it with the TI.

"In many cases it's more compact than a P-Box," he says. "There are a number of items which can be daisy-chained."

Walden notes that he has "close to 100" color monitors for the 99/4A, whereas he was expecting one or two dozen. He says he has a lot of documentation, "so people who purchase the 4A at a garage sale with nothing can get the docs from us."

For further information, contact Cecure Electronics, P.O. Box 222, Muskego, WI 53150; 1-800-959-9640 or (413) 679-4343 (voice); or (414) 679-3736 (BBS).

Support MICROpendium advertisers
They support you

THE ART OF ASSEMBLY — PART 39

More mysteries unraveled

By BRUCE HARRISON

In last month's column, we teased you a bit by mentioning that we'd gotten our compiler to report errors, breakpoints, and other messages by the line numbers from the original Extended BASIC program, but didn't say how that was done. Today we'll fill that gap for you, and also clear up some other "pending" mysteries we've mentioned in previous columns.

FOOLING XB

In our work on the compiler, we became concerned about the business of error reporting, and the fact that if errors were reported they would most often be reported as occurring in line 32767, because that's the line of the "shell" XB part of the compiled program that XB "thinks" it's executing. If we left matters like that, a user of our compiler would be left to guess where in his source XB program this error might be tracked down. Thus we set out to find some way of "fooling" XB into reporting the error as, for example, NEXT WITHOUT FOR IN 125, where 125 would be a line that doesn't exist in the compiled program, but that's where the original XB program would report this error.

We knew of course that XB has to "know" what line it's currently executing, so we got out some reference material to track down where that information gets stored. The address >832E was listed as "Pointer to current line number in line number table." That seemed a likely place to look, so we arranged a test by typing in a small XB program like this:

10 BREAK 20
20 GOTO 20

This program of course won't really do anything except run through line 10 and then stop with the report "BREAKPOINT IN 20." When that happened, we typed in CALL PG, to get into our P-GRAM program that allows us to examine memory. Sure enough, there was an address number in >832E that started with >FF. This looked promising, as the line number table for such a short program would surely be in the >FF area. We then looked at that address, and found there another number in the >FF range. This was most certainly not the line number. It was the address of the line itself. Sure enough, if we looked two bytes back from the address in >832E, we were at the location in the line number table that contained >0014, (decimal 20). That's what we were after.

Now we suspected that XB might determine the line number for breakpoints and such by taking the address from >832E, subtracting two, and then taking the number from that address to report the line number on the screen. Thinking that such is the case and proving it are, of course, two very different things. We set up an experiment in one of our test XB programs (this one always ends with an error because it was designed to do just that), then patched up the compiler's code so that the following would happen:

1. The compiler would record the current line number as a

(See Page 11)

Sidebar 39

```
* SIDEBAR 39
* FIRST IS A SMALL PIECE FROM
* SOURCE CODE CREATED BY THE
* COMPILER
*
* EACH LINE OF THE XB PROGRAM
* IS IDENTIFIED BY AN "L" PLUS
* THE LINE NUMBER BEING PROCESSED
* HERE IS SOURCE CODE CREATED BY
* LINES 20 THRU 30 OF A DEMO PROGRAM
*
L20    BL   @SETCL    USE SUBROUTINE SETCL
       DATA 20        TO SET CURRENT LINE NUMBER
       BL   @FORSET   SET UP A FOR-NEXT LOOP
       DATA 1         FROM 1
       DATA 5         TO 5
       DATA 1         STEP 1
       DATA IV0       VARIABLE IV0 (I IN XB)
       DATA >0000     ALL PARAMETERS JUST NUMBERS
LM0    DATA 0,0       STORAGE FOR LOOP COUNTING
       BL   @LIMCHK   CHECK LIMIT ON THIS LOOP
       DATA NX0       IF FINISHED, GOTO LABEL NX0
       BL   @IVTFP    CONVERT IV0 INTO FLOATING POINT I
       DATA IV0       (ALLOWS I TO BE USED BY FL #4, BELOW)
       BL   @TOGI     USE GPL INTERPRETER [READ A$(I)]
       DATA FL4       WITH FAKE LINE #4
       BL   @IVTFP    CONVERT AGAIN
       DATA IV0       IV0 TO F.P. VARIABLE I
       BL   @TOGI     USE GPL INTERPRETER
      DATA FL5       WITH FAKE LINE 5 [DISPLAY AT(I*2+5,6)...]
       BL   @INCLV    PERFORM "NEXT I"
       DATA LM0       WITH DATA AT LABEL LM0
NX0
       BL   @TOGI     USE GPL INTERPRETER
       DATA FL6       FOR FAKE LINE 6 [DISPLAY AT(24,6)...]
L25    BL   @SETCL    SET CURRENT LINE
       DATA 25        AT 25
       BL   @KEY      "CALL KEY" (SUBROUTINE KEY NOT SHOWN)
       DATA 0         0 (KEY-UNIT 0)
       DATA 18        K (KEY VARIABLE K)
       DATA 22        S (STATUS VARIABLE S)
       DATA >0220     K AND S ARE FLOATING POINT VARIABLES
       BL   @TOGI     USE GPL INTERPRETER
       DATA FI0       FOR FAKE IF #0 [IF S<1]
       MOV  R1,R1     CHECK R1 TRUTH INDICATION
       JNE  EL0       IF R1 NOT ZERO, STATEMENT FALSE
       B    @L25      IF TRUE, GOTO LINE 25 [THEN 25]
EL0
* IF STATEMENT WAS NOT TRUE, CODE FOLLOWING EL0 WILL EXECUT
L30    BL   @SETCL      SET LINE NUMBER
       DATA 30          AT 30
       BL   @ONGTS      USE SUBROUTINE ONGTS
       DATA NE0         NUMERIC EXPRESSION #0 [K-48]
       DATA >8405       CONTROL WORD
       DATA LU0         LOOKUP TABLE #0
* CONTROL WORD DECODES AS FOLLOWS:
* 8 MEANS THIS IS ON-GOSUB, NOT ON-GOTO
* 4 MEANS A NUMERIC EXPRESSION (K-48) AS ARGUMENT
* 5 MEANS THERE ARE 5 BRANCH LINE NUMBERS
*
* LU0 IS THE LOOKUP TABLE AS FOLLOWS:
LU0    DATA L120        ADDRESS OF LINE 120
       DATA L200          "   "   "  200
       DATA L230          "   "   "  230
       DATA L300          "   "   "  300
```

THE ART OF ASSEMBLY —

(Continued from Page 10)

DATA entry in the source code it creates.

2. The compiled program would place that data item at some convenient location, called CLNUM.

3. When an error was to be reported, the compiled program would load a register with CLNUM+2, then move that register to >832E before its BLWP @ERR.

In theory, this meant that the ERR routine would take that address from >832E, subtract two from it, then take the number from the resulting address and report that number on screen. The theory in this case was exactly right! The snippets in today's sidebar show how this process worked for error reporting, for breakpoints, and for error or warning messages in "Fake Line" processing by the compiled program.

THE ERR REPORT

Okay, here's another mystery we can clear up. On page 416 of the Editor/Assembler manual there's a long list of equates, some of which are left sort of unexplained. ERR EQU >2034 was one that baffled us for some time, especially since the next page contains a long list of possible error reports complete with addresses, but there didn't appear to be any way to connect the two things. By a mostly trial-and-error process, we have doped out just exactly how to use the numbers from that error message list and the ERR equate, so that, when we're operating from an XB environment, we can use XB to report the selected error message.

It's just this simple. Take that address from the table on page 417 for the message you want reported. (e.g >1E00 for BAD VALUE) Write this into your source code:

```
LI  R0,>1E00
BLWP @>2034
```

That's it! When these two lines execute, XB will produce the "boop" sound, and will report BAD VALUE IN XXX, where XXX will be the line number. The code that's used by the vector at >2034 takes whatever was in the caller's R0 and uses that to determine which message it will print.

SOME DAYS IT'S EASY

Shortly after discovering the answers we've just discussed, we were fooling around with one of our compiled test programs, in which we'd placed an INPUT statement to get a value for one of the variables used in the program. This INPUT would go into a numeric variable, which our program would then use as the limit value for a FOR-NEXT loop. In our compiler, INPUT is simply passed along to the GPL Interpreter to perform, since there's no speed advantage to be gained from replicating the INPUT process. Thus the source for the compiled program uses a BL @TOGI to execute a Fake Line (see last month's column) for the INPUT statement.

While running the compiled program, we decided to be daring and put an illegal entry in at this input prompt. We typed Q <Enter> at the prompt. What happened surprised and delighted us. We got the boop, a report saying WARNING - INPUT ERROR IN 95, and the prompt on the screen just below that. The number 95 was indeed the line number from the original XB program, but what would happen next? We typed in 20 <Enter> to answer the

(See Page 12)

```
      DATA L40              *  *  *  40
*
* THE ABOVE SECTION OF DEMO PROGRAM
* XB CODE LISTED IN 28 COLUMNS
* READS DATA TO PUT A MENU ON SCREEN
*
20 FOR I=1 TO 5 :: READ A$(I
):: DISPLAY AT(I*2+5,6):A$(I
):: NEXT I :: DISPLAY AT(24,
7)BEEP:"SELECT BY NUMBER"
25 CALL KEY(0,K,S):: IF S<1
THEN 25
30 ON K-48 GOSUB 120,200,230
,300,40
*
* NEXT PART IS FROM THE "RUNTIME" ROUTINES
* WHICH THE COMPILED PROGRAM USES
*
* SUBROUTINES USED IN COMPILED PROGRAM
* 24 SEP 93
* STORED AS STDSUB
*
TOGI   MOV *R11+,@>832C  PUT FAKE LINE ADDRESS AT >832C
       LI  R1,CLNUM+2    POINT AT CLNUM + 2
       MOV R1,@>832E     PLACE THAT ADDRESS AT >832E
       LWPI >83E0        LOAD THE GPL WORKSPACE
       B @>006A          BRANCH TO GPL INTERPRETER
*
* LABEL TRUE IS USED WITH AN IF-THEN
* SO THAT IF THE "IF" IS TRUE, OUR REGISTER 1
* WILL BE CLEARED.  OTHERWISE, THE "IF" WILL
* RETURN TO US AT LABEL BACK, AND R1 WILL STILL
* CONTAIN THE ADDRESS WE LOADED INTO IT IN TOGI
*
TRUE   CLR  @WS+2       CLEAR OUR REGISTER 1 IF STATEMENT WAS
TRUE
BACK   LWPI WS           RETURN FROM A FAKE LINE TO OUR WORK-
SPACE
       RT               THEN BACK TO ASSEMBLY CODE
SUBRET DECT R15          SUBTRACT 2 FROM OUR STACK POINTER
       MOV  *R15,R11     MOVE WORD INTO OUR R11
RETURN LIMI 2            ALLOW INTERRUPTS
       LIMI 0            THEN STOP THEM
       LWPI >83E0        LOAD THE GPL WORKSPACE
       BL   @>20         CHECK FOR FUNCTION-4 KEYPRESS
       JNE  RET0         IF NOT, JUMP AHEAD
       LI   R6,CLNUM+2   ELSE SET FOR BREAKPOINT
       MOV  R6,@>832E    WITH ADDRESS CLNUM+2
       LI   R6,CONLIN    SET FOR A "CONTINUE"
       MOV  R6,@>832C    WITH GPL INTERPRETER
      B   @>6A          THEN BRANCH TO GPL INTERPRETER TO BREAK
RET0   LWPI WS           LOAD OUR OWN WORKSPACE
       RT                THEN RESUME AT ADDRESS IN R11
SETCL  MOV   *R11+,@CLNUM MOVE THE DATA AFTER THE BL INTO
CLNUM
       RT                THEN RETURN
GENERR LI   R1,CLNUM+2   GET LINE NUMBER POINTER
       MOV  R1,@>832E    INTO >832E FOR ERROR REPORT LINE
       BLWP @ERR         (ERR EQUATED TO >2034, NOT SHOWN)


*  ONGTS ON-GOTO / ON-GOSUB
*  HARRISON COMPILER
*  25 SEP 1993
*  STORED AS ONGTS
*
*
ONGTS  MOV  *R11+,R2      FIRST DATA INTO R2 (ARGUMENT AD-
DRESS)
       MOV  *R11+,R13     SECOND INTO R13     (CONTROL WORD)
       MOV  *R11+,R3      THIRD INTO R3       (LOOKUP TABLE AD-
DRESS)
```

THE ART OF ASSEMBLY—

(Continued from Page 11)

prompt with a numeric response, and our compiled program went on about its business just as it normally would!

Why did this work? What XB apparently did after issuing the warning was to go back to the start of our INPUT fake line, and simply execute it again. When we answered the prompt with a legal input, XB continued the fake line, to wit :: GOTO 32767. That took us back into the Assembly code right where it should, and our program performed as expected. Flushed with success, we went back into our original XB test program, put in a line before No. 95 that said ON WARNING NEXT, then saved that in merge format for compiling.

When compiled, this too behaved exactly as it should. When the bad entry was made, no boop or warning message appeared, but the screen scrolled up and the input prompt was repeated. We made a correct entry, and the program continued from there. Incidentally, ACCEPT AT will behave in a similar manner, except that with the ON WARNING NEXT in effect, there will be no screen scrolling, and the input field will be cleared for a new input value.

AND THEN OTHER DAYS

Actually, it was the very same day, in this case. Having proved that our faking of line number reporting worked for WARNING messages as well as for ERROR reports, we tried yet another little test, starting with the original XB program just discussed. At the input prompt, we entered 0. The XB program then completely skipped the FOR-NEXT loop, and went on to what followed that. Unfortunately for us, that's *not* what happened in the compiled version.

Our Compiler's FOR-NEXT implementation was designed to handle almost every possible error, but in this case it went ahead and executed the FOR-NEXT loop once, then went on to the instruction after the NEXT. This happened because we were not checking the state of the index variable against the limit value until we got to the NEXT part of the loop. Once again it was back to the drawing board for our FOR-NEXT in the compiler.

That's just one small example of why things get messy when trying the impossible. Just before this happened, we had run through a similar exercise on the PRINT function. Just when we thought the PRINT was working okay, we remembered that there's a function called TAB. (Expletive Deleted!) Two whole days were eaten up with that little problem. While we were about it, though, we also took care of the cases like this:

PRINT ,"Hello There"

This skips over halfway across the screen before printing Hello There, unless there was a comma at the end of the last PRINT statement, in which case the screen will scroll and Hello There will appear at the start of the next line. You get the message by now. Once one decides to implement an XB function, one must go in and check out all the possible variations, including PRINT with no argument, PRINT :"Something," and so on.

This can lead to some surprising results. For example, does anyone know what happens if you ask XB to do the following?

PRINT TAB(33);"Hello"

(See Page 13)

```
       MOV  R11,R14      STASH R11 RETURN ADDRESS IN R14
       MOV  R13,R12      PUT R13 INTO R12
       SRL  R12,8        SHIFT RIGHT 8 BITS
       ANDI R12,7        MASK ALL BUT LOWEST THREE BITS
       JEQ  ONGTE1       IF ZERO, AN ERROR CONDITION
       CI   R12,1        COMPARE TO 1
       JNE  DECCV2       IF NOT ONE, JUMP AHEAD
       MOV  *R2,R2       ELSE GET INTEGER VARIABLE VALUE
       JMP  SETCV        THEN JUMP
DECCV2 CI   R12,2        COMPARE TO 2
       JNE  DECCV4       IF NOT, JUMP AHEAD
       AI   R2,VARTBL-2  SET UP FOR F.P. VARIABLE
       MOVB *R2+,R9      FIRST BYTE OF F.P. ADDRESS
       SWPB R9           SWAP
       MOVB *R2+,R9      SECOND BYTE OF F.P. ADDRESS
       SWPB R9           SWAP AGAIN
       BL   @MOVFAC      USE A SUBROUTINE (NOT SHOWN)
       MOV  @FAC,R2      GET INTEGER VALUE INTO R2
       JMP  SETCV        THEN JUMP
DECCV4 MOV  R2,@FAC      STASH ADDRESS OF NUMERIC EXPRESSION
AT FAC
       BL   @INTERP      INTERPRET EXPRESSION
       MOV  @FAC,R2      MOVE INTEGER IN TO R2
SETCV  ABS  R2           TAKE ABSOLUTE VALUE
       MOV  R13,R12      MOVE R13 TO R12
       ANDI R12,>00FF    USE ONLY LSB
       C    R2,R12       COMPARE
       JGT  ONGTE2       IF R2 > R12, ERROR (BAD VALUE)
       DEC  R2           DECREMENT R2
       JLT  ONGTE2       IF <0 THEN BAD VALUE
       SLA  R2,1         DOUBLE R2
       A    R3,R2        ADD LOOKUP TABLE ADDRESS
       MOV  *R2,R5       GET THE GOTO-GOSUB ADDRESS IN R5
       MOV  R13,R12      MOVE R13 AGAIN
       JGT  ONGSX        IF POSITIVE, THIS IS ON-GOTO
       MOV  R14,*R15+    ELSE IT'S ON-GOSUB, SO STACK RETURN
ADDRESS
ONGSX  B    *R5          BRANCH TO THE ADDRESS IN R5
ONGTE1 LI   R0,>0300     ERROR REPORT WILL BE "SYNTAX ERROR"
       JMP  ONGTEX       JUMP AHEAD
ONGTE2 LI   R0,>1E00     ERROR REPORT WILL BE "BAD VALUE"
ONGTEX B    @GENERR      BRANCH TO ERROR REPORTING CODE
*
* END OF COMPILER SUBROUTINE FRAGMENTS
*
* FOLLOWING IS SHORT XB PROGRAM
* USED TO TEST VPUSH/VPOP
* THROUGH LINKAGE TO ASSEMBLY
*
1 ! TEST OF VPUSH/VPOP
2 ! USE WITH PUPOP/O
3 ! EXTENDED BASIC ONLY
10 CALL INIT
20 CALL LOAD("DSK1.PUPOP/O")
30 A=48.254 :: PRINT A
40 CALL LINK("TEST",A)
50 PRINT A
*
* FOLLOWING IS SOURCE FILE PUPOP/S
* TO USE, ASSEMBLE INTO PUPOP/O,
* THEN RUN WITH ABOVE XB PROGRAM
*
* PUSH/POP TEST
* DSK1.PUPOP/S
* WITH XB
* ENTRY LABEL TEST
*
* REQUIRED EQUATES
*
NUMREF EQU  >200C        NUMERIC REFERENCE
NUMASG EQU  >2008        NUMERIC ASSIGNMENT
XMLLNK EQU  >2018        XML LINK VECTOR
```

THE ART OF ASSEMBLY—

(Continued from Page 12)

Harry Wilhelm knows because I told him, but who else out there knows? (The Shadow Knows!) If the number in the TAB argument is greater than 28, XB will simply subtract 28 from it, and keep doing so until it's less than 28, then will execute a tab to this number position in the current line. For the above case, 33-28=5, so the "H" in Hello will be at the fifth position on the bottom line of the screen. TAB(61) would produce exactly the same result, since XB would subtract 28 twice to arrive at the number 5. Things like this can just drive a person crazy. Harry Wilhelm was of the opinion that if the TAB has a number greater than 28, it should cause the screen to scroll enough times to make up the numbers above 28. Instead, we put a simple loop into our PRNTAB routine so that it will exactly match the performance of XB's TAB.

BUT WHAT IF...?

One of the problems in this process of compiling is the very richness of the language. This means that for each new thing we try to implement as a compiled function, there are many potential variations on how the programmer might use that function, and we have to consider each and every one. In almost every situation where a number can be used in a program, for example, that can be either a simple number, a numeric variable, or numeric expression. We have further complicated things for ourselves by creating the concept of the integer variable, so there are four possible things that can supply the numbers for our functions.

This means that for functions like the setting up of a FOR-NEXT or CALL HCHAR, or even CALL KEY, the compiler has to determine which kind of thing each parameter is, and then has to put something into its source file so that the "runtime" routine will know this, and can properly handle each possible kind of parameter.

To let our compiled program know what's up, we use a data word in which each nybble is coded separately, thus maximizing our memory efficiency. For example, the routine that does both CALL HCHAR and CALL VCHAR has a maximum of four parameters. Each of these can be a simple number, an integer variable, a floating point (XB) variable, or a numeric expression. To tell our routine what type each parameter is, we use the four nybbles of one word. The right nybble contains 0 if the ROW is a number, 1 if it's an integer variable, 2 if it's a floating point variable, and 4 if it's a numeric expression. Each of the other three nybbles is coded in similar fashion to indicate the kind of thing that COLUMN, CHARACTER, and REPEAT parameters are. This same process is applied to the parameters for the FOR setup in FOR-NEXT, and for the parameters in CALL KEY.

JUST ONE MORE THING

To quote Peter Falk as Columbo, just one more thing to clear up another of those little mysteries. The Equates on page 416 of the E/A manual include VPUSH and VPOP. We thought these might mean pushing and popping numeric values to and from a stack in VDP RAM. We performed a little experiment using the source code shown in the sidebar and, sure enough, VPUSH, executed through XMLLNK, took the floating point number from FAC and stashed it on a stack in VDP RAM. By clearing FAC, then doing a VPOP and reassigning this value to an XB variable, we proved that this worked. But don't get too confident about such things. If you've pushed a variable in this way and then returned control to XB before coming back to POP the variable, the chances are pretty good that you'll have lost your value. This happens because XB makes its own use of the VDP value stack, so by the time you're back in control, the pointers will have been reset, and your original value on the stack will be gone.

```
VPUSH  EQU  >000E        VDP PUSH
VPOP   EQU  >0010        VDP POP
FAC    EQU  >834A        F.P. ACCUMULATOR
*
* CODE SECTION
*
       DEF  TEST         DEFINE ENTRY POINT
TEST   LWPI WS           LOAD OUR WORKSPACE
       BLWP @NUMREF      GET PARAMETER (A FROM XB)
       BLWP @XMLLNK      USE XML
       DATA VPUSH        TO PUSH FROM FAC ONTO STACK
       CLR  @FAC         CLEAR FAC
       BLWP @XMLLNK      USE XML VECTOR
       DATA VPOP         TO POP VALUE A FROM STACK
       BLWP @NUMASG      RE-ASSIGN TO VARIABLE
       LWPI >83E0        LOAD GPL WORKSPACE
       B    @>6A         RETURN TO GPL INTERPRETER
*
* DATA SECTION
*
WS     DATA 0,1          PRELOADED REGISTERS 0 AND 1
       BSS  28           REST OF WORKSPACE
       END
```

Thus take a warning. You can use VPUSH to temporarily stash the value of a floating point number, and it can be recovered as long as your Assembly code is still in charge. However, if you return control to XB, you should first pop out that floating point number and put it somewhere else, lest it get lost while XB is using this same stack.

We have a planned topic for next month which may be of some interest to the non-programming public among our readers. We have recently devised a way of taking an existing Option-3 object file and converting it to an Option-5 program file without having access to the original source code, and without doing any exotic maneuvers with assembly. See you then!

Want to contact MICROpendium by phone?
Call us Saturday mornings, from 9 a.m. to noon, Central Standard Time
The number is 512-255-1512

Extended BASIC

Cardfile used to produce database using index card format

By LUCIE DORAIS

TI CARDFILE is an index card program, a database with only two fields: the title, or index, and the text. It's not unlike 3x5 index cards, but with the added facility of automatic editing, sorting and printing. CARDEDIT, a companion program, is a much better text editor (I kept the CARDFILE editor basic to save memory); CARDUTIL, another companion program, is a collection of utilities to change the size of the cards, merge, split and clean files, etc.

From the main menu, you can start a N)ew file or O)pen an existing one, and Q)uit. The cards can be B)ig (18 screen lines of text) or S)mall (9 lines); so if all you need can fit in nine lines, like an address book, you should use the small size to save space on the disk (each small record uses about one sector for the data, the big one just over two); the index, i.e. the title list and the pointers to the data records, is kept in a separate file. Each file has space for about 275 cards (if you have the disk space of course); the program does no checking of your available disk space, so it will be up to you to stay inside your system's limits.

When you load an existing file, you can A)dd to it, F)ind a specific card, S)ort the titles, P)rint all or part of the file and move from one card to another. When the title of the card you want to see is displayed on the screen, you can V)iew the card, then you can E)dit, D)elete or P)rint it, and again move to another card. Since the titles are kept in memory, moving through the file in the File menu is faster than moving in the Card menu, because each new card has to be read from disk.

The files, both RELATIVE and DISPLAY (so we can use LINPUT to read them: see XB manual at LINPUT), are organized as follows:

(Note that the Last RECord saved will not necessarily be the one corresponding to the total number of cards, because when you start deleting and adding, it will be different, see Fig. 1.)

NOTE: If you don't have a printer, remove the references to file #3 in lines 150 (OPEN), 220 and 1030 (CLOSE). Because the original program was published over two months, the comments on SORT (lines 540-600), PRINT (lines 610-670, 870, 1200-1230) and DELETE A CARD (lines 775-860 and 680-720) are placed after the comments on the main portion of the program and the subroutines.

We start with the safety measures and the DIM; in line 140, char. 136 is an octothorpe (#) in inverse video. The characters defined in line 170 using the strings in line 120 are the contours of the card; TOP$ and IND$ in line 140 are IMAGEs to be used later. The main menu is found in lines 190-220 (D$ is the default data disk number), then the N)ew file portion: SUB ASKF asks for the filename, OPIF and OPDF open the Index and Data files respectively. Then you are asked for the size, which will be kept in the BIG variable. All variables are reset by line 260 (DREC,LREC=minus BIG, therefore - 1 for a big file, so that the next record will be 1 when incremented by 2 in line 380); DEL$ is the deleted cards list, empty for now.

Fig. 1

```
2) DATA file: suffix "/D", FIXED 252:
^   REC(0): xxx nnnnnnnnnnn...
^           xxx .............  no/yes (del list)
^           ... nnnnnnnnnnn...  list (4 char/rec)
^   REC(1-n):  data, one record / card if BIG=0
^                  two records / card if BIG=1
```

When you O)pen an existing file, the error trapping routine will catch you if the filename you give does not exist; since naive Tex has already opened the files before it traps the error, the two new files will be erased (line 1020) before control is returned to you. If you gave a good filename, the file information is read from the /I file by line 290, and the titles put in the CT$ array for faster processing; then the delete list is retrieved from the /D file. ERCARD erases the card and GOSUB 1000 modifies the screen display, both according to the card size.

The File menu is displayed by lines 310-340, with info on the file name and sort status. GOSUB 960 uses the IMAGE TOP$ to display the title of the first card and its card number, preceded by the inverted "#". (This number is the order of the card in the file, not the actual data record number for the corresponding data in the /D file, which is permanent; the card numbers will, of course, change when you sort or delete cards.)

You fill a card (lines 360-480) when you start a N)ew file or when you A)dd to an existing one; C$ is a flag to tell Tex where to return when you're finished; of course the sort flag is reset since adding new titles will likely upset their order if any. The SUB ACC accepts a string at a row and pads it with spaces, for a total length of 23 for titles, 28 for text lines. ASCII 47 is the "/" you must enter to terminate the A)dd function. Line 380 increments the Number of cards, the Current Card number (last one of course) and GOSUBs 960 to display the new title and the card number. If the DEL$ list is empty, the next record in the /D file will also be the last one, and both are incremented by one for a small card, by 2 for a big one. Lines 390- 400 deal with a delete list if there is one.

The title is then formatted with its corresponding data record number (pointer) and promptly printed to the /I file as well as being kept in the CT$ array. The text of

(See Page 15)

CARDFILE—

(Continued from Page 14)

the card is then entered; for fast disk access, it will be saved in one or two strings of 252 characters each (nine screen lines); each screen line is accepted and padded with spaces. You must end with a "/", unless you are on the last line; that character is replaced with ASCII 127, which shows as a blank on screen, so that when you view a card it will not display, but will be there to put back a "/" on screen when you edit the card later. The value of P in line 430 is 1 or 2 according to the screen row, to fill the first and, if the card is big, the second string. You can then redo the title and the data if they are not OK. This is where the term "basic" applies to the editor: you have to ACCEPT all lines one by one again. CARDEDIT is faster, with the ability to Insert or Delete lines, and Reformat somewhat. If the data is OK, it is printed to the /D file at record number DREC; if the card is big, the second part is printed at record DREC+1.

Lines 490-510 are to F)ind a card: enter the first letter(s) of the index title (for example, in a sorted address file, you can reach the Cs by typing only "C"); if Tex finds a card, it goes to the V)iew function immediately, bringing the current card C value; if not, it beeps at you. Line 520 moves you quickly to the T)op (first card) or B)ottom (last) of the file, while line 530 moves forward (>) or backward (<); no need to press FCTN, SUB AKEY looks for"." and ",". All this is only at the index titles level, very quick because the titles are in memory. Line 730 will save the file information to the /I and /D files (GOSUB 990) before closing them and returning you to the main menu.

Lines 740-920 will work on an individual card. When you press "V" in the file menu, or you have used F)ind, the screen becomes white, the current title and card number are shown, and Tex reads the corresponding Data RECord kept in the right portion of the title in the array. Being nice, it will read the data from the disk and display it, according to the card size (GOSUB 930). The menu is simple: E)dit, D)elete, P)rint, M)enu, and "< >" to move forward or backward. Edit is as basic as above, but Tex will replace the end of text character 127 with a "/" (line 790) so you don't have to type it again, unless you add lines; Tex also resets the SORT$ flag if your new title is different from the old one, just in case... then you are sent back to line 410 to edit the text lines; flag C$ tells Tex to come back here when you are finished. Line 840 is used for moving among the cards; as said above, this takes longer here because each card has to be read from disk.

Notes on some subroutines: Line 970 is used with S)ort a file and D)elete a record (see below); in line 990, the variable CL is set to one when Tex needs to close the files (upon exiting to main menu or before deleting bad filenames). The Escape sub in line 1025 is used by SORT, PRINT and DELPACK functions.

In the error trapping routine (lines 1010-1030), the K reports the error number, the L the line number; an error number 130 is always an I/O error, in this program it should happen only when you try to open a file that does not exist, so we RETURN 270 after having closed and deleted the files opened by Tex by error; any other error will cause Tex to print the file information to the disk before it closes the files and breaks.

So that you don't have to worry about the Alpha Lock key, in the user-defined SUB AKEY (lines 1120-1150) a formula transforms any lower case letter into its upper case; in the sub ASKF, there are two dummy CALL KEYs: CALL KEY(3,...) causes Tex to change all lowercase to uppercase when entering a filename; to be able to accept lowercase later, another dummy CALL KEY(5,...) resets Tex to full BASIC Keyboard; our standard value of CALL KEY(0,..) simply tells Tex to use the *last* keyboard value used, which would be 3 if not for the second CALL KEY. In the SUB ASKF, there are eight spaces before the "/D", for the filename.

SORT (lines 540-600): it is the proven and fast (for TI) Shell Sort (Regena's *Guide to the TI99/4A*, p. 295); we reuse some variables that are not used otherwise in this portion of the program (P,S,K,X). The new title list is saved to disk (GOSUB 970), starting with the first card (S=1, line 570).

PRINT (lines 610-670, 870, sub 1200-1230): no word wrap (lines are identical to those on screen), but some fancy typesetting, and automatic page breaks between cards.

You can print all cards (default 1 and N in line 620) or a selected portion. The screen becomes pale red, and each printed card's title and number are displayed at the top (GOSUB 960); the data is retrieved from disk for each card (GOSUB 930). Lines 640-660 control page length (kept in L): by finding the "end of data marker" (ASCII 127), Tex calculates the number of lines needed in ND (data lines, plus two for the title and three empty lines after the card: see SUB PCARD below); if there is a marker in the first (or unique if small file) string, Tex goes immediately to line 660: the length of text (position of marker minus it) is divided by 28; if not, that portion has nine lines, so ND becomes 14 and Tex looks for the marker again. If there is none (the string has nine lines, or by accident there is no marker), P=253 makes sure that ND will calculate nine lines for the CD$(2) string. If the next card would cause page length L be beyond 62 lines, Tex prints a page break (12) and resets itself. Each card is then printed by the SUB PCARD; after a last page break, the first printed card title (C=S) is shown on screen.

You can also print a card while V)iewing it, but no page length checking (line 870). Finally, the SUB PCARD itself, lines 1200-1230: note that the CD$() array is passed whole to the D$() one.

The title and a line of "-" are printed in Bold (CHR$(27), ESCAPE, followed by "E" to start Bold and "F" to end it: TI/Epson codes, check your manual for other printers), then the card number preceded by a ">" at the right, then the text itself, indented to the right to make the printed cards easier to read; the text has to be taken from the CD$ strings by chunks of 28 char.; when Tex finds the end marker, it goes to line 1230 to put three empty lines before exiting the sub. (NOTE: Since the cards marked for deletion are also printed, you might want to use the D)el Pack function from the main menu to get rid of them.)

DELETE A CARD (lines 775-860, 680-720): Each card has to be deleted in-

(See Page 16)

CARDFILE—

(Continued from Page 15)

dividually, from the card menu (you *must* V)iew it first). Since the files can get pretty big, the program keeps track of the deleted records so that they can be used again (CARDUTIL has a C)lean file function); remember, the Data RECord number is always appended to the card title, and the order of the cards (the CT$ array) is totally independent in numbering. The DELETE function is done in two parts: first you "mark the card for deletion": a special code, "*D*", is inserted in the title and is visible on screen; you can always change your mind (recall a card) by pressing E)dit: clever Tex knows that it is deleted, and line 775 will GOTO 850 to edit only the title, after erasing the *D* from screen so that it is not ACCEPTed into A$.

The deleted cards, still part of your files, can be V)iewed, S)orted and P)rinted like any other, so at some point you will want to R)emove them: this is done from the file menu (lines 680-720).

The DEL$ (delete) list is retrieved if there is one already, otherwise initiated to "yes" (line 690). Tex reads all the cards; if the current card X is not marked for deletion (line 700), the "good cards" counter C is incremented and the next title in line, CT$(C), is filled with the "good card" title (this removes the deleted card title from the title array). If the card is marked for deletion, its corresponding Data RECord is extracted from the title, appended to the DEL$ list, and Tex writes "*del*" on the disk (line 710). At the end of the index file, the total number of cards N takes the value of the highest "good card" number C, and the new DEL$ list is promptly printed to disk (GOSUB 1050). Finally, Tex saves the redone title list to disk; to save time, GOSUB 970 starts at the S value: here it has the value of the first deleted card encountered in the loop, thanks to the S=N+1 in line 690, and to the MIN(S,X) function in line 710.

CARD

```
100 ! ***** CARD ***** L.Dor
ais/Ottawa UG/May 1993 !068
120 A$="000000" :: B$="18181
8" :: C$="1F1F" :: D$="F8F8"
 :: S$=RPT$(" ",10)!088
150 GOTO 170 :: B$,K,P,S ::
CALL KEY :: CALL CLEAR :: CA
LL SOUND :: CALL SCREEN :: C
ALL CHAR :: CALL VCHAR :: !@
P- !091
170 CALL CHAR(128,A$&C$&B$,1
29,A$&D$&B$,130,B$&C$&A$,131
,B$&D$&A$,132,B$&C$&B$,133,B
$&D$&B$,134,A$&"FFFF"&A$,135
,B$&B$&"1818")!078
190 CALL CLEAR :: CALL SCREE
N(12):: CALL VCHAR(1,2,135,2
2):: CALL VCHAR(1,31,135,22)
!037
200 CALL LINE(1,128,134,129)
:: CALL LINE(3,132,134,133):
: CALL LINE(22,130,134,131)!
229
210 DISPLAY AT(2,11):"TI CAR
D" :: DISPLAY AT(6,11):"F)il
e": :S$&"E)dit": :S$&"U)til"
: : :S$&"Q)uit" !226
220 CALL KEY(0,K,S):: IF S=0
 THEN 220 ELSE B$=CHR$(K)::
IF ASC(B$)>96 THEN B$=CHR$(A
SC(B$)-32)!184
230 P=POS("FEUQ",B$,1):: IF
P=0 THEN CALL SOUND(100,200,
0):: GOTO 220 !248
240 ON P GOTO 250,260,270,28
0 !213
250 CALL D("FILE"):: RUN "DS
K1.CARDFILE" !124
260 CALL D("EDIT"):: RUN "DS
K1.CARDEDIT" !136
270 CALL D("UTIL"):: RUN "DS
K1.CARDUTIL" !184
280 CALL CLEAR :: END !222
1060 !@P+ ==== user-def subs
 ===== !187
1070 SUB LINE(R,A,B,C):: CAL
L HCHAR(R,2,A):: CALL HCHAR(
R,3,B,28):: CALL HCHAR(R,31,
C):: SUBEND !148
1080 SUB D(A$):: DISPLAY AT(
24,5):"Loading CARD"&A$&"...
" :: SUBEND !181
```

CARDEDIT

```
100 ! ***** CARDEDIT ***** L
.Dorais/Ottawa UG/March 19
 !046
110 ON WARNING NEXT :: ON ER
```

(See Page 17)

CARDFILE—

(Continued from Page 17)

```
740 A$=SEG$(A$,1,Y+1)&SEG$(B
$,1,P-1):: CALL PAD(A$,28)!
add word !048
750 B$=SEG$(B$,P+1,28):: CAL
L PAD(B$,28)! remove word !2
50
760 IF B$<>E$ THEN Y=29 :: G
OTO 720 ! continue to look f
or words !007
770 FOR Y=X+1 TO 9+B9 :: L$(
Y)=L$(Y+1):: NEXT Y :: GOTO
710 ! rewrite data !117
780 L$(X)=A$ :: L$(X+1)=B$ :
: NEXT X :: L$(10+B9)=E$ ! p
rocess next two lines !129
790 FOR X=R-3 TO 9+B9 :: CAL
L D(X+3,L$(X)):: NEXT X :: C
ALL S(5):: GOTO 570 !090800
! ==== edit end ==== !200
810 CALL BIN(W$,E$,32):: CAL
L D(23,""):: CALL D(24,"Savi
ng the card...")!203
820 FOR X=9+B9 TO 1 STEP -1
:: IF L$(X)=E$ THEN L$(X+1)=
CHR$(127)&SEG$(E$,1,27):: GO
TO 840 !118
830 NEXT X ! put end marker
!227
840 CD$(1),CD$(2)="" :: FOR
X=1 TO 9+B9 :: Y=1-(X>9)! re
do data strings !119
850 CD$(Y)=CD$(Y)&L$(X):: NE
XT X !253
860 PRINT #2,REC(DREC):CD$(1
):: IF BIG THEN PRINT #2,REC
(DREC+1):CD$(2)!050
870 CALL ERCARD(BIG):: GOTO
300 ! back to file menu !037
920 ! ===== subroutines ====
= !008
930 DREC=VAL(SEG$(CT$(C),24,
4)):: LINPUT #2,REC(DREC):CD
$(1):: IF BIG THEN LINPUT #2
,REC(DREC+1):CD$(2):: RETURN
 ELSE RETURN !073
960 DISPLAY AT(2,1):USING TO
P$:SEG$(CT$(C),1,23),C :: RE
TURN !084
990 CLOSE #1 :: CLOSE #2 ::
RETURN !187
1000 IF BIG THEN RETURN ELSE
 CALL HCHAR(14,1,32,288):: C
ALL LINE(13,130,134,131):: R
ETURN !029
1010 CALL ERR(K,S,P,L):: IF
K<>130 THEN 1030 ELSE CALL S
(2):: CALL D(19," ** BAD FIL
ENAME **")!239
1020 GOSUB 990 :: DELETE D$&
"/I" :: DELETE D$&"/D" :: RE
TURN 270 !006
1030 PRINT "Error";K;"in lin
e";L :: GOSUB 990 :: BREAK !
085
1060 !@P+ ==== user-def subs
 ===== !187
1070 SUB LINE(R,A,B,C):: CAL
L HCHAR(R,2,A):: CALL HCHAR(
R,3,B,28):: CALL HCHAR(R,31,
C):: SUBEND !148
1080 SUB D(R,A$):: DISPLAY A
T(R,1):A$ :: SUBEND !073
1090 SUB PAD(A$,L):: A$=A$&R
PT$(" ",L-LEN(A$)):: SUBEND
!156
1100 SUB ERCARD(B):: CALL D(
2,""):: FOR R=4 TO 9*B+12 ::
 CALL D(R,""):: NEXT R :: CA
LL D(24,""):: SUBEND !218
1110 SUB S(N):: CALL SOUND(1
00,N*100,0):: SUBEND !172
1120 SUB AKEY(A$,P)!108
1130 CALL KEY(0,K,S):: IF S=
0 THEN 1130 ELSE B$=CHR$(K):
: IF ASC(B$)>96 THEN B$=CHR$
(ASC(B$)-32)!074
1140 P=POS(A$,B$,1):: IF P=0
 THEN CALL S(2):: GOTO 1130
!156
1150 SUBEND !168
1160 SUB ASKF(A$,D$):: DISPL
AY AT(17,2):A$&" FILE: "&SEG
$(D$,1,5)&"        /D" :: CA
LL KEY(3,K,S)!025
1170 ACCEPT AT(17,16)SIZE(-1
)BEEP:D$ :: ACCEPT AT(17,18)
SIZE(-8):F$ :: D$="DSK"&D$&"
."&F$ :: CALL KEY(5,K,S):: S
UBEND !187
1180 SUB OPIF(D$):: OPEN #1:
D$&"/I",DISPLAY ,FIXED 27,RE
LATIVE :: SUBEND !164
1190 SUB OPDF(D$):: OPEN #2:
D$&"/D",DISPLAY ,FIXED 252,R
ELATIVE :: SUBEND !204
1200 SUB BIN(W$,A$,C):: W$=A
$ :: CALL HCHAR(23,2,C):: SU
BEND !023
```

CARDFILE

```
100 ! ***** CARDFILE ***** L
.Dorais/Ottawa UG/March 1993
 !040
110 ON WARNING NEXT :: ON ER
ROR 1010 :: DIM CT$(275),CD$
(2)!186
120 A$="000000" :: B$="18181
8" :: C$="1F1F" :: D$="F8F8"
 :: S$="      " !209
130 END$="END with a '/' on
empty line" :: CMEN$="E)dit
D)el P)rint M)enu  > <" !006
140 TOP$=RPT$("#",23)&CHR$(1
36)&"####" :: IND$="# ### ##
## #" :: CALL CHAR(136,"FFD7
D783D783D7D7")!058
150 OPEN #3:"PIO" :: GOTO 17
0 :: BIG,C,CL,DEL$,DREC,K,L,
LREC,N,ND,P,R,S,SORT$,T$,X !
002
160 CALL HCHAR :: CALL VCHAR
 :: CALL GCHAR :: CALL CLEAR
 :: CALL SCREEN :: CALL ERR
:: !@P- !022
170 CALL CHAR(128,A$&C$&B$,1
29,A$&D$&B$,130,B$&C$&A$,131
,B$&D$&A$,132,B$&C$&B$,133,B
$&D$&B$,134,A$&"FFFF"&A$,135
,B$&B$&"1818")!078
180 ! ====== main menu =====
 !252
190 CALL CLEAR :: D$="DSK1."
 :: CALL SCREEN(4):: CALL VC
HAR(1,2,135,22):: CALL VCHAR
(1,31,135,22)!170
200 CALL LINE(1,128,134,129)
:: CALL LINE(3,132,134,133):
: CALL LINE(22,130,134,131)!
229
210 DISPLAY AT(2,9):"TI CARD
FILE" :: DISPLAY AT(6,7):"N)
ew file": : :S$&"O)pen a fil
e": : :S$&"Q)uit" !251
220 CALL AKEY("NOQ",P):: IF
P=3 THEN CALL CLEAR :: CLOSE
 #3 :: END ELSE IF P=2 THEN
270 !010
230 CALL ASKF(" NEW",D$):: C
ALL OPIF(D$):: CALL OPDF(D$)
! new file !111
240 CALL D(19,"  Card Size:
S)mall ( 9 li.)"):: CALL D(2
```

(See Page 19)

CARDFILE—

(Continued from Page 17)

```
740 A$=SEG$(A$,1,Y+1)&SEG$(B
$,1,P-1):: CALL PAD(A$,28)!
add word !048
750 B$=SEG$(B$,P+1,28):: CAL
L PAD(B$,28)! remove word !2
50
760 IF B$<>E$ THEN Y=29 :: G
OTO 720 ! continue to look f
or words !007
770 FOR Y=X+1 TO 9+B9 :: L$(
Y)=L$(Y+1):: NEXT Y :: GOTO
710 ! rewrite data !117
780 L$(X)=A$ :: L$(X+1)=B$ :
: NEXT X :: L$(10+B9)=E$ ! p
rocess next two lines !129
790 FOR X=R-3 TO 9+B9 :: CAL
L D(X+3,L$(X)):: NEXT X :: C
ALL S(5):: GOTO 570 !090800
! ==== edit end ==== !200
810 CALL BIN(W$,E$,32):: CAL
L D(23,""):: CALL D(24,"Savi
ng the card..."):!203
820 FOR X=9+B9 TO 1 STEP -1
:: IF L$(X)=E$ THEN L$(X+1)=
CHR$(127)&SEG$(E$,1,27):: GO
TO 840 !118
830 NEXT X ! put end marker
!227
840 CD$(1),CD$(2)="" :: FOR
X=1 TO 9+B9 :: Y=1-(X>9)! re
do data strings !119
850 CD$(Y)=CD$(Y)&L$(X):: NE
XT X !253
860 PRINT #2,REC(DREC):CD$(1
):: IF BIG THEN PRINT #2,REC
(DREC+1):CD$(2)!050
870 CALL ERCARD(BIG):: GOTO
300 ! back to file menu !037
920 ! ===== subroutines ====
= !008
930 DREC=VAL(SEG$(CT$(C),24,
4)):: LINPUT #2,REC(DREC):CD
$(1):: IF BIG THEN LINPUT #2
,REC(DREC+1):CD$(2):: RETURN
 ELSE RETURN !073
960 DISPLAY AT(2,1):USING TO
P$:SEG$(CT$(C),1,23),C :: RE
TURN !084
990 CLOSE #1 :: CLOSE #2 ::
RETURN !187
1000 IF BIG THEN RETURN ELSE
 CALL HCHAR(14,1,32,288):: C
ALL LINE(13,130,134,131):: R
ETURN !029
1010 CALL ERR(K,S,P,L):: IF
K<>130 THEN 1030 ELSE CALL S
(2):: CALL D(19," ** BAD FIL
ENAME **"):!239
1020 GOSUB 990 :: DELETE D$&
"/I" :: DELETE D$&"/D" :: RE
TURN 270 !006
1030 PRINT "Error";K;"in lin
e";L :: GOSUB 990 :: BREAK !
085
1060 !@P+ ==== user-def subs
 ===== !187
1070 SUB LINE(R,A,B,C):: CAL
L HCHAR(R,2,A):: CALL HCHAR(
R,3,B,28):: CALL HCHAR(R,31,
C):: SUBEND !148
1080 SUB D(R,A$):: DISPLAY A
T(R,1):A$ :: SUBEND !073
1090 SUB PAD(A$,L):: A$=A$&R
PT$(" ",L-LEN(A$)):: SUBEND
!156
1100 SUB ERCARD(B):: CALL D(
2,""):: FOR R=4 TO 9*B+12 ::
 CALL D(R,""):: NEXT R :: CA
LL D(24,""):: SUBEND !218
1110 SUB S(N):: CALL SOUND(1
00,N*100,0):: SUBEND !172
1120 SUB AKEY(A$,P)!108
1130 CALL KEY(0,K,S):: IF S=
0 THEN 1130 ELSE B$=CHR$(K):
: IF ASC(B$)>96 THEN B$=CHR$
(ASC(B$)-32)!074
1140 P=POS(A$,B$,1):: IF P=0
 THEN CALL S(2):: GOTO 1130
!156
1150 SUBEND !168
1160 SUB ASKF(A$,D$):: DISPL
AY AT(17,2):A$&" FILE: "&SEG
$(D$,1,5)&"       /D" :: CA
LL KEY(3,K,S)!025
1170 ACCEPT AT(17,16)SIZE(-1
)BEEP:D$ :: ACCEPT AT(17,18)
SIZE(-8):F$ :: D$="DSK"&D$&"
."&F$ :: CALL KEY(5,K,S):: S
UBEND !187
1180 SUB OPIF(D$):: OPEN #1:
D$&"/I",DISPLAY ,FIXED 27,RE
LATIVE :: SUBEND !164
1190 SUB OPDF(D$):: OPEN #2:
D$&"/D",DISPLAY ,FIXED 252,R
ELATIVE :: SUBEND !204
1200 SUB BIN(W$,A$,C):: W$=A
$ :: CALL HCHAR(23,2,C):: SU
BEND !023
```

CARDFILE

```
100 ! ***** CARDFILE ***** L
.Dorais/Ottawa UG/March 1993
 !040
110 ON WARNING NEXT :: ON ER
ROR 1010 :: DIM CT$(275),CD$
(2)!186
120 A$="000000" :: B$="18181
8" :: C$="1F1F" :: D$="F8F8"
 :: S$="      " !209
130 END$="END with a '/' on
empty line" :: CMEN$="E)dit
D)el P)rint M)enu  > <" !006
140 TOP$=RPT$("#",23)&CHR$(1
36)&"####" :: IND$="# ### ##
## #" :: CALL CHAR(136,"FFD7
D783D783D7D7"):!058
150 OPEN #3:"PIO" :: GOTO 17
0 :: BIG,C,CL,DEL$,DREC,K,L,
LREC,N,ND,P,R,S,SORT$,T$,X !
002
160 CALL HCHAR :: CALL VCHAR
 :: CALL GCHAR :: CALL CLEAR
 :: CALL SCREEN :: CALL ERR
:: !@P- !022
170 CALL CHAR(128,A$&C$&B$,1
29,A$&D$&B$,130,B$&C$&A$,131
,B$&D$&A$,132,B$&C$&B$,133,B
$&D$&B$,134,A$&"FFFF"&A$,135
,B$&B$&"1818"):!078
180 ! ====== main menu =====
 !252
190 CALL CLEAR :: D$="DSK1."
 :: CALL SCREEN(4):: CALL VC
HAR(1,2,135,22):: CALL VCHAR
(1,31,135,22)!170
200 CALL LINE(1,128,134,129)
:: CALL LINE(3,132,134,133):
: CALL LINE(22,130,134,131)!
229
210 DISPLAY AT(2,9):"TI CARD
FILE" :: DISPLAY AT(6,7):"N)
ew file": : :S$&"O)pen a fil
e": : :S$&"Q)uit" !251
220 CALL AKEY("NOQ",P):: IF
P=3 THEN CALL CLEAR :: CLOSE
 #3 :: END ELSE IF P=2 THEN
270 !010
230 CALL ASKF(" NEW",D$):: C
ALL OPIF(D$):: CALL OPDF(D$)
! new file !111
240 CALL D(19,"  Card Size:
S)mall ( 9 li.)"):: CALL D(2
```

(See Page 19)

CARDFILE—

(Continued from Page 18)

```
0,S$&S$&" B)ig   (18 li.)")!
004
250 CALL AKEY("SB",P):: BIG=
P-1 :: GOSUB 1000 !060
260 N,C=0 :: DREC,LREC=-BIG
:: DEL$="no" :: GOSUB 1050 :
: GOTO 360 !207
270 CALL ASKF("OPEN",D$):: C
ALL OPIF(D$):: CALL OPDF(D$)
:: C=1 !103
280 LINPUT #1,REC(0):A$ :: B
IG=VAL(SEG$(A$,1,1)):: N=VAL
(SEG$(A$,3,3)):: LREC=VAL(SE
G$(A$,7,4)):: SORT$=SEG$(A$,
12,1)!137
290 FOR X=1 TO N :: LINPUT #
1,REC(X):CT$(X):: NEXT X ::
GOSUB 1040 :: DEL$=SEG$(DEL$
,1,4):: CALL ERCARD(BIG):: G
OSUB 1000 !013
300 CALL SCREEN(12)! ==== wo
rk with a file ==== !112
310 DISPLAY AT(4,7):"V)iew":
S$&"A)dd"&S$&"T)op":S$&"F)in
d     B)ottom" !135
320 DISPLAY AT(8,7):"S)ort
   > forward":S$&"P)rint
< backward": :S$&"D)el Pack"
:S$&"E)xit (Save file info)"
 !108
330 GOSUB 960 :: CALL D(24,D
$&"    Sorted: "&SORT$)!241
340 CALL AKEY("AFSPVETB.,D",
P):: ON P GOTO 360,490,540,6
10,750,730,520,520,530,530,6
80 !234
350 ! ===== fill a card ====
= !039
360 C$="" :: CALL ERCARD(BIG
):: CALL SCREEN(16):: SORT$=
"N" !214
370 CALL D(24,END$):: CALL A
CC(2,T$,23):: IF ASC(T$)=47
THEN 300 !123
380 C,N=N+1 :: CT$(C)=T$ ::
GOSUB 960 :: IF DEL$="no" TH
EN DREC,LREC=LREC+1+BIG :: G
OTO 410 !107
390 GOSUB 1040 :: DREC=VAL(S
EG$(DEL$,5,4)):: B$=SEG$(DEL
$,9,244)!089
400 IF B$="" THEN DEL$="no"
:: GOSUB 1050 ELSE DEL$="yes
 "&B$ :: GOSUB 1050 !238
410 A$=STR$(DREC):: CT$(C),T
$=T$&RPT$(" ",4-LEN(A$))&A$
:: PRINT #1,REC(C):T$ !126
420 CD$(1),CD$(2)="" :: FOR
R=1 TO 9*(BIG+1):: CALL ACC(
R+3,A$,28):: IF ASC(A$)=47 T
HEN A$=CHR$(127)!051
430 P=1-(R>9):: CD$(P)=CD$(P
)&A$ :: IF ASC(A$)=127 THEN
450 !139
440 NEXT R !232
450 CALL S(5):: CALL D(24,"I
s it OK? (Y/N)"):: CALL AKEY
("YN",P):: IF P=1 THEN 470 !
194
460 CALL ACC(2,T$,23):: CALL
 D(24,END$):: GOTO 410 ! red
o title/data !223
470 PRINT #2,REC(DREC):CD$(1
):: IF BIG THEN PRINT #2,REC
(DREC+1):CD$(2)!050
480 IF C$="E" THEN 770 ELSE
360 !231
485 ! ===== work on file ===
= !136
490 CALL D(24,"Enter first l
etter(s)"):: ACCEPT AT(2,1):
A$ :: S=LEN(A$)! find !083
500 FOR X=1 TO N :: IF SEG$(
CT$(X),1,S)=A$ THEN C=X :: G
OTO 750 ! found !197
510 NEXT X :: CALL S(2):: GO
TO 330 ! not found !221
520 IF P=7 THEN C=1 :: GOTO
330 ELSE C=N :: GOTO 330 ! t
op/bottom !255
530 IF P=9 THEN GOSUB 940 ::
 GOTO 330 ELSE GOSUB 950 ::
GOTO 330 ! move for/backward
 !052
540 A$="SORT" :: GOSUB 1025
:: IF P=2 THEN 330 !224
550 CALL D(24,"... sorting .
..")):: SORT$="Y" :: P=1 ! so
rt !236
560 P=2*P :: IF P<=N THEN 56
0 !232
570 P=INT(P/2):: IF P=0 THEN
 S,C=1 :: GOSUB 970 :: GOTO
330 !210
580 FOR X=1 TO N-P :: K=X !1
44
```

(See Page 20)

1994 TI FAIRS

MAY

Lima Multi User Group Conference, May 13-14, Ohio State University Lima Campus, Lima, Ohio. Contact Lima Ohio Users Group, P.O. Box 647, Venedocia, OH 45894.

OCTOBER

9th International TI-Meeting, Oct. 14-16, Kirch I. Gemeindehaus Roshorf, German, sponsored by TI-Club Goettingen. For information, contact Jörg Kirstan, Mengershäuser Weg 5, D-37124 Rosdorf, Germany, tel. 01551/781153; Reinhard Obuch, Keplerstr. 5, D-37085 Göttingen, Germany, tel. 0551/46405; or Hans-Hartmut Kortry, Grüner Weg 10, D-37181 Hardegsen, Germany, tel. 05505/1470.

NOVEMBER

The TI International World's Faire, Nov. 12, Holiday Inn, Gurnee, Illinois. Sponsored by Chicago and Milwaukee users groups. For information, contact Don Walden (414) 679-2336.

1995 TI FAIRS

FEBRUARY

Fest West '95, Feb. 18, Fabulous Inn, San Diego, California. Contact Southern California Computer Group, P.O. Box 152535, San Diego,CA 92195, or call the SCCG BBS, (619) 263-9135, User No. 25, password FEST.

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.

CARDFILE—

(Continued from Page 19)

```
590 S=K+P :: IF CT$(K)>CT$(S
)THEN A$=CT$(K):: CT$(K)=CT$
(S):: CT$(S)=A$ :: K=K-P ::
IF K>0 THEN 590 !112
600 NEXT X :: GOTO 570 !251
610 A$="PRINT" :: GOSUB 1025
 :: IF P=2 THEN 330 !038
620 CALL D(24,A$&" from Card
 #1   to #"&STR$(N)):: ACCEP
T AT(24,18)SIZE(-3):S :: ACC
EPT AT(24,26)SIZE(-3):K !013
630 CALL SCREEN(10):: L=0 ::
 FOR C=S TO K :: GOSUB 960 :
: GOSUB 930 !178
640 ND=5 :: P=POS(CD$(1),CHR
$(127),1):: IF P THEN 660 !1
63
650 ND=14 :: P=POS(CD$(2),CH
R$(127),1):: IF P=0 THEN P=2
53 !160
660 ND=ND+(P-1)/28 :: L=L+ND
 :: IF L>=63 THEN PRINT #3:C
HR$(12):: L=ND !202
670 CALL PCARD(C,CT$(C),CD$(
),BIG):: NEXT C :: PRINT #3:
CHR$(12):: C=S :: GOTO 300 !
093
680 A$="PACK DELeted Cards"
:: GOSUB 1025 :: IF P=2 THEN
 330 !105
690 CALL D(24,"... packing .
..")):: C=0 :: S=N+1 :: IF DE
L$="yes " THEN GOSUB 1040 EL
SE DEL$="yes " !154
700 FOR X=1 TO N :: B$=CT$(X
):: IF SEG$(B$,20,3)<>"*D*"
THEN C=C+1 :: CT$(C)=B$ :: G
OSUB 960 :: GOTO 720 !217
710 S=MIN(S,X):: B$=SEG$(B$,
24,4):: DEL$=DEL$&B$ :: DREC
=VAL(B$):: PRINT #2,REC(DREC
):"*del*" :: IF BIG THEN PRI
NT #2,REC(DREC+1):"*del*" !0
52
720 NEXT X :: N=C :: GOSUB 1
050 :: GOSUB 970 :: GOTO 330
 !106
730 CL=1 :: GOSUB 980 :: CL=
0 :: GOTO 190 ! close file,
go to main menu !127
740 ! ==== view cards, work
on them ==== !141
750 CALL SCREEN(16):: GOSUB
960 :: GOSUB 930 !213
760 CALL D(4,CD$(1)):: IF BI
G THEN CALL D(13,CD$(2))!108
770 CALL D(24,CMEN$):: CALL
AKEY("EDPM.,",P):: ON P GOTO
 775,840,870,820,830,830 !17
8
775 IF SEG$(CT$(C),20,3)="*D
*" THEN 850 ! deleted card !
228
780 FOR R=4 TO 9*BIG+12 :: C
ALL GCHAR(R,3,K):: IF K=127
THEN CALL HCHAR(R,3,47):: GO
TO 800 ! put back the end /
!126
790 NEXT R ! no end / !014
800 CALL ACC(2,T$,23):: IF T
$<>SEG$(CT$(C),1,23)THEN SOR
T$="N" ! change? !024
810 CALL D(24,END$):: C$="E"
 :: GOTO 410 !021
820 CALL ERCARD(BIG):: GOTO
300 ! back to file menu !037
830 IF P=5 THEN GOSUB 940 ::
 GOTO 750 ELSE GOSUB 950 ::
GOTO 750 ! move > < !095840
A$=SEG$(CT$(C),1,19)&"*D* "
:: CALL S(3):: GOTO 860 ! ma
rk card for deletion !089
850 CALL HCHAR(2,22,32,4)::
CALL ACC(2,A$,23)! undelete
card (edit title) !202
860 CT$(C)=A$&SEG$(CT$(C),24
,4):: PRINT #1,REC(C):CT$(C)
:: GOSUB 960 :: GOTO 770 !10
6
870 CALL D(24,"...printing..
."):: CALL PCARD(C,CT$(C),CD
$(),BIG):: GOTO 770 ! print
card !034
920 ! ===== subroutines ====
= !008
930 DREC=VAL(SEG$(CT$(C),24,
4)):: LINPUT #2,REC(DREC):CD
$(1):: IF BIG THEN LINPUT #2
,REC(DREC+1):CD$(2):: RETURN
 ELSE RETURN !073
940 C=C+1 :: IF C<=N THEN RE
TURN ELSE C=N :: CALL S(2)::
 RETURN ! forward !096
950 C=C-1 :: IF C>=1 THEN RE
TURN ELSE C=1 :: CALL S(2)::
 RETURN ! backward !004
960 DISPLAY AT(2,1):USING TO
P$:SEG$(CT$(C),1,23),C :: RE
TURN !084
970 CALL D(24,"... saving ne
w index..."):: FOR X=1 TO N
:: PRINT #1,REC(X):CT$(X)::
NEXT X !099
980 PRINT #1,REC(0),USING IN
D$:BIG,N,LREC,SORT$ !169
990 IF CL THEN CLOSE #1 :: C
LOSE #2 :: RETURN ELSE RETUR
N !135
1000 IF BIG THEN RETURN ELSE
 CALL HCHAR(14,1,32,288):: C
ALL LINE(13,130,134,131):: R
ETURN !029
1010 CALL ERR(K,S,P,L):: IF
K<>130 THEN 1030 ELSE CALL S
(2):: CALL D(19," ** BAD FIL
ENAME **")!239
1020 CL=1 :: GOSUB 990 :: CL
=0 :: DELETE D$&"/I" :: DELE
TE D$&"/D" :: RETURN 270 !15
1
1025 CALL S(6):: CALL D(24,A
$&"? (Y/N)"):: CALL AKEY("YN
",P):: RETURN !017
1030 PRINT "Error";K;"in lin
e";L :: CL=1 :: GOSUB 980 ::
 BREAK !020
1040 CD$(1),CD$(2)="" :: INP
UT #2,REC(0):DEL$ :: RETURN
!024
1050 PRINT #2,REC(0):DEL$ ::
 :: DEL$=SEG$(DEL$,1,4):: RE
TURN !153
1060 !@P+ ==== user-def subs
 ===== !187
1070 SUB LINE(R,A,B,C):: CAL
L HCHAR(R,2,A):: CALL HCHAR(
R,3,B,28):: CALL HCHAR(R,31,
C):: SUBEND !148
1080 SUB D(R,A$):: DISPLAY A
T(R,1):A$ :: SUBEND !073
1090 SUB ACC(R,A$,L):: ACCEP
T AT(R,1)SIZE(-L):A$ :: A$=A
$&RPT$(" ",L-LEN(A$)):: SUBE
ND !149
1100 SUB ERCARD(B):: CALL D(
2,""):: FOR R=4 TO 9*B+12 ::
 CALL D(R,""):: NEXT R :: CA
LL D(24,""):: SUBEND !218
1110 SUB S(N):: CALL SOUND(1
00,N*100,0):: SUBEND !172
1120 SUB AKEY(A$,P)!108
1130 CALL KEY(0,K,S):: IF S=
0 THEN 1130 ELSE B$=CHR$(K):
```

(See Page 21)

CARDFILE—

(Continued from Page 21)

```
: IF ASC(B$)>96 THEN B$=CHR$
(ASC(B$)-32)!074
1140 P=POS(A$,B$,1):: IF P=0
 THEN CALL S(2):: GOTO 1130
!156
1150 SUBEND !168
1160 SUB ASKF(A$,D$):: DISPL
AY AT(17,2):A$&" FILE: "&SEG
$(D$,1,5)&"        /D" :: CA
LL KEY(3,K,S)!025
1170 ACCEPT AT(17,16)SIZE(-1
)BEEP:D$ :: ACCEPT AT(17,18)
SIZE(-8):F$ :: D$="DSK"&D$&"
."&F$ :: CALL KEY(5,K,S):: S
UBEND !187
1180 SUB OPIF(D$):: OPEN #1:
D$&"/I",DISPLAY ,FIXED 27,RE
LATIVE :: SUBEND !164
1190 SUB OPDF(D$):: OPEN #2:
D$&"/D",DISPLAY ,FIXED 252,R
ELATIVE :: SUBEND !204
1200 SUB PCARD(C,T$,D$(),B):
: PRINT #3:CHR$(27)&"E  "&SE
G$(T$,1,23):"  "&RPT$("-",30
)&CHR$(27)&"F    >";C ! bold
 title,crd # !242
1210 FOR I=1 TO B+1 :: FOR X
=1 TO 9 :: A$=SEG$(D$(I),28*
X-27,28):: IF ASC(A$)=127 TH
EN 1230 ! data !081
1220 PRINT #3:"    "&A$ :: N
EXT X :: NEXT I !131
1230 PRINT #3 :: PRINT #3 ::
 PRINT #3 :: SUBEND !237
```

CARDUTIL

```
100 ! ***** CARDUTIL ***** L
.Dorais/Ottawa UG/March 1993
 !070
110 ON WARNING NEXT :: ON ER
ROR 1030 :: DIM CD$(2),ST(3)
,EN(3),ND$(3)!207
120 A$="000000" :: B$="18181
8" :: C$="1F1F" :: D$="F8F8"
 :: S$="      " !209
130 F$(1)="MERGE" :: F$(2)="
SPLIT" :: F$(3)="DECREASE" :
: F$(4)="INCREASE" :: F$(5)=
"CLEAN" :: F$(6)="ANALYZE" !
188
140 TOP$=RPT$("#",23)&CHR$(1
36)&"####" :: IND$="# ### ##
## #" :: CALL CHAR(136,"FFD7
D783D783D7D7")!058
150 GOTO 170 :: BIG,C,DEL$,D
REC,K,L,LREC,N,P,R,S,SORT$,T
$,X,DD$,DV,E$,FM,N1,SPL,TN,Z
 !062
160 CALL HCHAR :: CALL VCHAR
 :: CALL CLEAR :: CALL SCREE
N :: CALL ERR :: !@P- !197
170 CALL CHAR(128,A$&C$&B$,1
29,A$&D$&B$,130,B$&C$&A$,131
,B$&D$&A$,132,B$&C$&B$,133,B
$&D$&B$,134,A$&"FFFF"&A$,135
,B$&B$&"1818")!078
180 ! ====== main menu =====
 !252
190 CALL CLEAR :: D$="DSK1."
 :: CALL SCREEN(4):: CALL VC
HAR(1,2,135,22):: CALL VCHAR
(1,31,135,22)!170
200 CALL LINE(1,128,134,129)
:: CALL LINE(3,132,134,133):
: CALL LINE(22,130,134,131)!
229
210 DISPLAY AT(2,9):"TI CARD
UTIL" :: DISPLAY AT(6,7):"M)
erge files":S$&"S)plit a fil
e" !128
220 DISPLAY AT(9,7):"D)ecrea
se size":S$&"I)ncrease Size"
: :S$&"C)lean a file":S$&"A)
nalyze a file":S$&"Q)uit" !0
83
230 CALL AKEY("MSDICAQ",P)::
 IF P=7 THEN CALL CLEAR :: E
ND !117
240 CALL ERCARD(1):: DISPLAY
 AT(2,10):F$(P):: A$=SEG$(F$
(P),1,4)!039
270 CALL ASKF(7,A$,D$):: CAL
L OPIF(1,D$)!201
280 LINPUT #1,REC(0):A$ :: B
IG=VAL(SEG$(A$,1,1)):: N=VAL
(SEG$(A$,3,3)):: LREC=VAL(SE
G$(A$,7,4)):: SORT$=SEG$(A$,
12,1)!137
290 CALL SCREEN(15):: ON P G
OTO 310,410,610,610,620,760
!242
300 ! ==== merge two files =
=== !136
310 DD$=SEG$(D$,1,5):: CALL
ASKF(12,"from",DD$):: CALL O
PIF(3,DD$):: INPUT #3,REC(0)
:A$ :: IF VAL(SEG$(A$,1,1))=
BIG THEN 330 !084
320 CALL S(2):: CALL D(15,"T
hey are not the same size!")
:: CLOSE #1 :: CLOSE #3 :: G
OTO 910 !155
330 N1=VAL(SEG$(A$,3,3)):: S
ORT$="N" :: Z=BIG :: DV=4 ::
 CALL OPDF(2,D$):: CALL OPDF
(4,DD$)!005
340 FOR C=1 TO N1 :: LREC=LR
EC+BIG+1 :: N=N+1 !040
350 LINPUT #3,REC(C):T$ :: G
OSUB 960 :: GOSUB 930 !029
360 GOSUB 970 :: CALL DPRINT
(1,2,N,T$,LREC,CD$(),BIG)!06
9
370 NEXT C :: CLOSE #3 :: CL
OSE #4 :: GOSUB 980 :: GOTO
190 !072
400 ! ==== split a file ====
 !056
410 ST(1)=1 :: CALL D(9,"Fil
e has "&STR$(N)&" cards"):::
CALL D(11,"Split in middle?
(Y/N)"):: CALL S(5):: CALL A
KEY("YN",P)!232
420 SPL=2 :: IF P=1 THEN FM=
INT(N/2):: EN(1)=FM :: ST(2)
=FM+1 :: EN(2)=N :: GOTO 460
 !145
430 CALL D(13,"SPLIT from Ca
rd #    to #"):: ACCEPT AT(1
3,18)SIZE(-3):FM :: ACCEPT A
T(13,26):L !071
440 IF FM>1 AND L<N THEN SPL
=3 :: EN(1)=FM-1 :: ST(2)=FM
 :: EN(2)=L :: ST(3)=L+1 ::
EN(3)=N :: GOTO 460 !175
450 IF FM=1 THEN EN(1)=L ::
ST(2)=L+1 :: EN(2)=N ELSE EN
(1)=FM-1 :: ST(2)=FM :: EN(2
)=N ! begin/end file !035
460 CALL D(15,"New split fil
es:"):: ND$(1),ND$(2)=SEG$(D
$,1,5):: CALL ASKF(17,"sp 1"
,ND$(1)):: CALL ASKF(18,"sp
2",ND$(2))!082
470 CALL OPDF(2,D$):: Z=BIG
:: DV=2 :: IF SPL=3 THEN ND$
(3)=ND$(1)!175
480 FOR X=1 TO SPL :: CALL D
(20," Copying to "&ND$(X)):
 IF X<3 THEN LREC=-BIG :: N=
0 ELSE LREC=L :: N=TN !127
490 CALL OPIF(3,ND$(X)):: CA
```

(See Page 22)

CARDFILE—

(Continued from Page 21)

```
LL OPDF(4,ND$(X)):: FOR C=ST
(X)TO EN(X):: N=N+1 !245500
LINPUT #1,REC(C):T$ :: GOSUB
 960 :: GOSUB 930 :: LREC,DR
EC=LREC+1+BIG :: GOSUB 970 !
101
510 CALL DPRINT(3,4,N,T$,DRE
C,CD$(),BIG):: NEXT C :: PRI
NT #3,REC(0),USING IND$:BIG,
N,LREC,SORT$ !042
520 PRINT #4,REC(0):"no" ::
CLOSE #3 :: CLOSE #4 :: IF X
=1 THEN L=LREC :: TN=N !069
530 NEXT X :: GOSUB 990 :: G
OTO 190 !048
600 ! ==== decrease/increase
 file size / clean file ====
 !077
610 IF P-3=BIG THEN CALL S(5
):: CALL D(12,"No need to ch
ange size!"):: CLOSE #1 :: G
OTO 910 ELSE BIG=P-3 !229
620 LREC=-BIG :: Z=0 :: DV=2
 :: CALL OPDF(2,D$):: DD$=SE
G$(D$,1,5)&"*"&SEG$(D$,7,7):
: CALL OPDF(4,DD$):: IF P=5
THEN 640 !205
630 IF BIG=0 THEN E$=RPT$("
",28)ELSE CD$(2)=CHR$(127)&R
PT$(" ",251)!255
640 FOR C=1 TO N :: LREC=LRE
C+1+BIG :: LINPUT #1,REC(C):
T$ :: GOSUB 960 :: GOSUB 930
 :: GOSUB 970 !156
650 IF BIG OR P=5 OR POS(CD$
(1),CHR$(127),1)THEN 670 ELS
E S=225 !069
660 IF SEG$(CD$(1),S,28)=E$
THEN S=S-28 :: GOTO 660 ELSE
 CD$(1)=SEG$(CD$(1),1,S+27)&
CHR$(127)!015
670 CALL DPRINT(1,4,C,T$,LRE
C,CD$(),BIG):: NEXT C !248
680 GOSUB 980 :: PRINT #4,RE
C(0):"no" :: CLOSE #4 :: IF
P=5 THEN 700 ELSE CALL D(11,
"INDEX FILE modified.")!143
690 A$="/D" :: DISPLAY AT(14
,1):"To use NEW DATA FILE,":
 :"1)  erase "&D$&A$: :"2) r
ename "&DD$&A$:S$&" to "&D$&
A$ :: GOTO 910 !175
700 CALL D(11," Cleaning ind
ex file..."):: CALL OPIF(1,D
$):: CALL OPIF(3,DD$)!019
710 FOR X=0 TO N :: LINPUT #
1,REC(X):A$ :: PRINT #3,REC(
X):A$ :: NEXT X :: CLOSE #1
:: CLOSE #3 !088
720 DISPLAY AT(14,2):"The cl
ean files are:":S$&DD$&"/I":
S$&DD$&"/D": :" They can be
used now, or    renamed (fir
st delete the   old files)."
 !069
730 GOTO 910 !224
750 ! ==== analyze a file ==
== !000
760 CALL ERCARD(1):: CALL OP
DF(2,D$):: GOSUB 1040 :: CAL
L D(2,D$)!175
770 DISPLAY AT(5,1):USING "#
## cards"&S$&S$&S$&"  ## lin
es of data"&S$&S$&" # sorted
":N,9+9*BIG,SORT$ !073
780 A$="DELETE LIST: " :: IF
 ASC(DEL$)=110 THEN DISPLAY
AT(9,1):A$&"no" ELSE DISPLAY
 AT(9,1):A$&"yes":SEG$(DEL$,
5,248)!117
790 CALL S(5):: CALL D(24,"S
EE Titles and DRECs? (Y/N)")
:: CALL AKEY("YN",P):: IF P=
2 THEN GOSUB 990 :: GOTO 190
 !242
800 CALL ERCARD(1):: S=0 ::
DISPLAY AT(2,1):D$;TAB(25);"
DREC" !064
810 FOR X=1 TO 18 :: IF X+S<
=N THEN LINPUT #1,REC(X+S):A
$ ELSE A$="" :: K=99 !196
820 CALL D(X+3,A$):: NEXT X
:: IF K=99 THEN GOSUB 990 ::
 GOTO 910 !066
830 CALL S(5):: CALL D(24,"M
ORE? (Y/N)"):: CALL AKEY("YN
",P):: IF P=2 THEN 190 !102
840 S=S+18 :: CALL D(24,""):
: GOTO 810 !153
910 CALL D(24,"Press any key
...")):: CALL KEY(0,K,S):: IF
 S=0 THEN 910 ELSE 190 !152
920 ! ===== subroutines ====
= !008
930 DREC=VAL(SEG$(T$,24,4)):
: LINPUT #DV,REC(DREC):CD$(1
):: IF Z THEN LINPUT #DV,REC
(DREC+1):CD$(2):: RETURN ELS
E RETURN !028
960 DISPLAY AT(2,1):USING TO
P$:SEG$(T$,1,23),C :: RETURN
 !097
970 A$=STR$(LREC):: T$=SEG$(
T$,1,23)&RPT$(" ",4-LEN(A$))
&A$ :: RETURN !195
980 PRINT #1,REC(0),USING IN
D$:BIG,N,LREC,SORT$ !169
990 CLOSE #1 :: CLOSE #2 ::
RETURN !187
1000 IF BIG THEN RETURN ELSE
 CALL HCHAR(14,1,32,288):: C
ALL LINE(13,130,134,131):: R
ETURN !029
1030 CALL ERR(K,S,P,L):: PRI
NT "Error";K;"in line";L ::
GOSUB 980 :: BREAK !222
1040 CD$(1),CD$(2)="" :: INP
UT #2,REC(0):DEL$ :: RETURN
!024
1060 !@P+ ==== user-def subs
 ===== !187
1070 SUB LINE(R,A,B,C):: CAL
L HCHAR(R,2,A):: CALL HCHAR(
R,3,B,28):: CALL HCHAR(R,31,
C):: SUBEND !148
1080 SUB D(R,A$):: DISPLAY A
T(R,1):A$ :: SUBEND !073
1100 SUB ERCARD(B):: CALL D(
2,""):: FOR R=4 TO 9*B+12 ::
 CALL D(R,""):: NEXT R :: CA
LL D(24,""):: SUBEND !218
1110 SUB S(N):: CALL SOUND(1
00,N*100,0):: SUBEND !172
1120 SUB AKEY(A$,P)!108
1130 CALL KEY(0,K,S):: IF S=
0 THEN 1130 ELSE B$=CHR$(K):
: IF ASC(B$)>96 THEN B$=CHR$
(ASC(B$)-32)!074
1140 P=POS(A$,B$,1):: IF P=0
 THEN CALL S(2):: GOTO 1130
!156
1150 SUBEND !168
1160 SUB ASKF(R,A$,D$):: DIS
PLAY AT(R,2):A$&" FILE: "&SE
G$(D$,1,5)&"        /D" :: C
ALL KEY(3,K,S)!062
1170 ACCEPT AT(R,16)SIZE(-1)
BEEP:D$ :: ACCEPT AT(R,18)SI
ZE(-8):F$ :: D$="DSK"&D$&"."
&F$ :: CALL KEY(5,K,S):: SUB
END !251
1180 SUB OPIF(D,D$):: OPEN #
D:D$&"/I",DISPLAY ,FIXED 27,
```

(See Page 23)

CARDFILE—

(Continued from Page 22)

```
RELATIVE :: SUBEND !229
1190 SUB OPDF(D,D$):: OPEN #
D:D$&"/D",DISPLAY ,FIXED 252
,RELATIVE :: SUBEND !0121200
 SUB DPRINT(D1,D2,C,T$,RC,CD
$(),B):: PRINT #D1,REC(C):T$
 :: PRINT #D2,REC(RC):CD$(1)
 :: IF B THEN PRINT #D2,REC(R
C+1):CD$(2)!122
1210 SUBEND !168
```

MICRO-REVIEWS

XB Compiler, Drawing Program, Video Titler, Font Converter, Turnfont, CALL LINKable XB Enhancements

By CHARLES GOOD

The TI community continues to be enhanced by the efforts of Bruce Harrison. Everything of his I am describing this month has been released by Bruce into the public domain. You can get these software packages from me for $1 per disk, which pays for the disk postage and mailer.

XB COMPILER
by Bruce Harrison

Most previous attempts at BASIC language compilers for the 99/4A have required extensive rewrites of existing software. Some were limited to only certain subsets of TI BASIC or Extended BASIC, and all were difficult to use. Not so with the Harrison compiler. It is not necessary to rewrite your favorite XB software before compiling, and the actual process of producing the compiled code is not difficult.

Bruce has taken certain XB operations and written assembly code that does the operation faster than XB's general programming language interpreter. Whenever a compiled XB program gets to one of these operations it uses Harrison's code for extra speed. If the particular XB operation is not one of those that have been speeded up by assembly code, the compiled XB program branches to GPL and the XB operation occurs at normal speed. This means that everything in a normal XB program works when the program is compiled. Some parts of the compiled program work at normal speed and some work at greatly accelerated speed. File handling (OPEN #, PRINT #, etc.) works at normal speed. XB programs that already have imbedded assembly routines can't be compiled, but you can sometimes CALL LOAD assembly object code to low memory and then CALL LINK to this code from a compiled XB program.

The following are speeded up by the Harrison compiler: CALL CHAR, CALL COLOR, PRINT (to the screen), ON ERROR, FOR/NEXT (including nested loops), CALL GCHAR, CALL HCHAR, CALL VCHAR, CALL KEY, ON GOTO, and ON GOSUB. If Bruce continues to work on his compiler this list may increase.

The process of compiling is multistep but not difficult. Bruce is well-known for writing user-friendly instructions and provides many on-disk examples of each step in the compiling process. First you get your XB program working just the way you want it. This XB program is saved in merge format. The merge format program is then run through the multistep compiling process. Once compiled, the XB program will OLD and RUN normally. If you BREAK the program (with FCTN/4) you will get the correct BREAKPOINT AT LINE XXX message. You can then type CON to continue the compiled program just like regular BASIC. You cannot, however, list or edit a compiled program. You have to go back to the uncompiled XB original code to do any editing and then run the edited code through the Harrison compiler again.

The main limitation to the compiler is program size inflation. The compiled program occupies much more memory and disk space than the uncompiled original. For each of the speeded-up operations listed above the compilation process adds some hidden assembly code to the program which increases memory requirements. If a speeded-up operation is not found in the original XB code, then the assembly code for that operation is not added. All of the speeded-up operations listed above add 21 sectors to program disk size compared to the original uncompiled XB code. Large XB programs may not be compilable because they run out of memory. This is one reason why Bruce hesitates to add more speeded-up operations.

The Harrison compiler is the best general use XB compiler available to the TI community. The compiler with on-disk documentation and sample files comes on a DSSD disk. Its source code is on a second DSSD disk.

DRAWING PROGRAM
by Bruce Harrison

As the name suggests, this lets you make multicolor drawings on screen and save them to disk. You can use the keyboard or joysticks to move the cursor around. You can either start from a blank screen or load in a previously created drawing. You can also load in TI-Artist "_P" and "_C" picture files. For text you can load in a TIA font or you can load any CHARA1 type of file (something TIA can't do). You can also load in TIA instances and place them where you want on the drawing screen. Pictures created with Drawing Program can be printed on almost any printer, including Star SG10 and 10X models. They can also be saved to disk, but graphics are *not* saved in TI-Artist compatible format.

Harrison's Drawing Program isn't nearly as fancy as TI-Artist. However, in most cases, if you need to create screen art work or manipulate graphics that already exist in TIA format, the Drawing Program will do nicely. Unlike TIA, which is commercial, Drawing Program is free. With source code it comes on one DSSD disk. One word of caution. Drawing Program doesn't work on my AVPC system. I don't know if it works with other types of 80-column systems.

VIDEO TITLER
by Bruce Harrison

Bruce was unimpressed with the text-only title screens I create for the Lima Multi User Group Conference videotapes, so he decided to help me make some really

(See Page 25)

MICRO-REVIEWS—

(Continued from Page 24)

fancy videotape title screens. Video Titler is the result. You are supposed to take the computer screen graphics manipulated with this software and record them onto a videotape. The results are impressive. As many of you know, the video output of a 99/4A can, using a monitor cable, be fed directly into the "video in" jack of a VCR, where it can be recorded onto videotape and displayed on the screen of a TV hooked up to the VCR.

Video Titler lets you store two full screens of graphics in memory and then rapidly switch back and forth between these pictures, displaying them one at a time on screen and on the videotape you are recording. First you load the pictures into memory. They can be either Drawing Program or TI-Artist pictures. Then you press "record" on the VCR and start switching between pictures. If the pictures in memory are only slightly different, rapid switching produces an animation effect. In addition to instantly switching between the two screen pictures, you can wipe the current image in various ways to display the second picture. Wipes can be left to right, right to left, top to bottom, bottom to top, or center to left/right. If you want to display more than two pictures, press "pause" on the VCR. Then load more pictures, replacing those already in memory. Then release the "pause."

I can send you Video Titler, some neat video pictures made by Bruce, and the Drawing Program (without source code, no room for it) all on one DSSD disk.

FONT CONVERTER
by Bruce Harrison

A few years ago Jim Peterson created almost 200 screen fonts for use in Extended BASIC software. Bruce Harrison has made an assembly program to convert all of these Peterson screen fonts into CHARA1 program files. The conversion process is somewhat lengthy, since each Peterson screen font has to be individually inverted. The result is a whole bunch of mostly five-sector CHARA1 fonts that can be used with Drawing Program. A unique feature of Drawing Program is its ability to import CHARA1 fonts. You can also use these converted CHARA1 screen fonts with word processing software, although many of them don't look very good used this way.

Font Converter comes on one SSSD flippy disk. The flip side of the disk contains some of Jim Peterson's original screen fonts for you to convert. The entire collection of Peterson's screen fonts for you to convert is available from me on three additional SSSD flippy disks.

TURNFONT
by Bruce Harrison

Here is something to do with all the strange CHARA1 fonts you make with Font Converter. Turn them on their sides, 90 degrees, or turn them again to make the characters upside down. Turnfont will input any CHARA1 font and output to a disk file the same CHARA1 font rotated 90 degrees either right or left. You can take a previously rotated font file, run it through Turnfont again, and get an upside-down CHARA1 font. The resulting rotated fonts can be used with Drawing Program, Funnelweb's central menus, or with word processing software. The results are really strange and humorous! The software and some sample turned fonts comes on one SSSD disk.

CALL LINKable XB ENHANCEMENTS
by Bruce Harrison

Each of the following Harrison public domain assembly utilities comes on a SSSD disk and can be merged into and CALL LINKed from your favorite XB programs. These can be added to your XB programs in any of three ways.: 1- Just CALL LOAD Bruce's utilities into memory and them CALL LINK to them. You do this while you are experimenting with the utilities 2- Use ALSAVE to imbed a utility into your program. 3- Imbed the utility in your program with "Hi Mem Loader" which, unlike ALSAVE, leaves all of low memory available.

Once these utilities are imbedded into your XB program (you can imbed several utilities into the same XB program) they are transparent. You OLD or RUN the program as you normally would from XB and the assembly code is automatically loaded ready for your program's CALL LINK. Each utility comes with source code, demo XB programs, the necessary software to imbed the utility into your XB program, and Bruce Harrison's user-friendly step-by-step documentation.

CALL FILES XB

This allows you to do CALL FILES from within a running XB program. Previously, CALL FILES could be executed only from command mode. Our computers normally default to CALL FILES(3) which lets us have three files open simultaneously. With this utility imbedded in your XB program you no longer have to do a CALL FILES(x) from command mode before running a program that needs more than three simultaneously open files or a program that needs the extra stack memory opened up by CALL FILES(1). Just OLD and RUN your XB program and the needed CALL FILES(x) is done automatically by the program,

TIME OUT

This puts a time limit for INPUT, ACCEPT AT, and CALL KEY. If data is not entered within the time allowed, the computer assumes just <enter> has been pressed and XB program execution continues on the basis of this null string input. The time limit is easily modified by changing a parameter of the CALL LINK statement in the XB program. An obvious use for this sort of time limit is in memory games where a player has only so much time to input an answer.

BACKGROUND MUSIC

This plays background music while waiting for user input at INPUT, ACCEPT AT, and CALL KEY. You can even have the music play while editing the program from command mode. Actually you can start and stop the music anytime you want. CALL LINK("CHIME") turns it on and CALL LINK("ENDSND") turns it off. Music is provided by a "sound list" as described in the Editor/Assembler manual. Such lists resemble, but aren't quite the same as, a series of CALL SOUND statements. Bruce provides three sound files in both source and object code for those like me who have no talent creating music and don't understand sound lists. Sounds in-

(See Page 26)

MICRO-REVIEWS—

(Continued from Page 25)

clude a nice three-part chime and 16 bars of a Vivaldi sonata.

An additional utility in the Background Music software package allows you to use music to time out at ACCEPT AT. The music plays for a defined length of time. If there is no user input during this time, the computer simulates pressing <enter> to yield a null string and the XB program continues from that point. Bruce includes a really cute demo of this feature in the form of a small Jeopardy television game simulation. The Jeopardy theme (the third sound file provided by Bruce) plays while the program displays an answer and waits for the user to input the proper question. When the music is finished, time is up, the opportunity for user input ends, and the user loses.

Bruce really appreciates your comments and suggestions. Feedback from TI users is why he enjoys providing us with all this great public domain software. Write him at 5705 40th Place, Hyattsville MD 20781. His phone number is (301) 277-3467. I can be reached at P.O. Box 647, Venedocia OH 45894. My phone is (419) 667-3131 and you can use cgood@lima.ohio-state.edu to send me internet email.

USER NOTES

NEW without CLEARing the screen

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

Jerry Keisler's LOADER100 program (June 1994 MICROpendium) is an excellent example of an Extended BASIC program that writes and Extended BASIC program. Unfortunately, the final screen message (statement 970) is erased as soon as NEW is entered. Did you foresee this and make a note of the remaining two instructions?

CALL LOAD(-31952,255,231,255,231) tells the TI that no program is present, but it doesn't disturb the screen. This should be followed with :: END. The entire program is still in memory, but the TI has been tricked into thinking that no program is present.

Before using CALL LOAD, CALL INIT must be in effect. It should either be done or have already been done, but it should never be re-done, so it must be checked with a CALL PEEK.

These four statements can be added to LOADER100 as an enhancement. Statements 969 and 971 replace and bypass the original 970. Line 972 does a CALL INIT only if it hasn't been done, and 973 fakes NEW without clearing the screen.

```
969 DISPLAY AT(15,1)ERASE AL
L:"An equivalent to NEW has
    been done.": :"Now, do t
he following:" :: GOTO 971 !
 Modified Program !127
971 DISPLAY AT(19,2):"MERGE
DSK";DISK$;".CAT":" SAVE DSK
";DISK$;".LOAD": :""RUN DSK
";DISK$;".LOAD"" to test." !
103
972 CALL PEEK(8198,K,S):: IF
 K<>170 OR S<>85 THEN CALL I
```

(See Page 27)

Coffey new manager of TI-NET

Jerry Coffey has become the new manager of the TI-NET, special interest group for TI Geneve users on Delphi, as of Sept. 1.

Coffey has been assistant manager for TI-NET for a number of years.

He replaces Jeff Guide, who has managed the SIG for the past seven years. Guide is moving on to an expanded role as Internet resource developer for Delphi, a position he has held since May.

Guide's new responsibilities include assisting other sysops in setting up Internet services and setting up special Internet features.

Guide notes that TI-NET recently ranked 101 out of 500 SIGs on Delphi.

BUGS & BYTES

Making exchanges

Persons wanting to avoid shipping charges for TI repairs can bring their defective items to Cecure Electronics in Muskego, Wisconsin, when they are in the neighborhood, according to Don Walden of the company, the official TI99/4A service center.

Also, anyone who is going to the Chicago-Milwaukee fair Nov. 12 in Gurney, Illinois, can do the same thing, but needs to let Walden know in advance what the item is that needs replacement. Walden says he can't bring everything with him, but will be happy to bring anything that he knows about. The customer also needs to bring the defective item to the fair.

Coming from all over

Speaking of fairs, Berry Harmsen of the TI Gebruikersgroep in the Netherlands says his group will be represented at the International TI-Meeting in Göttingen, Germany, Oct. 14, along with eight German clubs and groups from Belgium, France, Austria, Switzerland and Great Britain, plus the one remaining TI vendor in Europe.

USER NOTES

(Continued from Page 26)

```
NIT !232
973 CALL LOAD(-31952,255,231
,255,231):: END !242
```

Statements 38-40 of LOADER100 mention a possible problem with statement 580. You might try this addendum to statement 580 as a fix:

```
,OUTPUT ! Modified Program:
",OUTPUT" added to fix the
note in 38-40
```

TI-BASE date handling

This comes from Allen J. Rogers, of Springfield, Ohio. He writes:

In TI-BASE, it is sometimes convenient to handle the date that you put in on start-up (e.g. 12/25/94) so that it comes out as 25 December 1994. If you are using the latest version of TI-BASE (V 3.02, dated Aug. 28, 1990), you can place a file in the INSTALL area which will do this for you with a simple command. See Fig. 1 for the command file to use.

Install this command file under the name LDT in your INSTALL area. Change the disk number shown to the program disk number.

CREATE the following small database named MNTH and save it to the program disk:

```
Field  Descriptor Type Width Decimal
       MONTH      C    10
       NUM        N    3     0
```

Load this database with:

MONTH	NUM
January	1
February	2
March	3
April	4
May	5
June	6
July	7
August	8
September	9
October	10
November	11
December	12

After you load the database, sort it on NUM. Keep it on the program disk under the name MNTH.

When you write a command file in which you want the date to appear as 25 December 1994, type the following commands in the command file:

```
PRINT (LPT)
DO LDT
PRINT (LPT)
```

Where LPT is the printer command which resets the printer to standard mode. The date will be set 25 spaces from the left margin.

Fig. 1

```
LOCAL XDATE D 8
LOCAL LDT C 17
LOCAL MM N 2 0
LOCAL DD N 2 0
LOCAL YY N 2 0
LOCAL MTH C 10
REPLACE XDATE WITH .DATE
REPLACE YY WITH SUBSTR(XDATE,1,2)
REPLACE MM WITH SUBSTR(XDATE,3,2)
REPLACE DD WITH SUBSTR(XDATE,5,2)
SELECT 5
USE DSK5.MNTH
TOP
FIND MM
REPLACE MTH WITH MONTH
REPLACE LDT WITH DD | " " | TRIM(MTH);
  | " " | 19 | YY
PRINT (EMP) (25 ),"Date: ",LDT
CLOSE
SELECT 1
RETURN
```

Soundmaker ideal for experimenting

The following program, called Soundmaker, was written by the late Jim Peterson of Tigercub fame. The program runs in Extended BASIC. It requires a memory expansion. It is designed to create sound effects based on user input. Variables include tone, delay and frequency. Sound effects can be saved to disk.

SOUNDMAKER

```
100 CALL CLEAR :: GOTO 140 !
046
110 A;B;C;CA$;D;DEL(1);DL;DL
M;DLMA;DM;DMA;DR(1);DX;F;FL;
FLAG;FM;FMA;FR(1,1);G(1);J;K
;K$;L;LN;M$;N$;OP$;OP2$;P;P$
;PR;Q$;R;R$;S;T;V;VL(1,1);VM
 !018
120 @;@@;VMA;X;Y$;C$;D$;L$;L
1;L2;M$;F$;RF :: CALL COLOR
:: CALL SOUND :: CALL SCREEN
 :: CALL KEY :: CALL HCHAR :
: CALL CHAR :: CALL GCHAR !1
62
130 !@P- !064
140 CALL SCREEN(16):: FOR S=
2 TO 14 :: CALL COLOR(S,5,1)
:: NEXT S :: K$="," :: CA$="
CALL SOUND(" !001
150 CALL CHAR(94,"3C4299A1A1
99423C")!programmed by Jim P
eterson Dec. 1984 - copyrigh
t 1984 Tigercub Software, 15
6 Collingwood Ave., Columbus
 OH 43213 !058
160 DISPLAY AT(2,9):"SOUNDMA
KER": :"TCX-1137 ^ Tigercub
Software" :: DISPLAY AT(6,1)
:" This program will help yo
u":"to develop sound effects
." !081
170 DISPLAY AT(8,1):" The de
fault values for each":"opti
on will be those chosen":"th
e previous time, so that":"y
ou can easily experiment" !1
79
180 DISPLAY AT(12,1):"with m
odifying a sound.":" For ins
tance, you can use":"the ran
dom option until you":"hear
an interesting effect," !102
190 DISPLAY AT(16,1):"then s
witch to selective to":"refi
ne it.":" When you have perf
ected the":"effect, use the
```

(See Page 28)

USER NOTES

(Continued from Page 27)

```
selective" !135
200 DISPLAY AT(20,1):"and sa
 ve options to save it":"to d
 isk as a MERGE format":"file
 ." !172
210 DISPLAY AT(24,10):"Press
  any key" :: CALL KEY(0,K,S)
 :: IF S=0 THEN 210 !012
220 CALL CLEAR :: DISPLAY AT
 (3,1):" The MERGE file can t
 hen be":"converted to a RUNa
 ble pro-":"gram, or MERGEd i
 nto another" !162
230 DISPLAY AT(6,1):"program
 , by MERGE DSK1.(filename)"
 !234
240 DISPLAY AT(9,1):" If you
  select 4 tones, re-":"membe
 r that one of them must":"be
  a noise tone between -1":"a
 nd -8." !240
250 DISPLAY AT(14,1):" For b
 ass tones, select the":"4-no
 te option. Give the 1st":"an
  d 2nd notes a positive":"val
 ue; the frequency and" !171
260 DISPLAY AT(18,1):"volume
  may be either audible":"or
 inaudible. Give the 3rd":"no
 te a positive value below":"
 1000 and a volume of 30, and
 " !078
270 DISPLAY AT(22,1):"the 4t
 h note a -4 noise with":"an
 audible volume.":"
 Press any key" !191
280 CALL KEY(0,K,S):: IF S=0
  THEN 280 !093
290 DISPLAY AT(12,1)ERASE AL
 L:"This program is sold by":
 "TigercubÅSoftware for only"
 :"$3.00 -  if you've got $2.
 98":"morals  and a $2.98" !1
 57
300 DISPLAY AT(16,1):"consci
 ence,ögo ahead and":"steal i
 t!" :: DISPLAY AT(23,1):"Pre
 ss any key" !013
310 GOSUB 1280 :: CALL KEY(0
 ,K,S):: IF S=0 THEN 310 !082
320 CALL CLEAR :: D=500 :: :
 : A,B,C,R,T=1 :: OP$="S" ::
 Q$="C" :: P$,R$,Y$="Y" !pred
 efine initial default values
  !169
330 DISPLAY AT(1,1):"CHOOSE
 - ":" " :: DISPLAY AT(2,1):"
   (S)elective":"  (R)andom":
 "  (D)isable this option":"
 ":" ":" ":" ":" " :: DISPLAY
  AT(1,11):OP$ :: OP2$=OP$ !1
 53
340 ACCEPT AT(1,11)VALIDATE(
 "RSD")SIZE(-1):OP$ :: IF OP$
 ="R" THEN 820 :: IF OP$="D"
 THEN FLAG=1 :: OP$=OP2$ :: G
 OTO 330 !flag delete option
 !103
350 CALL HCHAR(1,1,32,320)::
  DISPLAY AT(1,1):"CHOOSE - "
  :: DISPLAY AT(2,1):"  (R) R
 epeat":"  (C) Change":"  (S)
  Save" !164
360 DISPLAY AT(1,10):Q$ :: A
 CCEPT AT(1,10)VALIDATE("RCS"
 )SIZE(-1):Q$ !183
370 IF Q$="S" THEN GOTO 1090
  ! to save as MERGE file !17
 6
380 IF Q$="R" THEN 560 ! to
 repeat last routine !157
390 GOSUB 670 !get input for
  change - how many repeats?
 !211
400 GOSUB 800 ! how many sou
 nds? !052
410 FOR J=@ TO A :: IF A<@@
 THEN 430 !116
420 X=J :: GOSUB 690 :: DISP
 LAY AT(1,1):N$;" sound -" !0
 78
430 D=DR(J):: IF D=0 THEN D=
 1 !252
431 GOSUB 610 :: DR(J)=D ! g
 et duration of each sound !2
 21
440 GOSUB 810 !how many tone
 s? !171
450 FL=0 :: FOR L=@ TO B ::
 IF B<@@ THEN 470 !107
460 X=L :: GOSUB 690 :: DISP
 LAY AT(4,1):N$;" tone -" !22
 3
470 F=FR(J,L):: GOSUB 620 ::
  FR(J,L)=F :: V=VL(J,L):: GO
 SUB 660 :: VL(J,L)=V :: IF B
 =1 THEN 490 ! get tones !058
480 NEXT L !226
490 IF R<@@ THEN 510 !141
500 DL=DEL(J):: GOSUB 680 ::
  DEL(J)=DL ! how long delay
 between sounds? !167
510 IF A=1 THEN 530 !010
520 NEXT J !224
530 CALL HCHAR(1,1,32,768)::
  PR=11 :: FOR P=1 TO A :: ON
  G(P)GOSUB 710,720,730,740 :
 : PR=PR-(LEN(M$)>28):: IF DE
 L(P)=0 THEN 550 !print CALL
 SOUNDS on screen !015
540 PR=PR+1 :: DISPLAY AT(PR
 ,1):"FOR DELAY=1 TO";DEL(P);
 ":: NEXT D" !print delays on
  screen !174
550 PR=PR+1 :: NEXT P !037
560 IF RF=0 THEN 350 !if fre
 quencies undefined !137
570 ON ERROR 1270 :: FOR T=1
  TO R :: FOR S=1 TO A :: ON
 G(S)GOSUB 760,770,780,790 ::
  IF DEL(S)=0 THEN 590 !play
  the sounds !171
580 FOR DX=1 TO DEL(S):: NEX
 T DX !for the delays !167
590 NEXT S :: NEXT T :: IF F.
 LAG=0 THEN 330 !go to 1st op
 tion if not deleted !148600
 IF OP2$="S" THEN 350 ELSE 82
 0 !if 1st option deleted, go
  to previous 2nd option !132
610 DISPLAY AT(2,1):"Duratio
 n? ";D:"  (-4250 to 4250)" :
 : ACCEPT AT(2,12)VALIDATE("0
 123456789- ")SIZE(-5):D :: I
 F D<-4250 OR D>4250 OR D=0 T
 HEN 610 ELSE RETURN !039
620 DISPLAY AT(5,1):"Frequen
 cy? ";F-(F=0)*110:"  (110 to
  44733 or -1 to -8)" :: ACCE
 PT AT(5,12-(F>0))VALIDATE("0
 123456789- ")SIZE(-5):F !172
630 IF F>-9 AND F<0 OR F>109
  AND F<44734 THEN 640 ELSE 6
 20 !check valid noise or fre
 quency parameters !168
640 FL=FL+ABS(F<0)*2 !flag s
 election of noise !205
650 IF L=4 AND FL=0 THEN 620
  ELSE RETURN !4-tone sound m
 ust contain noise !149
660 DISPLAY AT(6,1):"Volume:
    ";V;"(0 to 30)" :: ACCEPT
  AT(6,12)VALIDATE("012345678
```

(See Page 29)

USER NOTES

(Continued from Page 28)

```
9 ")SIZE(-2):V :: IF V<0 OR
V>30 OR V<>INT(V)THEN 660 EL
SE RETURN !019
670 RF=1 :: DISPLAY AT(1,1):
"Repeat how many times? ";R:
" ":" ":" " :: ACCEPT AT(1,2
5)VALIDATE("0123456789 ")SIZ
E(-3):R :: RETURN !031
680 DISPLAY AT(8,1):"Delay b
etween?     ";DL;"  (0 -1000
)" :: ACCEPT AT(8,21)VALIDAT
E("0123456789 ")SIZE(-4):DL
:: RETURN !225
690 IF X<4 THEN N$=SEG$("1st
2nd3rd",X*3-2,3)ELSE N$=STR$
(X)&"th" !150
700 RETURN !136
710 M$=CA$&STR$(DR(P))&K$&ST
R$(FR(P,1))&K$&STR$(VL(P,1))
&")" :: DISPLAY AT(PR,1):M$
:: RETURN !print 1-tone soun
d !092
720 M$=CA$&STR$(DR(P))&K$&ST
R$(FR(P,1))&K$&STR$(VL(P,1))
&K$&STR$(FR(P,2))&K$&STR$(VL
(P,2))&")" :: DISPLAY AT(PR,
1):M$ :: RETURN !2-tone !037
730 M$=CA$&STR$(DR(P))&K$&ST
R$(FR(P,1))&K$&STR$(VL(P,1))
&K$&STR$(FR(P,2))&K$&STR$(VL
(P,2))&K$&STR$(FR(P,3))&K$&S
TR$(VL(P,3))&")" :: DISPLAY
AT(PR,1):M$ :: RETURN !3 !16
3
740 M$=CA$&STR$(DR(P))&K$&ST
R$(FR(P,1))&K$&STR$(VL(P,1))
&K$&STR$(FR(P,2))&K$&STR$(VL
(P,2))!228
750 M$=M$&K$&STR$(FR(P,3))&K
$&STR$(VL(P,3))&K$&STR$(FR(P
,4))&K$&STR$(VL(P,4))&")" ::
 DISPLAY AT(PR,1):M$ :: RETU
RN !4-tone sound !238
760 CALL SOUND(DR(S),FR(S,1)
,VL(S,1)):: RETURN !play 1-t
one sound !240
770 CALL SOUND(DR(S),FR(S,1)
,VL(S,1),FR(S,2),VL(S,2)::
RETURN !2-tone !078
780 CALL SOUND(DR(S),FR(S,1)
,VL(S,1),FR(S,2),VL(S,2),FR(
S,3),VL(S,3)):: RETURN !3 !2
34
790 CALL SOUND(DR(S),FR(S,1)
,VL(S,1),FR(S,2),VL(S,2),FR(
S,3),VL(S,3),FR(S,4),VL(S,4)
):: RETURN !4-tone sound !15
1
800 DISPLAY AT(1,1):"How man
y sounds? ";A:" ":" " :: ACC
EPT AT(1,19)VALIDATE("012345
6789 ")SIZE(-3):A :: IF A>10
 THEN 800 ELSE RETURN !049
810 DISPLAY AT(3,1):"How man
y tones? ";G(J)-(G(J)=0):: A
CCEPT AT(3,18)VALIDATE("1234
")SIZE(-1):B :: G(J)=B :: RE
TURN !111
820 RF=1 :: RANDOMIZE :: CAL
L HCHAR(2,1,32,224):: DISPLA
```

(See Page 30)

USER NOTES

(Continued from Page 29)

```
Y AT(1,1):"Entirely random? (Y/N) ";R$ :: ACCEPT AT(1,24)VALIDATE("YN")SIZE(-1):R$ :: IF R$<>"Y" THEN 920 :: Y$="Y" !198
830 R=INT(5*RND+5):: A=INT(5*RND+1)!random 5-9 repeats, 1-5 sounds !033
840 FOR J=1 TO A :: DR(J)=INT(200*RND+10):: IF R$="N" THEN 850 :: B=INT(4*RND+1):: G(J)=B !218
850 FOR L=1 TO B !131
860 IF L=4 AND FL=0 THEN 880 !4-tone sounds must have a noise !061
870 IF FL=2 THEN 890 :: ON INT(2*RND+1)GOTO 880,890 !randomly frequency or noise !067
880 FL=2 :: FR(J,L)=-(INT(5*RND+4)):: GOTO 900 !random noise - turn on flag !033
890 FR(J,L)=INT(1500*RND+110)!random frequency 110-1609 !055
900 VL(J,L)=INT(5*RND)*2 :: NEXT L :: DEL(J)=INT(20*RND):: IF Y$="N" THEN 910 :: FL=0 !random volume 0-8, delay 0-19, turn off flag !227
910 NEXT J :: CALL HCHAR(11,1,32,448):: GOTO 530 !erase previous print - to print and play !160
920 GOSUB 670 :: GOSUB 800 :: J=0 :: GOSUB 810 :: FOR J=1 TO A :: G(J)=B :: NEXT J :: DISPLAY AT(3,1):"With noise? (Y/N) ";Y$ !restricted random !017
930 ACCEPT AT(3,19)VALIDATE("YN")SIZE(-1):Y$ :: IF (Y$="N")*(B=4)THEN 930 !020
940 IF Y$="N" THEN FL=2 !208
950 DISPLAY AT(1,1):"Restrict parameters? (Y/N)";P$ :: ACCEPT AT(1,27)VALIDATE("YN")SIZE(-1):P$ :: IF P$="N" THEN 840 !027
960 DISPLAY AT(2,1):"Duration - minimum? ";DM :: ACCEPT AT(2,22)VALIDATE("0123456789")SIZE(-4):DM :: IF DM<-4250 OR DM>4250 OR DM=0 THEN 960 !063
970 DISPLAY AT(3,1):" maximum? ";DMA :: ACCEPT AT(3,22)VALIDATE("0123456789")SIZE(-4):DMA :: IF DMA<-4250 OR DMA>4250 OR DMA=0 OR DMA<DM THEN 970 !027
980 DISPLAY AT(4,1):"Frequency - minimum? ";FM :: ACCEPT AT(4,23)VALIDATE("0123456789")SIZE(-5):FM !039
990 IF FM>-9 AND FM<0 OR FM>109 AND FM<44733 THEN 1000 ELSE 980 !119
1000 DISPLAY AT(5,13):"maximum? ";FMA :: ACCEPT AT(5,23)VALIDATE("0123456789")SIZE(-5):FMA !182
1010 IF FM<0 AND FMA>0 THEN 980 !103
1020 IF FMA>-9 AND FMA<0 OR FMA>109 AND FMA<44733 AND FMA>FM THEN 1030 ELSE 1000 !144
1030 DISPLAY AT(6,1):"Volume - minimum? ";VM :: ACCEPT AT(6,19)VALIDATE("0123456789")SIZE(-2):VM :: IF VM<0 OR VM>30 THEN 1030 !062
1040 DISPLAY AT(7,10):"maximum? ";VMA :: ACCEPT AT(7,19)VALIDATE("0123456789")SIZE(-2):VMA :: IF VMA<0 OR VMA>30 OR VMA>VM THEN 1030 !138
1050 DISPLAY AT(8,1):"Delay - minimum? ";DLM :: ACCEPT AT(8,18)VALIDATE("0123456789")SIZE(-3):DLM !254
1060 DISPLAY AT(9,9):"maximum? ";DLMA :: ACCEPT AT(9,18)VALIDATE("0123456789")SIZE(-3):DLMA :: IF DLMA<DLM THEN 1060 !137
1070 FOR J=1 TO A :: DR(J)=INT((DMA-DM+1)*RND+DM):: FOR L=1 TO B :: FR(J,L)=INT((FMA-FM+1)*RND+FM):: VL(J,L)=INT((VMA-VM+1)*RND+VM):: NEXT L !055
1075 IF B=4 THEN FR(J,4)=-(INT(8*RND+1))!175
1080 DEL(J)=INT((DLMA-DLM+1)*RND+DLM):: NEXT J :: CALL HCHAR(11,1,32,448):: GOTO 530 !130
1090 IF RF=0 THEN 350 :: DISPLAY AT(1,1)ERASE ALL:"Filename? DSK" !to save CALL SOUNDS in MERGE format !054
1100 ACCEPT AT(1,14):F$ !252
1110 DISPLAY AT(2,1):"Starting line number? " :: ACCEPT AT(2,23)VALIDATE(DIGIT):LN !088
1120 OPEN #1:"DSK"&F$,OUTPUT,DISPLAY ,VARIABLE 163 !035
1130 C$=CHR$(179)!212
1140 GOSUB 1260 !064
1150 FOR J=1 TO A :: M$=L$&CHR$(157)&CHR$(200)&CHR$(5)&"SOUND"&CHR$(183)!196
1160 M$=M$&CHR$(200)&CHR$(LEN(STR$(DR(J))))&STR$(DR(J))&C$ !CALL SOUND and duration !087
1170 ON G(J)GOSUB 1220,1230,1240,1250 !185
1180 M$=M$&CHR$(182)&CHR$(0):: PRINT #1:M$ :: GOSUB 1260 !close parenth., close line !245
1190 IF DEL(J)=0 THEN 1210 !else save delay routine !078
1200 PRINT #1:L$&CHR$(140)&"D"&CHR$(190)&CHR$(200)&CHR$(1)&"1"&CHR$(177)&CHR$(200)&CHR$(LEN(STR$(DEL(J))))&STR$(DEL(J))&CHR$(130)&CHR$(150)&"D"&CHR$(0):: GOSUB 1260 !191
1210 NEXT J :: PRINT #1:CHR$(255)&CHR$(255):: CLOSE #1 :: END !close MERGE, close file !039
1220 M$=M$&CHR$(200)&CHR$(LEN(STR$(FR(J,1))))&STR$(FR(J,1))&C$&CHR$(200)&CHR$(LEN(STR$(VL(J,1))))&STR$(VL(J,1)):: RETURN ! 1st note !024
1230 GOSUB 1220 :: M$=M$&C$&CHR$(200)&CHR$(LEN(STR$(FR(J,2))))&STR$(FR(J,2))&C$&CHR$(200)&CHR$(LEN(STR$(VL(J,2))))&STR$(VL(J,2)):: RETURN !068
1240 GOSUB 1230 :: M$=M$&C$&CHR$(200)&CHR$(LEN(STR$(FR(J,3))))&STR$(FR(J,3))&C$&CHR$
```

(See Page 31)

USER NOTES

(Continued from Page 30)

```
(200)&CHR$(LEN(STR$(VL(J,3))
))&STR$(VL(J,3)):: RETURN !0
82
1250 GOSUB 1240 :: M$=M$&C$&
CHR$(200)&CHR$(LEN(STR$(FR(J
,4))))&STR$(FR(J,4))&C$&CHR$
(200)&CHR$(LEN(STR$(VL(J,4))
))&STR$(VL(J,4)):: RETURN !0
96
1260 L1=INT(LN/256):: L2=LN-
256*L1 :: L$=CHR$(L1)&CHR$(L
2):: LN=LN+10 :: RETURN !alg
orithm to convert line numbe
r to MERGE format !109
1270 DISPLAY AT(21,1):"ERROR
! - CALL SOUND CANNOT":"CONT
AIN TWO NEGATIVE VALUE":"NOI
SES!" :: RETURN 350 !162
1280 CALL GCHAR(13,11,@):: @
=INT(SQR(@))-10 :: CALL GCHA
R(16,14,@@):: @@=INT(SQR(@@)
)-10 :: RETURN !028
```

IF-THEN in XBASIC

The following was written by Jim Swedlow in the User Group of Orange County (California) ROM newsletter.

A number of the Extended BASIC columns discussed alternatives to IF-THEN. Here is another.

Suppose that A$ depends on the value of I. You might use this code:

```
IF I=1 THEN A$="FRED" ELSE A
$="PAUL"
```

A simpler way is to use the SEG$ function:

```
A$=SEG$("PAULFRED",1-4*(I=1)
,4)
```

Will this work if the two variables have different lengths? Yes! Remember that SEG$ does not produce an error if the length of the new string (the last number) is longer than the source string. If our two names are "PAUL" and "SAM," this works:

```
A$=SEG$("PAULSAM",1-4*(I=1),
4)
```

MICROpendium pays $10 for items submitted by readers and used in this column. Send them to MICROpendium User Notes, P.O. Box 1343, Round Rock, TX 78680.

CLASSIFIEDS

FOR SALE

"THE WATCHAMACALIT"
From DBM TECHNOLOGIES

ATTENTION MYARC HFDC USERS! LIMITED SUPPLY $29.95 Each. GET YOUR HFDC UPGRADED NOW! From DBM TECHNOLOGIES 17725 22nd Street Ogden, Utah 84401-2112 Phone: (801) 782-1004 or (801) 394-6815. "THE WATCHAMACALIT"" is designed to Back-Up the clock on the Myarc HFDC. Reviewed in MICROpendium in August 1993. Comes complete with software. v11n10

PROGRAM INNOVATORS
SOFTWARE SALE

TOUCHDOWN: NFL Football Predictor $10.00 WALSTREET: Investment Package $30.00 USVBA Power Volleyball (ML-Game) $10.00 Klingon Invaders, Destroy Klaatu, Desert Rat (Machine Language Games) $10.00 Cutthroat Cribbage, Tltris, Cockroaches, Snomobile, Martian Missiles, $10.00 Multiple orders 10% discount 4122 Glenway, Wauwatosa WI 53222-1116

FOR SALE

Geneve 9640 card and manual. Call Jim (305) 294-7638

TI MONITOR

Original TI monitor. Asking $50. Call 512-255-1512.

The ONLY monthly devoted to the TI99/4A

Subscription Fees

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

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

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

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

Address Changes

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

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

Check/MO ❑ ❑ ❑ (check one)

Card No. _____

Expiration Date _____

(Minimum credit card order is $9)

Signature _____

(Required on credit card orders.)

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

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

Name _____

Address _____

City _____

State _____ **ZIP** _____

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

Disks, Etc.

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

_____

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

OUT OF STOCK: Vols. 1, No. 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 2.0** (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

SECOND CLASS