Covering the TI99/4A and the Myarc 9640

MICROpendium

Volume 14 Number 1 January/February 1997 $6

Extended BASIC

Writing assembly with an Extended BASIC program — Page 14

• • •

Disk and mailing labels from one program — Page 24

• • •

Creating a masterfile of D/V80 files — Page 32

New power supply

Ridding your cards and console of voltage regulators can be good for your system — Page 25

• • •

Of SCSI, RLL and MFM

TIers keep up with changes in technology — Page 29

Reviews

AMS Slideshow

Loadmaster V2.2

Contract Bridge V4.0

• • •

Geneve

Some advice on must-have software and hardware upgrades to keep you ahead of the game

Assembly

Sometimes you can't go by the book — Page 8

• • •

c99

Comparing c99 with Extended BASIC — Page 16

CONTENTS

MICROpendium

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

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

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

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

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

Telephone & FAX: (512) 255-1512

Delphi TI NET: MICROpendium

Internet E-mail: jkoloen@io.com

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.

TEX COMP LTD. FREEWARE ONLY

Public Domain and Shareware programs to meet your every need.

XB & 32K Req. for most programs

LISTED BY SUBJECT

$2.95 PER DISK

GRAPHICS, MUSIC, & ANIM.

#1 THE SINGING TI VOL.1 (S)
#4 PRINT ART (P)
#5A MUSIC/GRPAHICS
#6 XB MUSIC
#7 SPACE SHUTTLE MUSIC/GRAPHICS
#9 MONA LISA PRINTOUT (P)
#11 ANIMATED XMAS (WOODSTOCK)
#14 FIGURE STUDY (P)
#32 XB XMAS MUSIC
#41 VIDEO GRAFIS
#52 ANIMATION 99
#69 PLAYER PIANO
#93 KBGB GIRLIE CALENDAR (P) *
#103 SORGAN (ORGAN)
#107 STARTREK MUSIC ALBUM
#111 POPMUSIC & GRAPHICS
#114 PANARAMA
#115 GRAPHICS DESIGN
#120 BITMAC (P)
#230 THE SINGING TI VOL. 2 (S)
#231 THE SINGING TI VOL. 3 (S)
#246 THE SINGING TI VOL. 4 (S)
#313 3-D WORLD

COMPUTER & PRINTER UTILS. PROGRAMMING LANG.

#3 DUMPIT (E/A)
#15 SIRAT/EPSON DEMO (P)
#16 SIDEWAYS PRINTOUT
#18 TI DIAGNOSTIC (MM)
#28 LOADERS / CATALOGERS
#30 HOUSEHOLD BUDG. PRINTOUT (P)
#35 PROGRAMMING AIDS/UTIL VOL. 1
#42 FUNNELWEB FARM
#55 HACKER CRACKER
#58 SCREEN DUMP (P)
#62 DISK MANAGER II (M)
#75 DISK CATALOGER
#76 PROGRAMMING AIDS/UTIL VOL. 2
#78 ARTCON GRAPHICS CONVERTER
#79 DISK MANAGER 1000
#80 BIRDWELL DISK UTILITY
#85 AUTOBOOT UTILITY
#86 COLUMN TEXT III (P)
#87 ARCHIVER III
#89 PROCALC DEC/HEX CONV.
#96 STATS & SORT ROUTINES
#97 MEMORY MANIPULATOR
#101 ENHANCED DISPLAY PACK
#104 C99 COMPILER LIBRARY
#108 FULPLUS
#110 DISK + AID
#117 UNIVERSAL DISASSEMBLER
#119 RAG LINKER
#127 PIX GRAPHICS UTILITY
#243 OS/99 (Gram Device)
#251 PC TRANSFER TI/IBM (DD)
#254 NIBBLER/TURBO
#256 EXPLORER / DM1000
#260 TI FORTH (DISK ONLY)
(Add $12 for manual)
#17 TI FORTH DEMO
#116 TI FORTH TUTORIAL
#307 GRAPHICS CODE GENERATOR

BUSINESS, HOME, ACCOUNTING, DATA BASE, & WORDPROCESSING

#10 GOTHIC PRINTOUT (P)
#19 TI WRITER/MULTIPLAN UPGRADE
#20 ACCOUNTS RECEIVABLE
#21 DATA BASE DEMO
#23 WILL WRITER
#29 LABLE MAKER (P)
#36 STRICTLY BUSINESS
#56 SPREAD SHEET
#58 PR BASE
#59 GRAPH MAKER
#74 LABEL MAKER II (P)
#77 MICRODex 99
#81 HOME ACCOUNTING SYSTEM
#83 HOME APPLICATION PROGRAM
#90 JET CHECKBOOK MANAGER
#92 HOUSEHOLD INVENTORY
#109 TI WRITER MINI MANUAL
#112 INVOICE PACK
#113 LABEL MAKER TI (P)
#129 CASH DRAW "P O S"
#130 THE ORGANIZER
#145 CALENDER-NOTEPAD
#147 HOUSEHOLD BUDG. MANG. (M)
#30 BUS.M PRINT OUT (P)
#221 PERSONAL REAL ESTATE (M)
#249 MAPMAKER
#252 99 WRITER II (P)
#253 AMA MAIL LIST
#255 TRAK-A-CHECK
#257 DAILY DIARY
#258 EXER-LOG
#310 SELF HELP TAX CUT

TELECOMMUNICATIONS

#57 TELCO
#57D TELCO (for systems with DS drives)
#118 FAST TERM

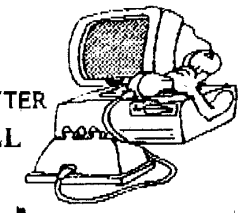
EDUCATION & PERSONAL DEVLP.

#22 ASTROLOGY
#24 ENGINEERING CALCS
#25 MEDICAL ALERT
#27 KIDS LEARNING VOL. I
#31 MORSE CODE TRAINER
#32 LAPD COOKBOOK
#40 ARTIFICIAL INTELIG.
#54 ASTRONOMY
#66 HEBREW TYPEWRITER
#67 GENEALOGY
#71 KIDS LEARNING VOL. II
#95 WEATHER FORECASTER
#138 FIREHOUSE COOKBOOK
#142 TOUCH TYP. TUTOR (M)
#184 FACE MAKER (M)
#194 ST. NICK (M)
#195 TINY LOGO
#196 JOTTO
#199 MILLIKEN ADDITION (M)
#200 MILLIKEN DECIMALS (M)
#203 MILLIKEN FRACTIONS (M)
#204 MILLIKEN INTEGERS (M)
#205 MILLIKEN LAWS MATH (M)
#211 MIND CHALLENGERS (M)
#212 MINUS MISSION (M)
#213 MILLIKEN PERCENTS (M)
#214 STORY MACHINE (M)
#215 BEGINNING GRAMMAR (M)
#216 METEOR MULTIPLE (M)
#217 HANGMAN (M)
#222 MUSIC MAKER (M)
#223 PHYSICAL FITNESS (M)
#225 ALIEN ADDITION (M)
#226 ALLIGATOR MIX (M)
#227 DEMOLITION DIVISION (M)
#229 DRAGON MIX (M)
#261 ASTRONOMY 2
#262 ASTRONOMY 3
#263 ASTRONOMY 4
#264 HIGHER MATH
#265 HIGHER MATH 2
#266 ELECTRONICS MATH
#268 CALCS. & CONVERTERS
#269 TRIGONOMETRY
#306 SPEAK & SPELL II (S)

GAMES

#2 WHEEL OF FORTUNE
#5 TI TREK
#8 LOOTO POKER
#12 STRIP POKER "TGT"
#26 R-RATED GAME
#33 CHECKERS & BACKGAMMON
#34 SOLITARE & SCRABBLE
#38 GREAT GAMES VOL. I
#39 GREAT GAMES VOL. 3
#43 BEST OF BRITAIN VOL. 1
#45 BEST OF BRITAIN VOL. 2
#46 SUPER TRIVIA 99
#47 INFOCOM RAPID LOADER
#48 GHOSTMAN
#49 DEMON DESTROYER
#50 OH MUMMY!
#51 BERLIN WALL
#60 FREDDY
#61 THE MINE
#63 ASTROBLITZ & MAZOG
#64 MAJOR TOM & SPACE STATION
#65 PERFECT PUSH
#68 CHESS (BARCON)
#70 TI RUNNER II
#72 CERBERUS
#73 CRYPTO (Gram)
#82 CROSSWORD PUZZLES
#84 GALACTIC BATTLE/SPY ADV.
#88 AUSSIE GAMES VOL. I
#91 THE MAZE OF GROG
#94 GREAT TI GAMES VOL. 3
#98 DAYS/DOORS OF EDEN (ADV.)
#99 GREAT TI GAMES VOL. 4
#100 ASSAULT THE CITY (TOD.)
#102 COLOSSAL CAVES (ADV.)
#105 KINGS CASTLE
#106 QUEST "D&D ADV."
#121 SUPER YAHTZEE & WHEEL II
#122 ADULT ADV. (ADV.)
#123 GREAT TI GAMES VOL. 5
#124 GREAT TI GAMES VOL. 6
#125 BLACKJACK & POKER (M)
#126 VIDEO CHESS (M)
#128 TETRIS
#131 COMPUTER CRAPS
#135 ARTURUS
#137 CROSS-FIRE (M)
#139 MOON MINE (M)
#140 MASH (M)
#141 MOONSWEEPER (M)
#144 STAR TREK (M)
#145 BUCK ROGERS (M)
#148 KENO & SLOTS
#149 GREAT TI GAMES VOL. 7
#150 ULTIMATE TRIVIA
#176 AMAZING (M)
#178 DEMO DEDYTROYER (M)
#179 POPEY (M)
#180 QUEBERT (M)
#181 METOER BELT (M)
#182 BLASTO (M)
#183 CARWARS (M)
#185 SUPER FLY (M)
#186 SPACE BANDITS (M)
#188 KILLER CATAPILLER (M)
#190 BLACK HOLE (M)
#191 GREAT TI GAMES VOL. 8
#192 GREAT TI GAMES VOL. 9
#193 SPYS DEMISE (M)
#197 PRO TENNIS (M)
#198 TI INVADER/TOMSTONE (M)/S
#202 CONNECT FOUR (M)
#206 SLTMOIDS (M)
#207 OTHELLO (M)
#208 PARSEC (M)
#209 SOCCER (M)
#210 SEWERMANIA (M)
#215 HUSTLE/FOOTBALL (M)
#219 CHISOLM TRAIL (M)
#220 ZERO ZAP (M)
#224 ATTACK (M)
#229 4-A FLYER (M)
#232 TUNNEL OF DOOM (M)
#233 MS ADVNTURE
#244 HOPPER (M)
#245 TREASURE ISLAND (M)
#247 STRIKE THREE (M)
#248 GREAT TI GAMES VOL. 10
#250 BARRAGE/SPOT SHOT (M)/S
#291 MAZE MANIA
#292 THE CASTLE
#319 ARCADE SPECIAL

KEY
P = PRINTER REQ.
G = GRAPHEX REQ.
S = SPEECH REQ.
M = MODULE BACKUP
MM = MINI MEMORY
E/A = EDITOR ASSEMBLER
ADV = ADVENTURE REQ.
* = TITLE IS $4.95
(To cover added costs of disk and label)



DON'T WAIT FOR YOUR COMPUTER TO CALL

ORDER BY PHONE

ORDERS *ONLY* (800) 846-3474
TECH HELP (818) 339-8924
(Call First) FAX (818) 858-2785

VISA & MASTER CARD EXCEPTED

ADD 5% SHIPPING ($5.00 MINIMUM) OUTSIDE USA INCLUDE 10% SHIP

ALL PRICES ARE FOR CASH/CHECK, ADD 3.5% FOR CREDIT CARD ORDERS

TEX COMP LTD.
AMERICAS NUMBER ONE TI COMPUTER RETAILER

425 E. ARROW HWY. #732
GLENDORA, CA 91740

COMMENTS

A good time to learn c99

Perhaps you've made a new year's resolution to learn a computer language. Or maybe you're interested in learning something new. Whatever your reason may be, this is a good time to take advantage of our newest column, Beginning c99, by Vern Jensen. One thing he's doing to make it easier on would-be c99 programmers is to offer a starter kit of c99 software that includes Clint Pulley's c99 compiler, documentation, libraries and sample programs. The cost is nominal and provides readers with everything that's needed to undertake a course in c99. The computer language "c" in its many manifestations is implemented on virtually every computer system, so learning the c99 variant will carry over if you decide to program in c on other machines — not that there would be any reason to do so. One of the problems with TI Extended BASIC, of course, is that what you learn about programming doesn't carry over to other more recent versions of BASIC as implemented on other machines. Those who learn TI assembly can carry those skills into other venues, but c99 is more accessible, especially to casual programmers. We're glad to have Vern writing the column. I think it's something that many readers can enjoy, even if they don't become c gurus.

HAMFEST CAN BE A GOLD MINE FOR TI'ERS

Also in this issue, frequent contributor Robert Carmany writes about his experience at a Hamfest he attended last year. I don't know about you, but I enjoy wandering through computer flea markets and the like. There's so much to see. Not to mention it's possible to come across outstanding bargains, sometimes even for equipment you have a use for. Looking for a hard-to-find TI part? Carmany advises that you not overlook local Hamfests. They're not just for Hams any more.

OUR WEB PAGE IS UP

MICROpendium finally has a web page. It's not going to knock your socks off, but at least it's there. It can be found at www.io.com/~jkoloen/. For those who use Internet browsers, it looks best on Netscape.

—JK

BUGS AND BYTES

It's a tough machine

This item appeared on comp.sys.ti on the Internet. It was written by Denis A. Dann.

Just thought I'd share a little story that most of the Milwaukee TI-99/4A User's Group have heard more than once.

A woman came to me and asked if I could check out her two boys' TI. It would turn on for about a minute then suddenly shut off by itself. Upon inspection I found a Chucky Cheeze token stuck in the vents at the bottom of the console. The coin was causing the unit to build up heat as it was shorting out the power supply. I removed the coin and the unit immediately fired up. Upon returning it to the woman, I mentioned that the unit looked like it had seen two world wars. She said, "Oh yes, my boys are only allowed to use one toy at a time in the living room. They bring the TI up from the basement to use it, but they are too lazy to carry it down again, so they dump it down the clothes chute. whether there were clothes already down there or not." I asked how often they did this. She said, "Oh, about three or four times a week for the last three years."

This. incident took place nine years ago. Last year, the woman and her family moved south. She gave the console to me as a present. It now looks as if it had not only seen two world wars, but Korea, Vietnam, and the war in the Gulf as well. But it still works!

Club donates computers

The Sooner 99ers in Oklahoma City have taken the recycling of old computers to heart. According to Barry Peterson, who posted a message on the TI newsgroup, the group has donated 50 computers to local schools. The donations included everything from single consoles to full-fledged systems. "We have kept title to the systems out of fear that a gift to the school would eventually be surplussed by some well-meaning administrative type who sees anything less than a Pentium-class PC as being obsolete."

Illinois club starts TI sub-board

This item appeared on comp.sys.ti, the Internet Usenet newsgroup dedicated to the TI.

The Knox Computer Club (Illinois) has started a TI-specific sub-board on its BBS (The Flipside, 309-343-7312). The TI files area is being built and additions are welcomed. The club is platform independent, with emphasis on early computers, including Commodore 64, Amiga, and others. Walid Maalouli, MD, is the lone TI club member, and invites other TIers to join or visit the BBS.

COMPETITION COMPUTER 800-471-1600

(Nationwide & Canada ORDERS ONLY)

350 MARCELLA WAY MILLBRAE, CALIFORNIA 94030

WE WILL BE AT FEST WEST 97 APRIL 5 IN SAN JOSE CA WHERE WE WILL BE RELEASING NEW HARDWARE AND SOFTWARE AND A NEW BIGGER CATALOG. MEANWHILE WE HAVE MORE WINTER READING AND SOFTWARE SPECIALS THIS MONTH.

YOU HAVE TOLD US THAT YOU WISH YOU COULD GET FAIRWARE AND SHAREWARE FROM A NONCOMMERCIAL SOURCE BUT YOUR LOCAL USER GROUP IS NO MORE. WELL NOW WE HAVE A SOLUTION: OUR LOCAL USER GROUP (THE SAME ONE THAT IS SPONSORING FEST WEST 97) WILL SELL DISKS FROM ITS LIBRARY FOR $2 EACH. FOR DETAILS AND A DISK CATALOG SEND A SELF ADDRESSED AND STAMPED BUSINESS SIZE ENVELOPE WITH 2 32 CENT STAMPS ON IT TO: SOUTH BAY TI USERS GROUP LIBRARIAN
C/O COMPETITION COMPUTER
350 MARCELLA WAY
MILLBRAE CA 94030

WE HAVE MANY TITTLES RELATED TO C LANGUAGE USEFUL FOR USERS OF TI C99 FOR $5 EACH SEND SASE FOR LIST.
STARTING FORTH BY BRODIE $5
MYARC WDS/100 USER'S MANUAL $10
SEAGATE ST506 MICROWINCHESTER OEM MANUAL $5
WESTERN DIGITAL WD1001 DISK CONTROLLER OEM MANUAL $8
(THESE LAST 3 MANUALS PROVIDE COMPLETE WDS INFO)
BEGINNING ASSEMBLY LANGUAGE FOR THE TI HOME COMPUTER BY YORK (RARE) $20
THE INNERMOST SECRETS OF THE TI 99/4A BY HOLCOMB $3
THE SOFTIES PRESENT: TUTOR,ASSEMBLY LANGUAGE TUTORIAL (ESPECIALLY GOOD IF YOU HAVE ONLY THE MINIMEM CART) $5
INTERFACE STANDARD & DESIGN GUIDE FOR TI99/4A PERIPHERALS 2ND EDITION BY TONY LEWIS WITH SOURCE CODE DISK $24.95
HACKING, THRASHING, FUN & GAMES $5
(THIS BOOK PROVES THAT BBS'S HAD THEIR BAD APPLES TOO. FOR INFORMATIONAL PURPOSES ONLY, DON'T TRY THIS STUFF!)
WORKBOOKS FOR SCHOLASTIC SPELLING LEVEL 3,4,5,+6 $5 EACH
READERS FOR SCOTT FOREMAN READING SERIES $2 EACH

READING FUN	READING RALLY	READING CHEERS
READING POWER	READING TRAIL	READING FLIGHT
READING ROUNDUP	READING WONDERS	

TEACHERS GUIDES FOR THE FOLLOWING:

READING POWER	READING TRAIL	READING RALLY
READING FLIGHT	READING ROUNDUP	FRACTIONS 1
ADDITION AND SUBTRACTION 3		$5 EACH

COMPETITION COMPUTER
350 MARCELLA WAY
MILLBRAE CA 94030
ORDERS:1-800-471-1600
INFO:415-697-1108
FAX(CALL FIRST)415-697-7406
WE ACCEPT MASTERCARD,VISA,AMERICAN EXPRESS
PERSONAL CHECKS(7 DAYS TO CLEAR) OR MONEY ORDERS
WE SHIP PRIORITY MAIL(2-3DAYS TO US ADDRESSES)
NO HANDLING CHARGES NO SRCHARGE FOR CREDIT CARDS

THIS AD WAS MADE USING:
A TI 99/4A SYSTEM RUNNING
NAVARONE CONSOLE WRITER
OUTPUT TO A CANON BUBBLE JET
(WE USE WHAT WE SELL)

DID YOUR CHILDREN OUTGROW YOUR TI EDUCATIONAL OR OTHER SOFTWARE? TRADE IT IN FOR OTHER TI SOFTWARE AND RECEIVE CREDIT EQUAL TO 50% OF OUR CURRENT RETAIL TOWARDS PURCHASE OF OUR TI SOFTWARE
OUR 2/96 CATALOG HAS 800 ITEMS FOR YOUR TI AND WE HAVE ADDED 100'S

TI-99/4A™ CD ROM

CD ROM UPDATE:
IT IS REGRETTABLE THAT THE TI COMMUNITY HAS PAID MORE MONEY FOR ANOTHER CD ROM THAN OURS THAT OFFERS LESS. IF YOU HAVE DONE THIS, WE ARE OFFERING OUR CD ROM WHICH CONTAINS 62% MORE FILES FOR ONLY $15 PLUS YOUR COPY OF THE OTHER CD.
OUR CD ROM IS STILL ONLY $35
CADD PC99 STAGE 3A: OUR CD ROM IS COMPATIBLE USING NEW PC99 UTILITY DLCONV.EXE!!
ORDER BOTH PC99 AND OUR CD ROM FROM US FOR ONLY $130

FLOPPIES FOR TIs:
BOX OF 10 FOR $1.30
INCLUDES SLEEVES+LABELS

CARTRIDGE SALE! $19.95 EACH!

ANTEATER	CHICKEN COOP
CONSOLE WRITER,DISK FIXER	
FROG STICKERS	HEN PECKED
KING OF THE CASTLE	
PRINCESS AND THE FROG	
ROMOX DEMO	ROTOR RAIDERS
SPEED READING A(TEENS+ADULTS)	
SPEED READING B(FOR CHILDREN)	
SUPER DUPER	TOPPER
TYPO-MAN	TYPO II
AMBULANCE	DRIVING DEMON
HENHOUSE	RABBIT TRAIL
SCHNOZ-OLA	ST. NICK
VIDEO VEGAS	TYPEWRITER 99
TRIS	TRIS 2
PAINT`N PRINT A(GP100,CP700)	
PAINT`N PRINT B(GP550,OKIDA)	
PAINT`N PRINT C(IBM,GEMINI)	
HOMEWORK HELPER+	
DATABASE SORT	DATABASE ENTRY

QUALITY 99 SOFTWARE TITTLES:
ANY TITTLE FOR ONLY $9.95

CHARTMAKER	XB-FORTH
DISK LABELER	QS-WRITER
DISK MANAGER3	DRAW`N PLOT
EZLOADER	SDUMP
QS-CLOCK	QS-SIDEWAYS
BANNER MAKER	DATA BASE 99
QUICK-COPIER2	OS-XREF

MasterCard VISA

FEEDBACK

Thanks to all

This is an open letter to the entire Chicago Users' Group. First, thanks for hosting the 1996 International Faire. Getting there and back was a chore, but participating was pure joy. Being able to demonstrate all my latest software for an enthusiastic crowd made my day.

My sincere thanks to all of you for giving me the Birdwell Award. I consider that perhaps the greatest honor in the "community," and it will hang in a place of honor here near my computer desk. There's a certain irony in this, as I've recently been relying very heavily on John Birdwell's DSKU to find out the contents of the "headers" on various file types. The information thus provided has been used to vastly improve the file identification feature in Load Master. Thus, John Birdwell's work has once again provided a significant benefit for the whole TI user community. It's also made Mickey Cendrowski and me very happy.

Bruce Harrison
Hyattsville, Maryland

Errors found in GEnie comments

I just recently had the opportunity to read your editorial comments regarding GEnie from your last issue of MICROpendium. If I may, I would like to correct some errors in the editorial.

At the end of January of this year, Youvelle Rennaisance, a subsidiary of IDT Corporation, purchased GEnie. As you correctly mentioned, the new owners immediately raised the rates. Estimates indicate GEnie lost nearly 80 percent of its members. Unfortunately, this was a bad business decision by the new owners and the sysops had no control over it.

During the changing of the guard, the TI and Orphans RoundTable and the CP/M RoundTable were merged into a new RoundTable called "The Golden Oldies RoundTable." This is a RoundTable to support all orphaned systems and platforms.

During the transition, files were moved from the TI and Orphans RoundTable to the Golden Oldies RoundTable. While the apparent file count indicated there were 5,000+ files, this was not the case. In fact, there have only been slightly over 3,000 TI and Geneve files. The reason for the discrepancy is due to the RoundTable also supporting other platforms and a 1988 "cleanup" after the introduction of Barry Boone's Archiver. During the transition to the new RoundTable, fewer than 1 percent of the files were lost. Those files that were lost were generally digests of conferences no longer suitable for use.

At the 1996 Chicago Faire, I promoted GEnie Interactive, the Web reincarnation of The Golden Oldies RoundTable. IDT spent $3 million and invested heavily to build a Web site. I fought with great effort to bring the Golden Oldies RoundTable to the Web site and to include a text-based interface system for the TIers. I won, and IDT was going to have a Web site where TIers could find 3,000+ files.

Unfortunately, the day after returning from the Chicago Faire, I phoned the president of GEnie Interactive and learned that corporate had made a business decision to close the whole site down. Corporate decided they did not want to compete with MSN or America On-Line. It was a personal loss to me, and a loss to the TI community where it would have allowed anyone with a modem to retrieve files or discuss TI/Geneve topics at the cheapest rate possible ($4.95/month unlimited access).

I'm always looking forward, and I can see the online services support of orphaned platforms decreasing very rapidly as their marketing departments justify other systems and platforms. Knowing the industry and desiring to insure all TIers have the opportunity to have a large resource of TI files, I made available a CD-ROM containing 3,000 archived files from the 9640 News BBS I run. The CD-ROM represents everything I am aware of released as shareware or freeware or public domain to the TI community in the last 10 years. BBS Sysops may place the CD-ROM online for their users if they have an IBM or MAC system. If the TI/Geneve ever has a CD-ROM reader that recognizes the most common ISO 9660 standard, the CD-ROM would also be useable on a TI/Geneve. No commercial softv is on the CD-ROM, unless the author/company released the software. If users are interested in the software, I have reduced the cost of the CD-ROM to $35 when it met with such favorable response and I felt I would be able to recover my costs for the equipment.

I don't know how much longer The Golden Oldies RoundTable will continue to survive on GEnie. Messages and new files are few and far between, just as they are everywhere for the TI community. I applaud the efforts of Tom Wills and Charles Good for their efforts in expanding Internet support with the listserve/mailing lists. I hope my efforts with the CD-ROM production will assist many TIers to maintain a permanent archive and maybe some TIer with a FTP server will be able to load those files onto the Web for any TIer without a CD-ROM.

I realize I have been quiet in the TI/Geneve Community. The CD-ROM production will probably be my last major project for the TI/Geneve community as I begin licensing my software to be distributed by one of your advertisers. My interest will always be with the TI/Geneve and I will forever remember the friends I have made and the support they have given me. Anyone wishing to contact me can generally catch me at one of the following three e-mail addresses: bw.miller@genie.com, beerym9@mail.idt.net, Beery_Asst@msn.com.

Beery Miller
Memphis, Tennessee

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

Western Horizon Technologies
3297 Woody Lane
San Jose, CA 95132
(408)-934-0352
e-mail: don@sonyx.com

The WHT SCSI Controller Card

Want more online storage for your 99/4a or Myarc Geneve? How does **2 GIGABYTES** sound?

The WHT SCSI card provides your computer with nearly instant access to hundreds of megabytes of storage on any standard SCSI hard drive. With lightening fast RAM Disk access speeds and nearly 2 gigabytes of capacity for programs graphics and other files! **Seven** hard drives are supported with each volume up to 250 megabytes!

Each SCSI controller kit comes complete with all manuals, software and cabling you need to install an internal hard drive in your PE-Box. Optional cases, cables and hard drives are available.

Head to Head with the HFDC

Feature	SCSI	HFDC
Industry Standard Interface	YES	Obsolete
Supports Seven Hard Drives	YES	NO
1.75 Gigabytes of Online Storage	YES	NO
10 MB/Second Drive to Host Transfer Rate	YES	NO
500 KB/Second Drive to RAM Transfer Rate	YES	NO
Faster then a Horizon Ram Disk	YES	NO
Easy to Find Off the Shelf Hard Drives	YES	NO
Interfaces to CD-ROM, Tape and CDR	YES	NO
Multiple Masters Sharing One Drive	YES	NO
0 Wait State Operation	YES	NO
PRICE	**$169.95**	**$199.95**

DRIVE SPECIALS
IBM 520 MB 3.5" SCSI HD - $129.95
Maxtor 213 MB 3.5" SCSI HD - $119.95

$169.95

The WHT AT Keyboard Interface

The WHT AT Keyboard Interface & ROM Upgrade provides your TI 99/4a with a true AT keyboard interface to connect your favorite 101+ keyboard to the console. This unique solution installs inside your computer allowing you to use BOTH the AT and console keyboards simultaneously!

Head to Head with the Rave XT

Feature	AT	XT
Industry Standard Interface	YES	Obsolete
Easy to find inexpensive keyboards	YES	NO
Supports 84, 101 and newer 104 keyboards	YES	NO
Simultaneous use of console keyboard	YES	NO
16 character type ahead buffer	YES	NO
Supports auto switching keyboards	YES	NO
Extends system CRU range from >600	YES	NO
User installable 64K 0-wait optional RAM	YES	NO
64K of upgraded system ROM	YES	NO

KEYBOARD SPECIALS
Mitsumi Win '95 104 - $14.95
Keytronics Win '95 104 - $25.95
Microsoft Natural 104 - $63.95

Installation not included. Install yourself or add $30 for installation fee.

$59.95

Digi-Port - The Sound Solution!

Plug this small device into your PIO port on your TI 99/4a or Myarc Geneve and now you can play digital sounds!! Ever want to have your Geneve "talk" to you on boot up? How about listening to your favorite music or sound effect clip? It can all be done with Digi-Port! Supports 99/4a and Geneve.

Supports many memory expansion devices $19.95

Interface Standards & Design Guide

This unique book by P.E. Tony Lewis is the hardware hackers Bible. Packed with information on the inner workings of the TI 99/4a, Peripherals and the P.E. Box, it is the essential resource for the programmer and hardware enthusiast. Spiral bound or in a binder for easy reading.

The ultimate hackers guide! $24.95

Geneve Turbo Video

This unique "one chip wonder" replaces your original Geneve PAL with a new faster version! This chip reduces the time necessary for the CPU to access the video chip enhancing access times by up to 20%! Easy to install in socketed Geneves. (Not compatible with some older Geneves)

Installation not included.
call for installation pricing. $24.95

Repair Services

WHT offers complete repairs for all your TI, Myarc and Cor-Comp peripherals. If it's broken, we can probably fix it! **FREE** estimates! Call us **FIRST** for your repair needs! Every repair comes with a 90 day warrantee!

Certain products can no longer be repaired due to a lack of parts. Call about your specific needs. Starting at $20.00

NOW SHIPPING!!!!

WHT is pleased to announce the shipment of the WHT SCSI card DSR v1.1 for the TI 99/4a and Myarc Geneve! This new upgrade, available FREE to registered WHT SCSI card owners, fixes all known bugs with 99/4a operations! Simple to install yourself, just plug it in!
Call or write to order yours today!
(S&H $2, $5 deposit on return of old eprom)

Due to popular demand, a new lot of SCSI's is now in production! All SCSI cards currently on back-order will be shipped asap!! New cards in stock soon!
Order yours today!

PC Products

WHT offers a complete line PC's and products for your home and office computing needs! Download our electronic catalog from our BBS or web site! Resellers welcomé, call for info.

Mini Tower Case - Perfect for a Geneve Conversion: $35.00
Medium Tower Case: $57.00
Full Tower Case: $71.00
4 MB, 72 PIN SIMM: $20
8 MB 72 PIN SIMM: $34
16 MB 72 PIN SIMM: $72
Pentium Motherboards starting at: $116.00
Pentium CPU's starting at: $41.00 (AMD)
CD-ROM's starting at: $72.00
PLUS MUCH MUCH MORE!! CALL NOW!

All prices are in U.S. Dollars and do not include applicable sales taxes or shipping charges. Please call or write us with your order before sending payment.

Call our BBS or visit our web site to download the latest DSR's & Manuals!

(408)-934-9682 FAX/BBS
http://www.sonyx.com/wht

We Honor:

Place your order with Competition Computer!!
1-800-471-1600

350 Marcella Way
Millbrae, CA 94030

THE ART OF ASSEMBLY — PART 62

Book Wrong!

Or, how you can't always believe what you read about computers

By BRUCE HARRISON

This article is about sprite operations, but we're going to start with a little anecdote from your author's undergraduate days at Penn State. How long ago was that? Here's a hint: Joe Paterno was an assistant coach for the football team, under Head Coach Rip Engle (early 1960s). My college roommate was a young man named Frank Ruhman. Frank was studying mechanical engineering, and always placed great faith in the accuracy of his textbooks. One time Frank's faith in textbooks was sorely tested when he got back a mid-term exam in which one of his solutions was marked wrong. He checked his solution carefully, and found that his method exactly matched what the textbook said, and that his math was all done correctly. He went to the instructor, a young man from India on some kind of teaching fellowship, and pointed out that his solution on the mid-term exactly matched the book. To that, the instructor gave a two-word reply: "Book wrong!"

HOW THE BOOK IS WRONG

There have been a few times before in this column where we've said that about the TI Editor/Assembler manual, and here's another such case. The book says that if you place the value >D0 in the Y-position byte for any sprite, that will delete this sprite and all higher numbered sprites in use. To quote from Porgy and Bess, that "Ain't Necessarily So."

If all your sprites are stationary, it's so. If, however, a higher numbered sprite is in motion, that sprite will remain visible and in motion after deleting the lower numbered one. If you really want all higher numbered ones to disappear, you'll also have to change the byte at >837A to a number equal to or lower than the sprite you've deleted. For example, let's say that you had three sprites in motion, numbered 0, 1, and 2. If you do the >D0 delete to number 1 without changing the value at >837A, you'll see that sprites 0 and 2 will stay on-screen and continue their motion.

If sprites are magnified, even the deleted one may still be visible in part at the top or bottom of the screen unless you've changed its color to transparent. For the benefit of our readers, we've put some source code in this issue's sidebar that will do all things necessary to really eliminate that sprite and all that follow it in the number sequence. The routine is called DELSPR, and gets used via a BLWP operation.

SPRITE SUBROUTINES

The sidebar is a whole series of subroutines for dealing with sprites. It includes one to "turn on" sprites, to delete them, and to change their patterns or motions, etc. We tried to include all of the CALLs from Extended BASIC that work for sprites. These are all designed as BLWP vectors, so that, with just one exception they won't affect your own registers. That one exception is the POSIT routine, which affects the calling program's Register 1. On return from POSIT, your R1 will contain the Y and X positions of the sprite, in its left and right bytes, respectively. Here's the list of routines that you can BLWP to:

SPRITE	puts a sprite on-screen
MOTION	changes the y and x velocities
PATTRN	changes the character pattern
LOCATE	changes the y and x positions
POSIT	reports the y and x positions in R1
COLOR	changes the color of a sprite
DELSPR	deletes a sprite and all following
MAGNIF	magnifies all sprites
COINC	determines coincidence
REVMO	reverses motion
REV1	reverses a single character left-right
REV4	reverses four characters left-right

The last three of these are not included in the sprite services of Extended BASIC. REVMO simply reverses the motion of a sprite in both vertical and horizontal axes. If a motion byte is 0, that is unaffected by REVMO. REV1 reverses a single character definition from left to right. REV4 does that for a four-character sprite definition.

All of these are set up for the "normal" sprite operations in graphics mode. Most of them will also work for the bit-map cases, provided only that the ATTLST and MOTBL equates are set correctly for those table locations in your own program's code, but DELSPR will need to be modified after the line that says JNE DELLOP. The code from there to DELEX should be replaced with the following:

```
MOV @SPMOT,R1
JEQ DELEX
C R1,R4
JLT DELEX
MOV R4,@SPMOT
DELEX RTWP
```

That will perform the same function for you as the code starting with MOVB @>837A,R1. This will make the deletion of sprites work just the way the book says, in spite of the book being wrong about how to delete sprites. Sprites should therefore be added in numerical order and deleted in inverse numerical order. To delete all sprites, just perform a BLWP @DELSPR with DATA 0.

Each of these BLWP routines requires at least one DATA line following it. In most cases, that data item must be the number of a sprite. Sprite numbers in assembly run from 0 through 31. The exception is the case of MAGNIF. In that case the DATA following the BLWP instruction must be a number from 0 through 3, and this affects all sprites in use. The meaning of 0 through 3 is exactly like the meaning of the numbers 1 through 4 used in CALL MAGNIFY in Extended Basic. (The pictures in the XB manual are incorrect. "Book Wrong" there, too.)

0	means single character, normal size
1	means single character, double size
2	means four-character, normal size

(See Page 10)

SAN JOSE
CAPITAL OF SILICON VALLEY

Fest West '97

A Texas Instruments 99/4a & Myarc Geneve 9640 Computer Users Convention

Admission $5

Saturday, April 5 - 9am to 5pm
San Jose Civic Auditorium,
Downtown San Jose

Fest West '97 is the BEST IN THE WEST!!! In past years, the Fest has been held in Salt Lake City, Phoenix, Anaheim, Tucson, San Diego, Las Vegas, and Los Angeles. For the VERY FIRST TIME, the Fest will be held in the HEART OF SILICON VALLEY - San Jose, California.

This exciting event is centrally located in the heart of Downtown San Jose. Many attractions are within just a few blocks walking distance, such as : The Tech Museum of Innovation, The Children's Discovery Museum, The San Jose Museum of Art, American Musical Theater of San Jose, San Jose Rep. Theater, American Museum of Quilts & Textiles, and many more!

Hotels in the downtown area include

Best Western Inn	455 S. Second Street	(408)-298-3500	Nice facilities, low rates
Townhouse Motel	475 S. Second Street	(408)-295-5558	Lower rates than Best Western
Fairmont Hotel	170 S. Market St.	(408)-998-1900	Luxury Hotel, expensive, but nice!
Hilton Hotel	300 Almaden Blvd.	(408)-287-2100	Adjacent to the convention center
Holiday Inn	282 Almaden Blvd.	(408)-998-0400	Adjacent to the Civic Auditorium

Transportation to and from the airport is available via local hotel shuttle buses, taxi, bus and airport super-shuttle. Transportation to and from Amtrak at the Tasman station is available via local light rail. The Light rail system can take you nearly anywhere within downtown San Jose, and can be taken to the Cal-train station to make a trip to San Francisco.

Adult admission is $5 for all day, children under 16 are free with a paid adult.

RAFFLE ***RAFFLE*** ***RAFFLE*** ***RAFFLE*** ***RAFFLE*** ***RAFFLE***

Hourly door prizes and a big raffle at the end of the day!! Raffle tickets are only $1, buy 10 and get in FREE! Raffle prizes include (more to come!!): WHT SCSI Controller, WHT AT Keyboard interface, Super AMS, Horizon RAM Disk, TI CD-ROM, Software and MUCH MORE!!!

For a Fest West '97 Vacation Package, please write or call:
Fest West '97, c/o South Bay TI Users Group, 3297 Woody Lane, San Jose, CA 95132, 408-934-0352

GET THE LATEST FROM THE WEB!!! http://www.sonyx.com/wht/ti/fest97.html

San Jose Downtown Association
http://www.sj-downtown.com

THE ART OF ASSEMBLY —

(Continued from Page 8)

3 means four-character, double size.

In the four-character cases, the sprite will consist of four character definitions, where the first one will be a character number that's evenly divisible by four. That first pattern will form the upper left part of the sprite. the second will form the lower left quarter, the third the upper right quarter, and the fourth the lower right quarter. (This is exactly the same as in the Extended BASIC four-character cases.)

If the character value given in the setup of the sprite is not evenly divisible by four, then the upper left quarter will be the next lower character that is evenly divisible by four. For example, if you've called for the sprite character to be A, B, or C, you'll get exactly the same result, with the @ symbol in the upper left quarter, the A in the lower left, B in the upper right, and C in the lower right. That is because the @ symbol's ASCII code of 64 (>40) is the next lower number evenly divisible by four below the ASCII values 65, 66, and 67 for A, B, or C.

Two other exceptions to the use of a sprite number as the DATA word are the character reversal routines REV1 and REV4. These each need one data words, that being the ASCII value of the character to be reversed. Thus to reverse the four-character definition starting at character 128, you would do this:

```
BLWP @REV4
DATA 128
```

The subroutine will work just like the sprite itself, in that it will ensure that the character number is divisible by four. If the DATA above is 129, 130, or 131, the subroutine will go to the address for character 128.

The subroutine has been set up on the assumption that the sprite pattern table has been set to coincide with the normal character table for E/A operations. We do that in many programs by LI R0,>601 and then BLWP @VWTR. If your sprite pattern table is not set up that way, you'll need to change that AI R0,>800 just after label REV in the sidebar.

In both cases, the Assembler will calculate the address in VDP RAM of the character pattern's address, and will put the appropriate hex value in the object file.

USING THE SIDEBAR

The sidebar has been set up so it can be used in several ways. You can simply insert this into your own program as is, taking care that no labels are duplicated. If you like, you can do it "module" fashion by assembling the sidebar, then inserting REFs in your own code so the routines will be treated as external to your own program. After assembling your own code, you'd load your own object file, then load the object file created by assembling the sidebar. That way, the linking loader will resolve all references for you, and the program in memory will have both your program and these subroutines. The third way to use this is to just put a copy directive into your own source file, as in COPY "DSK1.SIDEBAR62." Of course, then you have to make sure that the sidebar is in the disk drive called for in the copy directive when you're assembling your program.

Each of these routines in the sidebar is preceded by a "commented out" section that explains how to invoke that routine from the main code. Except for POSIT, none of these will affect any your own workspace registers. COINC will affect the status register, so that on return, you can use a JEQ for no coincidence or a JNE for coincidence. That would look something like this:

```
BLWP @COINC
DATA 1,3
JNE C13
(code to handle non-coincidence case)
...
C13 (code to handle coincidence case)
...
```

That's about it. We hope this makes it easier to handle all those sprite operations. Our next topic is undecided, but we hope you'll tune in for whatever it is.

SIDEBAR62

```
0001 * SIDEBAR 62
0002 *
0003 * HANDY SUBROUTINES FOR SPRITE
0004 * OPERATIONS IN ASSEMBLY
0005 * USING THE NORMAL GRAPHICS MODE
0006 * CODE BY Bruce Harrison
0007 * PUBLIC DOMAIN
0008 * 14 MAY 1995
0009 *
0010 * FOLLOWING ARE THE DEFAULT EQUATES
0011 * FOR GRAPHICS MODE - CHANGE THEM FOR
0012 * USE WITH THE BIT-MAP MODES
0013 *
0014 ATTLST EQU  >300            SPRITE ATTRIBUTE TABLE
0015 MOTBL  EQU  >780            SPRITE MOTION TABLE
0016 *
0017 * INSURE THE FOLLOWING REFS ARE IN YOUR PROGRAM
0018 * UNLESS YOU USE THE "OBJECT MODULE" APPROACH
0019 *
0020        REF  VSBW,VSBR,VWTR,VMBR
0021 *
0022 * THE FOLLOWING CAN BE USED AS REFS IN YOUR
0023 * PROGRAM IF YOU USE THIS AS A SEPARATE MODULE
0024 *
0025        DEF  SPRITE,MOTION,POSIT,LOCATE,PATTRN
0026        DEF  COLOR,MAGNIF,REVMO,COINC,DELSPR
0027        DEF  REV1,REV4
0028 *
0029 * SUBROUTINES FOR SPRITES
0030 * EXCEPT FOR POSIT, NONE OF THESE
0031 * WILL CHANGE ANYTHING IN THE CALLER'S
0032 * WORKSPACE REGISTERS
0033 *
0034 * INVOKE SPRITE BY:
0035 *      BLWP @SPRITE
0036 *      DATA SRPITE #       (0 THRU 31)
0037 *      BYTE YPOS,XPOS      Y POSITION, X POSITION
0038 *      BYTE CHAR,COLOR     CHARACTER VALUE, COLOR
0039 *      BYTE YVEL,XVEL      Y VELOCITY, X VELOCITY
0040 *
0041 SPRITE DATA SPWS,SPRCOD
0042 SPRCOD LI   R0,ATTLST      POINT AT ATTR. TABLE
0043        MOV  *R14+,R3       GET SPRITE #
0044        SLA  R3,2           MULT. BY 4
0045        A    R3,R0          ADD TO ADDRESS
```

(See Page 11)

THE ART OF ASSEMBLY —

(Continued from Page 10)

```
0046         LI   R2,4             FOUR ATTR BYTES
0047 SPLP    MOVB *R14+,R1         GET ATTRIBUTE
0048         BLWP @VSBW            WRITE TO TABLE
0049         INC  R0               NEXT SPOT
0050         DEC  R2               DEC COUNT
0051         JNE  SPLP             RPT IF NOT ZERO
0052         LI   R1,>D000         "DELETE" NEXT SPRITE
0053         BLWP @VSBW            WRITE THAT
0054         LI   R0,MOTBL         MOTION TABLE
0055         A    R3,R0            ADD OFFSET
0056         MOVB *R14+,R1         GET Y VELOCITY
0057         MOV  R1,R4            STASH IN R4
0058         BLWP @VSBW            WRITE THAT
0059         INC  R0               NEXT SPOT
0060         MOVB *R14+,R1         GET X VELOCITY
0061         MOV  R1,R5            STASH IN R5
0062         BLWP @VSBW            WRITE THAT
0063 CLR1    CLR  R1               ZERO R1
0064         INC  R0               IST AUX BYTE
0065         BLWP @VSBW            WRITE A ZERO
0066         INC  R0               2ND AUX BYTE
0067         BLWP @VSBW            ZERO THERE TOO
0068         MOV  R4,R4            CHECK FOR R4=0
0069         JNE  SETMO            IF <>0, JUMP
0070         MOV  R5,R5            CHECK FOR R5=0
0071         JEQ  SPREX            IF 0, EXIT
0072 SETMO   SRL  R3,2             DIVIDE R3 BY 4
0073         INC  R3               ADD ONE
0074         SWPB R3               SWAP R3
0075         CB   R3,@>837A        CHECK AGAINST SPRITES IN MO-
TION
0076         JLT  SPREX            IF LESS, EXIT
0077         MOVB R3,@>837A        ELSE SET THIS NUMBER IN MOTION
0078 SPREX   RTWP RETURN
0079 *
0080 * INVOKE MOTION BY:
0081 *       BLWP @MOTION
0082 *       DATA SPRITE #
0083 *       BYTE YVEL,XVEL        Y VELOCITY,X VELOCITY
0084 *
0085
0086 MOTION  DATA SPWS,MOTCOD
0087
0088 MOTCOD  LI   R0,MOTBL         POINT AT MOTION TABLE
0089         MOV  *R14+,R3         GET SPRITE #
0090         SLA  R3,2             MULTIPLY BY 4
0091         A    R3,R0            ADD OFFSET
0092         CLR  R4               CLEAR R4
0093         CLR  R5               AND R5
0094         MOV  *R14+,R1         GET DESIRED VELOCITIES
0095         MOVB R1,R4            Y VEL TO R4
0096         BLWP @VSBW            WRITE Y VEL
0097         SWPB R1               SWAP
0098         INC  R0               NEXT ADDR
0099         MOVB R1,R5            XVEL TO R5
0100         BLWP @VSBW            WRITE X VEL
0101         JMP  CLR1             JUMP TO END OF SPRITE
0102 *
0103 * INVOKE POSIT BY:
0104 *       BLWP @POSIT
0105 *       DATA SPRITE #
0106 *
0107 * ON RETURN, YOUR R1 WILL CONTAIN
0108 * Y POSITION IN LEFT BYTE
0109 * X POSITION IN RIGHT BYTE
0110 *
0111 POSIT   DATA SPWS,POSCOD
0112
0113 POSCOD  LI   R0,ATTLST       ATTRIB TABLE
0114         MOV  *R14+,R3         GET SPRITE #
0115         SLA  R3,2             MULT. BY 4
0116         A    R3,R0            ADD TO R0
0117         BLWP @VSBR            READ Y POS
0118         SWPB R1               SWAP
0119         INC  R0               NEXT ADDR
0120         BLWP @VSBR            READ X POS
0121         SWPB R1               SWAP
0122         MOV  R1,@2(R13)       PUT R1 INTO CALLER'S R1
0123         RTWP RETURN
0124 *
0125 * INVOKE LOCATE BY:
0126 *       BLWP @LOCATE
0127 *       DATA SPRITE #
0128 *       BYTE YPOS,XPOS        Y POSITION, X POSITION
0129 *
0130 LOCATE  DATA SPWS,LOCCOD
0131
0132 LOCCOD  LI   R0,ATTLST       ATTRIB TABLE
0133         MOV  *R14+,R3         GET SPRITE NUM
0134         SLA  R3,2             MULT. BY 4
0135         A    R3,R0            ADD OFFSET
0136         MOV  *R14+,R1         GET POSITIONS
0137         BLWP @VSBW            WRITE Y POS
0138         SWPB R1               SWAP
0139         INC  R0               NEXT ADDR
0140         BLWP @VSBW            WRITE X POS
0141         LI   R0,MOTBL+2       2 PAST MOTION TABLE
0142         A    R3,R0            ADD OFFSET
0143         CLR  R1               0 IN R1
0144         BLWP @VSBW            WRITE AUX BYTE 1
0145         INC  R0               NEXT ADDR
0146         BLWP @VSBW            WRITE AUX BYTE 2
0147         RTWP RETURN
0148 *
0149 * INVOKE PATTRN BY:
0150 *       BLWP @PATTRN
0151 *       DATA SPRITE #
0152 *       DATA CHAR             CHARACTER ASCII
0153 *
0154 PATTRN  DATA SPWS,PATCOD
0155
0156 PATCOD  LI   R0,ATTLST+2     OFFSET TO CHAR
0157         MOV  *R14+,R3         GET SPRITE NUM
0158         SLA  R3,2             MULT BY 4
0159         A    R3,R0            ADD TO ADDR
0160         MOV  *R14+,R1         DESIRED CHAR
0161         SWPB R1               SWAP
0162         BLWP @VSBW            WRITE
0163         RTWP RETURN
0164 *
0165 * INVOKE COLOR BY:
0166 *       BLWP @COLOR
0167 *       DATA SPRITE #         (0-31)
0168 *       DATA COLOR            (0-15)
0169 *
0170 COLOR   DATA SPWS,COLCOD
0171
0172 COLCOD  LI   R0,ATTLST+3     OFFSET TO COLOR
0173         MOV  *R14+,R3         GET SPRITE #
0174         SLA  R3,2             MULT. BY 4
```

(See Page 12)

THE ART OF ASSEMBLY —

(Continued from Page 11)

```
0175          A    R3,R0          ADD TO ADDR
0176          MOV  *R14+,R1       GET DESIRED COLOR
0177          ANDI R1,>000F       INSURE RANGE 0->F
0178          SWPB R1             SWAP
0179          BLWP @VSBW          WRITE
0180          RTWP RETURN
0181 *
0182 * INVOKE MAGNIF BY:
0183 *       BLWP @MAGNIF
0184 *       DATA MAGLEV         (0 THRU 3)
0185 *
0186 * WHERE:
0187 * 0 MEANS UNMAGNIFIED
0188 * 1 MEANS DOUBLE SIZE, SINGLE CHARACTER
0189 * 2 MEANS NORMAL SIZE, FOUR CHARACTERS
0190 * 3 MEANS DOUBLE SIZE, FOUR CHARACTERS
0191 *
0192 MAGNIF DATA SPWS,MAGCOD
0193
0194 MAGCOD MOV  *R14+,R0        DESIRED MAG
0195          ANDI R0,0003        INSURE 0-3
0196          ORI  R0,>01E0       ADD >1E0
0197          BLWP @VWTR          WRITE TO VDP REG 1
0198          SWPB R0             SWAP
0199          MOVB R0,@>83D4      PUT BYTE AT >83D4
0200          RTWP RETURN
0201 *
0202 * REVMO REVERSES THE MOTION OF SPRITE
0203 * BOTH HORIZONTAL AND VERTICAL
0204 * MOTIONS WILL REVERSE
0205 *
0206 * INVOKE REVMO BY
0207 *       BLWP @REVMO
0208 *       DATA SPRITE #
0209 *
0210 REVMO  DATA SPWS,REVMC
0211
0212 REVMC  LI   R0,MOTBL        MOTION TABLE
0213          MOV  *R14+,R3       GET SPRITE #
0214          SLA  R3,2           MULT BY 4
0215          A    R3,R0          ADD OFFSET
0216          BLWP @VSBR          READ Y MOTION
0217          SRA  R1,8           RIGHT JUST
0218          NEG  R1             MUL BY -1
0219          SWPB R1             SWAP
0220          BLWP @VSBW          WRITE
0221          INC  R0             NEXT ADDR
0222          BLWP @VSBR          READ X MOTION
0223          SRA  R1,8           RIGHT JUST
0224          NEG  R1             MUL BY -1
0225          SWPB R1             SWAP
0226          BLWP @VSBW          WRITE
0227          INC  R0             NEXT ADDR
0228          CLR  R1             0 IN R1
0229          BLWP @VSBW          WRITE AUX BYTE 1
0230          INC  R0             NEXT ADDR
0231          BLWP @VSBW          WRITE AUX BYTE 2
0232          RTWP RETURN
0233 *
0234 * INVOKE COINC BY:
0235 *       BLWP @COINC
0236 *       DATA SPR1,SPR2      SPRITE NUMBERS
0237 *
0238 * ON RETURN, STATUS WILL BE EQ IF NO COINC
0239 * OR NE IF COINC FOUND
0240 *
0241 COINC  DATA SPWS,COINCD
0242
0243 COINCD CLR  R8              CLEAR A REGISTER
0244          MOV  *R14+,R3       GET 1ST SPRITE #
0245          BL   @GP0           GET SPRITE POS
0246          MOV  R4,R6          Y POS TO R6
0247          MOV  R5,R7          X POS TO R7
0248          MOV  *R14+,R3       GET 2ND SPRITE #
0249          BL   @GP0           GET POSITION
0250          AI   R4,10          ADD TOLERANCE
0251          AI   R5,10          TO BOTH
0252          C    R6,R4          COMAPRE Y POSITIONS
0253          JGT  COINX          IF GREATER, EXIT
0254 TST5     C    R7,R5          COMAPRE X POSITIONS
0255          JGT  COINX          IF GREATER, EXIT
0256 TST42    AI   R4,-20         SUBTRACT 2X TOLERANCE
0257          AI   R5,-20         FROM BOTH
0258          C    R6,R4          COMPARE Y POSITIONS
0259          JLT  COINX          IF LESS, EXIT
0260          C    R7,R5          COMPARE X POSITIONS
0261          JLT  COINX          IF LESS, EXIT
0262          INC  R8             ELSE INC R8
0263 COINX    MOV  R8,R8          CHECK R8 FOR ZERO
0264          STST R15            STATUS REG TO R15
0265          RTWP THEN RETURN
0266
0267 GP0      SLA  R3,2           MULT. BY 4
0268          LI   R0,ATTLST      LOAD ATTLST ADDR
0269          A    R3,R0          ADD OFFSET
0270          BLWP @VSBR          READ Y POS.
0271          MOVB R1,R4          MOVE TO R4
0272          SRL  R4,8           RT. JUST.
0273          INC  R0             NEXT ADDR
0274          BLWP @VSBR          READ X POS.
0275          MOVB R1,R5          PUT IN R5
0276          SRL  R5,8           RT. JUST.
0277          RT   RETURN
0278 *
0279 * INVOKE DELSPR BY:
0280 *       BLWP @DELSPR
0281 *       DATA SPRITE #
0282 *
0283 * NOTE:
0284 *       DELSPR WILL DELETE THE
0285 *       SPRITE SPECIFIED AND
0286 *       ALL HIGHER NUMBERED ONES
0287 *
0288
0289 DELSPR DATA SPWS,DELCOD
0290
0291 DELCOD LI   R0,ATTLST       ATTRIB TABLE
0292          MOV  *R14+,R3       GET SPRITE #
0293          MOV  R3,R4          STASH IN R4
0294          SLA  R3,2           MUL BY 4
0295          A    R3,R0          ADD OFFSET
0296          LI   R1,>D000       "DEL" CODE
0297          BLWP @VSBW          WRITE THAT
0298          CLR  R1             ZERO IN R1
0299          AI   R0,3           POINT AT COLOR BYTE
0300          BLWP @VSBW          TRANSPARENT COLOR
0301          LI   R0,MOTBL       MOTION TABLE
0302          A    R3,R0          ADD OFFSET
0303          LI   R5,4           FOUR BYTES
0304 DELLOP   BLWP @VSBW          WRITE ONE 0
```

(See Page 13)

THE ART OF ASSEMBLY —

(Continued from Page 12)

```
0305          INC  R0              NEXT ADDR
0306          DEC  R5              DEC COUNT
0307          JNE  DELLOP          RPT IF NOT 0
0308          MOVB @>837A,R1       GET SPRITES IN MOTION
0309          JEQ  DELEX           EXIT IF 0
0310          SWPB R4              SWAP R4 (SPRITE #)
0311          CB   R1,R4           COMPARE
0312          JLT  DELEX           IF R1<R4, EXIT
0313          MOVB R4,@>837A       ELSE NEW NUM TO >837A
0314 DELEX    RTWP RETURN
0315 *
0316 * REV1 WILL REVERSE THE PATTERN OF A CHARACTER
0317 * REV4 WILL REVERSE THE PATTERNS OF A 4-CHARACTER SET
0318 *
0319 * INVOKE REV1 OR REV4 BY:
0320 *       BLWP @REV1           (OR REV4)
0321 *       DATA CHAR            CHARACTER ASCII VALUE
0322 *
0323 REV1    DATA SPWS,REV1CD
0324 REV4    DATA SPWS,REV4CD
0325
0326 REV1CD  LI   R2,8            EIGHT BYTES
0327          MOV  *R14+,R0        GET CHARACTER NUMBER
0328          JMP  REV             THEN JUMP AHEAD
0329 REV4CD  LI   R2,32           32 BYTES
0330          MOV  *R14+,R0        CHARACTER ASCII
0331          ANDI R0,>00FC        INSURE DIVISIBLE BY 4
0332 REV      SLA  R0,3            MULTIPLY BY 8
0333          AI   R0,>800         ADD OFFSET TO CHAR TBL
0334 ******************
0335 *
0336 * NOTE: THE >800 ABOVE ASSUMES YOU'VE SET THE
0337 * SPRITE PATTERN TABLE TO THAT VALUE.
0338 * IF YOU'VE LEFT THE DEFAULT 0 FOR VDP REG 6,
0339 * THEN OMIT THAT LINE. IF YOU'VE PUT THE SPRITE
0340 * PATTERN TABLE SOMEWHERE ELSE BY WRITING TO
0341 * VDP REG 6, THEN PUT THE APPROPRIATE VALUE
0342 * IN PLACE OF >800.
0343 *
0344 ******************
0345          LI   R1,CPBUFF       POINT AT BUFFER
0346          BLWP @VMBR           READ THE BYTES
0347          CI   R2,32           COMPARE R2 TO 32
0348          JLT  MOV19           JUMP IF LESS
0349          AI   R0,16           POINT R0 AT THIRD CHAR DEF
0350 MOV19    MOV  R1,R9           POINT R9 AT BUFFER
0351          MOV  R2,R5           COPY R2 VALUE INTO R5
0352 LD4      LI   R4,8            EIGHT BITS PER BYTE
0353          CLR  R1              R1=0
0354          MOVB *R9+,R3         GET ONE BYTE FROM BUFFER AND
INC R9
0355          LIMI 2               ALLOW INTERRUPTS
0356          LIMI 0               STOP INTERRUPTS
0357 SHFT1    SRL  R1,1            SHIFT R1 TO RIGHT 1 BIT
0358          SLA  R3,1            SHIFT R3 TO LEFT ONE BIT
0359          JNC  DEC4            IF NO CARRY, JUMP AHEAD
0360          ORI  R1,>8000        ELSE SET MSB OF R1
0361 DEC4     DEC  R4              DEC BIT COUNT
0362          JNE  SHFT1           IF NOT ZERO, REPEAT
0363          BLWP @VSBW           WRITE ONE BYTE
0364          INC  R0              POINT AHEAD ONE
0365          DEC  R5              DEC BYTE COUNT
0366          JEQ  DONE            IF ZERO, EXIT SUBROUTINE
0367          CI   R5,16           COMPARE R5 TO 16
0368          JNE  LD4             IF NOT EQUAL, JUMP BACK
0369          S    R2,R0           IF R5=16, SUBTRACT R2 FROM R0
0370          JMP  LD4             THEN JUMP BACK
0371 DONE     RTWP RETURN
0372 *
0373 * DATA SECTION
0374 *
0375 SPWS     BSS  32              SPRITE WORKSPACE
0376 CPBUFF   BSS  32              CHAR PATTERN BUFFER
0377 *
0378 * TO ASSEMBLE THIS AS AN OBJECT MODULE,
0379 * DELETE THE "*" FROM THE LINE BELOW
0380 *       END
```

Fest West plans readied as April date nears

Organizers for Fest West '97 have sent out vendor reservation forms for the event, to be held from 9 a.m. to 5 p.m. April 5 at the San Jose Civic Auditorium, in San Jose, California. This is the first time for the event to be held in that city. Host for the event is the South Bay TI Users Group.

Demonstrations will take place in the community room of the San Jose Public Library, adjacent to the community center.

Vendors need to make prepayment by March 5 to ensure table and speaking reservations. Credit card payments can be made at (408) 934-0352. Vendors, dealers and users groups may not share tables, the organizers say.

The 72x32-inch tables cost $25 for the first, $20 for each additional table. Each vendor table setup includes two admissions to Fest West.

General admission is $5 with children under 16 free when accompanied by an adult.

Vendors need to provide their own 10-amp, UL-approved grounded extension cords and outlet strips. No public performance of music is permitted.

Organizers ask that vendors who need to borrow equipment or can lend equipment for the event to indicate that on their reservation forms.

Setup for Fest West will be from 4 p.m. to 8 p.m. April 4, with takedown time from 5 p.m. to midnight April 5.

The vendor packet includes information on hotels, restaurants, transportation and other activities near the event.

For further information, contact Fest West '97, 3297 Woody Lane, San Jose, CA 95132; call Don O'Neil at (408) 934-0352; or visit the Web site at http:www.sonyx.com/wht/tifest97.html.

An XBASIC program that writes assembly

By JIM PETERSON

The Tigercub has dipped a cautious paw into the cold, dark, mysterious waters of assembly, while still keeping a firm grip on trusty, old Extended BASIC. The result in an Extended BASIC program that writes an assembly program!

The following subprogram, when merged into any program that has reidentified characters, and called after the characters have been reidentified, will write a source code which can be assembled into object code, loaded from Extended BASIC and linked to instantly access the character set. Save the program in merge format with SAVE DSK1.CHARSUB,MERGE.

The source code is based on 2FONTS/S by Barry Traver, who gives credit to Mac McCormick, David Migicovsky, and Karl Schuneman.

The subprogram, when merged into any program that has reidentified characters, and called after the characters have been reidentified, will write a source code which can be assembled into object code, loaded from Extended BASIC and linked to instantly access the character set.

CHARSUB

```
19000 SUB CHARSUB(HX$())!022
19001 DISPLAY AT(12,1)ERASE
ALL:"Source code filename?":
"DSK" :: ACCEPT AT(13,4)SIZE
(12)BEEP:F$ :: OPEN #1:"DSK"
&F$,OUTPUT !170
19002 DISPLAY AT(15,1):"LINK
ABLE program name?" :: ACCEP
T AT(16,1)SIZE(6):P$ !169
19003 DISPLAY AT(18,1):"Rede
fine characters from ASCI
I     to ASCII" !180
19004 ACCEPT AT(19,7)VALIDAT
E(DIGIT)SIZE(3):F !139
19005 ACCEPT AT(19,21)VALIDA
TE(DIGIT)SIZE(3):T !198
19006 PRINT #1:TAB(8);"DEF";
TAB(13);P$ :: PRINT #1:"VMBW
EQU >2024" :: PRINT #1:"STA
TUS EQU >837C" !081
19007 NB=(T-F+1)*8 :: CALL D
EC_HEX(A,A$)!090
19008 FOR CH=F TO T :: IF CH
<144 THEN CALL CHARPAT(CH,CH
$)ELSE CH$=HX$(CH)!247
19009 IF FLAG=0 THEN PRINT #
1:"FONT";:: FLAG=1 !087
19010 FOR J=1 TO 13 STEP 4 :
: M$=M$&"."&SEG$(CH$,J,4)&",
" :: NEXT J :: M$=SEG$(M$,1,
23)&" *"&CHR$(CH)!154
19011 PRINT #1:TAB(8);"DATA"
&M$ :: M$="" :: NEXT CH !143
19012 PRINT #1:P$;TAB(8);"LI
RI,FONT" :: PRINT #1:TAB(8);
"LI R0,>"&A$ :: PRINT #1:TAB
(8);"LI R2,>"&H$ !226
19013 PRINT #1:TAB(8);"BLWP@
VMBW":TAB(8);"CLR @STATUS":T
AB(8);"RT":TAB(8);"END" :: C
LOSE #1 !128
19014 SUBEND !168
19015 SUB DEC_HEX(D,H$)!080
19016 X$="0123456789ABCDEF"
:: A=D+65536*(D>32767)!025
19017 H$=SEG$(X$,(INT(A/4096
)AND 15)+1,1)&SEG$(X$,(INT(A
/256)AND 15)+1,1)&SEG$(X$,(I
NT(A/16)AND 15)+1,1)&SEG$(X$
,(A AND 15)+1,1):: SUBEND !1
36
```

Now to try it out. You probably know that CALL CHARSET will restore reidentified characters below ASCII 96 to normal form, but not those above. So, let's write a routine to restore those. Clear memory with NEW, merge in the above subprogram (MERGE DSK1.CHARSUB). Add the following line:

100 CALL CHARSUB(HX$())

Answer the prompt for a filename with DSK1.OLDLOW/S. Enter OLDLOW to answer the next prompt. Select ASCII 97 to 127.

When done, insert the Editor/Assembler module and its disk Part A. Select Assembler, "Y" to load the assembler, and enter DSK1.OLDLOW/S as the source code, and DSK1.OLDLOW/O as the object code. Press Enter at the next prompt, enter "R" for options. You should get 0000 errors.

Now key in the following routine to test your program:

CHARTEST

```
100 CALL INIT :: CALL LOAD("
DSK1.OLDLOW/O"):: FOR CH=33
TO 126 :: CALL CHAR(CH,"FF81
8181818181FF"):: PRINT CHR$(
CH);:: NEXT CH !113
101 CALL KEY(0,K,S):: IF S=0
THEN 101 ELSE CALL CHARSET
!160
102 CALL KEY(0,K,S):: IF S=0
THEN 102 ELSE CALL LINK("OL
DLOW")!205
110 GOTO 110 !189
```

Press any key to restore the uppercase characters by CALL CHARSET, any key again to use the CALL LINK.

You are now ready to use the routine to copy all kinds of character sets from programs in your library. You don't have any such programs? Not to worry. You don't have to reidentify characters one by one with one of those graphics editor programs. You can just manipulate the existing hex codes of the normal characters. I

(See Page 15)

CHARACTERS —

(Continued from Page 14)

have created nearly 50 character sets by that method!

The space occupied by a character on the screen is really an 8x8 square of 64 tiny dots. Various dots are turned on (colored) and off (transparent) to create a pattern just the opposite of light bulbs on a scoreboard.

And those on-and-off dots are really the binary numbers which the computer uses. Fortunately the computer lets us use hexadecimal numbers rather than binary. The following program will print out a reference chart of decimal to binary to hexadecimal. You can easily convert it to dump to a printer.

CHARFORM

```
10 DISPLAY AT(6,1)ERASE ALL:
"DEC BIN HEX" !215
100 FOR J=0 TO 15 :: CALL DE
C_BIN(J,B$):: CALL DEC_HEX(J
,H$):: DISPLAY AT(J+8,1):J;T
AB(5);B$;TAB(10);SEG$(H$,4,1
): NEXT J !222
21020 SUB DEC_BIN(D@,B$):: D
=D@ :: IF D=0 THEN B$="0000"
:: SUBEXIT !015
21021 IF D=1 THEN 21022 :: X
=D/2 :: B@$=STR$(ABS(X<>INT(
X)))&B@$ :: D=INT(X):: IF D>
1 THEN 21021 !055
21022 B@$="1"&B@$ :: B$=RPT$
("0",4-LEN(B@$))&B@$ :: B@$=
"" :: SUBEND !054
21039 SUB DEC_HEX(D,H$)!080
21040 X$="0123456789ABCDEF"
:: A=D+65536*(D>32767)!025
21041 H$=SEG$(X$,(INT(A/4096
)AND 15)+1,1)&SEG$(X$,(INT(A
/256)AND 15)+1,1)&SEG$(X$,(I
NT(A/16)AND 15)+1,1)&SEG$(X$
,(A AND 15)+1,1):: SUBEND !1
36
```

And the following routine will show you how each letter is formed, by binary zeros (off) and ones (on), for each key you press. I save it in merge format so it can be merged into any program and CALLed to examine the characters.

CHARVIEW

```
17000 SUB CHARVIEW !202
17001 !programmed by Jim Pet
erson, Feb. 1986 !219
17002 DISPLAY AT(1,1)ERASE A
LL:"CHARACTERS IN BINARY & H
EX":;:"Press any key to see
the binary representation
of the screen character and
its hexcode" !056
17003 DISPLAY AT(8,1):"Press
Enter to see the character"
!002
17004 CALL KEY(0,K,S):: IF K
=13 THEN 17005 ELSE IF S=0 O
R K<32 OR K>143 THEN 17004 E
LSE 17007 !108
17005 CALL CHAR(48,"FF"&RPT$
("81",6)&RPT$("FF",9))!044
17006 CALL KEY(0,K,S):: IF S
<1 THEN 17006 ELSE CALL CHAR
(48,"003844444444380010301
010101038"):: GOTO 17004 !01
1
17007 CALL CHARPAT(K,CH$)!13
7
17008 R=12 :: FOR J=1 TO 15
STEP 2 !219
17009 H$=SEG$(CH$,J,1):: CAL
L HEX_BIN(H$,B$)!197
17010 DISPLAY AT(5,8):B$ !20
4
17011 H$=SEG$(CH$,J+1,1):: C
ALL HEX_BIN(H$,B$)!128
17012 DISPLAY AT(R,12):B$ ::
DISPLAY AT(R,18):SEG$(CH$,J
,2):: R=R+1 :: NEXT J :: DIS
PLAY AT(22,6):CH$ :: GOTO 17
004 !158
17013 SUBEND !168
17014 SUB HEX_BIN(H$,B$):: H
X$="0123456789ABCDEF" :: BN$
="0000X0001X0010X0011X0100X0
101X0110X0111X1000X1001X1010
X1011X1100X1101X1110X1111" !
078
17015 FOR J=LEN(H$)TO 1 STEP
-1 :: X$=SEG$(H$,J,1)!114
17016 X=POS(HX$,X$,1)-1 :: T
$=SEG$(BN$,X*5+1,4)&T$ :: NE
XT J :: B$=T$ :: T$="" :: SU
BEND !249
```

And to reidentify a character, you just change the numbers and letters in the 16-digit hex code which represents the binary pattern. By writing little routines to switch those digits around, all kinds of things can be done.

For instance, the normal characters always have the top row of dots turned off, to provide spacing between lines of text on the screen. If you want taller characters, you will have to double-space the lines. You can create them by making the numerals and uppercase characters consist of the second through seventh rows, the seven row again, and the eighth row it just happens to work out.

HIGHCHAR

```
18000 SUB HIGHCHAR :: FOR CH
=48 TO 90 :: CALL CHARPAT(CH
,CH$):: CALL CHAR(CH,SEG$(CH
$,3,10)&RPT$(SEG$(CH$,13,2),
2)&SEG$(CH$,15,2)):: NEXT CH
:: SUBEND !225
```

I made the following subprogram so that you can merge it in and use it to modify other character sets.

```
100 CALL CLEAR :: FOR CH=33
TO 90 :: CALL CHARPAT(CH,CH$
):: FOR J=1 TO 15 STEP 2 ::
CH2$=SEG$(CH$,J,2)&CH2$ :: N
EXT J :: CALL CHAR(CH,CH2$):
: CH2$="" :: NEXT CH !000
110 DISPLAY AT(12,1):"?NWOD
EDISPU":"VT EHT DENRUT OHW !
YEH" :: GOTO 110 !101
```

The following subroutine shades characters on their left edge by turning on the pixel to the left of the leftmost "on" pixel, if any. Also, try it in combination with HIGHCHAR.

NEWCHAR3

```
18001 SUB NEWCHAR3 :: FOR CH
=48 TO 122 :: CALL CHARPAT(C
H,CH$):: FOR J=1 TO 15 STEP
2 !051
18002 CH2$=CH2$&SEG$("0367CD
EF",POS("01234567",SEG$(CH$,
J,1),1),1)&SEG$(CH$,J+1,1)::
NEXT J :: CALL CHAR(CH,CH2$
):: CH2$="" :: NEXT CH :: SU
BEND !250
```

The next subroutine uses HIGHCHAR to heighten the character and then blanks out three rows. Try following it with NEWCHAR3.

(See Page 16)

CHARACTERS —

(Continued from Page 15)

NEWCHAR10

```
18030 SUB NEWCHAR10 :: A$="0
0" :: FOR CH=48 TO 90 :: CAL
L CHARPAT(CH,CH$):: CH$=SEG$
(CH$,3,10)&RPT$(SEG$(CH$,13,
2),2)&SEG$(CH$,15,2)!073
18031 CH$=SEG$(CH$,1,4)&A$&S
EG$(CH$,7,2)&A$&SEG$(CH$,11,
2)&A$&SEG$(CH$,15,2):: CALL
CHAR(CH,CH$):: NEXT CH :: SU
BEND !200
```

The following subroutine works only on ASCII characters 97-122. It makes tall characters ridiculously elongated.

NEWCHAR20

```
18050 SUB NEWCHAR20 :: FOR C
H=97 TO 122 :: CALL CHARPAT(
CH,CH$):: CALL CHAR(CH,SEG$(
CH$,7,2)&RPT$(SEG$(CH$,9,2),
4)&SEG$(CH$,11,6)):: NEXT CH
:: SUBEND !233
```

The next subroutine has the characters raised by one line, widened by one column at left and two columns at right to make a full 8x8 character which must be double-spaced horizontally and vertically.

NEWCHAR27

```
18090 SUB NEWCHAR27 :: FOR C
H=48 TO 122 :: CALL CHARPAT(
CH,CH$):: CH$=SEG$(CH$,3,10)
&RPT$(SEG$(CH$,13,2),2)&SEG$
(CH$,15,2):: FOR J=1 TO 15 S
TEP 2 !078
18091 CH2$=CH2$&SEG$("014589
CD",POS("01234567",SEG$(CH$,
J,1),1),1)&SEG$("0129",POS("
048C",SEG$(CH$,J+1,1),1),1)!
186
18092 NEXT J :: CALL CHAR(CH
,CH2$):: CH2$="" :: NEXT CH
:: SUBEND !042
```

Those who have my Nuts & Bolts disks will see how valuable this assembly can be to make instantly available the routines for double-height and double-width characters, etc. If you have Todd Kaplan's ALSAVE routine, you can embed them in your Extended BASIC program for fast loading.

These assembly loads are compatible with my BXB program, so you can also load character sets into sets 15 and 16, ASCII 144-159. However, the CHARPAT statement cannot access ASCII above 143. In this case, you must dimension an array in the program you are copying from as DIM HX$(159), and place the hex codes in the array using the ASCII as the subscript number, such as:

```
CALL CHAR(CH+64,CH$) :: HX$(
CH+64)=CH$
```

This ensures that they will be passed to the subprogram. And don't CALL INIT after you have called BXB.

So now you can try to create your own screen fonts.

BEGINNING C99 - PART 2

Comparing c99 with Extended BASIC

By VERN JENSEN

Last issue you learned how to type in a c99 program and compile it. This issue we'll take a look at the differences between c99 and Extended BASIC by comparing a c99 program and Extended BASIC program that do the same thing. You'll also learn an easier way to load a c99 program, as well as how to create E/A Option 5 program files.

One of the first things you should do is to print some of the documentation that comes on the c99 documentation disk. You should print all four of the C99-DOC files, which tell how to use the c99 compiler, and GRF1DOC, which describes each of the routines in the GRF1 library (a library for displaying graphics, and sprites).

Some other files you might also want to print out are CLOADDOC, RANDOMDOC, SNDSAYDOC, and SOUNDOC. Use FunnelWeb's formatter (option 2 on the TI-Writer menu) to print the documentation. In addition, the file "C99-README" tells what's new in version 5.0 of the c99 compiler. If you previously used version 4.0, you may want to take a look at this. And if you want to know what any of the other files on the documentation disk are for, simply take a look at the "-README" file. There are also "-README" files on the "c99 Libraries" and "c99 Programs" disks, so if you ever wonder what a particular file is for, just take a look at the "-README" file on that disk.

Something I should clarify is that the $5 I charged for the c99 disk set was to cover S&H only; it doesn't pay for the compiler itself. If you use the c99 compiler, you should send Clint Pulley the $40 shareware fee. This is a very low price when you consider how much work he must have put into this excellent program. Clint Pulley's address is listed in the first C99-DOC file.

To get started, type in the program at the end of this article called "TEST2;C", and save it onto Disk 2 (the one you've previously saved c99 programs onto). When you are done, press FCTN-9 to return to FunnelWeb's menu, and type Q. However, this time, instead of hitting E to exit, type P to purge. Then type in the file called "TEST2;L" (also listed at the end of this article), and save it to Disk 2. Then purge the contents of the editor again, and type in the file called "TEST2;P", and save it to Disk 2 as well. We will use the TEST2;L file to load the program later, and we'll use the TEST2;P file to create an E/A Option 5 program file.

COMPILER OPTIONS

Once you're finished typing in and saving those files, exit the editor and start up the c99 compiler, and compile the TEST2;C

(See Page 17)

BEGINNING C99 —

(Continued from Page 16)

program, using the same options as we did last installment. ("No" for the first two options, "yes" for the third.) The first option, "Include C Source?", if turned on, will include the C source code as comments in the assembly language output file. This can be useful for debugging, since if you get an assembly error for a particular line, you can load the assembly code and see which C line is responsible for the problem.

This option can also be used just for fun, to see how many assembly language lines are produced by a single line of C code. However, since this option makes the output file slightly larger, you should normally leave this option off. (Pressing enter will accept the default response of "no.")

The second option, "Inline Push Code?", will make the assembled program slightly faster, but the program will also be larger as a result. Since the size of the program is generally more important than a little bit of extra speed (at least for large programs), you'll usually want to leave this option off as well.

The third option, "Assume Long Jump?", is a new option that appeared in version 5.0 of the compiler. It was an option that used to be built into the c99 optimizer, but Winfried Winkler, who updated the c99 compiler to version 5.0, decided to put this option in the compiler itself, since it is much easier for the compiler to do it than for the optimizer to do it. You should respond to this option with a "no" only when you are finished your program and are ready to optimize it, since, although it can make your program smaller and faster, it also requires a bit more work on your part before the program can be run. This option is completely documented in the file "J2BDOC" on the documentation disk, so I won't go into it here.

THE C-LOADER

Now start up the Assembler and assemble the TEST2;C program. You should be able to simply hit enter four times and then press FCTN-6, since FunnelWeb fills in all the fields for you. (Pretty neat, huh?) Once it is done assembling, instead of hitting 3 once back at the FunnelWeb menu to get to the Loaders menu, hit 5 for C-LOADER.

The C-LOADER is a great program by Tom Bentley that greatly simplifies the process of loading a c99 program. Instead of having to remember all of the files that need to be loaded (such as CSUP, GRF1, and TEST2;O), we can simply type in the name of a file that contains the names of all the files that need loading, and C-LOADER will do all the work for us. So once the C-LOADER has loaded, type in "DSK2.TEST2;L" and press enter, and sit back and watch it go to work!

When it is done, press enter to exit C-LOADER, then hit FCTN-3 to erase the text, and press enter to get to the "Program Name" screen. Select "Start" and press FCTN-6, and the program should start. Use the arrow keys to move the sprite around the screen, and the period key to change the sprite's color. Since two different keyboard scan codes are used to scan for the arrow keys and the period key, you can change the color of the sprite while moving it around.

When you're done playing around with the program, hit "Q" to quit. You will then be asked if you want to rerun the program . Type "Y" if you do, or "N" if you don't. This prompt to rerun the program is something that is built into c99; it can be useful if you want to stop a program and start it again from the beginning without having to reload it, which can be great for testing purposes. There are ways to disable this feature, however, if you should so desire. This is covered in the c99 manual, and we'll probably cover it in a future article as well.

This prompt to rerun the program is something that is built into c99; it can be useful if you want to stop a program and start it again from the beginning without having to reload it, which can be great for testing purposes.

If, after quitting the c99 program, your computer locks up, it is because the c99 program was loaded "on top" of FunnelWeb in memory and, therefore, when the TI tried to return to the FunnelWeb program, it locked up. This won't happen if you load the c99 program from the Editor/Assembler cartridge, because the E/A program is kept on the cartridge, not in RAM, and therefore can't be overwritten. This isn't terribly important because most programs don't have a "Quit" option anyway; they simply force you to reset your computer when you want to stop the program.

CREATING AN OPTION 5 PROGRAM FILE

I did promise that we'd cover how to create an Option 5 program file, so let's go ahead an do that now. However, before it can be created, you need to copy three files from the c99 Libraries disk over to your "Work disk" (the one in drive 2). The files that need to be copied are C99PFF, C99PFI, and SAVE. In order to create an E/A program file, you must load all the files you normally load when you want to run a program, but must "sandwich" them between the C99PFF and C99PFI files, and then load SAVE after everything else.

Go back into FunnelWeb and start up the C-LOADER once more (option 5 on FunnelWeb's Editor/Assembler menu). This time, however, type in "DSK2.TEST2;P" and press enter. This not only loads the program and all of the libraries it needs, but it also loads the files needed in order to create an Option 5 program file. Once it is finished, press enter, then FCTN-3, and then enter again to get to the Program Name screen. But this time do not select "START." Instead, you want to find the program name SAVE, which most likely won't be on the first screen of names. To get to it, press enter a few times until you see SAVE, and then use FCTN-D (the right arrow) to select it, then hit FCTN-6. This will take you to the screen for saving an Option 5 program file. Simply type in the name you want the program file to be saved as (such as "DSK2.TEST2") and press enter. That's it! You can then run this

(See Page 18)

BEGINNING C99 —

(Continued from Page 17)

program as much as you like, by simply using the E/A Option 5 "Run Program File."

A LOOK AT THE SOURCE CODE

Probably one of the first things you noticed when you typed in your first c99 program is that there are no line numbers. If you're an assembly language programmer, this won't seem very strange, but to an XB programmer, this can be quite a shock. (Note that the line numbers displayed by the editor are for your reference only; they aren't saved as part of the file.)

You may wonder, "But how can I write a program without any GOTO statements?". Don't worry; c99 has many flow control statements that Extended BASIC doesn't, and they more than compensate for the lack of a GOTO statement. Actually, c99 does have a GOTO statement, but it's generally considered bad programming practice to use it, since it can make your code hard to read. And once you get used to C's flow control statements, you'll never want to go back to using GOTO again.

Just as Extended BASIC programs start execution at the lowest line number, c99 programs start execution in the main() function. In c99, everything happens inside "functions." These are very similar to the Extended BASIC's subroutines, except that in C, the entire program is contained in functions, with "main()" being the function that is called automatically when the program first starts. (For this reason all C programs must have a main() function.) Functions can also accept parameters, just like XB subroutines. We'll cover functions in a future c99 article.

Another thing you'll notice about C code is the use of open braces ("{") and close braces ("}"). These are used to group several statements together, or to define a function. For instance, following main(), there is an open brace, and then at the end of main() there is a close brace. Everything in between these two braces make up the main() function. Braces are not only used for functions, but for many other things as well, such as if statements. Here's an example of an if statement that doesn't use braces:

```
if (numLives < 1)
  dead = true;
else
  dead = false;
```

But what if we wanted execute more than one statement if numLives is smaller than 1? Then we could write the code like this:

```
if (numLives < 1)
{
  dead = true;
  score = score - 50;
  DoGameOver();
}
else
  dead = false;
```

By using braces, we were able to group three statements together, so that "dead" would be set to true, "score" would be decremented by 50, and the function DoGameOver() would be called if the variable numLives were smaller than 1. Just as FOR and NEXT statements in Extended BASIC can be nested, so can braces be nested. The only rule is that every open brace must be matched with a close brace.

Another thing that XB programmers will notice about C code is that each statement ends with a semicolon, such as the statement "x = 1;". This is necessary because the C compiler totally ignores all spaces, so you could put more than one statement on a single line if you want. Because of this, there must be some way for the compiler to know where the end of a statement is, and the semicolon is used for this. As an example, the code above could have been written like this:

```
if (numLives<1) { dead=true; score=score-50;
DoGameOver(); } else dead = false;
```

This is exactly the same code as far as the C compiler is concerned. However, this is obviously much harder to read than the previous example. Therefore, you are free to write your code in whatever style you prefer. However, whatever style you adopt, you should be consistent, and should choose a style that makes the code easy to read. As an example of another method, some people prefer to write their code like this, putting the open brace at the end of the first line:

```
if (numLives < 1) {
  dead = true;
}
```

However, I prefer the other approach, since it makes it easy to visually line up the open and close braces.

Another main difference between BASIC code and C code is that in C code you have to tell the C compiler what variables you are going to use before you use them. BASIC does this for you by scanning through your code before your program is run and making a list of all the variables. In today's example program, the following lines at the beginning of the main() function tell the C compiler what variables we will be using:

```
int keyCode, status, myColor;
int rowVel, colVel;
```

The word "int" at the beginning of those statements tells the compiler that those variables are integers. There are two types of variables provided by c99: char and int. The char variables can contain any value from 0 to 255, and integers can contain any value from -32768 to 32767. Char variables take up half as much memory as an int, although the size of a variable doesn't matter much when you only have a few variables, like today's program. It is important to realize that these variables can only contains whole numbers; trying to assign a char or int a value such as 1.5 won't work. (There are ways around this, however, such as using fixed-point numbers. We may cover this in a future article if there is interest.)

Variables must be declared either at the beginning of a function, or outside of a function. If they are declared inside a function, they are called "local" variables, because they are "local" to that function; they can't be "seen" by any other function.

However, if a variable is declared outside of a function, such as at the beginning of the program (before main()), it is called a "global" variable, because all of the functions can access it. As I mentioned last issue, the c99 compiler only recognizes the first six

(See Page 19)

BEGINNING C99 —

(Continued from Page 18)

letters of any name, be it a variable name, function name, or a definition name. You can make your variables longer than six letters in order to make them more readable, but you must remember that only the first six letters are used by the compiler. So if you make two variable called "myValue1" and "myValue2", you will run into problems, because the compiler will turn them both into "myValu".

Variable and function names can be any mix of letters and numbers, but they must start with a letter. In normal C, capitalization does matter, so "myguy" is different from "myGuy", but in c99, your code is compiled into assembly language, where everything is in upper-case, so when writing c99 code, don't rely on capitalization to keep your variable names, function names, and definitions apart, since in c99 "myguy" and "myGuy" are both compiled into "MYGUY".

I'll explain two other things today and then call it quits. The first is that in C, to compare two values to see if they are equal, you must use two equal signs, like this:

```
if (x == 1)
   DoMyStuff();
```

This is because if you used only one equal sign, the value would be assigned to x instead of being compared with it, even though this is being done inside an if statement. In C, "x = 1" always means "assign 1 to x", no matter what the context is.

The second thing I'll explain is that whenever you see a character sandwiched between two apostrophes, that character will actually be converted by the C compiler into its ASCII equivalent, which is very similar so Extended BASIC's ASC() function. As an example, 'A' would be converted by the C compiler to 65. The great thing about this is that this conversion is done during compilation, *before* the program is run, so using a statement such as "if (x == 'A')" is just as fast as using "if (x == 65)", but is often much easier to read and use, since you don't have to memorize the ASCII value for A. This is just one of the many nice little perks of the C language.

If there is any other part of the C code in today's example that you don't understand, just take a look at the section of the Extended BASIC code also at the end of this article that does the same thing. Today's example program was written in both languages in order to give you something familiar to compare with something that is unfamiliar. Also take a look at the GRF1DOC; it explains some of the stuff you'll see in the c99 code (such as Grf1() and SpMct(1)) that isn't in the Extended BASIC code. And finally, feel free to experiment; the c99 compiler won't bite! If you want to try something out, go ahead! Experimenting will do much help you learn.

IN CONCLUSION

Today I've covered many aspects of the C language, and hopefully removed some of the mystery from it. I will continue to compare C code with XB code for the next few articles in order to get you up and running with C as quickly as possible, but there is no way I can cover everything about the language in just a few articles. Therefore, I strongly suggest that anyone interested in learning C purchase the book *The C Programming Language* by Brian Kerninghan and Dennis Ritchie. The great thing about this book is that it is short and to the point; you won't have to read 500 pages just to figure out how to assign a value to a variable. It also makes a *great* reference book. There will be some things in the book that won't work with the c99 compiler, but much of what the book has to say does apply to c99. In any case, I've found the book to be an indispensable reference, and I highly recommend it.

CONTACTING ME

Have any questions about c99, suggestions for future articles, or comments about past articles? Or maybe you want to take a look at my game Virus Attack, which was written completely in c99? Don't hesitate to write. You can contact me via e-mail at Vern_Jensen@lamg.com, or via snail mail at Vern L. Jensen, 910 Linda Vista Ave., Pasadena, CA 91103.

TEST2;C

```
#include "DSK2.GRF1;H"

main()
{
   int keyCode, status, myColor;
   int rowVel, colVel;

   Grf1();
   Screen(8);
   myColor = 5;

   Display(10,7,"Use the arrow keys");
   Display(11,7,"to move the sprite.");
   Display(13,3,"Press '.' to change color.");
   Display(24,8,"-Type Q to Quit-");

   ChrDef(64,"FFFFFFFFFFFFFFFF");
   SpMag(2);
   Sprite(0,64,myColor,80,112);
   SpMct(1);

   do
   {
      keyCode = Key(2,&status);
      if (keyCode == 13)
      {
         myColor++;
         if (myColor == 8)
            myColor++;
         else if (myColor > 16)
            myColor = 2;

         SpColr(0,myColor);
      }

      keyCode = Key(0,&status);
      if (keyCode >= 'a' && keyCode <= 'z')
         keyCode = keyCode - 32;

      rowVel = 0;
      colVel = 0;
```

(See Page 20)

BEGINNING C99 —

(Continued from Page 19)

```
        if (keyCode == 'E')
            rowVel = -16;
        else if (keyCode == 'X')
            rowVel = 16;
        else if (keyCode == 'S')
            colVel = -16;
        else if (keyCode == 'D')
            colVel = 16;

        SpMotn(0,rowVel,colVel);
    } while (keyCode != 'Q');
}


Display(theRow, theCol, string)
char theRow, theCol, *string;
{
    Locate(theRow,theCol);
    PutS(string);
}
```

TEST2;L

```
DSK2.TEST2;O
DSK2.CSUP
DSK2.GRF1
```

TEST2;P

```
DSK2.C99PFI
DSK2.TEST2;O
DSK2.CSUP
DSK2.GRF1
DSK2.C99PFF
DSK2.SAVE
```

TEST2;XB

```
100 CALL CLEAR
110 CALL SCREEN(8)
120 MYCOLOR=5
130 DISPLAY AT(10,7):"Use th
e arrow keys"
140 DISPLAY AT(11,7):"to mov
e the sprite."
150 DISPLAY AT(13,3):"Press
'.' to change color."
160 DISPLAY AT(24,8):"-Type
Q to Quit-"
170 CALL CHAR(64,"FFFFFFFFFF
FFFFFF")
180 CALL MAGNIFY(2)
190 CALL SPRITE(#1,64,MYCOLO
R,80,112)
200 !
210 CALL KEY(2,KEYCODE,STATU
S)
220 IF KEYCODE=13 THEN 230 E
LSE 260
230 MYCOLOR=MYCOLOR+1
240 IF MYCOLOR=8 THEN MYCOLO
R=MYCOLOR+1 ELSE IF MYCOLOR>
16 THEN MYCOLOR=2
250 CALL COLOR(#1,MYCOLOR)
260 CALL KEY(0,KEYCODE,STATU
S)
270 IF KEYCODE>=ASC("a")AND
KEYCODE<=ASC("z") THEN KEYCO
DE=KEYCODE-32
280 ROWVEL=0 :: COLVEL=0
290 IF KEYCODE=ASC("E")THEN
ROWVEL=-16 ELSE IF KEYCODE=A
SC("X")THEN ROWVEL=16
300 IF KEYCODE=ASC("S")THEN
COLVEL=-16 ELSE IF KEYCODE=A
SC("D")THEN COLVEL=16
310 CALL MOTION(#1,ROWVEL,CO
LVEL)
320 IF KEYCODE<>ASC("Q")THEN
 210
```

Tips you can use to get more from your 9640

Keeping up with upgrades is important

By DON JONES

This article first appeared in Chicago Times, *the newsletter of the Chicago TI User Group.—Ed.*

It has been some time since I have last done a "Geneve Support Article," and since that time, some significant developments have taken place. I shall herewith attempt to deal with most of them:

M-DOS, UNLEASHED, UNCHAINED

The most important development is that of a virtually bug-free operating system. Ever since the release of M-DOS version 2.0, we have had an operating system that does essentially all of the things that are listed in the Myarc Geneve owner's manual. (I might also mention that we also have some additional command options available to us, which were not foreseen by the original developers of the 9640.) We now have an operating system that allows the Geneve to do the things which we were promised that it would be able to do. I therefore feel confident that any subsequent releases of M-DOS will be versions that provide more extended support for additional devices; a good example of that is version 4.0S. This version provides complete support for SCSI and PFM and PFM-Plus.

We now have an operating system that allows the Geneve to do the things which we were promised that it would be able to do.

UPDATE YOUR EPROMS

In order to take advantage of the current operating systems, you really need to make sure that you have the latest EPROM installed on your Genny board. If you are using the original one, which came out quite a few years ago, you will have significant problems trying to run the current operating system and various programs. (I believe that the latest version of this software is version 0.98.)

In addition to having the above-mentioned EPROM, you also need to be aware of a persistent hardware problem on the Genny: I am referring to the problem of heat build-up. The most frequent hardware problem with the Genny has related to the

(See Page 21)

GENEVE SUPPORT —

(Continued from Page 20)

overheating of voltage regulators. The ones which came with the original boards were not sufficient for the number of transistors and chips that they were supposed to support and protect. More efficient, low voltage versions are now available. In fact, these newer versions will allow any Myarc, Texas Instruments, CorComp or Rave card to be used either in a standard PEB, or in a PEB or tower case with a switching transformer. By using the low-voltage regulators, you don't have to worry about "smoking" a card that was modified to work in an AT or a tower case. They are available from Cecure Electronics Inc.

MODS TO GENNY BOARD MAKE FOR A SEXY MACHINE

On a Genny board, it is possible to install 32K of no-wait state RAM, by "piggy-backing" an additional chip onto your board. It is possible to have an additional 192K of VDP RAM on your board by piggy-backing a series of small chips onto your Genny board. I would recommend that you have these modifications made, though you can more or less get by without them. They are not terribly expensive, and they tend to make Genny into the sort of dream machine that we have always wanted her to be.

I consider all of the above hardware modifications to be essential for any serious Genny user, and they can be quickly and efficiently done by Don Walden of Cecure Electronics. The emergence of Cecure Electronics as the sole repair facility for the TI-99/4A and for all Myarc products was a welcome bit of news for our community. It means that you can get your Genny repaired in a few days rather that months or years, as it literally came to be for some of us when Myarc was still in business.

It should also be mentioned that Cecure is the source of a modification that I would not want to be without. I am speaking of PFM and PFM-PLUS. (PFM stands for Programmable Flash Memory.) I know that it isn't necessary or required to own this modification, but it sure makes Genny into a sexy little machine when you do have it. I know of nothing that makes booting easier and faster. I just love my PFM+PLUS modification.

One other hardware modification or device that I really feel is essential to the Genny user is the addition of more memory. This can be done by acquiring a Myarc 512K or 128K memory/RAM card. This card can be modified by Cecure Electronics to become a memory expansion board for Genny so that it will provide additional RAM. (This RAM can be used for a larger print spooler or internal RAM disk.) If you don't own or can't find a Myarc card, Bud Mills Services sells a card called MEMEX, and this card is designed with the Genny memory expansion in mind. (I personally don't care for the GENMOD, which is necessary to provide more than 512K of memory expansion to your Genny, but this is really a matter of personal choice.)

HEAT KILLS

In the past, heat build-up has been a significant problem within our community, and I, at one point, had made significant modifications to my PEB so as to better dissipate the heat off of the various cards. A better solution for this problem is the use of an IBM tower case. Though the other hardware modifications, which I mentioned above, would be better left to the experts at Cecure Electronics, the tower case is a project that can be reasonably attempted by the hobbyist who likes to tinker. If you contact Mike Maksimik on the users group's BBS, he can give you what information you may need to do this project. A tower case can make your life so much easier. In the years which I have had mine, I have never had a heat or power related lock-up or shutdown of my system. It has performed flawlessly and quietly under the hottest conditions, and that is a very heavy statement to make. Please seriously consider placing you Genny system into a tower case.

HARD DISKS ARE ESSENTIAL FOR POWER COMPUTING

On the subject of hardware, we should consider the Myarc HFDCC. I consider this card to be integral to the Geneve, but I must admit that if you don't own one and you aren't able to find one, a SCSI card will probably take care of your hard disk needs. In order to use this card to its best use, you must make sure that you have the latest EPROM installed on this board. I believe that the latest HFDCC EPROM is version 11. Contact Cecure to make sure that you have the latest EPROMs on both your Genny and HFDC cards. It should also be mentioned that the latest versions of M-DOS now support 1.44 mb. drives with the HFDCC only. If you are using a Myarc floppy disk controller, a TI or CorComp controller, you are still unable to format the high density 1.44 mb. disks. I am not sure how floppy usage is handled with the SCSI card. I believe that it will allow up to 2.88 mb. high density floppy disks. I also believe that you would be able to read other 3.5-inch floppy disks.

FLOPPY DISK DRIVES ARE BASIC

Speaking of floppy drives, I recommend that you have at least one 3.5-inch high density drive with your Myarc HFDCC. These drives will be able to read 3.5-inch disks done on other TI or Genny systems, and you could also generate disks which others in our community would be able to read. In this way, you remain compatible with other members of the TI/Geneve community. I do not recommend having a 5.25-inch 1.2 mb. disk in your system. This drive would not read anything other than other 80-track disks, and it would not be able to generate disks for the 40-track drives which still abound in our community. In my own system, I have a high density 3.5-inch drive and a 40-track, 5.25-inch, DS/DD drive. This keeps me compatible with disks which I generated at an earlier date and with the other people's systems while allowing me the opportunity to use high density disks. I might also mention that some people find the 1.2 mb. 5.25-inch floppy disks to be somewhat unreliable.

Speaking of unreliability, I have lately had some problems with blown disks. In both cases, they were disks which had directories written to them. I don't know if I'm doing something wrong, as I have not heard of anyone else having the some problem. I therefore just warn you to be careful. If you must create directories on your disks, just make sure that the disks are backed up. I have lost some work files by not having done this.

(See Page 22)

GENEVE SUPPORT —

(Continued from Page 21)

SCSI FOR GENNY

Thanks to the work of Mike Maksimik, we now have a version of M-DOS that contains SCSI support. This means that SCSI works for Genny now! If a Genny user runs version 4.0S of M-DOS, he will have full access to any SCSI devices attached to the SCSI card. At this time, I am using that version of M-DOS with no ill effects or limitations. As it seems to be working well, I can recommend it. This version has full support for PFM and PFM-PLUS. Currently, my only SCSI device is an 80 mb. SCSI hard disk. I purchased it for the express purpose of backing up my two MFM hard disks.

MORE ON HARD DISKS

It is virtually impossible to find new MFM hard disks. As you know, we are unable to use either RLL or IDE hard drives. Almost any that you find will be reconditioned or used units. If you don't have a hard disk yet, you would be better off first getting your drive(s) before you decide whether you wish to get a HFDCC or a SCSI card. If you wish to purchase a HFDCC, contact Don Walden at Cecure Electronics. If you wish to purchase a SCSI card, contact Bud Mills at Bud Mills Services.. Though you will probably be satisfied with either SCSI or a MFM drive, the SCSI drives seem to have some advantages which the MFM hard disks don't, according to Maksimik. A HFDCC will control only two hard disks and up to four floppy disk drives. High density disk drives are supported. A SCSI card will also support three hard disks and I don't know how many floppy disk drives. It will also support a CD-ROM player, a SCSI tape backup drive, and many other SCSI devices. If someone gets around to writing the necessary code for them, there are many SCSI devices which may eventually work their way into our community.

John Johnson, a fine programmer in our community, has said that "Winchesters" tend to drop sectors at will, usually in a directory, causing the loss of an entire directory or volume. This has happened to me also. John surmises that the problem is with the HFDCC. Maybe it is, and maybe it isn't. One possible reason is the way in which the drives were being formatted; we need a good formatting program for MFM drives. Therefore, if you already have one or more MFM hard drives, you need Maksamik's fine program CFORM. The only purpose of this program is to format your MFM hard disks. This program seems to do this job much more efficiently than MDM5, as it will give you more sectors than MDM5. It is also faster, and it seems that drives formatted with this program have fewer crashes and other problems related to inefficient formatting.

AVOID USING MDM5

I recommend against use of MDM5. If you use Clint Pulley's fine program, Directory Manager, you will be able to do all of your copying of disks and files easily. It also allows you to create directories on both hard and floppy disks. It is a fine program that will take care of virtually all of your computer housekeeping needs. Maksimik's CFORM will take care of your hard disk formatting chores more quickly and more efficiently than MDM5 will. The program BACKUP MISER will allow you to back up your hard disks to floppy disks. With this program, the only thing that I wish is that it would support the use of high density disks for backup purposes. At this time, if you use high density disks, you won't have any more sectors available to you than if you used quad-density disks. There is another program, of which I have only heard, which will do string (title) searches on a hard disk. With the programs I have mentioned here, you can easily do without MDM5. It seems that this program causes more problems than it is capable of solving.

TELECOMMUNICATIONS

With regards to telecommunications, I suggest that you use nothing less than a 2400 baud modem, and if you can get a faster one with fax capability, I suggest that you get it. I say this because of the availability of PORT by Tim Tesch. PORT supports all of the above, ANSI graphics and more.

SOFTWARE

At this time, Barry Boone's EXEC is probably my most used program. It is most used because it is the one small file that allows me to do 98 percent of my computing work in native M-DOS mode, and this is the way that I feel Genny should be. The GPL environment exists only to allow Genny to emulate a 4A. My-Word remains my most used application. After this, I use TPA4MDOS and SPELL-IT! I use all of these programs in native M-DOS mode, and I am very happy with how they perform. I also use PR-BASE, but I am unable to run it in M-DOS. This is also the case with Multiplan, and I believe that this is also true of TI-BASE. I will occasionally use Page Pro. My most used utility programs are Directory Manager, DISKU, CYA, and Archiver. I occasionally use MEM-TEST, and I am currently using XCOPY quite a lot. I also use the program REMIND ME! almost every day. I will use other programs on an as-needed basis. As you can see, I have total use of those programs which are most important to me, and I don't really NEED any others, though I do enjoy cutting into nice new utility programs. Speaking of utility programs, I wish to speak about CYA.

CYA

I think that you are silly if you don't invest in this program. It is the best thing since sliced bread, and it allows you to pretty much do without an AUTOEXEC file or, if you do use one, you will need only three or four lines in it. This one program gives you more utility with your chosen version of M-DOS than anything else, and it's super cheap.

WHICH VERSION OF M-DOS TO USE

I recommend that everyone use version 4.0S of M-DOS. If you have a SCSI card, you must use this version. If you have limited or slow storage for the operating system which you will boot, and you do not have a SCSI card with any SCSI devices, then you can do very well with version 2.2 of M-DOS, but version 4.0S is for everybody who has the storage and speed to use it. It is the most debugged version of M-DOS around, and it is the version with the greatest range of support in our community. I also urge you to configure your chosen version of M-DOS with CYA. Anything else that you may choose to do will be harder.

DON'T FORGET THE HORIZON RAMDISK

I would not want to forget the Horizon

(See Page 23)

GENEVE SUPPORT —

(Continued from Page 22)

RAMdisk. It has long been a favorite of mine, and it still finds unqualified acceptance from me. It is a fine product and a great device to have in your system and, thanks to Jim Schroeder, it can be formatted into a 3.2-mb. storage device, if you have the appropriate number of memory chips on board. You can have two 3.2-mb. RAMdisks on board, if you have a single 4000 series HRD. I might also mention that the 4000 series board has RAMBO already built into it; you don't need a separate board for this.

Some ask why I choose to maintain my infatuation with a RAMdisk now that I have a SCSI hard disk, which is, admittedly, quite fast. The answer is related to my usage. I have two RAMdisks, and they are both on a single board. The 4000 series Horizon allows you to do this; you can partition your 4000 board into two physically discreet RAMdisks, and each can have up to 3.2-mb. of RAM memory. This is the reason I chose to go with the more expensive 512K chips rather than the cheaper 128K chips. It will cost me more, but I have a greater memory potential in using 512K chips. When I am working on a project, such as a newsletter, I am constantly saving to one of my RAMdisks. I just happen to prefer using RAMdisks rather than a hard disk when I need to save out a lot. It means that I am using electronic storage rather than mechanical storage. I know that the SCSI hard disks and floppy disks are fast, but they aren't as fast as my Horizon RAMdisk. I really love this device, and I can highly recommend it.

WHAT ARE YOU LOOKING FOR

Now is the time for all Genny users to ask themselves the questions, "What am I looking for in a computer? What do I want my computer to do for me? What is my current usage? What is my expected usage?"

Asking these questions will allow you to better decide what you need to do to upgrade your Genny, or whether you need to get rid of it and pop for a "klone." When you are looking for a computer, you should first find the software that does the job that you need to have accomplished. You then find the machines that will run your chosen software.

Genny is a fine machine that is capable of doing all of my computer-related work, with the exception of my music production work. I have no intention of dropping Genny, though I must curtail my general computer-related activities. This is one reason why I shall not write any more Geneve support articles. Another reason for this decision is the fact that I don't feel that there is a need for further Geneve support. Genny is alive, well, healthy, and kicking. I began writing these articles at a time when people who had already chosen to leave the TI community in favor of the klone kommunity were constantly attacking Genny for the most specious of reasons. They plied their excrementally exquisite arguments on various BBSes and in the group's newsletter. Eventually, they left to form their own groups. In spite of their attacks, Genny survived and prospered. It is clear to me that Genny is a worthy successor to the TI-99/4A.

The 4A is a good machine, but it is limited. Its limitations are built into it, in terms of speed, 40-column display, CPU RAM memory, VDP RAM, heat problems, a limited keyboard, a "fire hose" interface that is a source of program lockups, limited video resolution, etc. Various methods of overcoming and working around these built-in limitations have emerged: we have devices such as the RAVE Speech Card, the Horizon RAMdisk, the Myarc memory/RAM card, the RAVE keyboard, the RAVE IBM AT case, TIM, SOB, various 80-column cards, and diverse hardware projects to include 32K RAM in the console, increase the computer's speed, etc.

In spite of these various means of overcoming the 4A's limitations, they are unable to do one important thing: They are unable to change the fact that when you get finished with all of your modifications and additions, underneath it all, you still have a 4A with the same limitations. Genny was designed with the idea of eliminating most of those problems and limitations, she was therefore free to deal at a higher level of operating efficiency. This is why I chose to go with Genny, and I'm glad that I did.

It is clear that interest in Genny is a major reason why our community has remained alive. Few if any programs are being written for the 4A at this time while a considerable number are being written for Genny. Projects such as the SCSI card were able to generate compatibility with Genny well before that same compatibility was achieved with the 4A. In fact, the accomplishment of the compatibility of the SCSI card with Genny proved that such compatibility is possible with the 4A, and it was the successes with Genny that provided the inspiration and motivation to continue and finish the code for compatibility with the 4A. From my perspective, Genny is the only way to go in our community. It has the greatest potential, and it provides the greatest practical challenge.

I still love the 4A, but given the choice of machines, I would choose Genny. For this reason, I would continue to advocate the support of the 4A within our user groups, but I would oppose the treating of 4A users as second class citizens. At the same time, I would recognize that Genny is the future of the 4A/9640 community.

FAREWELL...

I guess that's it for me. It's time for me to move on to other things and into other areas of life. I never promised to stay here until I died. There comes a time in every man's life when change is called for, and this is that time for me. I will maintain my association with the Chicago Users' Group and with the TI/9640 community, but I doubt that I will be doing much or any more writing. It's time to pass this responsibility on to others. As Porky Pig says, "That's all folks!"

Cecure Electronics has new address

The new street address for Cecure Electronics Inc. in Muskego, Wisconsin, is S69 W15100 Cornell Circle, Suite 4. Mailing address is P.O. Box 22, Muskego, WI 53150-0022. Phone numbers are 800-959-9640 (orders only), (414) 422-9942 (technical assistance), (414) 422-9889 (fax) and (414) 422-9669 (BBS).

Disk and mailing labels from one program

By JIM SWEDLOW

The following article is excerpted from a column that originally appeared in the Orange County ROM *newsletter.—Ed.*

I have previously published this program, called Label, as two separate programs: one that prints disk labels and one that prints mailing labels. It got to be a chore changing programs, so I combined them.

CUSTOMIZING

Substitute your name and address in the DATA statements in line 170. Be sure to keep the leading and trailing commas, unless you use a four- or five-line address. You address lines should be no more than 27 characters long.

Lines 190 to 260 are the printer control codes. They work for most Epson-compatible printers. If your printer uses different codes, change them as needed.

If your printer name is not "PIO," modify line 280 as needed.

DEBUGGING

If you run into problems, delete the :: !@P- at the end of line 180 to keep prescan on for the entire program. Also change ON ERROR 580 to ON ERROR STOP in line 370. This should help you find any errors.

PROGRAM NOTES

Lines 550 and 560 do some interesting things:

ON POS is used to validate the correct key return. The value of "I" then is used, after returning from the subroutine, as the value in ON I GOTO ....

The second CALL KEY in line 560 remaps the keyboard as a 99/4A so that you can input lowercase characters for your labels.

By changing 13 and 32 to 80 (or "Enter" and "space bar" to "P") in line 550, you can start printing by pushing "P," or Enter, or the space bar.

When MICROpendium typed the Label program in, we changed a variable in line 520. The original line was:

.... ELSE D=VAL(E).

We changed it to:

.... ELSE D=VAL(E$).

The program didn't work properly with the original line. —Ed.

LABEL

```
100 ! LABEL !003
110 ! Extended BASIC V3.0 !0
93
120 ! By Jim Swedlow !163
130 ! 22 Mar 86 !213
140 ! Based on a program by
Phil Barnes !251
150 GOTO 180 :: IN$,SS$,DS$,
PI$ :: DIM Y$(4),A$(5):: C$,
D$ :: DIM F$(144),T$(144)::
B,J,D,E$,I,C,@ :: CALL KEY !
228
160 DATA "  FIRST CLASS
","  MAGNETIC MEDIA",DO
NOT BEND * DO NOT X-RAY,DO N
OT EXPOSE TO MAGNETISM !070
170 DATA Your Name,Street Ad
dress,City,STATE,ZIPCODE !06
7
180 @=1 :: CALL CLEAR :: FOR
J=0 TO 14 :: CALL COLOR(J,1
6,@):: NEXT J :: CALL SCREEN
(5):: !@P- !111
190 E$=CHR$(27)! ESCAPE !241
200 DS$=E$&"G" ! DOUBLE STRI
KE !249
210 C$=CHR$(15)! CONDENSED !
206
220 D$=CHR$(14)! DBL WIDTH !
173
230 PI$=CHR$(18)! PICA !177
240 SS$=E$&"S"&CHR$(0)! SUPE
RSCRIPT !191
250 IN$=E$&"@" ! INITIALIZE
PRINTER !123
260 E$=E$&"3"&CHR$(12)! CHAN
GE LINE FEED TO 12/144 INCH
!128
270 PI$=PI$&C$&D$ :: DS$=DS$
&C$ :: SS$=SS$&E$ :: Y$(@)="
DF" :: Y$(2)="DV" :: Y$(3)="
IF" :: Y$(4)="IV" :: D$="DSK
1" :: C$="" !135
280 OPEN #1:"PIO" :: DISPLAY
AT(3,9):"LABEL 3.0": :"Chec
k the position of the labe
ls before printing." !114
290 DISPLAY AT(20,@)BEEP:"<A
>ddress":"<C>ustom ":"<D>isk
":"<W>arning" !199
300 GOSUB 550 :: IF I>4 THEN
300 ELSE B=@ :: GOTO 330 !0
07
310 DISPLAY AT(10,@): : :"
Labels/Disk: ";B:"Driv
e:    ";D$:"Comment:":
TAB(6);C$: :!196
320 GOSUB 540 !110
330 ON I GOTO 470,500,310,47
0,370,340,600 !023
340 ACCEPT AT(12,20)SIZE(-2)
VALIDATE(DIGIT)BEEP:E$ :: IF
E$="" THEN 340 ELSE B=VAL(E
$):: IF B=0 THEN 340 !066
350 ACCEPT AT(13,20)SIZE(-@)
VALIDATE("123")BEEP:E$ :: IF
E$="" THEN 350 ELSE D$="DSK
"&E$ !127
360 ACCEPT AT(15,6)SIZE(-25)
BEEP:C$ :: GOTO 320 !074
370 ON ERROR 580 :: C=0 ::
ISPLAY AT(10,@):: DISPLAY AT
(20,@):"Initializing" :: OPE
N #2:D$&".",INPUT ,RELATIVE,
INTERNAL !245
380 INPUT #2:F$(C),I,I,I ::
T$(C)="FREE "&STR$(I)!012
390 DISPLAY AT(22,@):F$(C);"
";T$(C):: IF C=127 THEN 420
ELSE INPUT #2:F$(C+@),I,J,J
!161
400 IF F$(C+@)="" THEN IF C>
5 THEN 420 ELSE C=C+@ :: GOT
O 390 !149
410 I=ABS(I):: C=C+@ :: IF I
=5 THEN T$(C)="Prog" :: GOTO
390 ELSE T$(C)=Y$(I)&STR$(J
):: GOTO 390 !231
420 CLOSE #2 :: ON ERROR STO
P :: DISPLAY AT(20,@):"Print
ing": : : :: FOR B=@ TO B ::
J=0 :: D=8 :: E$=" " !090
430 PRINT #@:DS$;F$(0);E$;C$
;E$;T$(0);SS$: :!156
440 FOR I=J+@ TO J+D :: PRIN
T #@:F$(I);TAB(12);T$(I);T
(18);F$(I+D);TAB(29);T$(I+D)
;TAB(35);F$(I+2*D);TAB(46);T
```

(See Page 25)

LABELS —

(Continued from Page 24)

```
$(I+2*D):: NEXT I !217
450 J=J+24-6*(D=10):: IF C>J
THEN D=10 :: PRINT #@: : :
:: GOTO 440 ELSE PRINT #@:IN
$ !095
460 NEXT B :: B=B-@ :: FOR I
=@ TO C :: T$(I),F$(I)="" ::
NEXT I :: GOTO 320 !098
470 IF I=4 THEN RESTORE ELSE
RESTORE 170 !084
480 FOR C=@ TO 5 :: READ A$(
C):: NEXT C :: D=10 :: GOSUB
570 !194
490 GOSUB 540 :: ON I GOTO 4
70,500,310,470,530,510,600 !
082
500 FOR C=@ TO 5 :: A$(C)=""
:: NEXT C :: D=@ :: GOSUB 5
70 !197
510 FOR C=@ TO 5 :: ACCEPT A
T(C+9,2)SIZE(-28)BEEP:A$(C):
: NEXT C !020
520 ACCEPT AT(16,19)SIZE(-2)
VALIDATE(DIGIT)BEEP:E$ :: IF
E$="" THEN 520 ELSE D=VAL(E
$):: IF D THEN 490 ELSE 520
!058
530 DISPLAY AT(20,@):"Printi
ng" :: FOR I=@ TO D :: FOR C
=@ TO 5 :: PRINT #@:PI$;A$(C
):: NEXT C :: PRINT #@:IN$:
:: NEXT I :: GOTO 490 !195
540 DISPLAY AT(20,@)BEEP:"<A
>ddress <P>rint labels":"<C>
ustom <M>odify defaults":"<D
>isk <Q>uit":"<W>arning" !19
1
550 CALL KEY(3,I,C):: IF C<@
THEN 550 ELSE IF I=13 OR I=
32 THEN I=80 !168
560 I=POS("ACDWPMQ",CHR$(I),
@):: IF I THEN CALL KEY(5,C,
C):: IF B THEN DISPLAY AT(20
,@): : : : :: RETURN ELSE
RETURN ELSE 550 !037
570 FOR C=@ TO 5 :: DISPLAY
AT(C+9,@):">";A$(C):: NEXT C
:: DISPLAY AT(15,@): :"How
many labels: ";D :: RETURN !
173
580 DISPLAY AT(10,@)BEEP:D$;
" Could not be accessed" ::
ON ERROR 590 :: CLOSE #2 !12
0
590 ON ERROR STOP :: RETURN
320 !236
600 CALL CLEAR :: CLOSE #@ :
: END !061
```

HARDWARE PROJECT

Start by replacing power supply, but don't forget to remove regulators from cards and other places

This article comes from the September 1996 TIsHUG News Digest. *We believe they got it from the SouthWest 99ers originally. As with all hardware modifications, users undertake this project at their own risk.—Ed.*

The original "linear" power supplies provided by Texas Instruments for our peripheral expansion boxes have been faithful and obedient (if at times mischievous) servants for many years. It is time to begin to replace them. In the past year I have replaced three of them in various systems. The most drastic case was the one involving W. Leonard Taffs (pull up a chair, Holmes, while I fill you in with the details).

Leonard was having troubles with his disk drives. He had updated to two DSDD drives and his system began to act peculiarly. It would access and "read" his drives at random. It would do fine when he first turned the system on but after he had worked for some time it would occasionally not "read" or "write" to his drives correctly. Now, for me, this would be the normal state of affairs but it seemed to perturb Leonard to lose two hours of work. The trouble turned out to be a faulty power supply. Before Jack Mathis and I managed to pin this problem down, the low power it was producing caused damage to his 32K memory card, RS-232 card and a CorComp disk controller. We replaced his original TI power supply and the problems went away.

"Matt" Matthews had a power supply go bad in one of his boxes and we replaced it with another linear power supply.

Jack had a power supply go bad and he replaced it with a "switching" style "baby IBM" power supply. He now has, resident in his box, a 40-meg hard drive, a 5.25-inch floppy drive and a 3.5-inch floppy drive.

I began to have "read/write" errors with my computer (or at least more than normal for me) and attached a voltmeter to my original power supply. I noticed a severe drop in voltage when I was accessing my drives. Now, I have the lowest power requiring drives I can find so I knew it was not the drives' fault. I had been itching to install one of the switching power supplies so here was my chance. I installed the power supply in about four hours. This amount of time was not bad considering it was the first one I had changed by myself. I jumped and installed my cards, crossed my fingers and turned 'er on. Hmmmm, it didn't make any obnoxious noises or emit smoke! I must have gotten it right! I noticed immediately there was *no* fan noise. I had previously installed a low-noise fan but this still made some noise.

I also removed the transformer from inside the console and ran the power for the mother-board from the PEB transformer. I don't have the "ol' coffee warmer" any more. My console runs so cool, I can feel the heat from my XBASIC cartridge! I now run four disk drives, my console and the PEB from *one* power cord! My desk is

(See Page 26)

POWER SUPPLY —

(Continued from Page 25)

no longer an electrician's nightmare and I can run all my computer equipment from one five-outlet power strip. I even have a 110-volt outlet on the back of my PEB for a hard drive (future expansion plans)! My drives all run faster, quieter and smoother and my console has not locked up with XBASIC no matter what length of time I spend on it!

The one drawback I have noticed is the fact of not being able to turn off the console separately from the PEB. I have managed to locate a supply of three-pole double-throw mini-toggle switches so that problem can be remedied. They would work pretty much the same as the "reset" button on an Ingestor (of) Big Memory machine.

YOU TAKES YOUR CHANCES

If you undertake this project on your own I recommend you have a copy of the TI schematics for *all* cards and equipment you modify. I strongly recommend you get the June '93 edition of MICROpendium and read the article by Jack Miller on page 7. This is a "*must*"! I also recommend the *removal* of all jumpered regulators. You can jumper them and leave them on if you wish *but* they serve no purpose and are merely another component to go bad and short out. It's your money and you takes your own chances.

The author of this article does not claim to be an authorized service representative of any company and will assume no responsibility for any dire consequences.

I have done this modification and I can assure you, I ain't no genius. With a little care and precaution *you* can do it also.

Now, let's go to the most controversial steps in the power supply conversion procedure: the *removal* of the regulators from the "cards."

Texas Instruments' original design, with its linear power supply, was to ensure the steady supply of power to peripheral devices. In order to ensure that there would be sufficient power they heavily over-supplied the card strip bus. Each card would then regulate the voltage it required. When I say *heavily*, I *mean* heavily! TI's specs originally called for +16 volts on the +12 volt line and +8 volts on the +5 volt line. This was on an "ideal" power supply. I have found +25 volts on the +12 line and +13 volts on the +5 line. This is akin to using a 16-lb. sledgehammer to drive a thumbtack. Sure, it can be done, *but* you have to regulate the force.

There has been some talk of running the +12 volt line of the new power supply to the +5 volt line and leaving the regulators on the card. This will work, but I compare it to having a horse pull a working automobile. What's the point?

What we are trying to achieve is the supply of "clean" power and the removal of heat sources (the regulators). The removal of the regulators means if you place your "modified" cards into a standard TI PEB, you will have a new coaster to set your coffee cup on. It *will* be toast. *Never* swap cards from system to system *unless* they have the same type of power supply.

Ah, well, enough chit-chat, let's get on with the carnage.

TOOL LIST

- Low wattage soldering iron
- Small Phillips screwdriver
- Small flat-edge screwdriver
- Solder (small gauge, electrical)
- 24 gauge insulated wire
- Small diagonal-cut wire cutters
- Towel (to set your cards on)
- Small can (for screws, etc.)
- Ohmmeter

THE TI DISK CONTROLLER CARD

There are two handle clips on the "clam shell" covering. Remove them by lightly spreading the bottom of the loop apart. Set the clips in the can. There are four screws located in the corners of the case. Remove these screws. Some of the TI clamshells do not have the screws. Separate these style cases by depressing the locking "fingers" with your flat screwdriver.

Spread the towel on your work space. Turn the disk controller screws-side down and the screws should drop out. Place the screws in the can and lay the disk controller (henceforth to be called the "DC") with the drive connector to the right and facing up. The bus connector will be towards you.

Insert the flat screwdriver into the crack on the side where the shell halves meet. Twist the screwdriver, do not pry. The halves should separate. You might have to do this on both sides of the case. Do *not* put the screwdriver into either the opening for the bus slot or the one for the drive connector. The label on the top of the card might be stuck to both halves, but just fold it over.

In the lower right hand corner of the card is a flat, three-legged component. It will have the number TL780-05C, 78M05C or some combination of 78 and 05 on it. It is easy to spot. On the lower left, below the LED (that little yellow or green light which comes on when the card is activated) is one which looks like it except it says 78M12C. The one with the 05 is the +5 volt regulator and the one with the 12 is the +12 volt regulator. (Nope, don't take no rocket scientists 'round *here*!)

Remove the card from the shell. Set the shell aside. Plug in the soldering iron. Lay the card, component side up, so the LED is to your left and the drive connector is to your right. Insert your flat screwdriver beneath the outer/lowest leg of the +5 regulator. Heat the leg with the soldering iron and gently twist the screwdriver. The leg will lift free. Bend this leg up out of your way and do the same to the other two legs. Do the same to the +12 regulator.

Cut a 1-inch long piece of the 24 gauge wire. Strip and tin the two ends of the wire. Insert and solder one end of this wire into the closest/lowest hole where the +5 regulator was. Insert and solder the other end into the farthest/highest hole of the +5 regulator. Ignore the center hole (this is the ground terminal for the old regulator). Do the same for the +12 regulator. Turn the card over, component side down, and clip off any protruding wire from your jumper wires.

You can ohm-test your card if you wish. (I highly recommend it.) Look for a *direct* connection (0 ohms) to ground (the center hole of the old regulator). There will be a 750-ohm (approximately) between the +5 leg holes and ground. *This is normal.* There should be *no* direct (0 ohm) connection.

(See Page 27)

POWER SUPPLY —

(Continued from Page 26)

If you do have a connection, check your solder. Make sure the solder did not pool and make an overlap with any other traces (the small "lines" on the circuit board). Correct any problems and retest. Unplug your soldering iron.

Insert the card into your "IBM"-powered box (I never replace the clam-shells), Connect the drive controller cable and turn on the power. The drive motor should turn on and run for about 10 seconds. If the drive light comes on and stays on, you have the cable reversed. Turn it over and try again.

32K MEMORY CARD

This is the same as the drive controller with the obvious exception of not having a drive cable. If you stand your PEB on end, you might want to put the clam-shell back on this card. It has no protruding end such as the disk controller, The RS-232 card and flex cable interface have to hold them in place. I use a thin piece of plastic to insulate it from my DC card.

FLEX CABLE INTERFACE

Remove the shell the same way. Place the cards with the chips up and the "firehose" leading to the right. On the very lower right hand corner you will see the +5 regulator. (Hey, you're starting to get good at this.) Remove and jumper it. This is the only regulator on this card.

Replace the clam-shell on this card. You need the "tab" on the rear of it to attach it to your PEB. Do *not* take apart the end which attaches to your console. There may be no need to replace the regulator mounted on the small board in there. The end can be a real nightmare to put back together correctly. Try the card without removing the regulator first. My luck has been running about six to one towards removing this regulator. You can always pull the "foot" apart if yours needs to have the regulator removed.

RS-232 CARD

This card is a little trickier but it is still easy. Remove the shell. Lay the card component-side up and the cable connections to the right. The +5 and +12 regulators are in their usual spots. Remove and jumper them. Approximately 1 inch to the right and a half-inch above the +12 regulator (the one on the left by the LED) you will see a small three-legged component shaped like a transistor (it is shaped like a small black coffee can, flattened on one side, with three legs). It is marked "Q3" on the board. This is a -12 volt regulator. Remove it by grasping the plastic body and heating the legs until it comes free from the board. The three holes it was in form a triangle as opposed to the line formed by the other regulators. Insert and solder one end of a jumper wire into the *top* (towards the top of the card *away* from the bus strip) hole. Insert the other end of the jumper wire into the leftmost (towards the LED) and solder. Test your soldering with the ohmmeter. *Remember,* you are looking for a *direct* ground. Some cards will show a connection with a resistance. This is normal.

TRYING IT OUT

Place your cable card back into the PEB. Insert the mounting screw and tighten. Turn on the power. The LED in front should come on. Let it run for five minutes. No smoke? Turn off power. Install your RS-232. Power up again. There should be a light on the cable card and none on the RS-232. Do the five-minute test again. Do the same for your 32K memory card. The cable card LED should stay *lit* when power is on. The 32K card LED should flicker briefly at power-up and then go out.

Power down. Hook the firehose into your console. Turn on the PEB. There should be no LEDs lighted. Turn on your console. The flex cable LED should light. Insert your Extended BASIC cartridge. Go to the command prompt of the XBASIC (you know, hit any key and then select "2" for XBASIC). The computer should try to access the drive. Make sure you have drive No. 1 hooked up. The cable card LED should flicker, the disk controller LED should light, the 32K should flash and the disk tries to engage.

Now, put a disk in with a "Load" program and attempt to load it. It works? Fine, install the drives and close her up. If it does not work and the connections are correct you will need to disassemble the foot and remove the regulator inside. Remove the four screws and separate the halves. You should be able to spot and replace the regulator by now. Make sure you get the cable connector back into place and all the way forward before you tighten the screws.

AFTER-MARKET CARDS

Any after-market cards will need to have the regulators jumpered on them also. Most of them will have only a +5 regulator. They will almost always be in the lower right hand corner. Remove and jumper them as usual.

One notable exception is the Geneve card. This Cadillac has from five to seven regulators on it. They are removed and jumpered the same as other regulators. Be *very* careful with these cards. The traces on these cards are extremely delicate and must be handled with the utmost caution.

As with any project of this type, if you undertake this mission the company will disavow any knowledge of your actions. This tape will self destruct in five seconds (cut to theme music and close-up of smoking PEB).

CONSOLE POWER SUPPLY

Well, we have made it this far with no mishaps (that minor one where we had to call the fire department we won't discuss). Now it is time to go to the final step — the removal and replacement of the ol' "coffee warmer" power supply *inside* the TI console.

The tools needed for this part of the project are:

- Phillips screwdriver
- soldering iron and solder
- wire cutters
- electrical tape

You will also need approximately four feet of four-strand electrical wire. (Minimum of 20 gauge. I used "trailer" wire. You can find this item at almost any auto parts store or hardware store. It is color-coded and also plastic-coated for durability.) Two sets of four-pin connectors. (Two male and two female. I used Radio Shack connectors Nos. 274-204 and 274-205. These are stout connectors with a good strain relief attached. Matt Matthews uses a four-pin "microphone" style connector with good results. Any type of four-pin connector may be used.) Small, flat screwdriver, small container for screws, etc., Volt/Ohmmeter.

The first step in this procedure is to disconnect the power transformer from the wall and the console. Discard the transformer.

(See Page 28)

POWER SUPPLY —

(Continued from Page 27)

Turn the console switch to the off position. Turn the console upside down with the on-off switch facing you. Insert the flat screwdriver beneath the front edge of the slide switch and pry the slide off (those of you who are converting a tan console can ignore this step).

Now, take the Phillips screwdriver and remove the seven screws holding the console halves together. There should be three screws on the top or far end and four on the near or bottom end. I say "should" because when the console was originally shipped there were seven. However, in the process of cleaning or repairing your console over the years you may have, ahem, "forgotten" to replace all of them. It is said the difference between a "mechanic" and a "technician" is the fact that a mechanic will replace all screws, bolts and brackets removed while a technician does not need all those "extra" parts.

Lift the top part of the console (actually the "bottom" but I am not going to argue semantics at this late stage of the game) and set it to one side (or t'other). In the lower left of the console you will notice a small circuit board (approximately 3.5x4 inches). Remove the two screws holding it on. Do not remove the one screw which is recessed. Lift up this circuit board.

There will be a white plastic clip-on connector attached to the board. Disconnect this by squeezing the clip and pulling. Lift out the old power receptacle (connected to the wires still attached to the circuit board) and the wires which may be taped to the metal shield. Discard the circuit board and wires. Lift out and discard the remainder of the old on-off switch which you will find lying in the slot.

Now might be the time to give your computer that thorough cleaning you have been meaning to do for the last two years or so.

Hold the white plastic connector going to the computer with the retaining clip up and the wires going *away* from you. The voltages are, from left to right:

Pos.(+) 5 Ground Pos. (+)12 Negative(-)5 volts

Tape and label each wire about three inches away from the connector. Snip the wires at the connector and discard the connector. Strip and tin the ends of these wires. Cut a five-inch-long piece of the four-strand wire. Strip and tin both ends of this section of wires. (Yeah, yeah, I know I did not tell you to plug up your iron, so sue me.) Solder each wire leading from the console to one wire of the five-inch section (so help me, *nobody* better even call me and tell me he connected all the wires to *one* wire and now his console won't work.), and tape each connection.

Take the *male* connector and open it up. Remove the metal pin and they just slip apart. The strain relief will come off if you remove the screws and bend each metal bracket back. Slide the cover part of the connector over the wires. Now:

- Connect the *negative* 5-volt wire to pin number two *first.* BJ and I have fought this connector and she found the easiest way to do it is to solder the No. 2 pin first.
- Then connect the positive 12-volt wire to pin No. 1.
- Then the positive 5-volt wire to pin No. 3.
- Finish off with the ground wire to pin No. 4.

Slip the connector back together (there is a small pin which slides into the hole to hold it together). You might have to do a little touch-up on your soldering in order to make the connector fit. *Unplug* your soldering iron and put it into your hip pocket (just checking to see if anyone is paying attention) and scream. The setup you have now should be

Pin No. 1	Pin No. 2	Pin No. 3	Pin No. 4
+12 volts	(-)5 volts	+5 volts	Ground

String the new wire and connector out the cutout for the old power transformer plug. Tape the wires to the metal heatshield. Place the top/bottom of the console into place, replace and tighten all the screws (however many there might be). Place the console aside.

Take your soldering iron out of your hip pocket (don't worry, the heat will have cauterized the wound) and plug it in. Strip and tin both ends of the section of wires you have left. You can cut the length to whatever size you want. Connect one female end and one male end to the length of wires. Make sure you follow the color coding you used on the console. If you did not get color-coded wire, make *sure* you "ohm-out" *each* connection with its corresponding pin number. Set this cable aside.

Strip and tin the four wires you have extending from your PEB and attach them to the remaining connector. Unplug the soldering iron and put it away. Assemble all the connectors if you have not done so. Connect your voltmeter and turn on the PEB. Check the voltages on your PEB connector first. If they are hunky-dory, then power down and connect your extension cable Turn on the power and check the voltages at the end of this cable. OK? Power down again and connect your console to the cable. Connect the monitor cable and turn the power on. You should have the TI screen on your monitor. If you do, then power down again and connect the firehose cable. Power up and you are running with "real" power now. Your console will remain so cool, you will be able to feel the "heat" actually generated by your XBASIC cartridge.

There are many different things which could have been done in this article. I, myself, would recommend the use of "heat-shrink" tubing in place of electrical tape to cover the connections. I would use grommets for all holes drilled in metal where wires pass through. I use a Dremel tool with cutting wheels and various stones to make the holes and removal of the studs.

I have tried to write this article using the simplest methods which should be available to everyone in the TI community. I hope this article has helped you to learn more about this fine machine we are still learning about and enjoying. I believe you will enjoy this conversion to a switching style power supply, and your faithful TI will continue to serve you for many more years.

As per the usual warning, if something goes wrong, *tough!*

Groups change address

The new mailing address for the Portland Users of Ninety-Nines is c/o Ted Peterson, 8686 SW Oak St., Portland, OR 97223-5814.

Also, the new mailing address for the West Penn 99ers is c/o Paul A. Brock, General Delivery, North Apollo, PA 15673-9999.

SCSI — MFM — RLL — WHAT?

TIers keep up with changes in storage technology

By MICHAEL J MAKSIMIK

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

The incredible leaps in technology and demands for newer, faster hardware have become the new military-industrial complex. Old technology falls to let new technology grow and flourish. It is a good thing that our community has adjusted to this and has accepted some of the newer technologies in disk storage. But what is new, what is old, and what is the difference?

Hard disks, as you know, were in existence for years before microcomputers used them. Huge MFM drives on the IBM System/36 (the size of car tires) were used in the late 1970's at incredible cost by today's standards. In fact, some model 5365s (about the size of a dishwasher) are still in operation. Yes, they still run, and run on 220 volts! That much current is needed to keep the disk drives spinning. Those drives emit gravity when they spin!

Eight-inch hard disks were created using a standard which had existed in the form factor of the 8-inch floppy drive. It was an industry standard and microcomputer makers began seeing this as an affordable mode of storage. The standard was FM, Frequency Modulation, also used in FM radio. FM was based on two signals — one signal was a timing signal, a "clock," and the second signal was the data signal. By reading both signals we could store blocks of data onto the magnetic surface, using a magnetic "north" for a zero and a magnetic "south" for a one. The clock would be active at the center of every data cell, so we could "sample" the data line at precise intervals, thereby getting our data. FM was the standard for storage for many years. Single density floppy I/O is still done using FM encoding.

Later, we developed MFM, which embedded the clock signal within the data stream. Ones and zeroes were arranged in such a way that the clock could be extracted from the data stream itself, which also meant we could use the extra space on the drive for data (the *extra* space was used in FM encoding by the clock signal) thereby giving birth to "double" density. In reality, there isn't any doubling of density. In fact, the exact same number of flux transitions occur on double-density media. The difference is that only data is stored, no clock signal.

RLL encoding came later. RLL stands for "Run Length Limited," a technique which solved neighboring bit-cell corruption problems inherent in MFM storage, and at the same time increasing capacity of storage to about 1.5 times the normal capacity of MFM. RLL drives are still used today, but are disappearing in favor of more advanced encoding methods. We can write data into a space using multiple phases which allows 2, 4, 8, or even 16 bits in the same space that 1 bit would have taken. This technology allows a single platter to store what would have taken eight or more platters to store earlier.

NEW ENCODING TYPES

We are also seeing the emergence of new encoding and media types. Up to a few years ago, we just saw magnetic media, either with flexible media, as in floppy disks using a contact-type head, or rigid platter-type media with a flying head. Modifications of this method are used in tape drives (tapes can be streaming or blocked and accessed like floppy drives). I was even discussing with my brother, Tom, about using a VCR to back-up hard disk drives (this was actually used several years ago). In the VCR, the head rotates and the data is recorded in candy-stripe style diagonally across the width of the slow-moving tape. This method allows a high volume of data in a small area.

Lately we have seen new types — optical, magneto-optical, and RAID technology. With optical media, the bit cells are sampled using sensitive LED laser pick-ups which read microscopic pits on the surface of a smooth aluminum/polymer disc. When writing the data, a powerful laser is used. To read the data, a weak laser is used.

In magneto-optical, the media is still read using a magnetic head, but it is much smaller and more sensitive than normal read/write heads. To write data, the media is heated using an optical laser beam. This melts the surface of the media slightly, which allows the magnetic pattern being written to be "frozen" into the surface when the beam stops. I saw the inspiration for this when doing some research about earth plate tectonics (ocean surface). The earth's magnetic field was "frozen" in ferro-magnetic ores deposited on the volcanic rifts of the Pacific Ocean floor over thousands of years. This clearly showed

(See Page 30)

1997 TI FAIRS

APRIL

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

MAY

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

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.

DISK DRIVES —

(Continued from Page 29)

reversals of the earth's magnetic field many times. While these molten ores were solidified, they recorded the current state of earth's magnetic field. We can replicate this in a magneto-optical drive, with more practical results.

"Floptical" media is optical-guided media, where a beam is used to guide the read/write head in smaller increments thereby allowing more storage on the media. As we machine parts smaller and more precisely, we will be storing voluminous amounts of data in the tiniest of spaces. There is even research into storing data in crystals. By using lasers within crystals, we can record an electron-excited state within a unit cell of a crystal and, later, we can read it because the unit cell would change the polarization of light as it reflects. And since there are billions of unit cells in a crystal the size of a dime, we can readily imagine the amount of storage of this newly emerging technology.

Well, we know the various media types and their encoding strategies, but how are they connected to the computer?

COMMON INTERFACE TYPES

The interface types we have seen most frequently over the past few years are: ST506, ST412, ESDI, IDE, and SCSI.

ST506 and ST412 are similar to the Shugart floppy drive interface. The Shugart interface was a standard 5.25-inch drive interface (still used today) to allow control of up to four drives, FM or MFM, with clocking controlled by the media itself (hard-sectored) or the data (soft-sectored).

Hard-sectored disks have an index hole for every location where a sector should start. The index hole would trigger the start of bit cell sampling and that would control timing and synchronization of the disk media.

Soft sectoring used a special bit pattern which surrounded each sector, and that bit pattern would tell the disk controller where a sector began and where it ended. While a Shugart interface was fine for floppy disks (the signal to the head was single ended, that is, only one line was needed to read the bit cell and a logic level determined the value of the bit cell), we needed something better for hard disks.

Hard-sectored disks have an index hole for every location where a sector should start. The index hole would trigger the start of bit cell sampling and that would control timing and synchronization of the disk media.

Since hard disks moved data much faster than floppy disks, we had differential I/O for the reading and writing. In differential, we use a current loop to transmit the data to and from the read/write head. Current loop was immune to the problem of induction of signal logic, which affected single ended lines. ST506 and ST412 are very similar in this respect and, for the most part, are interchangeable. However, ST412 was used with 8-inch hard disk drives, while ST506 was not. ST506 is what we used for our Myarc HFDC controller. Many RLL and MFM drives use this format for system interface.

ESDI was a standard which was implemented in the IBM PS/2 and some clones. The drives were 5.25-inch (some 3.5-inch also) and it was a higher speed interface — typically it could handle enhanced RLL encoding methods and transfer speeds at double what MFM rates were. However, the interface was not compatible with ST506 or ST412. It was processor specific to PS/2 models and was not general enough for the rest of the computing world to adopt.

Integrated Drive Electronics (IDE) is popular for MS-DOS machines, and is specific to the architecture of Intel-based processor boards. IDE drives used a command-based structure, combining an intelligent 16-bit parallel interface with a disk drive. By placing a computer on board the drive itself, the system could relieve itself of the burdens of disk controlling and just issue commands to read/write, etc, and would let the drive do all the dirty work. IDE now allows for other devices (such as CD-ROM and tape drives) to be attached to the Intel-based processors. However, because the standard of IDE is closely tied to signal timing and bus behavior of Intel-based processors, it is difficult to implement on other platforms (including our own 99xx-based processors). The IDE standard has been expanded and, through the use of replicate interfaces, you can attach multiple devices onto an IDE bus. Still, many IDE adapters on PCs are limited to two drives on the bus.

SCSI IS VERSATILE

SCSI was developed in the mid-1980s as a peripheral interface for small computers (hence the name Small Computer Systems Interface). Through commands and handshaking, it could be used to exchange data between peripherals, between host computers, or a combination of both. Peripherals can communicate with each other (as in an unattended tape backup of a hard disk) or in a RAID disk unit (where multiple disk drives maintain small copies of each other, thereby providing data redundancy). Hosts can exchange data with each other if they were connected on a SCSI bus using high speed transfers, and it can provide a flexible interface for parallel processing.

SCSI today has emerged into new implementations. The first, SCSI-1, was used in 1986 to provide disk systems for Apple Macintosh computers, file servers, and high-end computers. Later, SCSI-2 was introduced as a way to refine the command structure of the interface, and to provide more standard commands on compliant devices. It also introduced faster timing methods and defined more device types. SCSI-3 is already out and defines wide data interchange for high transfer rates (40 megabits per second).

SCSI is implemented as an 8-bit parallel interface. There are many more control signals used than in a printer interface, and it is bidirectional, that is, data can be sent and received. Also, the data bit lines are also device select lines, and there are different negotiation phases used to provide an orderly flow of data between devices.

(See Page 31)

DISK DRIVES —

(Continued from Page 30)

There are also device priorities, and each device can itself control eight sub-devices, called logical units. A single SCSI bus can control 64 devices, including the host device and it's locally attached logical units. SCSI allows connection of disk drives, hard disks, removable media drives, printers, tape drives, network adapters, and other devices.

SCSI hard disks can use MFM, RLL or other encoding methods. Yes, you can have an MFM drive with a SCSI interface instead of an ST506 interface. Most SCSI drives are using RLL or multiphase storage encoding, few are using MFM (exept the older drives you can find at computer fairs). SCSI tape drives come in many varieties (streaming, rack, helical scan, reel-to-reel, 3490 cartridge, quarter-inch cartridge, etc.), but the nice thing is that they all share a common command set which makes them all somewhat interchangeable as far as data storage and retrieval is concerned.

SCSI is used to allow printers to get font information quickly from the host while leaving the parallel port free for print data. It is also used for CD-ROM drives and the abundance of cartridge-type media now available (Syquest, ZIP drives. EZ 135 drive, etc.).

SCSI offers several advantages over other media interfaces:

• It is non-computerspecific. This way, it is not tied to any chipset or manufacturer, so it can be easily implemented on any computer platform.

• It uses a high level protocol for data exchange, making your data very secure.

• SCSI devices are intelligent, often controlled by their own microprocessor. This relieves much of the burden of media I/O to the device's onboard hardware, so your computer can do more interesting things. It also means much faster transfer rates.

Our computers (the TI99/4A and the Myarc 9640) can use media types from FM diskettes to SCSI hard disks. We have adjusted to accept the newest media types, inexpensively. SCSI is used in IBM AS/400 midranges and mainframes. It is used in Apple computers and IBM computers, and now it is used with our computers as well. For those who can get them larger than 40 megabytes, the Myarc HFDC can control up to 3 MFM hard disks, and it is the only controller which can control high density floppy drives.

Few MFM hard disks survive in the open market nowadays, but they are still out there and they offer inexpensive storage. MFM drives are a bargain at computer fairs. If you aren't concerned about getting a used drive, or about size (MFM drives are hard to get in sizes larger than 40 megabytes) you can go for that storage media. The HFDC is still for sale and is an adequate system. It offers complete compatibility with the TI and the Geneve.

The SCSI market is expanding wide open, and you have many choices as far as media is concerned. The Western Horizon Technology SCSI controller for the TI/Geneve works well and is easy to hook up. A single cable bus connects all drives together and drives are available in fixed or removable format. In fact, I use the EZ-135 drive. Each of its cartridges yields 127 megabytes of formatted data storage. It is fast (as fast as any SCSI-2 buffered drive) and is cheap for the storage size (the drive is about $120, the cartridges run about $20). The SCSI bus can be shared between the TI and the Geneve, so you can use both computers on the bus simultaneously, sharing drives and media. SCSI has a great future ahead of it because it is ready for the faster processors on the market.

What size should I buy? Well, for now, our file system supports 31 sectors worth of bitmap, at a maximum of 16 sectors per allocation unit. This translates into 248 megabytes total storage per drive. This should be plenty for our uses since programs on the TI/Geneve are compact and we don't use a lot of space for program data.

On PCs, the story is different. The operating systems are graphically based, and many use virtual memory storage. This increases disk requirements on PC's to values higher than 248 megabytes, so we will see many hard disks in the 1 gigabyte or higher range, and fewer in the 248-800 megabyte range.

If you use a TI or Geneve, it really is not a good idea to invest in a very large drive. Save the money and get a cartridge drive, and just buy lots of cartridges. First of all, your storage requirements on the TI will not exceed 127 megabytes at a time. Trust me, you don't have that much data. If you decided to store the contents of software libraries on hard disk, well just separate them into 127-megabyte segments and store these groups on separate cartridges. Rarely will you need to access it all at once.

Storage media seems to be going in two directions: SCSI and IDE (or E-IDE, enhanced-IDE interface). SCSI is clearly at an advantage, because it is supported across the entire computer industry and is already available in 16- and 32-bit versions for wider bus requirements. E-IDE is suitable for the PC segment of the market but it will always be tied to that market. The market determines a final, single standard eventually, and E-IDE will be the next dinosaur. The market determined a standard for PCs and left all other machines in the dust. I am glad that the bandwagon we hopped on was the right one.

Competition produces cassette check program

Competition Computer has produced a new cassette-based checkbook manager for the TI99/4A. Kyle Crichton of the company explains that TI's Checkbook Manager was disk-based, and that the company responded to the needs of cassette-only users.

The program sells for $5, with a $1 shipping and handling charge. Contact Competition Computer at 350 Marcella Way, Millbrae, CA 94030, information (415) 697-1108, orders 1-800-471-1600.

Crichton also says he expects the company, which produces an IBM-compatible CD-ROM , to introduce some new products, both hardware and software, at Fest West in April.

EXTENDED BASIC

A program to create masterfiles of D/V80 files

By W. LEONARD TAFFS

The following comes from Feedforth, a column that appears in the newsletter of the Southwest 99ers User Group. — Ed.

After collecting numerous D/V80 files, it is desired to make up a sorted, composite master file of all of them. Many of these files may exceed the buffer of one's text editor. If using a TI-Writer editor, this necessitates dividing the larger files into buffer size.

As more and ore separate files accumulate, how does one get them into one sorted master file? It can be done with repeated editing in a text editor, and constant resorting, but there comes a time when one says "Enough! There must be a way to have the computer do this!"

My MASSREAD program was an earlier attempt to deal with this problem. It works well doing what it does (linking successive files to previous ones), but it requires a lot of editing if the end result is to be a complete, sorted composite. MASSREAD can not do any sorting by itself.

Can there be another, perhaps simpler, solution? What if you could simply run a program that would read two sorted files and produce a third output file tha tsummed them both in sorted order? Further, wouldn't it be nice if the program could eliminate duplicate records it found, thereby saving a lot of work? Programmers such as Jim Peterson or Earl Raguse have probably long since written such a program, but I have not heard nor seen of it.

After much trial and error, I found algorithms to handle this problem — they had eluded me for years. They are used in my program below, called MAKE1FROM2. The program processes approximately two records per second, which doesn't set any speed record when processing large files. Nevertheless, the results are well worth the time it takes. I am very happy to share this program with anyone able to use it.

MAKE1FROM2 reads two (sorted) D/V80 files and outputs a composite, sorted file. It takes items from each file and places them in order. It eliminates any duplication of records it finds. This does what MASSREAD could not. At last a program that can create a masterfile without requiring what can be an enormous and tedious effort sifting and patching bits and pieces of files. There is no worrying about text editor buffer limitations because the file has too many sectors. The only limit to the size of the masterfile composite you may make with this program is the limit of the number of sectors available on your disk.

MAKE1FROM2 reads two (sorted) D/V80 files and outputs a composite, sorted file.

Instructions are included in the program, with all input done through prompts. The most critical requirement for text files you want to use is that each one begin with a tilde (~) on the first line.—Ed.

MAKE1FROM2

```
1 REM [MAKE1FROM2] 11-13-96
By W. Leonard Taffs, SW99ers
Tucson, Arizona !093
2 !!131
3 ! To read two D/V80 files
and combine in sorted
order as a composite
output file. !155
4 !!131
5 ! The 2 files must have
been previously sorted
and include the tilde
"~" mark as first chr in
EOF rec. !012
6 !!131
7 ! OPEN #1:F1$
OPEN #2:F2$
OPEN #3:OF$
OPEN #5:"PIO" !166
8 !!131
9 ! NO CARRIAGE RETURN ADDED
!016
10 !!131
100 GOTO 150 !229
110 A,A1,B,B1,CL1,CL2,CTA,CT
AT,CTB,CTBT,F3,FCT,K,OF,OK,O
K2,PCT,PR,S,SK,U !090
120 A$,A1$,B$,B1$,D1$,D2$,D3
$,DT$,F1$,F2$,F3$,PR$,U$ !24
9
130 CALL CLEAR :: CALL KEY !
164
140 !@P- !064
150 CALL CLEAR :: DISPLAY AT
(1,8):"[MAKE1FROM2]": :" COM
BINE 2 D/V 80 FILES IN": :"
ALPHABETICAL ORDER!": :"B
y W. Leonard Taffs, SW99er"
!045
160 DISPLAY AT(9,3):"Both i
put files must be":"already
sorted and include a":"""~""
at the beginning of the":"l
ast record.": : :"Printer or
output file opts." !146
170 DISPLAY AT(19,1):"User (
0/1) response:": :TAB(5);"1=
YES";TAB(15);"0=NO" :: DISPL
AY AT(24,2):"Press <enter> t
o continue." !207
180 U$="User terminated prog
ram" :: PR$="Is printer onli
ne?" :: FCT=1 !211
190 CALL KEY(0,K,S):: IF S<1
THEN 190 !004
200 CALL CLEAR :: INPUT "Use
printer? (0/1) ":PR :: PRIN
T :: INPUT "Open out file? (
0/1) ":OF :: PRINT :: IF PR
THEN PRINT PR$ :: OPEN #5:"P
IO" :: PRINT #5: :!039
210 PRINT :: INPUT "Date? (O
pt.) ":DT$ :: PRINT :: CALL
CLEAR !021
220 PRINT :: INPUT "Read F1
NM$ ":F1$ :: PRINT :: INPUT
"F1 DSK: ":D1$ :: PRINT :: F
```

(See Page 33)

MAKE1FROM2 —

(Continued from Page 32)

```
1$="DSK"&D1$&"."&F1$ !013
230 INPUT "Read F2/NM$ ":F2$
:: PRINT :: INPUT "F2 DSK:
":D2$ :: PRINT :: F2$="DSK"&
D2$&"."&F2$ :: INPUT "O.K.?
(0/1) ":OK :: IF OK<>1 THEN
220 !240
240 PRINT :: IF OF THEN INPU
T "Save to/NM$ ":F3$ :: PRIN
T :: INPUT "Out F3 DSK: ":D3
$ :: PRINT :: F3$="DSK"&D3$&
"."&F3$ :: CALL CLEAR !016
250 DISPLAY AT(12,4):"In: 1
";F1$ :: DISPLAY AT(15,4):"I
n: 2 ";F2$ :: IF OF THEN DIS
PLAY AT(18,3):"Out: 3 ";F3$
ELSE INPUT "SURE? (0/1) ":OK
2 :: IF OK2<>1 THEN 220 !048
260 INPUT "SURE? (0/1) ":OK
:: IF OK<>1 THEN 220 !193
270 CALL CLEAR :: OPEN #1:F1
$,INPUT :: OPEN #2:F2$,INPUT
!045
280 IF OF THEN OPEN #3:F3$,O
UTPUT :: OF=1 ELSE OF=0 !249
290 DISPLAY AT(1,6):"~~ MAKE
1FROM2 ~~" !036
300 IF CL1 THEN GOSUB 660 EL
SE IF CL1 THEN 330 ELSE LINP
UT #1:A$ :: A=LEN(A$):: IF A
>1 THEN A=ASC(SEG$(A$,1,1)):
: DISPLAY AT(3,1):A$,A;CTAT
!220
310 IF POS(SEG$(A$,1,1),"~",
1)THEN CL1=1 :: CLOSE #1 ::
IF (CL1=1)*(CL2=1)THEN 590 !
018
320 IF (A$=A1$)*(A$=B1$)THEN
300 ELSE CTA=CTA+1 :: CTAT=
CTAT+1 :: FCT=FCT+1 :: IF CT
A=1 THEN A1$=A$ :: A1=ASC(SE
G$(A1$,1,1)):: CTA=0 !044
330 IF SK THEN 420 !064
340 IF CL2 THEN GOSUB 660 EL
SE IF CL2 THEN 300 ELSE LINP
UT #2:B$ :: B=LEN(B$):: IF B
>1 THEN B=ASC(SEG$(B$,1,1)):
: DISPLAY AT(6,1):B$,B;CTBT
!205
350 CALL KEY(0,K,S):: IF S<>
0 THEN 370 :: IF (K=81)+(K=1
13)THEN CLOSE #1 :: CLOSE #2
:: IF OF THEN CLOSE #3 ELSE
590 !158
360 CALL KEY(0,K,S):: IF S<>
1 THEN 360 !111
370 IF POS(SEG$(B$,1,1),"~",
1)THEN CL2=1 :: CLOSE #2 ::
IF (CL1=1)*(CL2=1)THEN 590 !
021
380 IF (B$=B1$)*(B$=A1$)THEN
330 ELSE CTB=CTB+1 :: CTBT=
CTBT+1 :: FCT=FCT+1 :: IF CT
B=1 THEN B1$=B$ :: B1=ASC(SE
G$(B1$,1,1)):: CTB=0 !086
390 CALL KEY(0,K,S):: IF S<>
1 THEN 410 :: IF (K=81)*(K=1
13)THEN CLOSE #1 :: CLOSE #2
:: IF OF THEN CLOSE #3 ELSE
590 !200
400 CALL KEY(0,K,S):: IF S<>
1 THEN 400 !151
410 SK=1 !086
420 IF (A$<B$)*(A1$<B1$)THEN
DISPLAY AT(11,1):F1$;" Prin
t #";FCT: :A$ !194
430 IF PR THEN IF (A$<B$)*(A
1$<B1$)THEN PCT=PCT+1 :: PRI
NT #5:TAB(5);PCT;A$ !077440
IF OF THEN IF (A$<B$)*(A1$<B
1$)THEN PRINT #3:A$ :: F3=F3
+1 :: DISPLAY AT(24,1):"Savi
ng: ";F3$;F3 !206
450 IF (A$<B$)*(A1$<B1$)THEN
300 !061
460 IF (A$>B$)*(A1$>B1$)THEN
DISPLAY AT(18,1):F2$;" Prin
t #";FCT: :B$ !205
470 IF PR THEN IF (A$>B$)*(A
1$>B1$)THEN PCT=PCT+1 :: PRI
NT #5:TAB(5);PCT;B$ !080480
IF OF THEN IF (A$>B$)*(A1$>B
1$)THEN PRINT #3:B$ :: F3=F3
+1 :: DISPLAY AT(24,1):"Savi
ng: ";F3$;F3 !209
490 IF (A$>B$)*(A1$>B1$)THEN
340 !103
500 IF (B$<A$)*(B1$<A1$)THEN
DISPLAY AT(18,1):F2$;" Prin
t #";FCT: :B$ !203
510 IF PR THEN IF (B$<A$)*(B
1$<A1$)THEN PCT=PCT+1 :: PRI
NT #5:TAB(5);PCT;B$ !078520
IF OF THEN IF (B$<A$)*(B1$<A
1$)THEN PRINT #3:B$ :: F3=F3
+1 :: DISPLAY AT(24,1):"Savi
ng: ";F3$;F3 !207
530 IF (B$<A$)*(B1$<A1$)THEN
340 !101
540 IF (B$>A$)*(B$ 1>A1$)THE
N DISPLAY AT(11,1):F1$;" Pri
nt #";FCT: :A$ !141
550 IF PR THEN IF (B$>A$)*(B
1$>A1$)THEN PCT=PCT+1 :: PRI
NT #5:TAB(5);PCT;A$ !079560
IF OF THEN IF (B$>A$)*(B1$>A
1$)THEN PRINT #3:A$ :: F3=F3
+1 :: DISPLAY AT(24,1):"Savi
ng: ";F3$;F3 !208
570 IF (B$>A$)*(B1$>A1$)THEN
300 !063
580 GOTO 300 !124
590 REM ** END PROGRAM ** !1
77
600 PRINT :"FILES CLOSED--EN
D OF PROGRAM": :!009
610 IF PR THEN PRINT #5: :TA
B(10);F1$;" had ";CTAT;" Rec
s.":TAB(10);F2$;" had ";CTBT
;" Recs. ";DT$ !055
620 PRINT "TOTAL READ: ";CTA
T+CTBT :: IF PR THEN IF OF T
HEN PRINT #5:TAB(15);"Outfil
e was: ";F3$;" ";F3;"Recs. "
;DT$ !104
630 IF OF THEN PRINT #3:"~ F
ile "&F3$&" "&DT$ :: CLOSE #
3 :: PRINT "Outfile was: ";F
3$:TAB(10);F3;" Recs." !085
640 PRINT :"FILES WERE:":F1$
,CTAT:F2$,CTBT :: IF U THEN
PRINT U$ !163
650 STOP !152
660 REM ** CHECK CLOSING **
!047
670 IF (CL1=1)*(CL2=1)THEN 5
90 !219
680 IF CL1 THEN 340 !018
690 IF CL2 THEN 300 !235
700 RETURN !136
```

Once you have created one composite of two D/V80 files, you can then use the composite as one of your two input files and create additional composite. By repeating this process for all of your files, you create the desired masterfile.

I have searched over the years for a program like this and never understood why none of my TI99 texts show any program even close to it, as far as I can see. I hope there are others in the TI community who can make use of this program. I'd appreciate hearing from any of you if you find it helpful.

MICRO-REVIEWS

AMS Slideshow, Loadmaster V2.2, Contract Bridge V4.0

By CHARLES GOOD

AMS SLIDESHOW
by Bruce Harrison

This is the second software product from Bruce designed for the extra CPU memory capabilities of the AMS card. The program is commercial and costs $7 directly from Bruce. There is a much slower public domain version of this (which I can send you for $1) for those without an AMS card. Other AMS software products from Bruce are currently in the thinking stage.

AMS Slideshow lets you load a bunch of TI-Artist pictures into memory and then display them one at a time in a slide show format. The common 256K AMS card will hold 20 pictures and if your card is upgraded to 512K you can load up 41 pictures including any combination of color or black-and-white TI-Artist pictures. These are loaded one at a time off of disk in the order that you want to display them. You can load the same picture several times if you want that picture to appear more than once in a sequence of pictures.

Once your pictures are all loaded you get to specify how many cycles you want to display, in other words, how many times you want to display the same sequence of in memory pictures. You are asked if you want to change pictures automatically or with a keypress. If you want automatic display you are asked for a display time length. This is the amount of time a picture stays on screen before the next picture is fetched from memory and displayed. This interval can be as little as 0.1 second. Actually it takes a little longer than 0.1 second to print a picture to the screen, so if you select an ultra short display interval your computer will spend more time changing pictures than it does displaying them. Changing pictures is done with a top-to-bottom wipe. With a set of slightly altered pictures and a very short display interval it is possible to generate screen animation.

The main advantage of AMS Slideshow over the non-AMS public domain version of the same thing is speed. The non-AMS software must fetch each picture off of disk each time that picture is displayed. Even if this is done off of a RAMdisk or hard drive the process is not as fast as getting the picture from bank-switched AMS memory.

This is a quality useful product. It is commercial and available only from Bruce, for $7. If you willingly paid your $100 for a 256K AMS card then you should be willing to fork over an additional $7 for good software that uses the card.

LOADMASTER V2.2
by Mickey Cendrowski and Bruce Harrison

Yes, I know I reviewed this twice already in MICROpendium. But it keeps getting better, much better. For those of you who don't remember, Loadmaster is for those wishing to organize their disk collections and find out what kind of software is on each disk. When you ask Loadmaster to bring up a disk directory you are told all the usual disk directory information such as file name, size, kind (program, DV80, etc.) and protection status. You are also told the file kind in a way that is not often possible to guess from an ordinary directory listing. You are told "Infocom game," "TI-Base data file," "Page Pro Banner," "Extended BASIC long file," etc., for every file on the disk. There are 52 file types that can be identified. This is very complete disk information not available from any other TI utility product. You can print disk labels, disk jackets, and create disk files that contain all this information.

New to v2.2 since my last review of v2.1 are speed, the ability to run any runnable file, more extensive and accurate file identification, and user-friendly error trapping. Bruce Harrison has added his assembly language skills to Mickey's Extended BASIC earlier versions so that Loadmaster can now do things that cannot be done with just Extended BASIC.

The most immediately obvious change is speed. In my previous review I noted that the entirely extended basic v2.1 was very slow. Version 2.2 really flies along. Code loads and executes quickly and screen refreshes are kept to a minimum. Sometimes if only half the screen needs refreshed then that part of the screen only is redone.

Loadmaster requires Extended BASIC and loads from an XB program called LOAD, but you don't have to boot it from DSK1. The product has boot disk tracking and will load from any floppy or Horizon RAMdisk drive. It also boots properly from the root directory (only) of either a SCSI or HFDC hard drive with the included Extended BASIC loader. Just put all the files on one drive and run LOAD. When it loads press C to catalog a drive. If you don't like the default drive press another number within 3 seconds to catalog that drive. After 3 seconds the default drive is automatically cataloged. This default can be changed from an options menu. Loadmaster will catalog any floppy drive, but not Horizon RAMdisk drives if the Horizon RAMdisk is not at CRU address 1000.

The catalog provides you with file identification information that is much more accurate and extensive than previous Loadmaster versions. All program files are now properly identified, which can only be done by reading file headers using assembly language support. TI BASIC/Extended BASIC, Extended BASIC Only, Editor/Assembler Option 5, Adventure, Tunnels of Doom, TI-Artist, and other types of Program files are all labeled as such. The inability to correctly distinguish between the different types of program files characteristic of earlier Loadmaster versions has been fixed using assembly language readings of file headers.

If "*" appears next to a file name in a

(See Page 35)

MICROREVIEWS —

(Continued from Page 34)

Loadmaster disk directory, this means Loadmaster can do something with the file. Move the cursor next to the file and press enter. DV80 text files are viewed with a word wrap 40-column display. Not all DV80 files are text files, and Loadmaster knows this! TI BASIC, Extended BASIC, EA5, and EA3 files can all be run. Just press enter and the file runs.

In the case of EA3 files, if there is only one entry name then the file starts running. If there are several entry names then these are displayed. You put the cursor next to the most promising of these entry names, press enter and see if the software starts properly. If not, you can try one of the other displayed names. Running EA3 software is much easier with Loadmaster than it is with the EA module or with Funnelweb, the only other ways I know to run such software. You don't have to remember the mysterious and often undocumented startup name needed for the EA module, and you don't have to first link the screen as is the case with Funnelweb. In starting assembly software Loadmaster clears the screen before running the software. Some assembly software does not clear the screen at the start of the assembly program. Such software produces a messy screen with other assembly loaders, but not with Loadmaster v2.2.

This is a really first-class product. Its main use for most of us will be printing disk labels or disk jackets with complete file information. The new version is also the best software I have seen for running software from a disk directory, better even than Funnelweb's Disk Review. All 99/4A users should evaluate this unique product. It is fairware. The title screen tells you where to send your "whatever you think it is worth" donation, which will be divided equally between the two authors. Send me $1 and I will send you the latest version. If you have an earlier version you should definitely upgrade.

CONTRACT BRIDGE V4.0

by John Bull

This was initially released about five years ago and reviewed at that time in MICROpendium. This new update has several new and improved features and plays a much more intelligent game than the original.

This software is sort of the Bridge card game equivalent of TI's Video Chess module. Both Bridge and Chess are games of skill rather than luck. Strategy, tactics and experience count for everything. Both Contract Bridge and the Video Chess module will play a good medium-level but not an expert-level game. In both cases the software can serve as a teacher/tutor of game strategy, in both cases the human and computer player can switch places, and in both cases you can set up and play specific game positions such as those obtained from newspaper articles. Other similarities exist.

When you start Contract Bridge you are given the options of Tutor, Rubber Bridge and Duplicate. The tutor is exactly that, it teaches you how to play the game of bridge in general and specifically how to use the Contract Bridge software. The rules of play and some suggested general tactics are presented. Then the learner is presented with a trial game with all the cards exposed. The learner bids and plays all four hands with no limitations except that the computer will not allow an illegal bid or card play.

Rubber Bridge lets one player play an ordinary game of bridge against the computer and see only his card hand. The human player plays south and also the Dummy hand if you win the initial bidding. Computer play is much more intelligent than in previous versions of Contract Bridge. The author considers himself a medium-grade bridge player and tells me that his program now beats him about one-third of the time. I am not a bridge player, so I cannot personally comment on the quality of the computer's play. The author believes that playing against the computer is now at least mildly challenging for experienced players. In previous versions, the author says, the computer often made dumb bids and bad plays.

(See Page 36)

MICROREVIEWS —

(Continued from Page 35)

The Duplicate option allows you to play with predetermined sets of hands, or boards. The game comes with 50 of these, each of which has already been played by the author. The object here is to beat the scores made by the author or others who have played with these same sets of hands. You can also set up your own custom boards and play them, such as those published in newspapers and magazines.

The screen display has the playing table in the center, the four hands on the NSEW side of the table, and bidding and score information at the top of the screen. In Rubber you see only your own hand unless you specifically check and ask for a (L)ook. You can do an assembly language screen dump to a printer and you can save hands as a new board. The software can keep track of 99 boards including the 50 that already come with the software. From within the program a directory of boards files is available. In Rubber you can change the hand you play by shifting all hands at once clockwise. This shift can be repeated as often as desired.

Contract Bridge is expensive shareware. I can send it to you on a DSSD disk to evaluate for $1. The registration fee is $20. That is a lot of money, but look what you get! There is nothing else like this in the TI world, and maybe not in other computer formats either. If you don't know how to play bridge you can learn from this software. If you do know, or once you learn, Contact Bridge will play an intelligent game with you with the emphasis on "intelligent." The author states that bridge is the most popular card game in the world and feels that many of the younger generation are missing out on the social pleasure and mental stimulation of the game. Tutor will teach you and Rubber will give you that pleasure and stimulation.

ACCESS: John Bull (Contract Bridge), 409 Blue Valley Lane, Knox TN 37922, e-mail bulljh@delphi.com

Bruce Harrison (AMS Slideshow), 5705 40th Place, Hyattsville, MD 20781. Phone 301-277-3467.

Mickey Cendrowski (Load Master v2.2), 100 Pine St., Russellton, PA 15076.

Charles Good (your humble reviewer), P.O. Box 647, Venedocia, OH 45894. Phone 419-667-3131. E-mail good.6@osu.edu Visit my Web site for the May 1997 TI MUG Conference at www.bright.net/ cgood/mug1997.html.

TI has 'official' Web site for 99/4A

Texas Instruments has an official site for the TI99/4A on its home page on the World Wide Web. Users can access the site through the Internet at http://www.ti.com/calc/docs/994a.htm.

NEWSBYTES

Tesch releases CYA as fairware

CYA, by Tim Tesch, is now fairware. The program, which works with the Geneve, is now at version 3.0.

The program was originally released commercially through Cecure Electronics Inc. The fairware offering is offered through S&T Software. Tesch said that users should pay no more than $2 for a copy of the program, no matter what the source. This amount should include shipping, handling, and media charges.

To receive a printed, laser-quality manual, or to show support, users may send donations to Tesch at S&T Software, 1856 Dixie Rd., Port Washington, WI 53074. He can also be reached via email at t.tesch1@genie.geis.com.

Tesch says several bugs have been fixed in CYA so that it will now properly reset the video blinking attribute region and will turn off the blink register. Programs such as Clint Pulley's QDE will look much better, he says. The program now supports MDOS 2.21, 4.00 and 5.00.

Tesch also noted that he plans to release two new programs in the near future. No details were available.

Suncoast defects

The Suncoast User Group of St. Petersburg, Florida, originally a TI users group and later a joint TI-IBM group, has now become an all-IBM users group, according to Ray Moglia, newsletter editor for the group.

Harrison adds Slideshow for AMS

Bruce Harrison has introduced another software product for use with the AMS (Advance Memory Systems) card, called AMS Slideshow. It uses TI-Artist Picture files, some samples included. According to Harrison, the pictures get loaded into the AMS memory, and thus changes of slide are nearly instant.

Two versions are available. For U.S. and Canadian users, the cost is $5 including shipping and handling. The European version, for 50-Hz [illegible] systems, sells for $7 including shiping and handling.

TI user finds Hamfest a bonanza for TI gear

By BOB CARMANY

Some years ago, I read that the TI was one of the best-selling computers ever made. There have been estimates of almost 2 million units being sold before the inventories ran out after "Black Friday." The fact is, they still keep showing up in some of the most unlikely places.

Every year around Thanksgiving, there is a large hamfest here in Greensboro, North Carolina. Since the mid-'80s about half of the floor space has been taken up by all sorts of computer supplies. This year, I decided to go and visit just out of curiosity — and the prospect of a deal or two on some electronics tools. As I wandered along the aisles of tables, I caught a glimpse of a TI console in the original packaging. The guy who was selling it had pulled it out of his garage, dusted it off and thought he might be able to sell it. The price was a paltry $5. Even though I have enough consoles to last well into the 22nd century (that's right, 22nd), I considered buying "just one more." Fortunately, reason set in and I passed the bargain by to look for something else that I could use.

The place was a virtual cornucopia of TI-compatible gear. There were tables and tables of floppy drives — everything from the old SSSD Shugarts to brand new 3.5-inch drives still in the shrink wrap. The prices were amazing! Some of the 3.5-inch 1.44-meg drives were going for as little as $15.

The prize of the show was a brand new Fujitsu SCSI II 670-meg drive for $75! The vendor had what appeared to be 50 or so of them. Lesser-sized SCSI drives were there as well for far less than one would pay for them at a computer store. In fact, just about any SCSI device that you could ever think of attaching to your Western Horizon SCSI controller was there, and all of them for next to nothing in price.

One vendor had a wide array of electronics parts for "less than Radio Shack" prices. Most any chip that you wanted could be found along that aisle. I picked up a couple of 9901s and a 9918 for a total of $7. I also managed to get a couple of 14-pin component carriers for a quarter each to make personality plugs for my eprommer. Besides that lot, I got a couple of hemostats and pliers that I needed. If you are into hardware hacking, the place was a treasure trove of opportunity.

About the only chips that I didn't run across were static RAMs. I couldn't find any 6264 (8K) or 62256 (32K) chips anywhere. I also didn't run across any 512K static RAMs to use to upgrade my AMS card to a full meg. I have seen all these at some other hamfests, though.

As I wandered around, I saw all sorts of other gear that could be used with my TI system. If you had one of the WHT AT-interfaces, there were keyboards for $5 still in the original packaging. The spectrum of joysticks was equally impressive. Almost any Atari-type joystick that you could imagine was there. There have been several articles in MICROpendium about building a cable, and for $5 or so you could have a set of joysticks to go with it.

I ended up spending a couple of hours walking around and looking, but I'm sure there was something I missed that I would have bought had I seen it. An analog RGB monitor would have been nice, or even one of the old Magnavox monitors, but I must have missed them somewhere amongst all the IBM gear.

I guess in the final analysis, I spent a rainy afternoon browsing through a virtually endless array of gear that could be used with your TI system and found a few bargains that appealed to me. Charles Good was wrong in his review of the Geneve — you can get the gear to set it up for far less than he paid for it. I would encourage all of you to get out to your local hamfest — there are probably some deals waiting for you there!

USER NOTES

TI secrets

The following is excerpted from Tigercub Tips Vol. 3.—Ed.

Here are some dark secrets Texas Instruments didn't tell us. The User's Reference Guide claims that the computer can produce frequencies up to 44722 Hz., "well above human hearing limits." It then admits that "the actual frequency produced may vary from zero to 10 percent depending on the frequency."

According to Jim Hindley, the highest frequency actually produced is 37287, which is certainly not above the hearing range of some humans, but neither is 44733. The maximum error rate far exceeds 10 percent because any frequency you call for from 31953 to 43733 ends up exactly 37287. Not to worry, the frequencies in the normal range of music are accurate enough and your TV speaker probably can't reproduce frequencies above 20000 anyway.

And did you know that TI really gave us only 15 volumes, not 30? Listen and count them using this program:

```
100 FOR V=0 TO 29 STEP 2
110 CALL SOUND(1000,500,V)
120 CALL SOUND(1000,500,V+11
)
130 FOR D=1 TO 500
140 NEXT D
150 NEXT V
```

And the duration values are just as inaccurate. Experimenting with a series of eight CALL SOUNDs in a loop repeated 100 times, I found that execution time was 40 seconds for any duration between one and 49, or a negative duration; 54 seconds for any duration between 50 and 66; 67 seconds between 67 and 83; 80 seconds between 84 and 99; 94 between 100 and 116; 106 between 117 and 133, and so on.

Accessing keyboard interface memory

The following item appeared on comp.sys.ti on the Internet. It was written by David Neiters in response to a question from Dan Eicher regarding methods of accessing memory on a Western Horizon Technologies keyboard interface. It may be useful to other TIers who have a WHT keyboard interface.—Ed.

Accessing the memory in the WHT keyboard is done with CRU instructions. The base CRU address for the AT keyboard interface is located at >400. The memory is banked in at addresses 0000-1FFF (8K) and replaces the console ROM.

You can swap between EPROM and optional RAM on the interface by setting bit 2. Setting this bit to 0 selects the EPROM. Setting this bit to 1 selects the optional RAM.

There are eight banks of ROM and eight banks of RAM. The bank selection is controlled by bits 12-15 for RAM and 8-11 for ROM. You may notice that there are more bits than are needed to select eight banks. Don O'Neil left room for future expansion to double the amount of memory on the card.

Anyway, lets say you want to select bank two of the RAM. Your code would look something like this:

```
LI R12,>400
SBZ 2 Turn on the RAM
LI R12,>400+24
LI R1,>0200 Select RAM bank
LDCR R1,4
```

When you are done, you want to enable bank 0 of the EPROM again.

```
LI R12,>400
SBO 2 Turn on EPROM
LI R12,>400+16
CLR R1Select ROM bank 0
LDCR R1,4
```

Color blending in XBASIC

The following item was written by Janice E. Brooks of the Dallas TI User Group.

Have you ever felt limited by the mere 15 colors in the standard TI99/4A? Ever wished you could coax the system into producing more? Well, you can, and it's actually pretty easy. I'm not talking about fancy bit-mapped graphics or assembly programming — just a simple BASIC or XBASIC routine.

How does it work? Just by redefining a given character in a checkerboard pattern, assigning different foreground and background colors, and printing a block of the character with the HCHAR function. The idea is old hat to skilled programmers. I saw the underlying routine in one of Steve Davis' Programs for the TI Home Computer books, which used blended colors to suggest various landscapes, such as tan (gray and light yellow) for a desert.

I've seen the effect in some fairware programs as well, but the concept was so striking to me that I figured there had to be other comparative novices out there who would be interested to see how color blending works. So I wrote this Extended BASIC program to display all the possible combinations. The bottom of the screen shows the various blended colors while the top keeps track of the colors used to make the blends.

Over 100 new colors are made possible by blending the original ones. Unfortunately, you can use only a few colors at a time in BASIC or Extended BASIC graphic programs, since you have to use CALL COLOR on whole character sets. At least some new options are available. The colors look a bit grainy, but some are quite nice. I'm especially partial to the new shades of purple. Be careful with the spacing of the colons when keying in the program so Extended BASIC knows when you mean "multiple statement line" and "skip a print line."

COLORBLEND

```
50 !Public domain, By Janice
Brooks !123
100 CALL CLEAR :: DIM A$(16)
:: FOR R=2 TO 16 :: READ A$(
R):: NEXT R !130
110 PRINT TAB(12);"BLEND": :
:"This program cycles throu
gh the 120 colors made possi
bleby blending the 15 standa
rd TI colors." !000
120 PRINT :"Press any key to
continue." !198
130 CALL KEY(0,K,S):: IF S=0
THEN 130 !198
140 CALL CLEAR :: CALL SCREE
N(16):: CALL CHAR(128,"AA55A
A55AA55AA55")!050
150 FOR I=2 TO 16 :: FOR J=
TO 16 :: IF I>J THEN 200 !2
53
```

(See Page 39)

USER NOTES

(Continued from Page 38)

```
160 DISPLAY AT(1,1):A$(I);"(
";STR$(I);")" :: DISPLAY AT(
2,3):"and" :: DISPLAY AT(3,1
):A$(J);" (";STR$(J);") "
!016
170 DISPLAY AT(5,3):"<Press
any key>" !143
180 CALL COLOR(13,I,J):: CAL
L HCHAR(12,1,128,384)!076
190 CALL KEY(0,K,S):: IF S=0
THEN 190 !002
200 NEXT J !224
210 NEXT I :: CALL CLEAR ::
INPUT "REVIEW AGAIN (Y OR N)
? ":G$ :: IF G$="Y" OR G$="y
" THEN 140 ELSE STOP !128
220 DATA BLACK,MEDIUM GREEN,
LIGHT GREEN,DARK BLUE,LIGHT
BLUE,DARK RED,CYAN !162
230 DATA MEDIUM RED,LIGHT RE
D,DARK YELLOW,LIGHT YELLOW,D
ARK GREEN,MAGENTA,GRAY,WHITE
!033
```

ogram outputs screen image to printer

The following program was written by Stephen Shaw of the TI99/4A user group in England.

Many printers let you download a screen display using 8-pin bit image mode. Most of the Epson-compatible printers support this. The following program in TI BASIC does this.

Because the printer defines its characters vertically, but the computer defines its characters horizontally, this program will produce a picture of the screen on its edge. It is easier to do this than to rotate the image.

If the screen contains only text, there is no advantage to using this program: use GCHAR to obtain the characters and print them one row at a time.

This program will print the characters as they appear on the screen, in black only. "On" pixels are printed. "Off" pixels are not. Sprites are not copied.

PRNTSCREEN

```
100 OPEN #1:"PIO.CRLF" !087
110 REM OR EQUIVALENT RS232
FILE NAME !086
120 PRINT #1:CHR$(27);"A";CH
R$(8)!115
130 FOR [A=1 TO 32 !192
140 PRINT #1:CHR$(27);"K";CH
R$(192);CHR$(0)!211
150 FOR [B=24 TO 1 STEP -1 !
048
160 CALL GCHAR([B,[A,[CHAR)!
084
170 IF [CHAR<33 THEN 300 !14
6
180 CALL CHARPAT([CHAR,DEF$)
!251
190 IF DEF$="000000000000000
0" THEN 300 !178
200 FOR [SEG=16 TO 2 STEP -2
!208
210 [HEX=ASC(SEG$(DEF$,[SEG-
1,1))!213
220 GOSUB 430 !255
230 [PRINTDEF=[HEX !181
240 [HEX=ASC(SEG$(DEF$,[SEG-
1,1))!213
250 GOSUB 430 !255
260 [PRINTDEF=[PRINTDEF+[HEX
*16 !033
270 PRINT #1:CHR$([PRINTDEF)
;!246
280 NEXT [SEG !208
290 GOTO 340 !164
300 FOR []=1 TO 7 !173
310 PRINT #1:CHR$(0);!056
320 NEXT [] !078
330 PRINT #1:CHR$(0)!132
340 NEXT [B !051
350 PRINT #1:CHR$(13);CHR$(1
0)!219
360 NEXT [A !050
370 PRINT #1:CHR$(27);"@" !1
22
380 PRINT #1:CHR$(7)!139
390 CLOSE #1 !151
400 REM NOW GO WHERE YOU WIS
H !119
410 REM USING RETURN OR GO T
O AS APPROPRIATE !149
420 STOP !152
430 [HEX=[HEX-48+([HEX>64)*7
!091
440 RETURN !136
450 END !139
```

program goes here

To permit the routine to be added to any program, the square bracket has been used in front of each variable — [B, for example. The bracket is accepted as a valid character in variable names.

ESC K ((CHR$(27);"K") is used in Epson printers, and some others, to select normal density, 8-pin bit image mode.

CLASSIFIED

FOR SALE

GENEVE 9640

With keyboard, MY-Word, MY-Art, Advanced BASIC, About 50+ disks full ames and Geneve programs worth $100+. Buy it all for $175. Roger, 317 664-6401, 10 a.m.-2 p.m. v14n1

DISK CONTROLLER AND MORE

CorComp disk controller, $75; TI RS-232 card, $50; Signalman Mark XII (1200 baud) modem, $7; CorComp Triple Tech card, $25; Commodore 1702 composite color monitor (works with TI). Call 512-255-1512, email jkoloen@io.com.

Devoted to the TI99/4A since 1984

Subscription Fees

❑ 6 issues, USA, **$35**

❑ 6 issues, Mexico, **$40.25**

❑ 6 issues, Canada **$42.50**

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

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

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

Address Changes

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

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

Check/MO ❑ ❑ ❑ (check one)

Card No. _____

Expiration Date _____

(Minimum credit card order is $9)

Signature _____

(Required on credit card orders.)

No sales tax on magazine subscriptions. Texas residents add 7.75% sales tax on other items, including back issues and disk subscriptions. Credit card orders add 5%.

Name _____

Address _____

City _____

State _____ **ZIP** _____

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

Disks, Etc.

❑ **Back Issues**,$3.50 each to March 1996, later $6 each. List issues: _____

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

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

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

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

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

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

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

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

GENEVE DISKS (SSSD unless specified)

❑ **MDOS 2.21** (req. DSSD or larger (for floppy & hard drive systems) $4.00

❑ **GPL 1.5** $4.00

❑ **Myarc Disk Manager 1.50**.....$4.00

❑ **Myarc BASIC 3.0**.....$4.00

❑ **MY-Word V1.21** $4.00

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

GENEVE PUBLIC DOMAIN DISKS

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

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