Covering the TI99/4A and the Myarc 9640

MICROpendium

Volume 11 Number 6 | July 1994 | $3.50

Arranging lists into columns

◆ ◆ ◆

Resequencing and rearranging fields

◆ ◆ ◆

Option 5 revisited

◆ ◆ ◆

Anatomy of a header file

◆ ◆ ◆

Memory defined

◆ ◆ ◆

Using mail merge in TI-Writer

◆ ◆ ◆

Reviews of AEMS Macro Assembler/Linker, DORTIG Disk #2, Kanji Drill, P-GRAM Utilities Ver. 2

CONTENTS

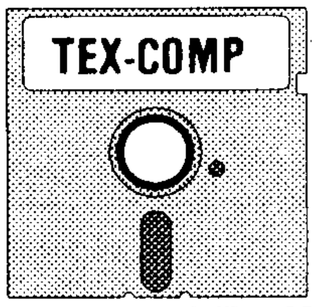
MICROpendium

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

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

*READ THIS

Here are some tips to help you when entering programs from MICROpendium:
1. Most BASIC and Extended BASIC programs are run through Checksum, which places the numbers that follow exclamation points at the end of each program line. Do not enter these numbers or exclamation points. Checksum is available on disk from MICROpendium for $4.
2. Long Extended BASIC lines are entered by inputting until the screen stops accepting characters, pressing Enter, pressing FCTN REDO, cursoring to the end of the line and continuing input.



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

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

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

GRAPHICS, MUSIC & ANIMATION
#1 THE SINGING TI VOL. 1 (S)(2)
#4 PRINTART (2)(P)
#5A MUSIC/GRAPHICS
#6 EXBASIC MUSIC (2)
#7 SPACE SHUTTLE MUSIC/GRAPHICS
#9 MONA LISA PRINTOUT (P)
#11 ANIMATED XMAS (WOODSTOCK)
#14 FIGURE STUDIES (P)(PG)
#32 EXBASIC XMAS MUSIC (2)
#41 VIDEO GRAPHS (M)
#52 ANIMATION 99' (2)
#69 PLAYER PIANO/KEYBOARD ANALYSIS
#93 KBGB GIRLIE CALENDAR (4)(P)
#103 SORGAN THE TI ORGAN
#107 STARTREK MUSIC ALBUM
#111 POP MUSIC & GRAPHICS
#114 PANORAMA
#115 GRAPHICS DESIGN SYSTEM
#120 BITMAC (2)(P)
#230 THE SINGING TI VOL. 2 (S)(2)
#231 THE SINGING TI VOL. 3 (S)(2)
#246 THE SINGING TI VOL. 4 (S)(2)
#313 3-D WORLD (CAD for the TI)

COMPUTER UTILITIES, PRINTER UTILITIES & PROGRAMMING LANGUAGES
#3 DUMPIT (E/A)
#15 STAR/EPSON PRINTER DEMO (P)(2)
#16 SIDEWAYS PRINTOUT (P)(2)
#18 TI DIAGNOSTIC (MM)(2)
'28 LOADERS & CATALOGERS
#30 HOUSEHOLD BUDGET PRINTOUT (P)
#35 PROGRAMMING AIDS & UTILITIES I
#42 FUNNELWEB FARM (SHELL UTILITY)
#53 HACKER CRACKER
#55 SCREEN DUMP (P)
#62 DISK MANAGER II (M)
#75 DISK CATALOGER
#76 PROGRAMMING AIDS & UTILITIES II
#78 ARTCON GRAPHIC CONVERSION
#79 DISK MANAGER 1000
#80 BIRDWELL DISK UTILITY (2)
#85 AUTOBOOT UTILITY
#86 COLUMN TEXT III (P)
#87 ARCHIVER III
#89 PROCALC DECIMAL/HEX CONVERTER
#96 STATISC & SORT ROUTINES
#97 MEMORY MANIPULATOR
#101 ENHANCED DISPLAY PACKAGE
#108 FUNLPLUS (PLUS!)
#110 DISK+ AID
#117 UNIVERSAL DISASSEMBLER
#119 RAG LINKER CONVERSION
#127 PIX GRAPHICS UTILITY
#243 OS/99 (GD)
#251 PC TRANSFER TI/IBM (DD)
#253 THE EXPLORER/DM1000
#254 NIBBLER/TURBO
#260 TI FORTH (DISK ONLY- add $8 for manual)
#17 TI FORTH DEMO
#116 TI FORTH TUTORIAL
#5079 FORTH SOURCE CODE
#104 C99 COMPLIER & LIBRARY
#5007 TEACH YOURSELF TI BASIC
#5019 TEACH TOURSELF EX-BASIC
#5067 BEGINNING BASIC TUTOR
#307 GRAPHICS CODE GENERATOR
#3912 SUPER BUGGER
#3913 BETTER BANNERS
#3914 CERTIFICATE 99
#3915 HOROSCOPE MAKER
#3916 GRAPHX+ PRINT SHOPPE
#3917-3720
GRAPHX COMPANIONS 1-4
#3721 MAC FLICK (G)
#3722 PRINTING TO GO (G)
#3723 GRAPHX DINOSAURS (G)

BUSINESS, ACCOUNTING, WORD PROCESSING, DATA BASE HOME OFFICE & HOME
#10 GOTHIC PRINTOUT (P)
#19 TI WRITER/MULTIPLAN UPGRADE
#20 ACCOUNTS RECEIVABLE (2)
#21 DATA BASE DEMO
#23 WILL WRITER
#29 LABEL MAKER I (P)
#36 STRICTLY BUSINESS (2)
#56 SPREAD SHEET
#58 PR BASE (database)
#59 GRAPH MAKER
#74 LABEL MAKER II (P)
#77 MICROdex 99 (database)
#81 HOME ACCOUNTING SYSTEM
#83 HOME APPLICATION PROGRAMS (2)
#90 JET CHECKBOOK MANAGER
#92 HOUSEHOLD INVENTORY
#109 TI WRITER MINI MANUAL
#112 INVOICE PACK
#113 LABEL MAKER III
#129 CASH DRAWER (point of sale)
#130 THE ORGANIZER
#147 CALENDAR-NOTEPAD
#177 HOUSEHOLD BUDGET MANAGEMENT (M)
#30 HBM DATA PRINTOUT (P)
#221 PERSONAL REAL ESTATE (M)
#249 MAPMAKER
#252 99 WRITER II (TI WRITER)(P)
#253 AMA MAILING LIST
#255 TRAK-A-CHECK
#257 DAILY DIARY
#257A EXER-LOG
#23 WILL WRITER
#310 SELF HELP TAX CUT

TELECOMMUNICATIONS (MODEM)
#57 TELCO
#57D TELCO (FOR SYSTEMS WITH DS DRIVES)
#118 FAST TERM

EDUCATION & PERSONAL DEVELOPMENT
#22 ASTROLOGY
#24 ENGINEERING CALCULATIONS (2)
#25 MEDICAL ALERT
#27 KIDS LEARNING I (2)
#31 MORSE CODE TRAINER
#37 LAPD COOKBOOK (2)
#40 ARTIFICIAL INTELLIGENCE (ELIZA)
#54 ASTRONOMY
#66 HEBREW TYPEWRITER
#67 GENEALOGY (2)
#71 KIDS LEARNING II (2)
#95 WEATHER FORECASTER +
#138 FIREHOUSE COOKBOOK
#142 TOUCH TYPING TUTOR (M)
#184 FACE MAKER
#194 ST. NICK/GHOSTLY SPELLING (M)
#195 TINY LOGO
#199 MILLIKEN ADDITION (M)
#200 MILLIKEN DECIMALS (M)
#203 MILLIKEN FRACTIONS (M)
#204 MILLIKEN INTEGERS (M)
#205 MILLIKEN LAWS OF MATH (M)
#211 MIND CHALLENGERS (M)
#212 MINUS MISSION (M)
#213 MILLIKEN PERCENTS (M)
#214 STORY MACHINE (M)
#215 BEGINNING GRAMMAR (M)
#216 METEOR MULTIPLICATION (M)
#217 HANGMAN (M)
#222 MUSICMAKER (M)
#223 PHYSICAL FITNESS (M)
#225 ALIEN ADDITION (M)
#226 ALLIGATOR MIX (M)
#227 DEMOLITION DIVISION (M)
#228 DRAGON MIX (M)
#270 TRIGONOMETRY
#271 CALCULATORS & CONVERSIONS (2)
#272 HIGHER MATH (2)
#273 ASTRONOMY 2
#306 SPEAK & SPELL II (S)(EX)

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

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

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

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

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

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

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

COMMENTS

TI Emulator! off the boards

Texas Instruments has pulled the plug on one of the most exciting programs developed for the TI in recent memory. Edward Swartz, a young programmer from Texas, developed a program called TI Emulator! and released it as shareware on electronic bulletin boards. Only a matter of months latter, TI put its foot down and told Swartz to stop distributing the program because it included the content of some of its ROMs.

The ROMs are TI property and apparently are available for use under licensing agreement. That's how CaDD Electronics does it, which also developed a TI emulator called PC99 that works on a PC. The cost of the licensing does not seem to be exorbitant, but the hassle of doing it and keeping records probably isn't worth the trouble for someone like Swartz.

It doesn't strike me that TI acted unreasonably. The company is simply trying to protect its patent and copyright. Obviously, they didn't do it to put the squeeze on a young programmer. If they knowingly let Swartz slide, they'd surrender their rights, rights which probably cost millions of dollars when you consider how much the company spent to develop software for the TI. And we're not just talking about modules here. This in cludes the console ROM and GROM as well.

It is unfortunate, however. Swartz had developed a reall fine program that brought excitement to a lot of wizened T veterans.

NEW HFDCs AVAILABLE

Don Walden of Cecure Electronics has informed us that h has received a shipment of new Myarc Hard and Floppy Dis Controllers from Lou Phillips. The HFDCs upgraded for Gen eve will sell for $250 while the standard HFDCs sell for $23(He also says TIm Tesch will soon release a program calle CYA (configure your autoexec). It will allow you to preconfig ure System/Sys with any remaps you normally do in your au toexec, allowing users to preassign your drive and pre-rema where your devices will be, and define your prompt ahead c time, screen colors and default drive and other things, Walde says. Cecure will distribute the commercial program. Walde says it should sell for under $15.

—Jl

READER TO READER

❑ Martin Marcinko, 3628 Chalet Ct., Orlando, FL 32818, writes:

I need some help in getting some information on the PHP 2300 video controller that I have. I need the cable for the remote cable connector on the right side of the controller that goes to the remote control on the videocassette recorder. This cable works only with the particular brand. If I can get one of these cables I will find the videotape player for that cable.

❑ Jerry Clasby, 612 Meandering Rd., Frederick, OK 73542, writes:

I've just come upon a printer buffer I hope will work with the TI99/4A to Gemini connection. It is a buffer from Global, model C4684. The jacks are marked with "parallel output" and "parallel input." However, there is an 8-pole dip switch on the bottom of the box which I believe is baud rate for a serial buffer, right? I tried the buffer's self test (which also says that it is a parallel buffer) and the printer responded correctly. Would a serial buffer do that?

Also, I seem to remember reading somewhere that Texas Instruments swapped some wires to make their system incompatible with other systems of the day. If so, can someone enlighten me so that if I do come across a parallel buffer I could make the changes?

❑ Joanne Corney, 301 Country Club Circle, Midwest City, OK 73110, writes

The TI Sooner 99ers User Group has undertaken to supply special education teachers with TI99/4A computers and educational cartridges. We have received donations of equipment to do this. The biggest problem is lack of TV/monitors.

We are in possession of three 13-inch TI99/4 color monitors, Model No. DC-13 PF4, made by Zenith in 1980. Other numbers appearing on the back of one are Serial# 9655660, Run# 916A045, 31A088. Each has a different trouble that prevents its use. We would like to have a set of schematics for this monitor. TI CARES no longer has them, so I hope someone, somewhere in the TI community, does have access to a set so that these can be repaired and put to good use.

❑ This copy of a reply to Michael Scheller's request in Reader to Reader was sent to us by the B.C. 99er User Group, c/o Ron Warfield, 216 10th Ave., New Westminster, BC, Canada, V3L 2B2:

We saw your letter in the MICROpendium, and wish to introduce ourselves. We are the B.C. 99er User Group from the Vancouver area of British Columbia, Canada. Our group is very active and meets every Thursday. We have about 35 members and exchange newsletters with 16 other groups. If your group would like to exchange newsletters please let us know.

We would also extend the invitation to other groups to exchange newsletters. The B.C. 99er User Group also runs a BBS called DIAL A TI which is a modified version of Paradigm

(See Page 5)

FEEDBACK

Cecure goes above and beyond

It isn't often that we give credit and praise to people from whom we purchase products or services even when they have gone beyond what is normally expected of them.

When the United Parcel Service botched up a delivery of an item I was trying to purchase from Cecure Electronics, I contacted Cecure to determine the status of my order. Don Walden answered my inquiry personally. During our conversation I mentioned the reason I wanted to purchase a new disk drive controller for my TI. Don suggested that I might be able to effect a better fix in simpler manner even though he faced a loss of 80 percent of the sale.

As we continued our conversation, Don made several other suggestions of how I might expand my TI system, and as a result I changed my original order and we both profited by his call — I with my better system and he with a sale of additional hardware.

As it was, Don did not lose part of the sale as the result of his suggested remedy. His willingness to help someone, even at a possible loss of income, is commendable. It is not often that we find someone willing to act in such a manner.

William H. Berendts
Camdenton, Missouri

Problem with PFM caused by hard drive

In my review of the Cecure Electronics PFM modification for the Geneve, which appeared in the June issue of MICROpendium, I stated that I was unable to load SYSTEM/SYS from my hard drive with the PFM device installed on my Geneve. The hard drive in question was a 40-megabyte Mitsubishi MR535 model. The MR535 model is a combination MFM/RLL type drive. After discussing this problem with Don Walden and Jim Schroeder, I learned that the problem was related to the SYSTEM/SYS file being fractured on the hard drive. After reformatting the hard drive and copying SYSTEM/SYS back to the root directory I am now able to boot the system from hard drive. PFM does not like a fractured MDOS file!

One other correction I would like to make is that MDOS version 2.0 will boot from a 1.44 megabyte floppy disk with PFM and a Myarc HFDC located at CRU 1000, 1100 or 1200. In the June PFM review I inadvertently only reported CRU location 1100 as useable for these purposes as this is the CRU location that my HFDC resides at. Again, Don Walden cleared up this point for me!

Cal Zanella
Bozeman, Montana

Another way

In the June issue I'm concerned with the way Cal Zanella writes about the Geneve EPROM. I have my System-Sys on a Horizon RAM Card and I'm up and running in GPL Extended BASIC in 12 to 13 seconds. I don't use M-DOS very much so if I need to it's fast just to Control, Shift, Shift, and then Escape, there's the A> prompt ready and waiting.

John Taylor
Fort Erie, Ontario, Canada

See above letter.—Ed.

Keeping your disks cool

The other day I received a 5.25-inch disk from Beery Miller (many thanks this way!). When I put it into my 5.25-inch drive, MDOS reported "Device Error," as it does if the disk is not formatted. I then tried John Birdwell's DSKU 4.2, which reported the same error.

I took the disk out and intended to examine optically by mirroring the ceiling light with the disk. Sometimes, when a disk is mechanically damaged, you can see pits or something else on the surface of the disk with this method. But, trying to rotate the disk by hand, I realized that the disk rested so tight in its envelope that the drive simply couldn't rotate it.

What to do?

I remembered that when I used the original SSSD TI drive I carefully cut apart a damaged envelope, took out the disk and was able to read the naked disk in the drive without the envelope. But the DSDD drive I use now cannot be closed without a disk envelope in it.

So I cut the envelope carefully on three sides, leaving alone the side with the read-write hole. Then I was able to read the disk. I quickly copied it to another, and everything went well.

I suppose the disk was exposed to heat during its transport by mail. These days we had about 34 degrees centigrade (93 F) here in Vienna, so temperatures, for example in a mailman's car, could go up to 60 centigrade (140 F) or more. Perhaps the disk envelope lost its shape this way and squeezed the disk.

This is a warning to all computer users to take good care of their disks and better not mail them during periods of hot weather. The disk itself may be worth rubbish, but the software stored on it is perhaps irrecoverable, not to mention the waste of time and labor trying to restore a damaged disk.

Alfred Slovak
Vienna, Austria

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

READER TO READER

(Continued from Page 4)

BBS software. Our BBS is 300, 1200 or 2400 at 8N1, 24 hours a day. It runs on a 20 meg hard drive and is restricted to TI or Geneve users only.

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

Extended BASIC

Arranging lists into columns with LISTCOL

By LUCIE DORAIS

Ready for some heavier stuff? Since I am using TI-BASE, I was sometimes producing long lists that took only the left part of the paper, therefore a useless waste of space. Why not write a program that would take a slim list and arrange the data so that it could be printed into many columns on the page?

LISTCOL will analyze your list, previously saved as a D/V80 file, and will calculate the best arrangement: How many columns, how many pages, etc. It will tell you its suggestion, but you can then modify things, like the space between the columns, and the left margin. If you print on a printer, you can choose between 80 and 132 columns (condensed). As a bonus, a graphic will give you, on screen, an idea of the arrangement on paper (or into a new TI-Writer file). This was not really needed, but I thought it was a nice programming challenge: How to represent an 80- or 132-column page on a screen which has only 32 columns.

To simplify, I decided to reduce the intended printed line by four: To display 80 columns, we need 20 screen columns (and 32 columns will display only 128 printer columns, but I solved that, sort of). The clue is in the XX$ string in line 130: With its 1s and 0s it has about all possible combinations of word columns (represented by the 1s) and space between them (the 0s). We will use this string when we play with our list further on. But let's start at the beginning.

In line 130, DIM I$ will hold the 60 lines that will be printed on each page (you can also print to disk so that you can refine your list in TI-Writer). Variables WS and LM are the default Width-of-Space and Left-Margin; PW is the default Printer-Width.

In lines 170-190, we redefine characters 120-135 to represent all combinations of columns and spaces that correspond to our XX$ string. By finding the position of one combination (a segment of four characters) into XX$, we can relate it to CHR$(119+POS), therefore "0000" will be represented on screen by char.^120 (all 0s), "0001" by char.^121, "0011" by char.^122, etc.

Before Tex can present you with a good arrangement of your list data, it needs to analyze the list itself! So the first part of the program is the setup/analyze: You first give the filename of your list to reshape, and you tell Tex right now if you want the new list on printer or on disk. Of course, you must then tell Tex the name of your printer or of your new file (Tex even suggests one!).

Tex will then display line 270, and you must answer another question: Do you need to analyze the list (to get the total number of data lines and the maximum column width) or not? If you already know the answers to these questions, answer "N" here, and Tex will accept those two values with the CALL As in line 290. If you need to analyze the list, Tex will open the file and read it until the end, displaying the result on screen. After it LINPUTs a line, it checks to see if the first character (ASC) is a control character, which means that it is the TAB line if you saved the file with "SF" in TI-Writer. If it is not a control character, it gets the length of the data and puts the MAXimum value (previous Word-Width or new length) into WW; the CTR line counter increments by one, until Tex reaches a line of control codes or the End Of File marker.

Tex now has all the information it needs to calculate the best arrangement on the page, and it will display it in a new "info screen:" The list filename FM$ with its CTR total lines, and the new list TO$ (or printer name). A line below the graphic will tell you what Tex thinks is the best number of COLumns, total LInes and PAges for your new list (line 370). Then some more useful information is displayed (lines 380-390); line 400 just puts a "<" besides the variables that you can then modify (Width-of-Space between columns, Left Margin and, if you want the new list on printer, Printer-Width).

We now GOSUB 650 to calculate the best arrangement. Follow me... CW is the total Column Width, i.e. Width-of-Word plus Width-of-Space; NC is the Number of Columns, calculated by dividing the Printer-Width, less Left Margin, by the column width CW. The Right Margin is than calculated (the right column CW will already include some spaces after the words, so we subtract them). The Total Line is simple: take the total of data lines in your first list, CTR, and divide it by the number of columns, with some adjustment. Then the PAge number is simply the total printed lines, TL, divided by 60, our default total lines per page.

Line 410 DISPLAYs the resulting values on screen: Width-of-Space, Left Margin, Right Margin and Printer Width are all displayed in column 25, with the use def. sub CALL D. Number of COLumns, Total LInes and PAges are all displayed on row 9, and we use CALL E. Finally Tex builds the graphic on the screen. Again, follow me with a GOSUB 670.

To compare each indented printed line with the string XX$, we must first build an A$ string of 0s and 1s to represent it. It will have 80 or 132 characters, depending on our choice of printer width. Disk file width is always 80.

Lines 670-680 is simply code to erase a larger graphic (for 132 characters) before we build a slimmer one (for 80 characters). If there is no need to do that, Tex goes to line 690, where our Width-of-Word is represented by 1s and the Width-of-Space by 0s. The Left Margin (all 0s) is added in front and each column repeated for our Number of Columns NC. Finally, the right margin is added. The S variable used in this portion of the program holds the previous Printer Width, useful when we go through line 670 to refresh the width of the screen graphic.

We now need to analyze our long string to compare it with XX$. If the Printer Width is 80, we read by segments of four

(See Page 7)

LISTCOL—

(Continued from Page 6)

until we reach position Y=77 (if PW=132, then Y=129). K is to calculate the column to display our characters on screen (if PW=132, we start in column 1; if PW=80, we start in column 7: see the CALL VCHAR in line 720). The IF in line 710 is because if we get a result of 17 for the position, finding the segment "1011" in XX$, that would mean to display character 119+17, i.e. 136. But, in line 160, we have redefined it to be a magenta line used to enhance our display screen, so we will use character 131 (119+12) instead.

Line 720 CALL VCHARs our characters on the screen (we will see three lines of our proposed list). If the reading position is below 129 (always if our Printer Width is only 80), we can put everything on one line; the last character for a 132-wide line will be displayed at the right of row 9, preceded by a "+" sign (ASCii 43).

With our graphic on screen, we can now modify the Width-of-Space, the Left Margin and, if we print to a printer, the Printer Width (lines 430-460). The calculations are displayed, and the graphic will be refreshed accordingly when we GO back TO 410.

Finally, we print the new list — SP$ and LM$ are simply the space between columns and the left margin translated into strings for easier handling. At this stage, you can give a T$ title to your printed list; Tex will also print the page number in the form "page 3/4 of TITLE". Line 510 opens the file with the corresponding width (DISPLAY VARIABLE 80, default, or 132); if we are in 132 mode, CHR$(15) will put your printer into condensed (compressed) mode. Each page PP is now processed and printed, which takes a while. Have a cup of coffee now!

Line 540 prints the information on top of your new list: Title and page number are printed in double-width, put on with CHR$(14), put off with CHR$(20). We now reuse variable S to hold the total lines per page (60 if total lines of new list is higher than 60, or the total TL if it is smaller). Tex needs that info to know when to stop reading the old list and print the page! The variable C here simply counts the LINPUTed lines of your old list so you can keep track of what is happening.

Each page is held in the I$ array, so when we build column 1 (Y=1), we need to erase the previous data (if we print more than one page). We then LINPUT a line of data from our old list FM$. Again, we close the file if we encounter a line of control characters (TABs), or the End-Of-File (EOF). Otherwise, we need to "pad" each data line with spaces: K will tell us how many spaces for each line (Maximum Width-of-Word WW less actual length of our linputed line). The result is then added at the end of the I$(X) string. This is done for the Number of Columns NC multiplied by the total lines per page S (FOR-NEXT Y and X, lines 560 and 590).

When the page is built in memory, Tex will print it in one loop (line 600). When a page is done, Tex prints a form feed, CHR$(12). If we still have data to print, i.e. if TL>60, then we decrement that total by 60 and we GOTO 530 to do the next page. Otherwise, we close the file; if our printer width was 132, we reset the printer to normal (ESC "@@") before closing it.

LISTCOL

```
100 ! *** LISTCOL *** L.Dora
is / Ottawa UG / Oct. 1990 !
070
110 !!131
120 DISPLAY AT(1,7)ERASE ALL
:"LIST TO COLUMNS" :: ON WAR
NING NEXT !171
130 DIM I$(60):: WS,LM=4 ::
PW=80 :: XX$="00001111000110
011011" !153
140 CALL COLOR(12,14,16,13,1
4,16,14,14,1):: LL$=RPT$(CHR
$(136),28):: A$="0000" !106
150 GOTO 160 :: C,CW,CTR,FM$
,K,NC,PA,PO,PP,S,SP$,T$,TL,T
O$,WW,X,Y :: CALL KEY :: CAL
L CHAR :: CALL HCHAR !@P- !1
44
160 CALL CHAR(35,"0101010101
010101",37,"8080808080808080
",136,A$&"00FFFF")!150
170 CALL CHAR(120,RPT$("0",1
4)&A$&"030303030303"&A$&"0F0
F0F0F0F0F"&A$&"3F3F3F3F3F3F"
)!017
180 CALL CHAR(124,"00FFFFFFF
FFFFF"&A$&"FCFCFCFCFCFC"&A$&
"F0F0F0F0F0F0"&A$&"C0C0C0C0C
0C0")!042
190 CALL CHAR(130,"003C3C3C3
C3C3C"&A$&"CFCFCFCFCFCF"&A$&
"C3C3C3C3C3C3",135,"00F3F3F3
F3F3F3")!030
200 !!131
210 ! ** setup/analyze ** !2
23
220 DISPLAY AT(4,1):"ENTER T
IW LIST FILENAME":"to divide
 in columns:": :" DSK" :: AC
CEPT AT(7,5)SIZE(12)BEEP:A$
:: FM$="DSK"&A$ !158
230 DISPLAY AT(10,1)BEEP:"NE
W LIST on  [D]SK [P]RINTER"
!118
240 CALL KEY(0,K,S):: IF S=0
 OR(K<>68 AND K<>80)THEN 240
 ELSE D$=CHR$(K)!073
250 IF K=68 THEN DISPLAY AT(
10,1):" "&SEG$(FM$,1,14)&"1"
 :: ACCEPT AT(10,5)SIZE(-12)
BEEP:TO$ :: TO$="DSK"&TO$ ::
 GOTO 270 !255
260 DISPLAY AT(10,1):"PIO" :
: ACCEPT AT(10,1)SIZE(-28)BE
EP:TO$ !081
270 DISPLAY AT(17,1):"NEED *
 ANALYZING * TO FIND:": : :"
TOTAL LINES IN LIST":TAB(27)
;"Y":"MAX LENGTH IN LINES" !
042
280 ACCEPT AT(21,27)SIZE(-1)
BEEP:A$ :: CALL HCHAR(21,29,
32):: IF A$="Y" THEN 300 !21
4
290 CALL A(20,CTR):: CALL A(
22,WW):: GOTO 360 !059
300 DISPLAY AT(17,1):"     *
 ANALYZING *" :: OPEN #1:FM$
,INPUT !022
310 LINPUT #1:A$ :: IF ASC(A
$)>127 THEN 330 ELSE WW=MAX(
WW,LEN(A$)):: CTR=CTR+1 !102
320 CALL D(20,CTR):: CALL D(
22,WW):: IF EOF(1)THEN 330 E
LSE 310 !131
330 CLOSE #1 !151
340 !!131
350 ! ** info screen ** !215
360 DISPLAY AT(1,1)ERASE ALL
:"fm: "&FM$;TAB(18);CTR;"lin
es":"to: "&TO$:LL$ !210
```

(See Page 8)

LISTCOL—

(Continued from Page 7)

```
370 S=132 :: DISPLAY AT(9,5)
:"COL      LI      PA" !25
1
380 DISPLAY AT(11,1):LL$:"WI
DTH OF WORD COLUMN:":"SPACE
BETWEEN COLUMNS:": :"LEFT MA
RGIN:":"RIGHT MARGIN" !154
390 DISPLAY AT(17,1):LL$:"TO
TAL CHAR/LINE:":LL$ :: CALL
D(12,WW)!198
400 CALL HCHAR(13,31,60):: C
ALL HCHAR(15,31,60):: IF D$=
"P" THEN CALL HCHAR(18,31,60
)!010
410 GOSUB 650 :: CALL D(13,W
S):: CALL D(15,LM):: CALL D(
16,RM):: CALL D(18,PW)!087
420 CALL E(1,NC):: CALL E(11
,TL):: CALL E(20,PA):: GOSUB
 670 !235
430 DISPLAY AT(24,1):"INFORM
ATION OK? N  (<=mods)" :: AC
CEPT AT(24,17)SIZE(-1)VALIDA
TE("YN")BEEP:A$ :: IF A$="Y"
 THEN 490 !026
440 DISPLAY AT(22,1):"":"":"
" :: CALL A(13,WS):: CALL A(
15,LM):: IF D$="D" THEN 410
!205
450 DISPLAY AT(24,1)BEEP:"EN
TER ONLY:  80 or 132" :: CAL
L A(18,PW)!248
460 IF PW<>80 AND PW<>132 TH
EN 450 ELSE DISPLAY AT(24,1)
:"" :: GOTO 410 !159
470 !!131
480 ! ** print/save new list
 ** !220
490 SP$=RPT$(" ",WS):: LM$=R
PT$(" ",LM)!151
500 DISPLAY AT(20,1):"title:
" :: ACCEPT AT(21,1):T$ !064
510 OPEN #1:FM$,INPUT :: IF
PW=80 THEN OPEN #2:TO$,OUTPU
T ELSE OPEN #2:TO$,OUTPUT,VA
RIABLE 132 :: PRINT #2:CHR$(
15)!212
520 DISPLAY AT(22,1):"":"REA
DING INPUT LINE":"PRINTING P
AGE" !007
530 PP=PP+1 :: DISPLAY AT(24
,14):PP !161
540 A$=LM$&"page  "&CHR$(14)
&STR$(PP)&"/"&STR$(PA)&CHR$(
20):: IF T$<>"" THEN A$=A$&"
  of  "&CHR$(14)&T$ !163
550 PRINT #2:A$:"":"" :: IF
TL>60 THEN S=60 ELSE S=TL !0
95
560 FOR Y=1 TO NC :: FOR X=1
 TO S :: C=C+1 :: DISPLAY AT
(23,20):C :: IF Y=1 THEN I$(
X)="" !135
570 LINPUT #1:A$ :: IF ASC(A
$)>127 THEN 600 !103
580 K=WW-LEN(A$):: I$(X)=I$(
X)&A$&RPT$(" ",K)&SP$ :: IF
EOF(1)THEN 600 !086
590 NEXT X :: NEXT Y !095
600 FOR X=1 TO S :: PRINT #2
:LM$&I$(X):: NEXT X !130
610 PRINT #2:CHR$(12):: IF T
L>60 THEN TL=TL-60 :: GOTO 5
30 !212
620 IF PW=132 THEN PRINT #2:
CHR$(27)&"@" :: CLOSE #2 ELS
E CLOSE #2 !172
630 END !139
640 ! ** subs ** !072
650 CW=WW+WS :: NC=INT((PW-L
M)/CW):: RM=PW-NC*CW-LM+WS :
: TL=INT(CTR/NC):: IF TL<>CT
R/NC THEN TL=TL+1 !109
660 PA=INT(TL/60)+1 :: RETUR
N !228
670 IF S=PW OR PW=132 THEN 6
90 ELSE CALL HCHAR(9,30,32,2
)!025
680 FOR X=5 TO 7 :: CALL HCH
AR(X,1,32,32):: NEXT X :: CA
LL VCHAR(5,6,35,3):: CALL VC
HAR(5,27,37,3)!161
690 A$=RPT$("1",WW)&RPT$("0"
,WS):: A$=RPT$("0",LM)&RPT$(
A$,NC):: A$=A$&RPT$("0",RM-W
S)!210
700 S=PW :: IF PW=80 THEN Y=
77 :: K=27 ELSE Y=129 :: K=3
 !152
710 FOR X=1 TO Y STEP 4 :: P
O=POS(XX$,SEG$(A$,X,4),1)::
IF PO=17 THEN PO=12 !226
720 IF X<129 THEN CALL VCHAR
(5,(K+X)/4,119+PO,3)ELSE CA
LL HCHAR(9,30,43):: CALL HCHA
R(9,31,119+PO)!007
730 NEXT X :: RETURN !248
740 !!131
750 !@P+ ** u-d subs ** !041
760 SUB D(R,V):: DISPLAY AT(
R,25):V :: SUBEND !098
770 SUB E(C,V):: DISPLAY AT(
9,C)SIZE(-3):USING "###":V :
: SUBEND !001
780 SUB A(R,V):: ACCEPT AT(R
,26)SIZE(-3)BEEP:V :: SUBEND
 !102
```


Hayes lowers prices

Hayes Microcomputer Products has announced 10-17 percent reductins in the estimated retail prices of five modems in the companies Accura line. New prices range from $149 to $179; the old prices ranged from $179 to $199.

In May, Hayes reduced the estimated retail price of Accura 288 V.FC _ FAX, Hayes' fastest Accura modem, to $299.

For information, call: (404) 441-16.., United States; (519) 746-5000, Canada; +33 1 34 22 30 15, France; +44 252 775544, Europe.

THE ART OF ASSEMBLY — PART 37

Option 5 Revisited

By BRUCE HARRISON

Some time ago, we presented in this column a couple of ways to make your Assembly program over into an Option-5 Program File format. This makes programs load much faster and, of course, more conveniently than is the case for Option-3 loading. The drawback is that the utility subprograms like VMBW, VSBW, and such, do not get loaded into low memory under Option-5 as they do in an Option-3 loading process. (Funnelweb's loader for Option-5 does set things up properly, but TI's does not. Once again Tony McGovern did the right thing.) In our earlier columns on this subject, we used very crude but effective methods to load the low memory with the utilities.

This month we'll present two more methods for getting your Option-5 program to load its utilities. The first method came from work we were doing to create a compiler for Extended BASIC programs. (That will be covered further next month.) In the process of our "research" toward a compiler, we were playing around with the idea of executing CALL operations from within Assembly routines. To execute a call, we used a DSRLNK, complete with a PAB and the DATA >A. What we found in our little experiments was that we could write an Option-3 program that would perform a CALL INIT that re-loaded everything in low memory to the normal "startup" situation for an Option-3 load. That is, the low memory would look as if no loading of the Option-3 program had taken place. It then occurred to us that this might be a way to get things set up for Option-5 programs also.

Let's digress for a moment to describe how our own TI is configured. We use a "normal" TI with two floppy drives, but have added a Horizon RAMdisk and Horizon P-GRAM card. The P-GRAM is kept loaded with Extended BASIC and Editor/Assembler. The P-GRAM also gives us the very handy CALL PG, which allows us to inspect RAM, ROM, GROM and VDP memories very easily. Thus, when we got our little CALL INIT experiment put together, we added a CALL PG to the program so that we could go and inspect the memory as it was left by the CALL INIT operation. (The high memory expansion and parts of VDP memory get changed by the CALL PG, but low memory is mostly left alone.) Sure enough, a quick examination of the low memory showed that after our little Option-3 program had run, low memory looked just the way it would if we went into console BASIC, did a CALL INIT, then a CALL PG, with the singular exception that the device name PG showed up in the low memory after the DSRLNK to perform the CALL PG.

MAKING A CALL

As you can see from Part One of the sidebar, the call is actually made by a DSR linkage operation, just like a file operation, except that this time the DATA statement that follows the BLWP to DSRLNJ is DATA >A, not the usual DATA 8. The reason for this is simple. In the Device ROMs and GROMs, there are linked lists for the device drivers present in that ROM or GROM. The pointer to the beginning of the list for "disk" type devices is at eight bytes from the start of the ROM or GROM space. The list pointer for the "call" type services is at 10 (>A) bytes from the start of the ROM or GROM space. In the case of the TI disk controller, for example, the ROM space (CRU BASE >1100) starts at >4000. The ROM contains both disk device drivers and call drivers, so there is a pointer at >4008 to the start of the disk devices list, and a pointer at >400A to the start of the call drivers. In the case of the TI Disk Controller, the call drivers include some which take care of direct sector-access operations and CALL FILES, while the disk devices list takes care of devices DSK, DSK1, DSK2 and DSK3.

The DSR linkage actually searches these linked lists for the named device or call, as appropriate, then executes when it finds the address for the desired device or call driver. In the sidebar, we are showing the Warren/Miller DSRLNK, which actually uses a GROM routine through GPLLNK to do this searching for us, so the searching in this case is "transparent" from the point of view

(See Page 10)

Sidebar 37

```
* SIDEBAR 37
* EXPERIMENTS WITH "INIT"
* USING THE E/A MODULE
* CODE BY B. HARRISON
* ************************************ * FIRST PART - USING
INIT AS A CALL * * 16 JUL 1993            *
* PUBLIC DOMAIN              *
************************************ *
PAB       EQU     >1000       PAB LOCATION IN VDP RAM
STATUS    EQU     >837C       GPL STATUS BYTE
GPLWS     EQU     >83E0       GPL WORKSPACE
GR4       EQU     GPLWS+8     GPL REGISTER 4
GR6       EQU     GPLWS+12    GPL REGISTER 6
STKPNT    EQU     >8373       STACK POINTER
LDGADD    EQU     >60         LOAD GROM ADDRESS
XTAB27    EQU     >200E       STORAGE SPOT
GETSTK    EQU     >166C       GET STACK
GRMRA     EQU     >9802       GROM READ ADDRESS
GRMWA     EQU     >9C02       GROM WRITE ADDRESS
          REF     VMBW,DSRLNK REF UTILITIES
          REF     VDPWA,VDPWD REF ADDRESSES
          DEF     PGX         DEFINE ENTRY POINT
PGX       LWPI    WS          LOAD OUR WORKSPACE
          LI      R0,PAB      POINT AT PAB LOCATION
          ORI     R0,>4000    MAKE IT A WRITE ADDRESS
          SWPB    R0          SEND LOW ORDER BYTE FIRST
          MOVB    R0,@VDPWA   TO VDPWA
          SWPB    R0          SWAP BYTES
          MOVB    R0,@VDPWA   SEND HIGH ORDER BYTE
          LI      R1,PABDT    POINT AT PAB DATA
          LI      R2,15       15 BYTES TO SEND TO VDP
WMB       MOVB    *R1+,@VDPWD WRITE ONE BYTE
          DEC     R2          DECREMENT COUNT
          JNE     WMB         IF NOT ZERO, REPEAT
          LI      R0,PAB+9    GET PAB +9 ADDRESS
          MOV     R0,@>8356   PUT AT >8356
          CLR     @STATUS     CLEAR GPL STATUS
          BLWP    @DSRLNJ     USE MILLER DSRLNK
          DATA    >A          WITH DATA >A FOR "CALL" OPERATION
*
* CODE FROM HERE ON MAKES A CALL TO PG IN THE P-GRAM CARD * THIS
```

THE ART OF ASSEMBLY—

(Continued from Page 9)

of our source code. The fact that our CALL INIT worked correctly presented something of a mystery in itself.

Here's why. Our P-GRAM card has all of the GROM for both Extended BASIC and Editor/Assembler loaded into it. The XB part starts at >6000 and runs through >DFFF, while the E/A GROM starts at >E000 and runs through >FFFF. Why, then, when we executed our DSRLNK looking for INIT, did the DSR not find the INIT that's in the XB's GROM and execute that, instead of doing what we intended, executing the CALL INIT that's in the E/A GROM? A little probing around in the GROM has given us a potential answer to this mystery.

The linked lists of calls in all but the XB GROM are set up like this: The first pointer points to a group of bytes that looks, typically, like this:

>6B3E,>6B82,>04,INIT,>6B50,>6BD8,>04,LOAD

In each such grouping, the first two bytes point to the next entry in the list (0000 for the last entry). The next two bytes are the address of the named routine for this entry, then there's a byte for the length of the name, followed by the name of the routine. The numbers we've shown here are the actual numbers found in an E/A's GROM. For ease of reading, we've shown the names in "plain English" but, of course, in the GROM they'd show up as hex characters.

(See Page 11)


```
MUST BE LEFT OUT IF YOU DON'T HAVE THAT CARD
*
CALLPG      LI      R0,PAB          POINT TO PAB AGAIN
            LI      R1,PABDT2       DATA FOR CALL PG
            LI      R2,15           15 BYTES
            BLWP    @VMBW           USE THE VMBW THAT INIT HAS PUT
IN PLACE
            AI      R0,9            ADD 9
            MOV     R0,@>8356       PUT AT >8356
            CLR     @STATUS         CLEAR GPL STATUS
            BLWP    @DSRLNK         USE E/A DSRLNK
            DATA    >A              WITH DATA >A TO CALL PG
*
* YOUR MAIN PROGRAM CODE WOULD GO HERE
* ANY GPLLNK CALLS MUST USE THE MILLER GPLLNK
* THAT'S ALREADY INCLUDED BELOW (NO REF TO GPLLNK) * IT MUST
EXIT WITH THE FOLLOWING:
*
EXIT        BLWP    @0
*
* DATA FOR PART ONE
*
WS          BSS     32              OUR WORKSPACE
PABDT       DATA    0,0,0,0,>0004           PAB DATA FOR 'INIT'
            TEXT    'INIT  '        FILE NAME
PABDT2      DATA    0,0,0,0,>0002           PAB DATA FOR 'PG'
            TEXT    'PG    '        FILE NAME
*
* FOLLOWING IS THE WARREN/MILLER GPLLNK AND DSRLNK *
GPLLNK      DATA    GLNKWS          UTILITY VECTOR - WORKSPACE
            DATA    GLINK1          VECTOR CODE WORD
RTNAD       DATA    XMLRTN          R9 OF GLNKWS
GXMLAD      DATA    >176C           R10 OF GLNKWS
            DATA    >50             R11 OF GLNKWS
GLNKWS      EQU     $->18
            BSS     >08
GLINK1      MOV     *R11,@GR4       MOV @>50,@GR4
            MOV     *R14+,@GR6      MOV DATA INTO GR6
            MOV     @XTAB27,R12     STASH VALUE FROM >200E
            MOV     R9,@XTAB27      PLACE ADDRESS XMLRTN AT >200E
            LWPI    GPLWS           LOAD GPL WS
            BL      *R4             BL
MOV         @GXMLAD,@>8302(R4)      PLACE >176C AT >8302 +
            INCT    @STKPNT         INC BY TWO AT >8373
            B       @LDGADD         B @ >60
XMLRTN      MOV     @GETSTK,R4      MOV @>166C, R4
            BL      *R4             BL THERE
            LWPI    GLNKWS          LOAD GLNKWS
            MOV     R12,@XTAB27     REPLACE VALUE FROM >200E
            RTWP                    RETURN WITH WORKSPACE POINTER
PUTSTK      EQU     >50
TYPE        EQU     >836D
NAMLEN      EQU     >8356
VWA         EQU     >8C02
VRD         EQU     >8800
GR4LB       EQU     >83E9
GSTAT       EQU     >837C
DSRLNJ      DATA    DSRWS,DLINK1 RE-NAMED DSRLNJ, NOT TO CON-
FLICT
DSRWS       EQU     $
DR3LB       EQU     $+7
DLINK1      MOV     R12,R12
            JNE     DLINK3
            LWPI    GPLWS
            MOV     @PUTSTK,R4
            BL      *R4
            LI      R4,>11
            MOV     R4,@>402(R13)
            JMP     DLINK2
            DATA    0
            DATA    0,0,0
DLINK2      MOVB    @GR4LB,@>402(R13)
            MOV     @GETSTK,R5
            MOVB    *R13,@DSRAD1
            INCT    @DSRADD
            BL      *R5
```

THE ART OF ASSEMBLY—

(Continued from Page 10)

In Extended BASIC's GROM, the numbers are arranged in a different order, so an entry similar to the above would look like this:

>C02B,>04,INIT,>C2BA,>C034,>04,PEEK,>C2CA...

Here, the pointer to the next entry is followed immediately by the name length, then the name, and after that is the address for the routine. Thus, when the search is taking place, the DSR doesn't find the name where it should be in the linked list, so each item found in the XB GROM gets ignored, and the search continues until it finds INIT "correctly" listed in the E/A's GROM. Obviously the XB interpreter must execute its own search for call names in a different fashion, so it can find its own calls, not those in other GROMs. (The calls linked list in BASIC's GROM are arranged like those in E/A's GROM.)

It was probably arranged this way on purpose by TI so that XB would skip over the console BASIC version of any CALL name, and find its own version for execution. This difference turned out to be convenient for us, in that it meant our new method for dealing with Option-5 would work even though both the XB and E/A GROMs are "present" in our P-GRAM. For ROM calls, XB seems to be able to find those even though their linked lists are constructed like the BASIC and E/A GROM lists. Please don't ask just how XB manages that. We don't know!

After proving to ourselves that we could execute a CALL INIT and then a CALL PG from within an Option 3 program, we added the labels SFIRST, SLOAD, and SLAST to our source file, performed a SAVE operation to make an Option-5 Program File called OPT5, and then ran that. That worked perfectly, getting us quickly into our PG call. It worked when run directly from our RAMdisk menu, just as well as when run from Option-5 under E/A. Having had some success, we decided to try another experiment, and found a flaw in our "perfect" solution.

Instead of doing a CALL PG, we decided to try simply putting some text on the screen, executing a "key loop" to let us see that, then returning to E/A control. DISASTER! Everything worked until we pressed a key, then our return to E/A went bonkers. A bunch of stuff appeared, scrolled up the screen, and then the screen went blank and eventually returned us to "startup" conditions, with our RAMdisk menu on the screen.

The only "safe" return from this little experiment was to BLWP @0. That did what it should do, returning us to the RAMdisk menu without any intervening trouble. We assume that this return problem has something to do with the GROM address, but saving and restoring that did absolutely nothing. Obviously, TI's "hidden agenda" has struck us again, and we don't know why.

So long as you don't need to return to E/A at the end of your program, the CALL INIT process will work just fine to load the E/A utilities for your Option-5 program. Just be sure that any exit point executes a BLWP @0, and you'll go safely back to startup condition, with the appropriate title screen or menu, depending on how you're equipped.

Having the P-GRAM available makes our life easier in many ways, and we'd like to thank Bud Mills and company for this

(See Page 12)

```
            LWPI    DSRWS
            LI      R12,>2000
DLINK3      INC     R14
            MOVB    *R14+,@TYPE
            MOV     @NAMLEN,R3
            AI      R3,-8
            BLWP    @GPLLNK
DSRADD      BYTE    >03
DSRAD1      BYTE    00
            MOVB    @DR3LB,@VWA
            MOVB    R3,@VWA
            SZCB    R12,R15
            MOVB    @VRD,R3
            SRL     R3,5
            MOVB    R3,*R13
            JNE     SETEQ
            COC     @GSTAT,R12
            JNE     DSREND
SETEQ       SOCB    R12,R15
DSREND      RTWP
            END
*
*****************************
****  END OF PART ONE  **** ***************************
*
***********************************
* SECOND PART - GETTING UTILITIES *
* DIRECTLY FROM THE E/A GROM      *
* 18 JULY 1993                    *
* PUBLIC DOMAIN                   *
***********************************
*
GRMRA       EQU     >9802       THE GROM READ ADDRESS
GRMWA       EQU     >9C02       THE GROM WRITE ADDRESS
GRMRD       EQU     >9800       THE GROM "READ DATA" ADDRESS
            REF     KSCAN,VMBW,GPLLNK    REFERENCE VECTORS
            DEF     SFIRST,SLAST,SLOAD   DEFS FOR OPTION 5
            DEF     INITX3      DEFINE ENTRY POINT
*
* THE "LONE LABELS" SFIRST AND SLAST GET THE SAME * ADDRESS AS
INITX3, SO THAT THE "SAVE" UTILITY
* WILL WORK CORRECTLY
*
SFIRST
SLOAD
INITX3      LWPI    WS          LOAD OUR WORKSPACE
            LI      R3,>6000    POINT AT >6000 IN GROM
NXTGRM      MOVB    R3,@GRMWA   WRITE HIGH BYTE OF ADDRESS
            SWPB    R3          SWAP
            MOVB    R3,@GRMWA   WRITE LOW BYTE OF ADDRESS
            SWPB    R3          SWAP AGAIN
            MOV     R3,R4       MOVE R3 INTO R4
            CB      @GRMRD,@VALBYT  CHECK FOR BYTE >AA AT START
            JNE     ADDGRM      IF NOT EQUAL, SKIP AHEAD
            AI      R4,6        ELSE ADD 6 TO ADDRESS IN R4
            MOVB    R4,@GRMWA   SEND THE HIGH BYTE
            SWPB    R4          SWAP
            MOVB    R4,@GRMWA   SEND LOW BYTE
            SWPB    R4          SWAP BACK
            MOVB    @GRMRD,R4   READ A BYTE FROM X006 IN GROM
            SWPB    R4          SWAP
            MOVB    @GRMRD,R4   READ NEXT BYTE
            SWPB    R4          SWAP AGAIN
            MOV     R4,R4       MOVE R4 TO ITSELF TO SET STATUS
REGISTER
            JEQ     ADDGRM      IF ZERO, SKIP THIS GROM
            LI      R5,4        ELSE LOAD R5 WITH 4
            AI      R4,5        ADD 5 TO ADDRESS IN R4
            LI      R9,EATITL   POINT R9 AT TITLE "EDIT"
*
* AT THIS POINT R4 HAS ADDRESS OF THE TITLE OF THIS GROM * WHILE
R9 POINTS AT THE WORD "EDIT"
* AND R5 HAS THE NUMBER OF LETTERS TO COMPARE
*
            MOVB    R4,@GRMWA   SET TO READ FROM
            SWPB    R4          SWAP R4
```

THE ART OF ASSEMBLY—

(Continued from Page 11)

nifty little card, and especially for the PG call that makes probing into TI's secrets much easier. While we were at it, we decided to find out just where E/A's CALL INIT gets its data to dump into the low memory. It was easy enough to find, using the "search" function from the PG call.

We found the data that's dumped into the low memory starting at >F000 in our P-GRAM's GROM memory. In the normal E/A cartridge, we found this data at GROM address >7000. The data is there in the form of three strings, each with a "length" word, then a target location word, followed by the string of data itself. The word at >7000 (or >F000) contains the number 8, followed by a word containing >2000, then the eight bytes that the INIT routine places at that location. The next two bytes are the length of the second string (>654), followed by the destination for that string (>2022), and then the 1620 bytes that go into low memory starting at >2022. Finally, after those 1620 bytes is the length (>00C8) and destination (>3F38) for the initial state of the REF/DEF table that goes at the high end of the low memory. This is followed by those definitions for all the pre-defined labels like VMBW, VSBW, DSRLNK, and so on.

METHOD TWO

Having thus found the data that's used by CALL INIT, and doped out how it's organized, we have a second method for putting this data into low memory. We can take the length from the first two bytes of the GROM at >7000 (or >F000), place that into a register, take the target address from the next two bytes, place that in another register, then simply loop a GROM read cycle for the number of times in that first register, and we'll have what we were after for the first "string". When that finishes, we'll be "pointing" at the first of the two bytes for the second group's length. Thus we can make a nested loop to take the three strings needed as shown in today's sidebar. This will work fine for the case of a user with an actual E/A cartridge, and will not cause a problem upon return to E/A, but will cause a problem when the user has a P-Gram with E/A starting at a different address in GROM space.

Since we don't know, in general, whether the user of our program will be using an actual E/A module, or some device like the P-GRAM, we've added a routine in Part Two of the sidebar to search the GROM address space until the E/A GROM is found, and then advance the address to the "equivalent" of >7000 in the real E/A module.

What's shown there is a kind of shortcut method. Starting at >6000, it first checks the 0th byte in that block for >AA, and if that's okay, it gets the address from the 06 and 07 bytes for the title "lookup table" starting point. Since the E/A GROM has only one title, we can take that address from the 06/07 bytes, then just add five so we're pointing at the title itself. Here there's a comparison loop that's another shortcut, in that it checks the title from the GROM for only the first four letters, to wit "EDIT". That's more than enough to distinguish the E/A module from TI Extended BASIC or TI-Writer, and such. The loop that compares the title is set up so that as soon as a byte is found not to match the text

(See Page 13)

```
            MOVB    R4,@GRMWA       ADDRESS IN R4
            SWPB    R4              SWAP R4
READTL      CB      @GRMRD,*R9+     COMPARE ONE BYTE TO "EDIT"
            JNE     ADDGRM          IF NOT EQUAL, WRONG GROM
            DEC     R5              DECREMENT COUNT
            JNE     READTL          IF NOT ZERO, TRY NEXT BYTE FROM
GROM
            ANDI R3,>F000           ELSE WE'VE FOUND EDITOR/ASSEMBLER
            MOV     R3,R7           STASH "ROOT" GROM ADDRESS IN R7
            AI      R3,>1000        ADD >1000 TO ROOT ADDRESS OF GROM
            JMP     GETUT           THEN GO GET THE UTILITIES
ADDGRM      AI      R3,>2000        POINT AHEAD BY 2000
            JNE     NXTGRM          IF NON-ZERO, JUMP TO INSPECT
NEXT BLOCK
            BLWP    @0              ELSE WE HAVE NO E/A GROM
GETUT       MOVB    R3,@GRMWA       LOAD GROM ADDRESS HIGH BYTE
            SWPB    R3              SWAP BYTES
            MOVB    R3,@GRMWA       LOAD GROM LOW ADDRESS
            LI      R5,3            SET COUNTER AT 3 (3 BLOCKS OF DATA)
GQTY        MOVB    @GRMRD,R4       GET ONE BYTE FROM GROM
            SWPB    R4              SWAP
            MOVB    @GRMRD,R4       GET LOW ORDER BYTE
            SWPB    R4              SWAP AGAIN
            MOVB    @GRMRD,R3       GET LOCATION HIGH ORDER BYTE
            SWPB    R3              SWAP
            MOVB    @GRMRD,R3       GET LOCATION LOW ORDER
            SWPB    R3              SWAP
GETGR       MOVB    @GRMRD,*R3+     GET A BYTE OF DATA, PLACE AT AD-
DRESS IN R3
            DEC     R4              DECREMENT NUMBER OF BYTES THIS
BLOCK
            JNE     GETGR           IF NOT ZERO, DO ANOTHER
            DEC     R5              DECREMENT NUMBER OF BLOCKS
            JNE     GQTY            IF NOT ZERO, START NEXT BLOCK
*
* NEXT SECTION SETS UP THE CORRECT GROM ADDRESS FOR USING
* THE TI GPLLNK - WITHOUT THIS, OPTION 5 PROGRAMS CAN'T
* USE THAT UTILITY EVEN THOUGH IT'S LOADED INTO LOW MEMORY *
            AI      R7,>0892        ADD OFFSET >892 TO ROOT GROM
ADDRESS
            MOVB    R7,@GRMWA       WRITE HIGH BYTE
            SWPB    R7              SWAP BYTES
            MOVB    R7,@GRMWA       WRITE LOW BYTE
            SWPB    R7              SWAP BACK
            LI      R5,>061C        PUT STANDARD GPL RETURN ADR IN R5
            MOV     R5,@>2030       PLACE THAT AT >2030
*
* FROM THIS POINT ON, ALL E/A UTILITIES ARE AVAILABLE
* FOR USE BY YOUR PROGRAM, INCLUDING TI'S GPLLNK
* YOUR PROGRAM MAY EXIT THROUGH THE CODE AT QEXIT, SHOWN BELOW *
DISPL       LI      R0,11*32+13  POINT AT ROW 12, COL 14
            LI      R1,EATITL    THE WORD "EDIT"
            LI      R2,4         FOUR CHARACTERS
            BLWP    @VMBW        WRITE THOSE
            CLR     @>837C       CLEAR GPL STATUS BYTE
            BLWP    @GPLLNK      USE TI'S GPLLNK
            DATA    >34          TO MAKE A BEEP
SCAN        BLWP    @KSCAN       SCAN KEYBOARD
            LIMI    2            ALLOW INTERRUPTS
            LIMI    0            THEN SHUT THEM OFF
            CB      @ANYKEY,@>837C       HAS A KEY BEEN STRUCK?
            JNE     SCAN         IF NOT, SCAN AGAIN
QEXIT  LWPI >83E0    LOAD GPL WORKSPACE
            B       @>6A         BRANCH TO GPL INTERPRETER
*
* DATA SECTION FOR PART TWO
*
WS          BSS     32           OUR OWN WORKSPACE
VALBYT      BYTE    >AA          VALIDATION BYTE
EATITL      TEXT    'EDIT'       COMPARISON NAME FRAGMENT
ANYKEY      BYTE    >20          COMPARISON FOR KEYSTROKE
SLAST       EQU     $            END OF "SAVE" FOR OPTION 5
            END
*
*****************************
****  END OF PART TWO  **** ****************************
```

THE ART OF ASSEMBLY—

(Continued from Page 12)

at label EATITL, the program will exit the comparison loop and go on to the next >2000 group in the GROM space.

As we were getting ready to put a "wrap" on this month's sidebar, one of those "almost forgots" sprang to mind. The astute reader will now go back to Part 26 of this series with us. If the program that's going to use this "startup" includes a REF to TI's GPLLNK, then we have to correct the GROM address for GPLLNK before the program can BLWP to it. Thus we've pulled that little trick out of Part 26 and added it to this month's sidebar Part Two so that the user's program can BLWP to TI's GPLLNK without any problem. At least that's what we thought, until we tried testing part two of the sidebar with an actual GPLLNK call in it. Another disaster! It took quite a while to unravel this new mystery, but eventually we got the answer.

In our older methods, we had "captured" the low memory contents after an Option 3 loading process, so there had to be something else of great importance that we were missing. After much probing around, we found that the Option 3 load sets the word of memory at >2030 to the value >061C. That address is part of the UTLTAB area, and is called "GPL Return Address" in the big book. When we executed our Part Two code to get low memory loaded from the GROM, address >2030 was left with zero as its value. Thus we added the short section of code that puts >061C to R5, then moves that value into >2030. Hooray! This did the trick. As shown in the sidebar, then, Part Two is a complete program that works from Option 5, puts the word "EDIT" on the screen, beeps, and then waits for a keypress, after which it exits to E/A's PRESS ANY KEY TO CONTINUE prompt. In other words, it does exactly what we intended. We tested this from Option 3, from Option 5 under E/A, with both our P-GRAM E/A and a real module, and as run directly from our RAMdisk menu. Worked in all cases.

THE EXCEPTIONS

You always know that in our column we'll point out carefully any known "exceptions" to what we've presented as a "solution," and sure enough, here's a "biggie." In the previous methods we showed for setting up the utilities in low memory, the data needed was "embedded" in the program as saved to the disk. That way, the program could be loaded (assuming some loader was at hand) and run without the E/A module being available. The method we've covered this month is not like that. In the sidebar, (Part Two) we've shown an "emergency exit" that will get out of the program if the E/A module is not found, and that's an essential feature in any program file that works this way. We ran a test in which only the XB module was available, and sure enough the program worked as expected, exiting back to startup. Either of the two methods shown in this month's sidebar will work just fine so long as the E/A GROM is available somewhere in the GROM address space. Neither will work if E/A is not present.

We should also point out that the Part One section, which uses the Warren/Miller DSRLNK to execute the CALL INIT, will probably not work on the Geneve. So far as we know, the second method shown will work just as well on the Geneve as on a TI. We'll ask a friend who has a Geneve to test this, and in due course will let you know.

Next month we'll write some about our "impossible" project, making a Compiler for Extended BASIC. If you thought this month's topic was "heavy," just wait one month, and this will look like "duck soup."

Anatomy of a header file

By JEFF H. WHITE and DAN H. EICHER

There we were again, hacking for truth, justice, and the American way. Jeff had just showed me his nifty modification to adding hardware handshaking to a Myarc RS232. He was making LEDs light up and go dark with a single CRU instruction. Nifty!

At this point, I was ready to fire up Al Beard's TIC compiler and write a Kermit protocol with full hardware handshaking when I suddenly remembered — "Hey, TIC and TIC lib don't have any functions defined to allow me to manipulate CRU address, nor play directly with memory (a must for true systems programming work!)". [Actually, Dan, memory is easily manipulated in C through the use of pointers, as I explained in a Delphi TI Net message to you several months ago.]

Quickly I ran to my bookshelf and pulled out my copy of Software Development handbook by TI. Yep, they included example commands for direct memory and CRU manipulations, so I had to have these to!

It was off to another adventure, because I had never written my own function in C. Jeff had never written anything in C, but tons in assembler. So it was going to be a real learning experience.

The first hurdle that must be overcome when using TIC is the documentation. While all the information you could ever really need is included in between the documentation provided with TIC and TICLIB, it is rather spartan and doesn't handhold the novice.

Looking over the docs that come with TIC, we managed to discover the register usage used. The really nice thing about TIC is that it doesn't directly compile down into executable code. Instead, it produces assembler source code, which you can use to go in and debug your software. In the accompanying C and assembler source code you will see how to access values from your main C program and how to call your own assembler functions — without the need to resort to "IN LINE" assembler code.

L.D.O.M. 03.27.94

(See Page 14)

ANATOMY OF A HEADER FILE—

(Continued from Page 13)

```
#include <STDIO_H>
#include <VIDEO_H>
#include <TIME_H>
#include <STRING_H>
#include "nstdio_h"

struct rw_port_block cru_access;      /* Creates new variable definition */
struct rw_port_block port_access;

main(argc,argv)
      int argc;
      char *argv[];
{
      char mybuff[81];  /* our character buffer */
      char A,B;         /* char is basically Unsigned Int... */
      int  C,D;
      int  i;

      printf("\f Starting routine \n");
      A=0x0;      /* A value of 0, means write          */
      B=0x3;      /* 3 bits to write                    */
      C=0x130A;   /* base address of RS232 primary CTS  */
      D=0x0007;   /* 3 lsbits are all 1 to turn on CRU  */
                  /* bits                               */

/* Since B is a value greater than >1, the CRU bits are packed in D */
/* and right-most (lsbyte) justified.  Bit 0 is the right-most bit  */
/* and bit 15 is left-most.                                         */

      cru_access.read_write = A;   /* We could have just directly assigned */
      cru_access.length = B;       /* values to these variables, but it    */
      cru_access.address = C;      /* is educational to study the          */
      cru_access.value = D;        /* assembler output of TIC              */

      crumod( cru_access );        /* Call our new function.               */

      /* the above function will activate the extra hardware handshaking  */
      /* lines described in the accompanying article.                     */

      cru_access.read_write = 0x0; /* write a value                       */
      cru_access.length = 0x0;     /* a single bit write                  */
      cru_access.address = 0x1100; /* Base Cru address of disk controller */
      cru_access.value = 0x0001;   /* offset from base in msbyte          */
                                   /* turn on (lsbyte <> 0 )              */

      crumod( cru_access );        /* call function.                      */

/* The above function turns on the led of any Floppy Disk Controller */

/* Ok, lets do something with memory                                  */

port_access.read_write = 0xFF; /* Non-zero value, so it's a read     */
port_access.length = 0x1;      /* 1 byte involved.                   */
port_access.address = 0xF111;  /* Our address.                       */
port_access.value = 0x0000;    /* Store value in Port_access.        */

/* In the above example, we are reading from address >F111 which is  */
/* the address for mapper register 1.                                 */

mem( port_access ); /* call our new function. */

D=port_access.value;
printf("Our Mapper Value %x\n",D);

/* turn off the rs232 handshake lines */
cru_access.read_write = 0x0;
cru_access.length = 0x3;      /* 3 CRU bits to change               */
cru_access.address = 0x130A;  /* CRU Address >130A - primary CTS    */
      cru_access.value = 0x0000;    /* Write 3 zeroes                 */
      crumod( cru_access );         /* de-activate the lines          */

      /* turn off disk controller led  */
      cru_access.read_write = 0x0;
      cru_access.length = 0x0;      /* single CRU bit                 */
      cru_access.address = 0x1100;  /* CRU base of disk controller    */
      cru_access.value = 0x0000;    /* offset 0 bits from base (msbyte) */
                                    /* turn off (lsbyte = 0)          */
      crumod( cru_access );

      exit(0);    /* clean exit, no errors, all files closed */
}
/* L.D.O.M. 03.27.94 */


/* Header file for NSTDIO_H - For Non-Standard I/O by Dan H. Eicher */
/*                                                  and Jeff White */
/* Two functions are defined here.                                 */
/* MEM(A,B,C,D); - Write directly to an address in memory.         */
/*   A = Read/Write Flag. 0=Write, anything else is a read.        */
/*   B = Number of bytes to access at this location.               */
/*   C = Address to access bytes.                                  */
/*   D = Value to store last two/or odd bytes                      */
/*                                                                 */
/* CRUMOD(A,B,C,D);                                                */
/*   A = Read/Write Flag. 0=Write, anything else is a read.        */
/*   B = Single/Multiple bit CRU Operation. 0,1 = Single Bit,      */
/*       greater than 1 is multiple-bit (maximum 16 accessed)      */
/*   C = Base CRU Address.                                         */
/*   D = Location of desired CRU value(s)                          */
/*                              MSbyte          LSbyte             */
/*      single bit operation    offset from base  0=off 1=on       */
/*      multiple bit operation  packed CRU bits word-justified     */
/*                                                                 */

void mem();
int  crumod();

struct rw_port_block {
                    char read_write; /* read or write? */
                    char length; /* number of accesses */
                    int  address; /* address */
                    int  value; /* value */
                   };

/* mem is called as follows:

   int mem( struct rw_port_block *myparams );

*/

/* crumod is called as follows:

   int crumod( struct rw_port_block *myparams );

*/
/* Why did we call this NSTDIO_H? It stands for NON-Standard IO!!! */
/* L.D.O.M. 03.27.94                                               */
```

```
WS     BSS  32      Our workspace.
SAVR11 DATA 0       Save R11 of caller here to restore upon exit.
RWFLAG BYTE 0       This block duplicates our structure, so as to
LENGTH BYTE 0       preserve our calling parameters.
ADDR   DATA 0
VALUE  DATA 0
```

(See Page 15)

ANATOMY OF A HEADER FILE—

(Continued from Page 14)

```
MEM    EQU  $              Label for our def statment to point to.
       LIMI 0              Disable interupts.
       MOV  R11,@SAVR11     Save R11 of caller.
       BL   @GETPAR        GoSub GETPAR
       BLWP @MEMAXS         GOSUB MEMAXS
       BL   @PUTPAR        GOSUB PUTPAR
       MOV  @SAVR11,R11     Restore R11.
       LIMI 2              Enable interupts.
       B    *R13           Return to main program.

MEMAXS DATA WS              Defines workspace for the sub program.
       DATA MEMPC           Defines first instruction for this subprogram.
MEMPC  MOVB @LENGTH,R0       Copy number of iterations to Register 0.
       SRL  R0,8           Move it to the least significant byte.
       MOV  @ADDR,R1        Put the Source/Target address into register 1.
       LI   R2,VALUE        Point to address VALUE with register 2.
       MOVB @RWFLAG,@RWFLAG  Is RWFLAG=0, therefore being a write to memory?
       JEQ  WRTMEM          Yes, go write.
       BL   @READ           Nope, lets READ!
       RTWP                All Done Here. Return and Restore former WP&PC
WRTMEM BL   @WRITE           Perform subroutines to do a write.
       RTWP                All Done Here. Return and Restore former WP&PC

READ   MOVB *R1,*R2          Move Byte from the address in R1 to address in R2
       SWPB *R2             Swap Bytes at that address.
       DEC  R0             Decrement read counter by 1.
       JNE  READ            If not zero, do loop again.
       RT                  All done, return home.

WRITE  SWPB *R2              Swap Byte at the address pointed to by R2.
       MOVB *R2,*R1         Move Byte at address held in R2 to R1 held address
       DEC  R0             Subtract 1 from number of times to write.
       JNE  WRITE           If not zero, do loop again.
WREX   RT                  All done, return home.

CRUMOD EQU $                Label for Def to point to.
       LIMI 0              Disable interupts, until we are finished.
       MOV  R11,@SAVR11     Save R11 of caller.
       BL   @GETPAR         Subroutine to get parameters.
       BLWP @CRUAXS         Our program to Access CRU address.
       BL   @PUTPAR         Subroutine to pass parameters back.
       MOV  @SAVR11,R11     Restore R11.
       LIMI 2              Enable interupts.
       B    *R13           Return to main program.

CRUAXS DATA WS              Our working registers.
       DATA CRUPC           Our program counter.
CRUPC  MOV  @ADDR,R12        Move base address into register 12.
       MOVB @LENGTH,R0      Move Length of transfer in # of Bits.
       SRL  R0,8           Move it to Least Significant Byte.
       JNE  CMPTO1          If it is not zero, check if it is 1.
SNGBIT MOVB @VALUE,R1        Move value to be read/written to Register 1
       SRA  R1,8           Adjust "for CRU usage".
       A    R1,R1          Double offset, "for CRU usage".
       A    R1,R12         Get address by adding offset + base address
       BL   @SNGCRU         Branch to single CRU operation.
       RTWP                All done here, Return and restore old PC & WP.
CMPTO1 CI   R0,1             Compare Value of Register 1 to the number 1.
       JEQ  SNGBIT          If equal, then it is a single bit cru operation.
       BL   @MLTCRU         else, multiple CRU operation.
       RTWP                All done here, Return and restore old PC & WP.

SNGCRU MOVB @RWFLAG,R1       Move read/write flag to Register 1
       JEQ  WRTBIT          If R1 = 0 then goto write bit....
       TB   0              Else... We are testing the status of this bit.
       STST R0             Store the Status (0 or 1) in Register 0.
       ANDI R0,>2000        Clear all but the EQual bit ( if set )
       SRL  R0,5            Left justify in MSbyte
       MOVB R0,@VALUE+1     Store 0 or 1
       RT                  End of this routine, return to caller.
WRTBIT MOVB @VALUE+1,R1      Test for a zero value.
       JEQ  CLRBIT          If bit to be written is 0, goto CLRBIT
       SBO  0              Set that CRU bit ON.
       RT                  Return to caller.
CLRBIT SBZ  0               Set that CRU bit Off....
       RT                  Return to caller.

MLTCRU MOVB @RWFLAG,R1       Set Status Word's Zero pointer.
       JEQ  WRBITS          If it is zero, then we are going to write bits.
       STCR R1,0            Else STore CRu bits in register 1.
       CI   R0,8           Compare R0, is number of bits to be stored > 8?
       JGT  RMLTEX          If it is go to RMLTEX
       SRL  R1,8            If =  or < 8, move CRU bits to least significant
*                          byte.
RMLTEX MOV  R1,@VALUE        Store the CRU bits.
       RT

WRBITS MOV  @VALUE,R1        Get bits to write in register 1
       CI   R0,16           Is it less than 16?
       JL   NOTALL          Yes, go to next check.
       CLR  R0             All 16 bits to write.
       LDCR R1,0            Write CRU bits.
       RT                  Return to caller.
NOTALL CI   R0,8             Is it greater than 8.
       JGT  LOADIT          Yes, go write bits
       SWPB R1             Get right-justified byte ready.
LOADIT LDCR R1,0             Write out CRU bits.
       RT                  Return to caller.

GETPAR MOV  @-12(R7),R0      This is a generic routine to get our values
       MOVB *R0+,@RWFLAG     passed from the main routine, into our
       MOVB *R0+,@LENGTH     temporary values. This keeps us from walking
       MOV  *R0+,@ADDR       on the values in our structure as they are
       MOV  *R0,@VALUE       used.
       RT

PUTPAR EQU  $               This routine will copy results from this
       MOV  @-12(R7),R0      program ( our data area ), back to the
       MOVB @RWFLAG,*R0+     structure of the calling routines.
       MOVB @LENGTH,*R0+
       MOV  @ADDR,*R0+
       MOV  @VALUE,*R0
       RT
       DEF CRUMOD,MEM,RWFLAG,LENGTH,ADDR,VALUE      ( See Note )
       END
*
* You may be wondering why we have included the ability to have a multiplica-
tion
* factor applied to a single memory address, without autoincrementing to the
* next address? This was provided as a way to read and write memory mapped
* devices like grom and VDP ports, where you wouldn't want to increment to
* the next address.
*
* The statement "for CRU usage" seems somewhat vague. In assembler, in
* order to read the fire button, which is at CRU address >3, you could
* left-justify the 15-bit CRU address into R12.  This is the same as
* doubling it or shifting it left 1 bit.  In general, a 15-bit CRU base
* address is left justified in R12.  If the joystick fire button is to
* be tested at the CRU base address with "TB 0" (test CRU bit at 0-offset
* from base), R12 should be loaded with >3 times 2, or >6.  If the CRU
* base R12 were loaded with >0, the fire button would be tested with "TB 3"
* (test CRU bit at 3 bits from base).
*
* The DEF statement must have these values defined or Header/Link-Loader
* will not be able to find the there names.
*
* L.D.O.M. - 03.27.94
```

Back to basics

Memory defined

By R.A. GREEN

The following article has appeared in several user group newsletters. The author is from Ontario, Canada.—Ed.

In this article we will have a look at the memory system of the TI99/4A. You have all heard the terms ROM , GROM, GRAM and RAM used constantly in conversations at user group meetings. Maybe you are confused by this jargon, or are not quite sure what they all mean. I hope things will be clearer by the ned of this article.

To begin, there are three different kinds of memory in the TI. They are:

CPU memory
Video display memory
Graphics memory

CPU memory is the most important. It is the only memory that belongs to the 9900 microprocessor (the CPU). All machine language programs must be in CPU memory before they can be executed by the CPU.

Graphics memory is the memory where the Graphics Programming Language (GPL) programs are stored. Like the video display memory, graphics memory cannot be accessed directly by the CPU.

Video display memory is the memory used mainly to generate the TV picture. It belongs to the Video Display Processor (VDP). Video display memory cannot be accessed directly by the CPU. The CPU must ask the VDP to fetch or store bytes in the video display memory.

Graphics memory is the memory where the Graphics Programming Language (GPL) programs are stored. Like the video display memory, graphics memory cannot be accessed directly by the CPU. The CPU must ask for bytes to be fetched or stored one at a time. This type of memory is unique to TI. I assume it was intended to help prevent — dare I say it — pirating.

Now we have looked at the three kinds of memory in the TI99/4A. Each of the three kinds of memory could exist in two forms: Read Only, which cannot be destroyed, or Read Write, which can be changed.

Now, we have a small problem with terminology. Originally, in the computing field, RAM meant Random Access Memory. However, since the advent of the micro/home computers, the term has come to mean Read/Write Random Access Memory. The term ROM is used for Read Only Random Access Memory.

We have taken a few steps along the way to understanding and now is the time for a little table to organize our thoughts:

#	Owner	Type	Comment
1	CPU	ROM	read only
2	CPU	RAM	read/write
3	VDP	ROM	read only
4	VDP	RAM	read/write
5	Graphics	ROM	read only
6	Graphics	RAM	read/write

Oops! In the TI99/4A, there is no VDP ROM, although there could be. It might be used to hold the character definitions as is done in some other micros, such as the Commodore Vic 20 and IBM PC.

To refine our terminology a bit, let's look at the common terms used by the TI enthusiast:

ROM — means the CPU ROM
RAM — VDP RAM
CPU RAM — means the CPU RAM (strangely enough)
GROM — means Graphics read only memory
GRAM — means Graphics read/write memory

Let's redo our table just to keep things straight:

#	Owner	Type	Common Name
1	CPU	ROM	ROM
2	CPU	RAM	CPU RAM
3	VDP	RAM	VDP RAM
4	Graphics	ROM	GROM
5	Graphics	RAM	GRAM

Remember that the CPU, the 9900 microprocessor, can only execute programs in the CPU ROM or CPU RAM. Anything in the other kinds of memory is just data for some program executing in the CPU.

Now we know the five kinds of memory in the TI99/4A, and we have the terminology to talke about them. This is a good time to ask, "Where is all this memory?"

Here are some answers:

Console (bare bones, with no expansion)

CPU ROM — 8K bytes
CPU RAM — 256 bytes
VDP RAM — 16K bytes
GROM — 18K bytes

Cartridge (if it were full)

CPU ROM — 8K bytes
GROM — 30K bytes

Memory expansion

CPU RAM — 32K bytes

I/O devices (RS232 and disk)

CPU ROM — 8K bytes

Now that we are taking sizes, let's talk about the maximum sizes. As we all know, and like to brag about, the TI99/4A is a 16-bit machine. This means that it can count (or address) bytes from zero to 65536 (64K). So we say it can address 64K or memory (i.e. CPU memory). The TI99/4A also uses a 16-bit word to tell the VDP which byte it wants. Two bits of this word are used as indicators so that there is only 14 bits of address. So the maximum VDP memory is 16K bytes. a 16-bit address is also used for Graphics memory (with no indicators) as its maximum is again 64K.

(See Page 17)

MEMORY DEFINED—

(Continued from Page 16)

We are now ready to take another step, and find out what is in all this memory.

CPU ROM contains three main things, all written in assember:

- The low-level operating system. This code is necessary to make the machine do anything.
- The interpreter for GPL programs.
- The Device Service Routines (DSR) which provide high level access to the input/output devices.

GROM contains three main things:

- The high level operating system, written in the GPL language. The high level operating system gives you the master title screen, the first level menus and provides for calling other programs written in the various languages.
- The TI BASIC interpreter, written in GPL.
- Tables of data for initializing the VDP.

VDP RAM contains three main things:

- The TV screen image, character, color and sprite tables.
- Work areas for programs and device service routines.
- BASIC programs, which are interpreted by the TI BASIC or Extended BASIC interpreter.

CPU RAM contains three main things:

- Work space fo the GPL interpreter (the 256 bytes that are in the console).
- Assembler language programs.
- Extended BASIC programs, which are interpreted by the Extended BASIC interpreter.

So there you are! We have looked at the kinds of memory, the owner of the memory, where the memory is, what is in the memories, and the terminology used to talk about all of this. Of course, there are a few things we didn't look into, such as bank switching. But can wait for another day.

Extended BASIC programs resequence, rearrange fields in text files

The following article was written by the late Jim (Tigercub) Peterson. It is one of the many Tips from the Tigercub columns he wrote over the years. This was No. 63 in the series.—Ed.

Several articles have been published on the subject of using Funlweb as a simple fixed-field database. Sometimes you might want to rearrange the sequence of fields in such a file. This mini-program will quickly change the position of any field in a D/V80 file.

FIELDSWITCHER

```
100 DISPLAY AT(3,8)ERASE ALL
:"FIELDSWITCHER":"":"      b
y Jim Peterson":"":"To chang
e sequence of fieldsin a DV8
0 fixed fieldfile   created
by Funlweb or other means" !
226
110 DISPLAY AT(23,6):"PRESS
ANY KEY" :: DISPLAY AT(23,6)
:"press any key" :: CALL KEY
(0,K,S):: IF S=0 THEN 110 EL
SE CALL CLEAR !118
120 DISPLAY AT(8,1):"FILENAM
E? DSK" :: ACCEPT AT(8,14):F
$ !061
130 OPEN #1:"DSK"&F$,INPUT !
094
140 DISPLAY AT(12,1):"MOVE F
IELD STARTING AT WHAT POSITI
ON?" :: ACCEPT AT(13,11)VALI
DATE(DIGIT):N !133
150 DISPLAY AT(15,1):"LENGTH
 OF FIELD?" :: ACCEPT AT(15,
18)VALIDATE(DIGIT)BEEP:L !12
6
160 DISPLAY AT(17,1):"TO WHA
T POSITION?" :: ACCEPT AT(17
,19)VALIDATE(DIGIT)BEEP:T !0
29
170 IF T>N+L-1 OR T<N THEN 1
90 !001
180 CALL SOUND(400,110,0,-4,
0):: DISPLAY AT(23,1)BEEP:"C
ANNOT MOVE FIELD WITHIN ITSO
WN PARAMETERS!" :: GOTO 140
!178
190 DISPLAY AT(19,1):"OUTPUT
 FILENAME? DSK" :: ACCEPT AT
(19,21)BEEP:OF$ !246
200 OPEN #2:"DSK"&OF$,OUTPUT
 !019
210 LINPUT #1:M$ :: M$=M$&RP
T$(" ",80-LEN(M$)):: IF T<N
THEN M$=SEG$(M$,1,T-1)&SEG$(
M$,N,L)&SEG$(M$,T+1,255)!085
220 IF T>N THEN M$=SEG$(M$,1
,N-1)&SEG$(M$,N+L,T-N-L)&SEG
$(M$,N,L)&SEG$(M$,T+1,255)!0
62
230 PRINT #2:M$ :: IF EOF(1)
<>1 THEN 210 ELSE CLOSE #1 :
: CLOSE #2 !231
240 DISPLAY AT(12,1)ERASE AL
L:"ANOTHER? Y/N" :: ACCEPT A
T(12,14)VALIDATE("YN")SIZE(1
)BEEP:Q$ :: IF Q$="Y" THEN 1
20 ELSE CALL CLEAR :: STOP !
016
```

And this one will make it easy to completely rearrange the sequence of any number of fields.

REARRANGER

```
100 DISPLAY AT(3,9)ERASE ALL
:"REARRANGER":"":" by Jim Pe
terson" !173
110 DISPLAY AT(7,1):" To rea
rrange the sequence  of fiel
ds in a DV80 file of fixed f
ields created by     Funlweb
 or otherwise." !118
120 DISPLAY AT(24,7):"PRESS
```

(See Page 18)

TIGERCUB—

(Continued from Page 17)

```
ANY KEY" :: DISPLAY AT(24,7)
:"press any key" :: CALL KEY
(0,K,@):: IF @=0 THEN 120 !0
12
130 DIM L(20),S(20),F$(20)::
 CALL CLEAR !023
140 DISPLAY AT(8,1):"INPUT F
ILENAME?":"":"DSK" :: ACCEPT
 AT(10,4)BEEP:I$ :: OPEN #1:
"DSK"&I$,INPUT !150
150 DISPLAY AT(12,1):"OUTPUT
 FILENAME?":"":"DSK" :: ACCE
PT AT(14,4)BEEP:O$ :: OPEN #
2:"DSK"&O$,OUTPUT !154
160 DISPLAY AT(16,1):"HOW MA
NY FIELDS?" :: ACCEPT AT(16,
18)VALIDATE(DIGIT)SIZE(2):F
:: CALL CLEAR !081
170 FOR J=1 TO F :: DISPLAY
AT(12,1):"FIELD #";J;"LENGTH
?" :: ACCEPT AT(12,20)VALIDA
TE(DIGIT)BEEP:L(J):: NEXT J
:: FOR J=1 TO F !125
180 DISPLAY AT(12,1):"IN FIE
LD #";J:"":"PLACE FIELD #" :
: ACCEPT AT(14,15)VALIDATE(D
IGIT)BEEP:S(J)!130
190 IF S(J)<1 OR S(J)>F THEN
 CALL SOUND(300,110,0,-4,0):
: GOTO 180 !222
200 IF POS(E$,CHR$(S(J)),1)=
0 THEN E$=E$&CHR$(S(J)):: GO
TO 220 !137
210 CALL SOUND(300,110,0,-4,
0):: DISPLAY AT(16,1):"FIELD
#";S(J):"HAS ALREADY BEEN PL
ACED!" :: GOTO 180 !249
220 NEXT J !224
230 LINPUT #1:M$ :: M$=M$&RP
T$(" ",80-LEN(M$)):: P=1 ::
FOR J=1 TO F !194
240 F$(J)=SEG$(M$,P,L(J))::
P=P+L(J):: NEXT J !084
250 FOR J=1 TO F :: N$=N$&F$
(S(J)):: NEXT J :: PRINT #2:
N$ :: N$="" !090
260 IF EOF(1)<>1 THEN 230 EL
SE CLOSE #1 :: CLOSE #2 :: S
TOP !217
```

If you need to use either of these programs on files with a record length other than 80, just add VARIABLE (or FIXED) and the record length to all the file opening statements, and change the "80" in line 230.

The following subprogram, in which X=28 for a 28-column screen or whatever width you want, will reformat a string of almost any length to print on screen without breaking words, and will return in L the number of lines required to print it, which can be used to space DISPLAY AT statements:

FORMAT

```
31993 SUB FORMAT(X,M$,L):: Y
=X !017
31994 IF LEN(M$)<Y+1 THEN 31
996 ELSE IF LEN(M$)<Y+X+1 AN
D SEG$(M$,Y,1)=" " THEN 3199
6 ELSE IF LEN(M$)<Y+X+1 AND
SEG$(M$,Y+1,1)=" " THEN 3199
6 ELSE P=Y-1 !036
31995 IF P<1 THEN 31996 ELSE
 IF SEG$(M$,P,1)=" " THEN M$
=SEG$(M$,1,P)&RPT$(" ",Y-P)&
SEG$(M$,P+1,255):: Y=Y+X ::
GOTO 31994 ELSE P=P-1 :: GOT
O 31995 !091
31996 L=INT(LEN(M$)/X)-(LEN(
M$)/X<>INT(LEN(M$>/X)):: SUB
END !209
```

The following little program, plus the magic of Funlweb, should be all the mailing list program that most people would need for home use. Just use Funlweb to create a file with name on the first line, address on the second line, city and state on the third; or use four or even five lines for the address if you need to, but the sixth line must be either blank or contain selection codes. These codes can be anything you want, such as "C" for everyone you want to send a Christmas card to, or B11 to send a birthday card in November, or whatever.

You can put as many codes as you want on the line, separated or strung together. But be sure not to use a code that is part of another code. For instance, if you use B11 for those November birthdays, don't us "B" or "1" or "B1" or "11" for something else.

Then continue with the next address in another block of six lines. Just be sure that the line number of the line just above the first address line is always a multiple of six.

FIELDSWITCHER

```
100 DISPLAY AT(12,1)ERASE AL
L:"Filename? DSK" :: ACCEPT
AT(12,14)BEEP:F$ :: OPEN #1:
"DSK"&F$,INPUT :: OPEN #2:"P
IO" !158
110 DISPLAY AT(14,1):"Print
addresses with code -":"":"(
to print all addresses,    j
ust press Enter)" !251
120 ACCEPT AT(15,1)BEEP:X$ !
253
130 LINPUT #1:A$ :: LINPUT #
1:B$ :: LINPUT #1:C$ :: LINP
UT #1:D$ :: LINPUT #1:E$ ::
LINPUT #1:F$ !251
140 IF POS(F$,X$,1)<>0 OR X$
="" THEN PRINT #2:A$:B$:C$:D
$:"":"" !237
150 IF EOF(1)<>1 THEN 130 EL
SE CLOSE #1 !065
```

In Tips No. 62 I reported on the weird behavior of the CALL LOAD(-31961,149), when used to clear all defaults and search for a LOAD file on DSK1. I have since found that if you put this call at the beginning of a program, it will not execute until an END or STOP is reached — but if you break the program with FCTN-4, it will not be in memory!

I stated that after this CALL LOAD was executed, any number taken to the power of 0 (which should be a value of 1) acquired a value of 220.5727273. I was led astray by the INT in the formula in which I first found this puzzle. Actually, it is 220.57000101, which prints to the screen in the peculiar format Fo.57000101.

If a number between one and nine is added to that, it is printed as 1< followed by the number being added, followed by the decimal part. For a number between 10 and 19, the < is changed to = and between 20 and 29 it becomes > (note the ASCII sequence). From 30 to 35 it becomes a question mark but from 36 to 99 the decimal portion is preceded by 0 to 63, respectively. One hundred is 2<0.570001 and the pattern continues.

Although these are not valid representations of numbers, they are treated as such. Run a program to give N the power of 2^0, then break the program and experiment in immediate mode.

Print N gives that strange F0.57000101. Print N+1, or whatever, gives values rep-

(See Page 19)

TIGERCUB—

(Continued from Page 18)

resented in the format described above. PRINT N*1 will give the true numeric value 220.57000101 but multiplying by other values gave me results in the odd format, as did dividing.

Peter Walker pointed out to me that trying to subtract from N within a program resulted in printing a value followed by a crash reporting a Syntax Error (in the line which had just been executed!) followed by a jump to a non-existent line zero!

N-1 should be 219.57 ... of course, but in immediate mode PRINT N-1 results in 63.57000101. In the format in which added values are printed, this would be 319.57000101 but the 63 ... is actually a decimal value, as can be proved by PRINT CHR$(INT(N-1)). When I tried to get a zero value by PRINT N-64.57000101, the computer blew its mind.

An IBM program called DOC-SMASH, which sells for about $35, will read a D/V80 file and output it to a printer in full carriage-width lines of elite condensed subscript, thereby getting up to 216 lines per page. Bud Wright wrote a TI version, with assembly links, to let us do the same thing for free. His version wouldn't work on my trusty, old Gemini 10X, which does not support condensed elite, so I wrote this mini-program. It is not as fast as Bud's, but it does the job.

TEXTSMASHER

```
100 DISPLAY AT(3,9)ERASE ALL
:"TEXTSMASHER":"":"For the G
emini 10X printer, to print
D/V80 text in linesof 136 ch
aracters closely  spaced, i
n subscript." !183
110 DISPLAY AT(20,1):"Press
Enter to end input" :: DIM F
$(20)!080
120 F=F+1 :: DISPLAY AT(12,1
):"FILE #";F:"DSK" :: ACCEPT
 AT(13,4)BEEP:F$(F):: IF F$(
F)<>"" THEN 120 !027
130 OPEN #2:"PIO",VARIABLE 2
55 :: PRINT #2:CHR$(27)&CHR$
(83)&CHR$(1)!113
140 PRINT #2:CHR$(15)&CHR$(2
7)&CHR$(51)&CHR$(12);:: LN=1
36 !048
150 FOR J=1 TO F-1 :: OPEN #
1:"DSK"&F$(J),INPUT !216
160 LINPUT #1:M$ !199
170 IF LEN(T$)>0 THEN M$=T$&
" "&M$ :: T$="" !150
180 IF LEN(M$)<LN+1 AND POS(
M$,CHR$(13),1)<>0 THEN PRINT
 #2:M$ :: GOSUB 260 :: M$=""
 :: GOTO 230 !118
190 IF LEN(M$)=LN THEN PRINT
 #2:M$ :: GOSUB 260 :: M$=""
 :: GOTO 230 !255
200 IF LEN(M$)<LN AND EOF(1)
<>1 THEN LINPUT #1:X$ :: M$=
M$&" "&X$ :: GOTO 170 ELSE I
F LEN(M$)<136 THEN PRINT #2:
M$ :: GOSUB 260 :: GOTO 240
!198
210 P=LN !168
220 IF SEG$(M$,P,1)=" " THEN
 T$=SEG$(M$,P+1,255):: M$=SE
G$(M$,1,P):: PRINT #2:M$ ::
GOSUB 260 :: M$="" :: GOTO 2
30 ELSE P=P-1 :: GOTO 220 !1
65
230 IF LEN(T$)<LN+1 AND POS(
T$,CHR$(13),1)<>0 THEN PRINT
 #2:T$ :: GOSUB 260 :: T$=""
 !219
240 IF EOF(1)<>1 THEN 160 !0
71
250 CLOSE #1 :: NEXT J :: ST
OP !019
260 X=X+1 :: IF X<121 THEN R
ETURN !221
270 X=0 :: FOR K=1 TO 8 :: P
RINT #2 :: NEXT K :: RETURN
!085
```

For that to work properly, your paragraphs must end in carriage returns, and so must the title line, etc. If such is not the case, try Bill Wood's method — load the file into Funlweb, enter RS for Replace String, then /. /.X/ but instead of X type CTRL-U Shift-M. At the first prompt, enter A for All. If your text has any paragraphs ending in question marks or exclamation points, get your cursor back to the beginning, change that first period to ? or !, and do it again. You might also need to manually add carriage returns to titles, etc. Just type CTRL-U, then Shift-M wherever a CR is needed.

Without having printers to test it on, I think the program can be modified for the SG-10 by changing line 140 to:

```
140 PRINT #2:CHR$(27)&"B"&CH
R$(4)&CHR$(27)&CHR$(51)&CHR$
(12);:: LN=160
```

And for old Epson-type printers that don't support elite condensed, try the following line:

```
140 PRINT #2:CHR$(27)&CHR$(7
7)&CHR$(27)&CHR$(51)&CHR$(18
);:: LN=132
```

And for new Epson compatibles, try this:

```
140 PRINT #2:CHR$(27)&CHR$(7
7)&CHR$(15)&CHR$(27)&CHR$(51
)&CHR$(18);:: LN=160
```

You might also have to change that 8 to 12 in line 270 — my old Gemini seems to think that 11*12=128.

COMPLETELY out of memory

Jim Peterson

1994 TI FAIRS

MAY

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

NOVEMBER

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

1995 TI FAIRS

FEBRUARY

Fest West '95, Feb. 18, Fabulous Inn, San Diego, California. Contact Southern California Computer Group, P.O. Box 152535, San Diego,CA 92195, or call the SCCG BBS, (619)263-9125, User No. 25, password FEST.

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

TI-Writer

Using Mail Merge can save time when you've got a lot of letters to send

By CHICK DE MARTI

The following article has appeared in user group newsletters.—Ed.

A TI-Writer Mail Merge is a convenient way to mail a form letter to many people without manually typing in the names, addresses, etc. This is extremely useful when various forms of communication must be sent to the same group of people on a monthly, quarterly or other regular basis. So the obvious thing, then, would be to create the mailing list. Here is an example:

```
1 Mr.
2 Tom
3 Jones
4 2341 Any Street
5 Somewhere, CA 91123
6 Jones family
*
1 Ms
2 Jane
3 Smuthers
4 7777 Lucky Lane
5 Sameplays, CA 91119
6 Smuthers family
*
1 Mrs.
2 Eunes
3 Somuch
4 2468 'Preciate Ct.
5 Fairytale, CA 91121
6 Somuch family
*
```

You'll have to press Enter after each line. Yes, the numbers 1-6 must be entered. Notice the asterisk separating each group of names. TI-Writer recognizes it as a separator.

Save the file as DSKx.MYFILE/1.

Next, let's create a form letter. This is the one I created for a user group meeting. A typical head would look similar to this:

```
.FI;AD;LM 8; RM 72
.IN +32
Chick De Marti
18028 Falda Ave.
Torrance, CA 90504
<blank line>
August 26, 1991
.IN +0
<blank line>
<blank line>
*1* *2* *3*
*4*
*5*
<blank line>
```

The +32 will print the lines that follow near the center of the page.

This cancels the +32 indent.

1=Title, 2=first name, 3=last name; 4=address; 5=city, state and ZIP code.

```
Dear *2*,
<blank line>
```

Now you would type the body of the letter, for example:

```
  This year we are going to have many in-
teresting demonstrations, and surprise
guests. We expect some of the leading
programmers in the TI community to be
displaying their wares, as well as their
expertise. There will be many bargains
available in software, as well as some
'steals' in hardware.
  It would be my pleasure to greet the
*6* at the door personally. So do come to
the meetings and enjoy yourself, as well
as learn many interesting things.
<blank line>
.IN +32
Yours truly,
Chick De Marti
```

Save this form letter as DSKx.FORMLET/1.

This is about as simple as it can be. By doing the mailing list the TI-Writer editor, you can make changes, corrections, as well as additions with little or no trouble.

Note that when you are typing your form letter, before leaving the editor, make a quick copy of it using the Print File command. The resulting copy will include all 'dot' commands and will allow you to check the finished copy with your original notes.

To make a Print File copy while still in the editor, press FCTN-9 to get to the command line, type PF and press Enter, type PIO and press Enter to print a copy of the form letter. There's no need to add an LF (line feed) after PIO.

PRINTING THE LETTERS

Printing the letters is very easy. Get into the TI-Writer formatter and follow these prompts:

Program	DSKx.FORMLET/1
Printer	PIO.LF
Use mail list?	Y
Page(s)/All	A
Pause after each page?	N
Mail list names	DSKx.MYFILE/1

That's all there is to it. When the letters are printed, TI-Writer will replace *1* *2* *3* with *Mr. Tom Jones* and the *4* with his address. The *5* will be replaced by the city, state and ZIP code, and the *6* will insert *Jones family* to help personalize the letter even more. Neat, huh?

ANOTHER NEAT FEATURE

TI-Writer has another feature you might like to try. It is a manual insert option. Using this option you could insert personal information anywhere within the letter. This could be used by a busi-

(See Page 21)

MAIL MERGE—

(Continued from Page 20)

ness for billing purposes, or for a club to make its members aware of changes, awards, etc. What follows is a quick how-to.

Let's make a minor change to our mailing list. Load MYFILE/1 into your editor and then delete line the lines that begin with the number 6, i.e. Jones family, etc. When finished, resave the file as DSKx.MYFILE/2.

Now load FORMLET/1.

Normally, this would probably be a different heading, but for the sake of expediency let's use what we have. Go to the line with Dear *2*. Change it to Dear *1* *3* (Mr. Jones). Now, to the letter itself. Type in the following:

```
  Thank you for your recent payment. Your
balance is *6*. If you have any ques-
tions, please give my office a call.
.IN +32
Yours truly,
Ida Dungood
SMALL LOAN CO.
```

Delete the rest of the text from the letter and save it as DSKx.FORMLET/2. Don't forget to make a copy using Print File.

TI-Writer matches all numbers within the asterisks with the respective lines in your mailing list. If it does not find the corresponding number, it halts the printing and asks for your input. Now print your letters, answering the prompts as you did for FORMLET/1. The printer will stop and at the message *Enter input:* enter the following: $1150.90. When you read the letter, the second sentence will read: "Your balance is $1150.90."

There doesn't seem to be any limit to the number of inputs you may have. With that in mind, another opportunity to use the Mail Merge might be at golf or other tournaments. Still using the mailing list MYFILE/2, type in the following form letter:

```
Dear *2*,
<blank line>
  Thank you for participating in the Ju-
nior/Senior Golf Tourney. Your official
handicapped score was *6*, earning you
*7* as your portion of the 'Winner's
Booty.' Looking forward to seeing you
again next year.
.IN +32
Bobby Sox, treas.
Good Guys Golf Gang
```

The inputs in the above example are obvious, as you can see. The occasions to use a Mail Merge are endless. So try it out for kicks, that's how I learned how to use Mail Merge.

AEMS Macro Assembler/Linker

A major new tool

REVIEW

By BRUCE HARRISON

When writing a review, there's always the old problem of whether to start with the good news or the bad news. This time the coin came up tails, so your curmudgeonly author will start with the good news. Art Green still knows how to write an excellent assembler. For some years now, we've been using his RAG Macro Assembler (Version 8) as our one and only means of assembling source files. Even for somebody who never uses a macro, Art's assembler has earned its keep, in our opinion, by the superb way it reports errors on the screen. The new one is, if anything, even better in this respect than his earlier version. As part of our testing, we took one of our own source files and purposely changed the spelling of a label reference. CLR @FKLCNT became CLR @FLKCNT. That seemed a good test case, as transposing letters in a label name is probably our most frequent "detectable at Assembly" error. This error is in a file that is "copied" by the main assembler source file, not in the main file itself. Sure enough, the AEMS Macro Assembler reported as follows:

LINE 10 COMPCD2
FLKCNT UNDEF SYMBOL

That's really superb error reporting, since it not only tells us what's wrong, but shows the offending symbol and tells in which line of which file the error was found. Nobody who's into Assembly programming should have any trouble using this product, once it's up and running, that is.

THE BAD NEWS

(See Page 22)

REPORT CARD

Performance	**A**
Ease of Use	**B-**
Documentation	**A-**
Value*	**?**
Final Grade	**A-**

Cost: $119.95

Publisher: Asgard Peripherals — Asgard is out of the TI business, though the AMS system may be available second hand from other users. Rights to the system have been granted to an unnamed person or company.

Requirements: TI-99/4A Console, Monitor, DSDD disk drive, AMS, AEMS or AMS2 memory system.

* Value is hard to assess - see text.

AEMS MACRO ASSEMBLER/LINKER—

(Continued from Page 21)

You knew this was coming, so here's the first part of it. When Laura Burns first asked me to review this product, I asked about the hardware required to use this Assembler, namely the Asgard Expanded Memory System (AEMS). Yes, John Koloen had the card, and agreed to loan it to me for the review. There's the first part of the bad news, that this Assembler is totally useless to anyone who does not have the AEMS installed in his Peripheral Expansion Box.

The package arrived, with the card and two disks. One disk in the package was an obviously incomplete early version of something, but no assembler. The second disk would not catalog on any of the four disk drives on our two TI machines. That, as we suspected, was because this disk (the "correct" version of the package) was a DSDD disk, which our TI controllers can't handle. Fortunately, in the box with the card and disks was a slip of paper (very elegant looking) with a phone number for Asgard. From the area code, I guessed that this number would be answered by Chris Bobbitt, and I was right.

SOME MORE GOOD NEWS

Chris quickly agreed to send me a set of two DSSD disks, and these arrived after the usual mail delay for this area. These disks turned out to be rather tightly packed with files, most of them archived. A copy of the Archiver had been thoughtfully provided on each disk, and each included a short unarchived README file, so I'd know what I was doing.

Five DSSD disks were required to unpack all the contents of those two disks. Except for an I/O ERROR #6 problem in unpacking the Linker ARCFILE, all went fairly smoothly. We found the primary "doc" file that we sought, and found that there was even a "tailored" version of the TI-Writer Formatter provided on the disk, in which the file name for the Assembler's User's Guide came up as the default in the filename field. (A really nice touch, since I always have trouble remembering file names like AEMSMADUG when the formatter is asking me that question. I'm pretty sure that's not the correct name, but thanks to the pre-defaulted formatter, I didn't need to know.)

GETTING IT STARTED

When the Assembler User's Guide finished printing its modest six pages (a surprise in itself), I started reading. Being the kind of impatient person who never really reads instructions, I skimmed down to the part where it said to run the Assembler from file AMAC1. I failed to notice what was in the line just above that, to wit: "... and the AEMS System Software." Again being impatient, and knowing that the AEMS 128K card was sitting there in my P-Box, I plunged right ahead into Option-5 for DSK1.AMAC1. I/O ERROR 07 happened. Okay, let's try loading it from the RAMdisk menu's RUN option. This loaded up the Assembler, and gave me a nice screen for entering source, object, and so on. Since I had a source file right there on my RAMdisk, I typed in the source and object names, and pressed Enter. BOOM! The screen went blank, then nothing. Only the On-Off switch had any effect. Why? (Hint: see that part above about "failed to notice.")

In my little brown book I keep lots of phone numbers for people in the TI community, so I turned to the Gs and called Art Green in Gloucester, Ontario. Art straightened me out in just a minute. One of those other disks had a file called ABOOT, and that has to be run in order to load and run the Assembler. Sure enough, running ABOOT from Option-5 gave me a menu which included the Macro Assembler, and this allowed me to run the Assembler with complete success. Of course none of my source files includes macros, and none of my programs assembles to anything even close to 128K of mem[illegible] use, but the Assembler seemed to work exactly as expected.

THE LINKER

We said early on in this review that the Assembler is very easy to use. The Linker, on the other hand, is a good deal more complex, even though the user interface is fairly simple. The main reason for this is that the Linker is the part of this product which resolves the matter of "paging" memory, so that apparently "seamless" programs of more than 32K length may operate from the AEMS. The linker's documentation is many pages long, but clearly written, and is designed to allow the user to use only what he needs among its many features.

The only real confusion we suffered was that if only a single object file is to be linked, that must be entered at the prompt for "CONTROL FILE." (Maybe it should say CONTROL/OBJECT.) We've alway[illegible] been able to get along without Linkers or Libraries on the TI but, of course, that's because we've been able to make each program we write into just a single object file, and where necessary have written our own "saver" code to make Option-5 program files from our object code.

The new Linker bears a striking similarity to Art Green's earlier RAG Linker, which was a companion to his RAG Assembler. It offers so many options that it would be easy to get bogged down even trying to describe the possibilities. Suffice it to say that when you're writing programs large enough to need AEMS, you're going to have to study the Linker's docs closely just to get started on the right foot.

We've studied the Linker docs to some extent, and found that the Linker places restrictions on the construction of long pro[illegible] grams. In essence, the linker requires a highly structured approach, using the 4K

(See Page 23)

AEMS MACRO ASSEMBLER/LINKER—

(Continued from Page 22)

module concept, with a strictly vertical "tree" form between the main module and its subroutines. As our regular readers know, this is not the kind of approach we follow in our own programs, but many Assembly writers do adhere to this method. In the case of programs designed to run on the normal TI, it makes no difference whether the program is organized into modules, nor whether there is any structure at all. Everything will work if the instructions are correctly written. This is definitely not the case for programs written in the AEMS system.

We tried taking one of our own object modules and linking it with the AEMS Linker. To be fair, this object file was designed strictly for Option-3 operation under E/A and, without some more work on the source file, it wouldn't have run under E/A's Option-5 either. The resulting program file would not run in the AEMS environment. Yes, this was a stupid and unfair test, but we mention it just to point out that the transition from the "normal" TI Assembly program to the AEMS environment is not a trivial matter. There will be a learning curve.

THE DOCS

Copious documentation is included with the AEMS package, including reference material on the 9900 Assembly language, detailed instructions for using the Linker, etc. As with any new product, there are some errors in the docs, but these are more an annoyance than a real problem. There is some inconsistency about the status of the Assembler itself. Its User Guide states on the cover page that this manual and the AMES Macro Assembler are copyrighted products of RAG Software, but the very next page says that "This package is being made available via the Fairware concept." Which is right? We are not fond of the Fairware concept, and how can a copyright be enforced when the manual clearly and explicitly encourages the buyer to "... distribute complete copies to your friends." As we understand copyright law, giving the buyer "carte blanche" to copy the product and distribute it would void any future infringement claim by the author. In any case, the Assembler's User Guide is easy to read and mercifully short.

SYMBIOSIS

John sent a short note with the package containing the AEMS card and disks, in which he asked that we review the Assembler and Linker, but not the card, or else make the card a separate review. We're trying to isolate this review to just the Assembler and Linker, but that's difficult because neither can be used without the card and its "system software". The reverse is not entirely true, because a card purchaser could use the system software and other people's programs without ever using the Assembler and Linker which were supplied with the card. Of course nobody can write software that makes use of the card without using the Assembler and Linker, but that's not quite the same thing. As we understand it, Asgard's word processor First Draft/Final Copy has been designed to use the card's memory if that's available, and to run on a "normal TI" if the card is not there. If AEMS becomes very popular, this may become the trend for Assembly authors. We are biased on this point, because there's nothing we've ever wanted to do on the TI that wouldn't fit into the 32K memory with room left over.

A LIMITED AUDIENCE

As the years go by, fewer and fewer Assembly programmers are available to the TI "community". Some subset of this group will be sufficiently interested to purchase AEMS and produce software which takes advantage of its memory capacity. Another subset of the non-programming "TI public" will be owners of the AEMS, and will thus form a customer base for that subset of the Assembly programmers. Our own opinion is that both of these subsets will be small minorities, and that the work of programming for the AEMS will become a "labor of love," not a "profit center."

MINOR FLAWS

Every new product has to have them, and there are a couple we found. The new Assembler starts out with the cursor in the SOURCE FILE input field, but there are default answers already on-screen for the MACROS FILE and OPTIONS fields. Neither of those defaults was to our liking, so we had to back the cursor up the screen and change them. In the documentation, we found that Art provided a "patcher" program that would allow us to change the defaults in the Assembler program, by simply editing the contents of a D/V80 patch file. We tried that, but all we got was a screenful of error messages, the last being "FILE NOT PATCHED." After several unsuccessful tries, we dug out a copy of Funnelweb 4.4 that had John Birdwell's DSKU on it, and used its sector search-replace function to change the entries for the defaults in Art's Assembler. This worked, so that our "working copy" of the new Assembler has our choice of defaults in place. On the older RAG Assembler, there was a simpler installation process which worked quite well to tailor the defaults. Perhaps Art should have kept that older concept.

Another minor flaw was noted in the System Software. There's an ability provided to bring programs into the memory on a "stay resident" basis, and then to restart them later. These resident programs are added to the system software's menu, and stay available while you're in the AEMS system. However, you can exit back to the TI title screen, then re-enter AEMS through a program called ASHOE. The "resident" programs will still be listed on the menu, but will not work.

A final minor annoyance is the presence on the System Menu of certain selections(PEDIT, for example) that are NOT included in the package. We were able to edit the script file that provides these as default selections, but why should the user have to delete things that the vendor left out?

Of course these latter two problems

(See Page 24)

AEMS MACRO ASSEMBLER/LINKER—

(Continued from Page 23)

should not reflect on the Assembler or Linker, but people who use these will be exposed to the AEMS system, so we thought it only fair to include them in this review. We said right at the outset that Art Green has made an excellent Assembler, and we meant that. The Linker will take some significant study to master, but anyone who's writing software large enough to need AEMS should be willing to invest the time to learn the Linker's rules. Our minor gripes about the system software are just that, not by any means to diminish the artistry that went into the AEMS Macro Assembler and Linker.

TERMINOLOGY

Throughout this review, we have used the acronym AEMS for Asgard Expanded Memory System, just to be consistent internally. There is a family of such products, so the names AMS, AEMS, and AMS2 may be found to apply to various members of this family of products. The hardware implementation is somewhat different in the three related products, but what this review has to say about the Assembler and Linker should apply whether the product you're considering goes by the name AMS, AEMS, or AMS2, as the software is common to all three.

FINAL THOUGHTS

The idea appears sound. With this new Assembler/Linker and the capability to greatly expand directly-usable memory, the TI can reach new heights in performance. If enough Assembly authors can be convinced to buy the AEMS package and produce new programs for it, TI users with AEMS will be able to eclipse t[illegible] Geneve, and maybe rival the growing PC and Mac trends. It's really a shame that this couldn't have come along 10 years ago, but at least now the door is open for "Windows" or "Lotus Symphony" systems running on the TI. Maybe somebody will be willing, given this major new tool, to invest the year or two of effort required to create the "Mac Emulator" for the TI.

It's very hard, without a crystal ball, to assess the value of this new Assembler/Linker software, since its principal value lies in what people do with it. By itself, it can't be used on the standard TIs that we own, but given the right environment it has the potential for being the most valuable thing ever created for this machine.

MICRO-REVIEWS

DORTIG Disk No. 2, Kanji Drill, P-GRAM Utilities Version 2

By CHARLES GOOD

DORTIG DISK No. 2 by Dorset TI User Group

To the best of my knowledge there are two TI user groups in the United Kingdom. A national group, chaired by Stephen Shaw, publishes a nice newsletter which is well known to many in the U.S. The smaller group is limited to the county of Dorset, and communicates with the outside world via its humorous disks of public domain utilities. Mr. J. Murphey, Treavor and the other lads of DORTIG have recently released their second such disk.

Everything is in Extended BASIC. When you first DSK1.LOAD you are given the choice of a full introduction with "a lot of rubbish" or a menu of applications with "not so much rubbish." The "lot of rubbish" choice is so funny that the first time I saw it I burst out laughing several times during its execution. I was in my living room at the time, and the rest of my family wanted to know what was so funny. I couldn't describe in words the cause of my chortles. All I could say was, "Come here and look at this thing."

Here is the "lot of rubbish" scenario. Text of the DORTIG lads' conversation appears at the bottom of the otherwise black screen. You quickly figure out that the lads are *inside* the disk drive. Ten pairs of eyes appear. It is determined that someone out there wants to see a menu on the disk. One of the lads is assigned the task of putting the needle on the disk. Another is supposed to work the little red disk drive light. The rest get on the tread mill to start spinning the disk. You see the eyes of the lads moving faster and faster across the screen as the disk gradually gets up to speed. Finally the (on screen) red light starts to glow and an application menu appears. This all is really quite funny, with a definitely British flavor (oops, flavour) to the humor.

In addition to several graphic demos, there are some useful applications on the disk. My favorite is the program to randomly assign matches to teams in a round robin bowling tournament. This is for British lawn bowling, not our familiar 10-pin bowling, but it will work for any multi-game league team sporting event such as baseball or basketball. The soccer World Cup series, currently touring the United States, could have used this software to determine which country plays which in each round. You enter the names of the teams and the number of games each team is to play. The computer then randomly creates play lists matching teams against each other for the required number of games. This may be a unique application in the TI world. I don`t remember seeing another TI program that does this. The DORTIG No. 2 disk also has a nice program to keep track of automobile expenses and gasoline (oops, petrol) mileage on a monthly basis. Ac-

(See Page 25)

MICRO-REVIEWS—

(Continued from Page 24)

counts are maintained for business and personal car expenses, and the program handles petrol amounts entered either as gallons or liters. Data are displayed on screen and can be printed at any time with a screen dump.

I'll send you the DORTIG No. 2 for $1. It is public domain. However the lads give their address in the "lots of rubbish" section and would be grateful for any correspondence or tokens of appreciation you want to send their way.

KANJI DRILL
by Don Shorock

We live in a small world with a global economy. It is not possible nor desirable to isolate our country's economy from those of other countries because much of our prosperity depends on international trade. For example, look inside 99/4A console or peripheral sometime and note how many of the chips are from Malaysia, El Salvador and other far away places. As competitors in a world economy we cannot assume that the rest of the world will deal with us on our own terms. Specifically, we cannot assume that the rest of the world speaks English. If we want to sell our exports in other countries we need speak and write in the local languages of these countries. Many citizens of European and pacific Asian countries are multilingual. In these countries, training in a second language is required in high school. Not so in the U.S. In fact, you can earn a liberal arts bachelor's degree from many of our best universities without any kind of foreign language requirement. Most U.S. citizens are not multilingual and this puts us at a disadvantage in a global economy. One of the most important participants in this global economy is Japan.

Don Shorock has written foreign language drill programs in a number of languages for the 99/4A. His on-screen Japanese Kana Drill software that uses the Terminal Emulator II to speak Japanese and his Japanese Writing System for printing Japanese to a printer have received superb reviews. His newest offering is Kanji drill. There are 881 separate Kanji symbols, the majority of which are available for on-screen viewing in Kanji Drill.

The user is presented on-screen with four Kanji symbols, one English translation, and asked which symbol matches the English meanings. If you select the wrong symbol you get to try again until you are correct. You then get another random group of four Kanji symbols to analyze. This is repetitive, rote memory drill, and it works!

In his cover letter to me that accompanied my review disk, Don says, "You say you don't know how to read Japanese. Play with this program for a while and you'll learn a lot of it." He's right. After 30 minutes I was correctly guessing the majority of the Kanji symbols presented to me on the first try. After an hour, my correct percentage approached 90 percent.

There is no easy way to learn to read a second language. It takes lots of time and drill, the kind of drill you get from Kanji Drill.

I know of no similar Japanese language software for other computers. This is because Kanji symbols cannot be entered from computer keyboards. They each have to be drawn pixel by pixel. Don has taken the time to do this for us, so only on the TI can you display (with Kanji Drill) and print (with Don's Japanese Writing System) the numerous Kanji symbols.

If you want a computer to help teach you Japanese, the 99/4A and Don's software may be the only way to go. Kanji Drill is fairware with a suggested donation of $20. You can get the latest version directly from the author by sending your money to P.O. Box 501, Great Bend, KS 67530-0501. If you just want to try it out, send me $1 and I will send you a version of Kanji Drill which has 360 of the 881 Kanji symbols.

Don has many foreign language vocabulary drill packages with musical and graphic rewards for correct answers. Offerings include Vietnamese, Bulgarian, Croatian, Czech, Danish, Dutch, Finnish, French, German, Greek, Hebrew, Hungarian, Icelandic, Indonesian, Italian, Latin (with roman numerals), Norwegian, Polish, Portuguese, Rumanian, Russian, Serbian (Cyrillic), Serbo-Croatian (with Latin alphabet), Slovak, Spanish, Swedish and Turkish. What a list! These all come with several starter vocabulary files and have room to add your own words. These are all fairware, with a suggested donation of at least $10 per package. Write Don for details, or go ahead and send him $10 and a disk with paid return mailer for each of the languages that interests you.

P-GRAM UTILITIES VER. 2
by Tony Knerr

When you purchase a P-GRAM card from Bud Mills you get a disk labeled "P-GRAM Utility" which contains some modules you might want to put on your P-GRAM. This is Version 1. Tony Knerr has created P-GRAM Utility Version 2 with more software to load into the card. In most cases you just load this stuff into any P-GRAM page without modification or patching, and it is there waiting for you at the first menu that follows the color bar title screen. All the Version 2 utility modules allow exit back to the title screen with FCTN-9. Such clean exits often don't occur with many custom GRAM modules created with GRAM Packer. Here are the modules you get:

• The TI-Writer module with editor and formatter files all stored together as one GRAM module. You don't need a disk to boot the editor and formatter. You can load TI-Writer from either BASIC with CALL TIW, and you don't have to modify either BASIC to do this.

• The Editor/Assembler module with its editor and assembler all in one GRAM module. You don't need a disk to boot the E/A editor or assembler, and you can call the E/A module from either BASIC with CALL EA.

• DSKU. This is my favorite. It is not normally possible to pack John Birdwell's DSKU into a GRAM module using GRAM Packer. Many, including myself, have tried and failed. Now DSKU is available as a GRAM file and everything works properly.

• XBPATCH. This is a patch to your existing Extended BASIC which adds such CALLs as HONK, BEEP, QUIT, LRGCPS (loads the large capital char set used in the color bar title screen into XB) and CHARA1 (loads the TI-Writer char-

(See Page 26)

MICRO-REVIEWS—

(Continued from Page 25)

acter set into Extended BASIC). If you already have a modified (patched) XB, this new set of patches may lock up your computer. Keep a disk copy of your old XB before you try out XBPATCH.

Some nice cosmetic changes come with Utilities Version 2. The anemic white on light blue colors of the P-GRAM DSR menus are changed to a much more readable white on dark blue. Also, the GMENU menu can be centered on screen. Previously, if you had many fewer than 24 GMENU items, these started at the top of the screen and left lots of blank space at the bottom of the screen.

And, finally, if you have a CorComp RAMdisk (one of the least useful pieces of hardware made by CorComp) you can now load the P-GRAM DSR with the RAMdisk in the Peripheral Expansion Box. Previously this was not possible.

From the list above I am now using with my P-GRAM+ the complete E/A module package, DSKU, and the easier to read DSR menus. Probably all P-GRAM owners will find something useful in Util[illegible] Version 2. It is public domain. I'll send it to you for $1, or you can send a little donation along with a disk and paid return mailer directly to the author: Tony Knerr, 17 Marshall Circle, Downington, PA 19335.

I am easy to contact. My mail address is P.O. Box 647, Vendocia, OH 45894; my Internet email address is cgood@magnus.acs.ohio-state.edu, and you can phone me evenings by voice at 419-667-3131.

NEWSBYTES

MUNCH issues Adventure II as fundraiser

MUNCH (Massachusetts Users of the Ninety-nine and Computer Hobbyists) has released Adventure II as its 1994-1995 fund-raiser.

Adventure II continues the group's Adventure Compendium series. This two-disk double-sided, double density set of more than 15 adventures costs $6.95 plus $2 for postage and handling.

The group offers the Adventure Compendium of more than 70 adventures and Adventure II together for $14.95 and $3 postage and handling. According to the group, this is a savings of more than $12.

The Educational World of Tony Falco, the group's 1993-1994 fund-raiser, is still available. This five-disk single-sided, single density set contains more than 60 program aimed at children's education. The set costs $8.95 plus $3 postage and handling.

To order, or for further information, contact MUNCH, c/o J.W. Cox, 905 Edgebrook Dr., Boylston, MA 01505.

Tler William Hay dies

William Hay of Jefferson City, Missouri, a long-time [illegible] and Geneve user, was killed along with his dog in a weather-related accident that occurred over the July 4 weekend. His wife was also injured by the tree which fell on them while they were camping.

Computing conference seeks participation

The 16th National Educational Computing Conference, scheduled June 17-19, 1995, at the Baltimore Convention Center in Maryland is seeking participation.

The theme of the conference is "Emerging Technologies — Lifelong Learning," referring to all the ways learners and teachers use technology in the educational process today, and how they may be using it in the future. The conference is designed to provide a forum for discussing and sharing ideas, techniques and research findings among educators and administrators in both training and education.

Individuals who would like to participate are invited to submit papers, proposals for workshops, poster or project session, or classroom demonstrations. Research seminar proposals were due July 15. The deadline for most NECC '95 submissions is Oct. 15 and poster session proposals are due Dec. 1.

A Call for Participation with detailed information on the conference is available from Donella Ingham, NECC/NECA Coordinator, 1787 Agate St., Eugene, OR 97403-1923; phone (503) 346-2834; fax (503) 346-5890; E-mail NECC95@ccmail.uoregon.edu.

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

USER NOTES

Using Multiplan

This comes from Donald N. Andrews, of Williamsburg, Virginia. He writes:

My tips are only the tip of an iceberg, which are really a cry for help!

There are a few "no-no's" that I've learned the hard way I'll share with you, but for each single tid-bit of knowledge, there will be at least two questions raised. My topic is Multiplan and I will share my experience with this valuable tool in the Geneve environment.

My initial reaction was that the time saved in loading data was lost when the file was saved; it seemed to take as long to save the file to disk as it did to print (see addendum).

As with the TI, there are a few things you must remember to do before loading a file:

1. Always select the No Option for Recalc, as this will allow you to make all your entries without the delay of bringing the file up to date after each entry. You may use the (FCTN-8) to update all data if you wish to check it prior to S)aving the file, but the program will do this automatically if you care to skip it.

2. Always transfer your files to the RAMdisk or, better still, to a Horizon RAMdisk so they will not be lost when you turn the machine off. Of course, if you have only the built-in RAMdisk, you will find that by transferring all your files to RAM, then using Disk Utilities (or DM 1000, or any file copy program) to copy the updated files back to the physical disk, you will save time both in the Load/Save process of the files, as well as with data entries.

I have one file (that because of X)ternal data copied from other files) takes over four minutes to load with the Geneve from a physical drive, and loads in under 20 seconds from RAM — don't ask me to time it with the 4A! I had to get the docs out to remember how to tell Multiplan to look for the data in drive 8 rather than drive 1, but I did not find how to have it remember that it was loaded from drive 8. It always wants to search drive 1, and will go through a series search of the physical drives 5 times before it starts on the RAM. It will eventually load on the fifth search, but the program still wants to return to physical drives for "instructions," and you will get numerous error messages to "Enter Y to retry access to MP disk."

I have always liked Multiplan and I keep coming back to it. Perhaps I got so familiar with it before some of the others came out that I just did not take the time to familiarize myself with some of the newer spreadsheets. Even with the problem with the printer, and having to cycle through the physical disks (I usually leave a TIMP disk in one of the physical drives for the program to return to for "instructions" rather than continue to receive the error messages) which was a simple, but temporary "fix"). I hope to find a patch for this, soon.

I mentioned earlier to see the addendum regarding the printer. Well, this is it. While this may be seem to be a different subject, I have found that if using a simple BASIC program, ad infinitum for infinite lines, I have no problem with the printer accepting the commands. Here is the program:

```
100 OPEN #1:"PIO";
110 PRINT #1:"NAME 1"
120 PRINT #1:"ADDR 1"
130 PRINT #1:"PHONE 1"
140 PRINT #1:"NAME 2"
150 PRINT #1:"ADDR 2"
160 PRINT #1:"PHONE 2"
```

Or, if I use TI-Writer, I can use the formatter to I)nclude F)ile, and again print the Encyclopedia Britannica without interruption. However, using Multiplan, the printer prints only about eight or 10 lines of data, and I must again use the Print command for the data to continue. If its a full sheet of data, I may have to execute the Print command numerous times to complete the document.

Here's the problem: For each time the Print command was executed, that is the number of files stored in the buffer, and if you had planned to print several files, you must remember to load the second file prior to entering the second Print command, etc. There must be an easier way!

I've mentioned this problem at my user group meetings and received many suggestions, such as changing the speed, or using another printer (I have both the STAR SG-10 and STAR NX 1000). However, I get the same result with both.

I have the same problem printing data from Archiver. I use Archiver to print a catalog of my disks on the disk jacket. I like the feature Archiver uses of printing only so many lines and giving you the prompt MORE__. I always pause at the second prompt and move the printer head over, and the jacket back to top position, in order to print more columns. Archiver does the same thing in only printing limited lines, as in Multiplan, which is very frustrating.

Program writes checks

The following program, Check Writer, was written by Harry Allston, of Reedley, California. He writes:

There are numerous check writing, bank balancing programs. I find them cumbersome and time-consuming when all I want to do is "print a check."

The program is straightforward. Just follow the input requirements. Some checks, like mine, position the "money box" in a different plane than the other lines. Correct spacing can be had by adjusting the line spacing in line 440. The program was written for a Epson printer. Printer Modifications can be made in line 410.

Printing the check will take some experimenting. I set the check in the printer with just a tad of the check showing and press the key to print. The check advances and "out of paper" alarm sounds. I then place the paper bar down and cause the printer to print by bypassing the alarm. For me, this works out fine. Try some note paper and compare to the check.

CHECK-WRTR

```
1 ! SAVE DSK1.CHECK-WRTR !01
3
100 CALL CLEAR :: CALL BLUE
```

(See Page 28)

USER NOTES

(Continued from Page 27)

```
!228
110 CALL CHAR(96,"0000FF")!2
49
120 DISPLAY AT(2,5)ERASE ALL
:"CHECK WRITER" !105
130 DISPLAY AT(3,5):"``````
`````" !180
140 DISPLAY AT(4,1):"Check D
ate i.e. 00-00" !013
150 DISPLAY AT(6,1):"Check Y
ear i.e. 94" !159
160 DISPLAY AT(8,1):"Paid to
?" !241
170 DISPLAY AT(11,1):"Check
amount in $$$-->" !023
180 DISPLAY AT(13,1):"Check
amount in cents>" !112
190 DISPLAY AT(15,1):"Writte
m DOLLARS of check" !160
200 DISPLAY AT(16,1):"
  ``````" !008
210 DISPLAY AT(19,1):"ÇENTS
of check i.e. (XX)" !111
220 DISPLAY AT(20,1):"````"
 !057
230 DISPLAY AT(22,1):"A memo
 at the bottom" !132
240 ACCEPT AT(4,17)SIZE(-2)V
ALIDATE(DIGIT)BEEP:DT$ :: IF
 DT$="" OR DT$="00" THEN 240
 !184
250 ACCEPT AT(4,20)SIZE(-2)V
ALIDATE(DIGIT)BEEP:DT1$ :: I
F DT1$="" OR DT1$="00" THEN
250 !079
260 ACCEPT AT(6,17)SIZE(-2)V
ALIDATE(DIGIT)BEEP:DT2$ !223
270 ACCEPT AT(9,1)BEEP:NAME$
 :: IF NAME$="" THEN 270 !24
0
280 ACCEPT AT(11,23)VALIDATE
(DIGIT)SIZE(-4)BEEP:AMT1$ ::
 IF AMT1$="" THEN 280 !168
290 ACCEPT AT(13,23)VALIDATE
(DIGIT)SIZE(-2)BEEP:AMT2$ ::
 IF AMT2$="" THEN 290 !180
300 ACCEPT AT(17,1)SIZE(-25)
BEEP:DLR$ :: IF DLR$="" THEN
 300 !011
310 ACCEPT AT(19,22)SIZE(-2)
VALIDATE(DIGIT)BEEP:CENT$ ::
 IF CENT$="" OR CENT$="XX" T
HEN 310 !056
320 ACCEPT AT(23,1)BEEP:MEMO
$ !210
330 CALL HCHAR(22,1,32,84)!2
27
340 DISPLAY AT(2,1)ERASE ALL
:"YOU ARE GOING TO PRINT:" !
072
350 DISPLAY AT(4,1):"DATE: "
;DT$;" ";DT1$;" ";DT2$ !142
360 DISPLAY AT(6,1):TAB(1);N
AME$;TAB(18);"$"&AMT1$&"."&A
MT2$ !057
370 DISPLAY AT(9,1):DLR$: :"
DOLLARS & "&CENT$&"/100" !10
0
380 DISPLAY AT(13,1):MEMO$ !
225
390 DISPLAY AT(22,3):"READY
TO PRINT? [Y-N]" !061
400 ACCEPT AT(22,20)SIZE(-1)
BEEP:N$ :: IF N$="Y" OR N$="
y" THEN 410 ELSE 120 !143
410 OPEN #1:"PIO" :: PRINT #
1:CHR$(27);"M";CHR$(27);"G"
!243
420 PRINT #1: : :TAB(44);DT$
&"-"&DT1$;TAB(54);DT2$;!181
430 PRINT #1: : :TAB(8);NAME
$;!201
440 PRINT #1:CHR$(27);"3";CH
R$(13):: PRINT #1:"";!215
450 PRINT #1:TAB(57);AMT1$&"
."&AMT2$ !112
460 FOR V=1 TO 3 :: PRINT #1
 :: NEXT V !202
470 PRINT #1:CHR$(27);"M";CH
R$(27);"G";"******"&DLR$&" DO
LLARS & "&CENT$&"/100"&"****
*" !068
480 FOR V=1 TO 10 :: PRINT #
1 :: NEXT V !249
490 PRINT #1:CHR$(27);"8" !1
14
500 PRINT #1:CHR$(27);"G";TA
B(7);MEMO$ !164
510 PRINT #1:CHR$(27);"@" ::
 CLOSE #1 !147
520 DISPLAY AT(10,1)ERASE AL
L:"Print another check? [Y-N
]" :: ACCEPT AT(10,23)SIZE(-
1)VALIDATE("ynYN")BEEP:A$ !2
23
530 IF A$="N" OR A$="n" THEN
 CALL CLEAR :: STOP !107
540 IF A$="y" OR A$="Y" THEN
 120 !215
550 SUB BLUE !149
560 CALL SCREEN(5):: FOR C=0
 TO 14 :: CALL COLOR(C,16,5)
:: NEXT C :: SUBEND !179
```

## Configure printers using Funnelweb

The following article was written by Fred Moore. It is based on using Funnelweb V4.10. It has appeared in several user group newsletters.

1. Make a copy of Funnelweb. Leave off the write-protect tab and place this new copy in DSK1.

2. Autoload Funnelweb up the first menu (selection 1-9 and I0.

3. Select option "I" (Configure). It may also be reached by selecting User List from the Editor menu. DSK1.CF and CG will load and display "CONFIGURATIO" the prompts: (?) HELP, (c/C) BACK (F-7) DIR.

4. Hit any key. First window: SYSINFO, QUIT, INSTALL.

5. Hit "S" (SYSINFO) and get the second window: LOAD, EDIT, SAVE.

6. Hit "L" (LOAD).

7. Hit Enter. This loads SYSCON from DSK1.

8. Hit "E" (EDIT). This will display the third window: LOADING, DEVICES, COLORS, MENU, XB LIST and UL LIST.

9. Hit "D" (DEVICES). This will display the fourth window: EDIT PRINTER, FMTR PRINTER, OBJECT FILE, WORK FILE, 7 PROGRAM.

10. Hit "E" (EDIT PRINTER). The following instruction window will appear: ENTER/FILE DEVICE.

11. Type in your ptiner name: PIO or RS232.BA=xxxx.LF, etc.

12. Press enter. The easiest way to change the formatter printer is to repeat steps 10, 11 and 12 as follows before going any further:

10a. Hit "F" (FMTR PRINTER).

11a. Type in PIO.LF or RS232.BA=xxxx, etc.

12a. Hit enter.

13. Hit FCTN-9 (BACK) or CTRL-C.

14. Hit FCTN-9 or CTRL-C again.

(See Page 29)

# USER NOTES

(Continued from Page 28)

15.Hit "S" (SAVE).

16. Hit enter to save the change in DSK1.SYSCON.

17. Hit FCTN-9 or CTRL-C to get back to the top of the menu.

18. Hit "I" (INSTALL). The sixth window gives two choices.

19. Hit "L" (LOAD-XB/XBII).

20. Enter. source program (DSK1.LOAD).

21. Enter target program (DSK1.LOAD).

22. Hit FCTN-9 or CTRL-C.

23. Hit Q (QUIT).

24. Test from scratch by reloading Funnelweb.

## Care on the information highway

If you find it inconvenient to reach TI's consumer relations department at 1-800-TI-CARES, you can reach them via modem on the Internet. The E-mail address is ti-cares@lobby.ti.com — an advantage to those who cannot telephone during office hours, for instance.

## Experimenting with sound

The following article and program were written by W. Leonard Taffs. It first appeared in the newsletter of the SouthWest Ninety-Niners of Tucson, Arizona. He writes:

What a miracle it is to be able to hear! Imagine an auditorium filled with several hundred or a thousand or more people. There is a piano on stage — it could be any instrument — and the auditorium has quieted down to such a point that you could hear a pin drop, hall accoustics permitting. A person comes out on stage and plays one single note on the piano. Mind you, though the person has played only a single note or sound, the entire atmosphere of the auditorium has changed in an instant.

Silence itself is an auditory phenomenon, even if nit-pickers want to waste time disputing that, and sound is another. Insensitivity to the perception of the single sound phenomenon is one of the greatest deprivations that can be inflicted on a human being. Even the pin drop is significant.

The appreciation of minute sounds is easily lost in the cacophony of today's environment. The power of silence and the lack of it was acutely demonstrated in the recent coverage of the "Juice's" dilemma.

The following program uses a CALL LOAD for the TI's sound register.

### SOUND/EXP

```
1 REM (SOUND/EXP) 6-15-94
  EXPERIMENT BY W.L. TAFFS
  USING NORTH COUNTY CALL
  SOUND REGISTER CALL LOAD.
!137
```

(See Page 30)

# USER NOTES

(Continued from Page 29)

```
2 CALL INIT !157
10 REM!154
15 REM INPUTS A AND B WILL
   RESULT IN NUMBERIC OVER-
   FLOW IF YOU EXCEED +/-
   32767 OR -32767 IF LINE
   110 IS REMOVED. !253
20 REM!154
25 REM POS OR NEG INCREMENTS
   SET BY LINES 190-22.
   (BOTH CAN BE + OR -, OR +
   AND MINUS, OR - AND +) !1
53
30 REM!154
35 REM UN-REMARK LINE 260 TO
   STEP THROUGH CALLS !123
40 REM!154
45 REM A=300, B=200 IS ONE
   SUGGESTED SETTING !111
50 REM!154
55 CALL CLEAR !209
65 DISPLAY AT(15,3):"ENTER v
alues for A and B": : :"Read
 Remarked lines to make Chan
ges in Parameters A1,B1" !05
0
70 INPUT "Press <ENTER> to S
tart ....":K$ !198
100 CALL CLEAR !209
110 ON ERROR 280 !034
120 INPUT "A: ":A !237
130 PRINT :: INPUT "B: ":B !
013
140 CALL CLEAR !209
150 DISPLAY AT(10,8):"YOU EN
TERED: ": :"A=";A,"B=";B !12
4
160 CALL LOAD(-31740,A,B)!10
9
170 FOR E=1 TO 50 ! OPTIONAL
 DELAY !001
180 NEXT E ! OPTIONAL DELAY
!115
190 A1=25 ! CHANGE INCREMENT
 TO YOUR WISH !252
200 A=A+A1 !115
210 B1=-100 :: ! CHANGE INCR
EMENT TO YOUR WISH !108
220 B=B+B1 !118
230 DISPLAY AT(1,8):"INCREME
NTS:": :"A by ";A1,"B by ":B
1 !034
240 DISPLAY AT(15,6):"VARIAB
LES CALLED:": : :"A=";A,"B="
;B !082
250 IF A>32000 THEN STOP !14
3
260 ! CALL KEY(0,K,S):: IF S
<1 THEN 260 ! OPTIONAL PAUSE
 TO STEP THROUGH CALLS !157
270 GOTO 160 !239
280 END !139
```

The program will not take much time to enter. As an experiment, this is virtually a cornucopia of possibilities. The CALL LOAD specifies two variables, A and B. User input determines the numbers for the program to start with. These variables are varied as the program runs by the parameters specified in lines 190-220, which you can change to any numbers you wish within the numeric limits of the TI. The program will run more quickly if you eliminate the optional delay (170-180), and if you set large increment values in lines 190 and 210.

For ideal results, it would be advantageous to have your sound output from your modulator hooked up to feed through a hi-fi system. But even without this luxury you can hear some interesting results.

**SOME TIPS**

If you let the program run its course, it will stop when the maximum values are reached. The ON ERROR in line 110 sees to this. You may notice that even though the program has stopped, the sound may still continue. Nothing is wrong — just enter characters randomly in COMMAND mode to get a syntax error and this will stop the sound. This could be a distinct advantage over using a CALL SOUND routine in some programs.

If you do not use the delay, the sounds may go past certain critical points before you can read what the values were. Anothing method of going slowly through the value changes is to use the optional CALL KEY in line 260. Of course, this might take an infinitely long time to step through each CALL. My suggestion is to use the delay routine until you get in the neighborhood of the numbers you are interested in. Then stop the program and un-REMark line 260. Then run the program again and enter A and B values in the neighborhood of the values were were curious about.

Another thing, as the program progresses and the program passes through certain variables that were called, the sounds remain continuously. This continuing sound can cover succeeding areas of more delicate sounds. The answer is to periodically BREAK into the program using FCTN-4 and then use CON to continue the program. You variables will continue from where they left off and you have eliminated the previous sound.

I found this to be an interesting experiment. For the first time I heard some of the sounds the TI emits sometimes when a program crashes, or when the villain running under the alias of Super Extended BASIC goes crazy, or similar sounds to those produced in games I have heard.

What useful purpose could this program serve? It can enable you to enter a CALL LOAD at any time in your program or game to produce a certain, specified sound. Antoher CALL LOAD can be inserted in the program where you want the sound terminated or suspended. Without a knowledge of what specific parameters to enter in your CALL LOAD — this sou$4 register being one — you will have no way of guessing what the numbers should be.

Does this program produce sounds a CALL SOUND statement can't? I don't know the answer, as I have not tried it. I have found it easier to experiment using this program — and it takes less patience — that thumbing through CALL SOUNDs. I have not heard some of the sounds produced with the program from any CALL SOUNDs, though I guess it may be possible. Perhaps this experiment might spur some advanced programmer to come up with a more ambitious development of this idea.

## Asgard Peripherals no longer developing TI hardware

According to Jim Krych, director of research and development for Asgard Peripherals, the company is no longer involved in hardware design for the TI99/4A. In a note posted to an electronic bulletin board, Krych said that SUPER-

(See Page 31)

# USER NOTES

(Continued from Page 30)

AMS and the technology has been given to an unnamed third-party.

"They have the prototype(SUPERams with PSRAM chips) and all of the needed information, custom chip programming, schematics, and theory of operation," he wrote.

## D/V 80 is the TI workhorse file format

The following item originally appeared in the newsletter of the Orange County, California, TI user group.

The Display Variable 80 file is TI's workhorse. TI-Writer uses this format. If you open a file without specifying a type — OPEN #1:"DSK1.MYFILE" — it will default to D/V80.

Assembly language source code files are D/V80. Here are some interesting aspects of D/V80 files as they concern TI-Writer and Multiplan:

First, you can save a Multiplan spreadsheet as a D/V80 file on disk. Then, you can use that D/V80 file for printing or for merging into a TI-Writer file. You choose Print, and then File. You must be careful to use a different filename than the one you used to save your spreadsheet as, unlike Transfer Save, Multiplan does not warn you if you are about to overwrite an existing file.

Just as when printing to a printer, you can control the margins and page format with Print Margin. One of the items that Print Margin lets you set is print width. If you set this to a number greater than 80, Multiplan will write a D/V80 file wherein each record is longer than 80 characters.

Should you attempt to read such a file with a BASIC program, your system will produce a strange error code and lock up. Apparently the folks at TI thought that a D/V80 file could not have a record longer than 80 characters, so their error handling language does not consider this possibility.

TI-Writer, however, will read this illegal file. It will input only the first 80 characters in each record, but it is just about the only way to access the file. Another is to use a disk sector editor.

Incidentally, TI-Writer is very forgiving when reading files. You may find that loading a text filethat has a glitch into TI-Writer and then saving it under a different name will make it readable. If you run into a D/V80 file that gives you such a problem, load it into TI-Writer and see what happens. It can't hurt.

# The ONLY monthly devoted to the TI99/4A

## Subscription Fees

❑ 12 issues, USA, **$35**

❑ 12 issues, Mexico, **$40.25**

❑ 12 issues, Canada **$42.50**

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

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

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

## Address Changes

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

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

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

Card No. _____

Expiration Date _____

**(Minimum credit card order is $9)**

Signature _____

**(Required on credit card orders.)**

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

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

**Name** _____

**Address** _____

**City** _____

**State** _____ **ZIP** _____

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

## Disks, Etc.

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

_____

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

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

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

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

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

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

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

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

### GENEVE DISKS (SSSD unless specified)

❑ **MDOS 2.0** (req. DSSD or larger (for floppy & hard drive systems) .....$4.00

❑ **GPL 1.5** .....$4.00

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

❑ **Myarc BASIC 3.0**.....$4.00

❑ **MY-Word V1.21** .....$4.00

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

### GENEVE PUBLIC DOMAIN DISKS

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

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

**SECOND CLASS**