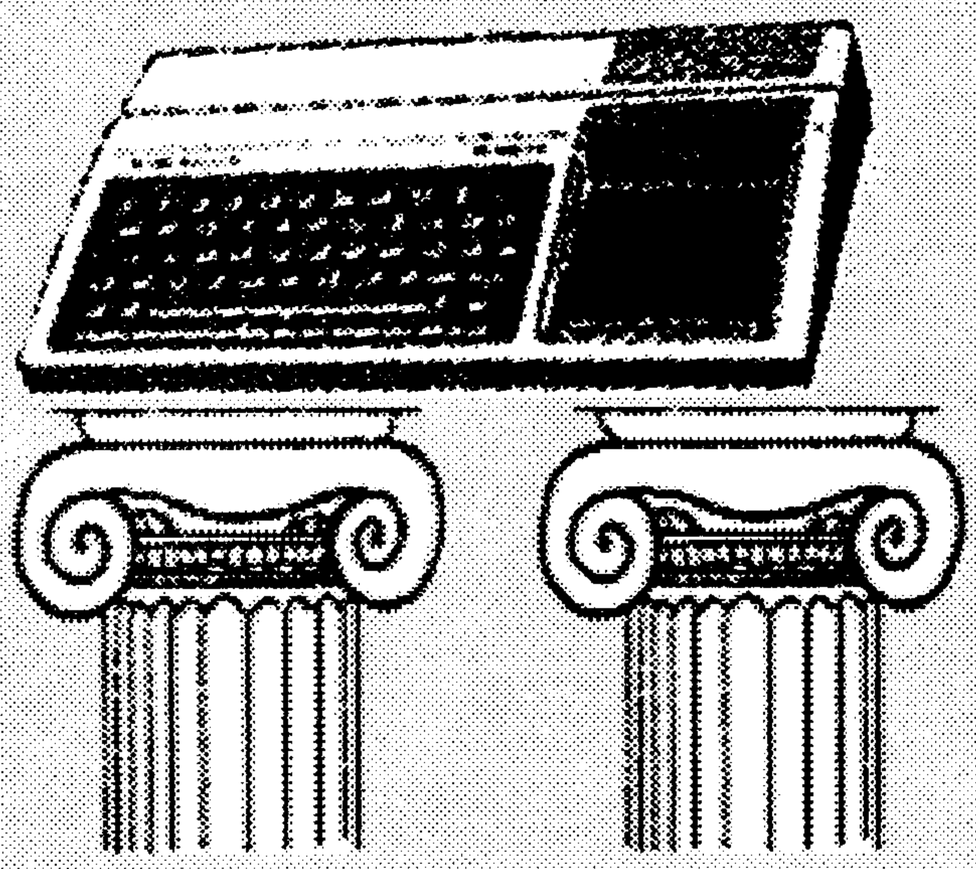
Covering the TI99/4A and the Myarc 9640

MICROpendium

Volume 11 Number 7 August 1994 $3.50



Cecure to do all repairs for TI home computer products

see story page 5

☆ ☆ ☆

Extended BASIC

Testing your visual perception
Mail merge envelope printer
Printing in multiple columns

The art of assembly

Trying the impossible

Reviews

PC99 emulator
Asgard Memory System

Also inside

Pix Pro and graphics conversions
Understanding error messages

☆ ☆ ☆

CONTENTS

MICROpendium

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

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

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

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

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

Cecure takes over 4A repairs

Enough supplies remain for service for at least five years, Walden says..... Page 5

Extended BASIC

Testing your visual perception .....Page 6
Mail Merge envelope printer.....Page 14
234 COL.....Page 17

Newsbytes

A vendor list, a new address and BBS number for VAST, and a fair in Germany .....Page 10

The Art of Assembly

Trying the impossible.....Page 11

Graphics

Pix Pro and graphics conversions.....Page 20

Reviews

Asgard Memory System.....Page 21
PC99.....Page 24

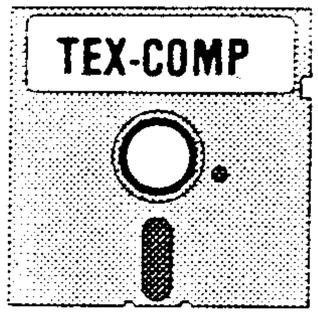
User Notes

Understamding Extended BASIC error messages, lining up decimals, and adding numbers with Castnines.....Page 26

Classified .....Page 31

*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 ARICON 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 YOURSELF EX-BASIC
#5067 BEGINNING BASIC TUTOR
#307 GRAPHICS CODE GENERATOR
#3912 SUPER BUGGER
#3913 BETTER BANNERS
#3914 CERTIFICATE 99
#3915 HOROSCOPE MAKER
#3916 GRAPHX+ PRINT SHOPPE
#3917-3720
GRAPHX COMPANIONS 1-4
#3721 MAC FLICK (G)
#3722 PRINTING TO GO (G)
#3723 GRAPHX DINOSAURS (G)

BUSINESS, ACCOUNTING, WORD PROCESSING, DATA BASE HOME OFFICE & HOME
#10 GOTHIC PRINTOUT (P)
#19 TI WRITER/MULTIPLAN UPGRADE
#20 ACCOUNTS RECEIVABLE (2)
#21 DATA BASE DEMO
#23 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 SPEAR & 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 BETA
#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 BYPHONE
(818) 366-6631 24 HOURS A DAY

Authorized Dealer
TEX COMP
America's Number One TI computer retailer
P.O. Box 33084, Granada Hills, CA 91344

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

COMMENTS

TI made the right choice

Texas Instruments' selection of Cecure Electronics as its designated repair facility for 99/4A products should make the TI/Geneve community very happy. Another company could have gotten the nod, and that would have been unfortunate.

Cecure is a part of the TI community and has been for years. Don Walden knows the TI and its customer base and runs a tight ship. He's been involved in developing innovative products for Geneve users and has proven to be reliable and conscientious. Having Cecure repair 99/4A products is the next best thing to having TI do the work.

AMS REVIEWS

We've been publishing several reviews of the Asgard Memory System hardware and software recently, even though Asgard is no longer around to support it. We're doing this because users may find it available on the second hand market and because the system represents a bold breakthrough for TI users who want to extend the power of their systems.

There's no telling how significant the development of this system would have been several years ago when the only option to extend the memory limitations of the TI was to purchase the Geneve, which wasn't a TI. Unfortuntely, AMS came along rather late and by then Asgard lacked the resources to market and support it effectively. The only glimmer of hope is that the rights to the system have been assigned to another company. Anyone who has tried out the AMS would agree that it deserves an extended life span. And I hope it gets it. But that will depend on how many TIers are interested in further development of this technology. In any case, Bruce's reviews definitely paint an accurate picture of a technology with enormous promise, and there aren't many orphan computers about which one can say that.

—JK

READER TO READER

❑ Robert Schulz, Kirnsteinstrasse 20 a, 83026 Rosenheim, Germany, writes:

One of my most used software on my TI is TI-BASE. In TIB we have the feature to redirect the printer-output to a file (e.g., SET PRINTER=DSK.1.FILE). This works fine from the dot prompt. But I want this feature to be selectable from my main menu. To no avail so far, I assigned a local:

LOCAL C OUT 10

REPLACE OUT WITH DSK1.FILE (in the command file read in through redstring)

SET PRINTER=OUT

The printer device now is OUT, not DSK1.FILE.

How can I make TIB realize that I don't mean the STRING "OUT" but the contents of the variable OUT?

Can any reader help?

P.S.: There was a small mistake in the July issue: Ron Warfield states that TIB reads only 40 characters. That's not exact. The command file interpreter of TIB can well handle lines up to 80 characters, but the command file editor won't. If loading a command file with lines longer than 40 characters into the TIB-Editor, all characters exceeding 40 will be truncated. As an example you may take the command file TUTOR/C that came with your TIB package. I have done quite a lot of work so far with TI-Base and would like to get in contact with readers who have problems or suggestions. Please write to me!

❑ Jerry Keisler, 2221 College Dr., Paris, TX 75460, writes:

I am trying to put disk error traps in my TIW-ENV program but am having trouble with the error control. The following is part of a program to read disk catalogs. When I try to read a drive, it runs the error routine. But it will not read any good disks once an error is detected. It will not even try to read a drive. It just goes back to the error routine.

```
190 DISPLAY AT(10,1)ERASE AL
L:"CATALOG TI-WRITERS FILES
ON": :"DISK DRIVE";DR :: ACC
EPT AT(12,12)BEEP SIZE(-1)VA
LIDATE("123456789"):DR
200 DK$="DSK"&STR$(DR)&"."
210 ON ERROR 355 :: OPEN #3:
DK$,INPUT,RELATIVE,INTERNAL
220 INPUT #3:A$(0),J,N,K ::
ON ERROR STOP
....
355 DISPLAY AT(24,1):"DISK E
RROR      PRESS ENTER" :: CAL
L KEY(3,K,S):: DISPLAY AT(24
,1):"" :: IF S=0 THEN 355 EL
SE ON ERROR STOP :: RETURN 1
90
```

I have tried several approaches but get the same results or an error message from the computer. Can anyone help?

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

FEEDBACK

P-GRAM banking

Here's some furthern information relating to Bruce Harrison's "Art of Assembly — Part 37" from July '94. Relating to the second part of the example assembly code, this routine will only search page one of the P-GRAM card for the E/A module. To search the other three pages it is necessary to change the GROM "Read Data" address to concide with the other pages and to bank in the proper page of the P-GRAM by changing the word at c>80FA. The other page addresses are: Page 2 — >9804, Page 3 —>9808, Page 4 — 980C. To bank in the proper page at the P-GRAM, use the following code:

```
LI      1,>9804 (or >9808 or >980C)
MOV     1,@80FA
```

The assembly programmer can include this code to set the proper page as well as changing the GROM "Read Data" equate in Bruce's program segment if he knows what page the E/A module is on. To search all four pages, go through the changes for all four pages until the module is found and don't report any errors or return to the title screen until all four pages have been searched. I krrp my E/A module on page 2 and the utility works when the GROM "Read Data" address is changed to >9804 and the above two lines are included aftger the line:

INITX3 LWPI WS

I hope this clears up any problems any P-GRAM owners may have had trying to use this routine knowing that the E/A module was there but not finding it.

Tony Knerr
Downington, Pennsylvania

Cecure to repair TI products

By LAURA BURNS

Texas Instruments has contracted for Cecure Electronics to take over as the official repair center for the TI99/4A, its peripherals and cards. TI has provided the repairs since its introduction of the home computer in the early 1980s.

Don Walden of Cecure says the official changeover date is Sept. 1.

"We've been going over the list of parts and equipment TI possesses now that we are going to want," Walden says.

TI's 1-800-TI-CARES line will be telling callers to send their equipment to Cecure's address, Walden notes. He says customers should get the same type of turnaround and warranty on repairs as when TI was doing them.

Walden says Cecure will acquire new storage facilities for what he says is "literally a ton" of new parts and equipment for repairs. TI chose Cecure over other contractors as the exclusive facility for doing warranty repair and out-of-warranty repair, he says, noting that TI representatives visited the facilities to assure themselves as to the quality of customer service.

TI has continued to provide service for the 99/4A "11 years after the fact and 6 years after they were legally required to," Walden notes, commenting that it is a company which "wants to take care of its customers. I think that says a lot for their integrity."

He praised the support TI is providing for the changeover.

He says that customers for repairs "will be getting the same type of service as from TI" from Cecure Electronics. The only difference, he says, is that TI would bill customers for repairs after making them.

"We will need a card number, prepayment or a C.O.D. That's basically the only difference," he says. He adds that the price for some types of service may be lowered in the future.

He estimates that TI is providing Cecure with enough supplies to take care of 99/4A customers for at least the next five years.

Walden notes that TI is contracting repairs for its calculators to a Dallas company as well as contracting the 4A repairs to Cecure.

He also notes that the Myarc hard-and-floppy disk controllers recently received for sale by Cecure are "new, they're not anyone's used cards" that have been reconditioned.

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

Extended BASIC

Testing your visual perception

By LUCIE DORAIS

This program is about Visual Perception, or how your eyes can deceive your brain; or is it the opposite? Some of the problems in this program are classics, some are not.

The program is modular. That is, each section (module) is totally independent from the others, with its own sub(s) at the end of each module. The only things in common are the menu at the beginning and the two global subs at the end. Actually, each module can be used by itself, provided that each has the global subs.

Let's start with the pre-scan and the MENU (lines 100-170). The dummy DATA in line 110 is there only because at least one DATA line has to precede the pre-scan. The CALL SCR(8) is a call to our only user-defined sub: it clears the screen and changes its color (for variety, each module has its own color). As you can see, the program will also use sprites, a good way to refresh your memory of them.

And now the two global subs (lines 1000-1030): One is a normal GOSUB, the other the user-defined SUB SCR mentioned above.

The first problem of Visual Perception (lines 180-490) is based on the ability of the brain to retain a color, and then to mix it with another color right in the brain. The problem is often reproduced in books, but each time it is given in only one color contrast. Thanks to Tex, we can have a choice of 10 contrasts (five times two colors). I tried to use only the colors which offered the stronger contrasts. The instructions and explanations are in the program itself.

Lines 200-240 just display a screen of explanations of the problem. When typing text, always make sure that no word will be cut at the right side of the screen. To make sure, always align the first word of a new line under the first character of the preceding line. Each screen line has 28 characters, of course. Then the GOSUB 1010 asks you to press any key to start.

In this program, all text is in capitals, and the small letters are used for the graphics. Here, letter "A" is redefined as a block. We put the definition into variable A$ because we will use it again — why bother to type it twice? The DATA in line 260 is for our five color contrasts, read into A(X) and B(X) in the next line. RESTORE 260 will allow Tex to be able to find the right data line each time it runs that portion of the program. To save running time, we use the same loop to CALL COLOR sets 10 to 14, and to redefine the first four characters in each set. Variable Y takes the value of the first character of each set: 104, 112, 120, 128 and 136. Line 280 CALL CHARs these characters and the three next in each set.

The first problem of Visual Perception (lines 180-490) is based on the ability of the brain to retain a color, and then to mix it with another color right in the brain.

We now need to display the Color Contrast Menu, then to change the screen into the chosen contrast. All this is done by the SUB in lines 380-440. Again, Y will take the value of the first character in each color set. The color contrasts are displayed on two rows of five contrasts each. For the first row, displayed at row R=3, C is the column, and D the value of character 64, to which we will add the value of X to get the menu choices A-E. The CALL HCHARs in the secondary sub in line 450 first display three rows of three times character Y (we recycle variable K, usually kept for the call keys).

Then, in the middle of that square, (R+1,C+1), we call char the character next to Y, a smaller square in the contrasting color. The menu letter D+X is then displayed under the big square.

RETURNing to our main sub, back to line 390, we increment the Y character by two (to get the third and fourth char. in the set), the row becomes 11, the default character D becomes 69, and back we go to draw a contrasting square on the bottom row. This is done five times, then you press a key (A-J, ASCII 65-74) for your choice. If you pressed a key in the top row, letters A-E, the character color CC will be A(X), the screen color SC will be B(X). If your key was in the bottom row (F-J), the colors are inverted. The screen is then CLEARed, then another secondary sub (line^460) colors it with SC and set 9 is colored with C (remember above, we defined "a" as a block?). Why don't we use the SUB SCR to do part of this (clear and color screen)? Because later we will need to recolor the screen without clearing it...

Finally, before exiting the main SUB, Tex displays a big square all made up of "aaaaa"s in the middle of the screen. Upon coming back to the main program (we are still in line 290), we GOSUB 470 for a short delay, and from there we GOSUB 1020 to tell Tex to wait for us to press a key. The delay is just to prevent you from pressing the second key too quickly.

When you press a key, the square in the middle of the screen will disappear, this is done very fast by coloring it with SC. That is, character color CC becomes the same as screen color SC. We then GOSUB 460 to recolor set 9. The CALL SCREEN(SC) will have no visible effect, since SC was not changed.

Now, the GOSUB 480 in line 300 lea[us to another delay and call key (GOSUB 470), but this time when you press a key

(See Page 7)

VISUAL PERCEPTION—

(Continued from Page 6)

the screen is cleared and colored grey (15), and Tex asks you if you want another contrast. Upon return to line 300 in the main program, if you said "Yes," you are taken back to line 290 to display the contrast menu again. You can now run and play with that part of the program.

When you are bored, ask yourself the question DISPLAYed in line 310, and type the remaining portion of the module to get the answer! (Tex, being polite, does not force the second exercise on you: If you answer "No" in line 320, you are taken back to the main menu.)

In line 330 we again GOSUB 380 to display the color contrast menu, followed by the delay and the call key. This time, when you press a key, the colors of the square and of the screen background will be inverted. To do this, we need to store the screen color SC in a temporary variable D (we save memory by recycling this now useless variable), then SC becomes CC, and CC takes the value of D. The GOSUB 460 will re-color both screen and square (set 9), and the GOSUB 480 will perform the delay, call key, and ask you for another contrast. If you say "Yes," back to the beginning of the line.

If you say "No," you are given a screen of explanations (lines 340-360), followed by a note to press a key, then you GOTO 140 to the main menu.

We continue our explorations with three more puzzles: Which is the longest line? (a classic), Say the color (a sort of game) and another classic, What do you see?

WHICH IS THE LONGEST

Lines 500-650 displays two lines on the screen and asks you which is the longest. This is often reproduced in books, but with Tex we can actually demonstrate it. After you have entered your answer, the two lines will move closer to each other so you can study them better.

To confuse the viewer, the two lines are "decorated" with endings, that we redefine in lines 530-540. We then define the starting rows for the top line, R, and the bottom line, RR; then their left column, C and CC. The lines will be black. We GOSUB 640 to display them on the screen, then Tex asks you the question. The GOSUB 1020 will get your answer.

The text is then erased (line 580) and the two lines start their movement towards the center. Their present position at rows R and RR are first erased, then the top row is incremented by two while the bottom row is decremented by two. The columns are, respectively, incremented and decremented by one only, until the top line sits at column 6. This is where they stop. Since Tex kept your answer in memory (key pressed), he can tell you if you were right or wrong.. and give you the answer.

The computer displays words that tell a color, but with a twist: The words themselves are written in various colors, and not necessarily in the color told by the word.

So you see there is no challenge for you in this puzzle, but there was some in programming it!

SAY THE COLOR

Lines 660-830 are more like a game. I saw it in a television program about psychology and found it more difficult than it sounded.

The computer displays words that tell a color, but with a twist: The words themselves are written in various colors, and not necessarily in the color told by the word. The game is to try to tell the color the word is written with, NOT the color told by the word.

Since I wanted the words big but did not want to redefine a lot of characters, I settled for sprites. We will use the normal TI character definitions, but magnified twice, so they appear twice as big. Sprites have one fault though: You cannot have more than four on the same line, otherwise one disappears.

Also, when you color or move them, this is done sequentially so, if you want quick action, you have to keep them low in number. Therefore my choice of three letters maximum for each color: RED, BLU [blue], GRN [green] and YEL [yellow]. Another choice was made concerning them: Since the CALL SPRITE statement, with all its parameters, takes a long time to be interpreted and done, we call the four sprites only once. Then we hide them below the bottom row of the screen when we don't need them. CALL LOCATE, in a sub, moves them around.

The DATA in line 720 is for the letter sprites. The numbers are the ASCII values of characters B, L, U, G, etc. Before we READ the data, we need to RESTORE Tex to the first one, just in case Tex had got lost in another module. Each letter is then CALLed as a SPRITE numbered 1 to 12, colored transparent, and hidden in pixelrow 193, which is below the screen's field of display (the screen has 24 x 8 = 192 pixelrows). This being done while you read the text displayed by lines 690-710, Tex now GOSUBs 1010 for the key press.

In line 720, there are four DATA values at the end that are not ASCII values: They are the four colors that we will use (5=blue, 3=green, 12=yellow, 7=red). The screen is made black in line 740 (a call to our user-defined SUB) but you can try other colors that might look better on your monitor (mine does not like to be painted white at all). These four colors are put into the array A(X). In the same loop, Y takes the value of X; in this game, Y is the variable used for the word that will be displayed, while X is the variable for the color it will be written with. To give your brain some exercise, while Tex fills A(X), it shows you the four words into their true color (RED in red, GRN in green).

Then the game starts with a CALL KEY, because you can end it any time by pressing any key. If none is pressed (S=0), you go to the next line, where both the word Y and its color X are chosen randomly. Of course they can sometimes correspond, but more often than not they don't.

(See Page 8)

VISUAL PERCEPTION—

(Continued from Page 7)

The actual display of the colored words is done by the sub in lines 810-820. CC takes the COLOR value kept in the A(X) array. SC is the sprite number corresponding to the first sprite of each group of three, therefore Y determines the WORD displayed (if Y=2, SC will be 4, and sprites No. 4 to No. 6 correspond to characters 71-82-78, GRN; Y=3 is for YEL, etc). Pixelrow R and pixelcolumn C are also chosen randomly. Our three sprites, from SC to SC+2, are then colored by CC.

The actual display of the sprites is done by the CALL LOCATE sub in line 830. Since our sprites are magnified twice, the distance between each character is 16 pixelcolumns (two TI normal characters). The D delay in line 820 gives you a few seconds to read the word and tell its color (not its meaning!) before our sprites are sent back into hiding at row 193.

When you press a key to end the game (line 750), the sprites disappear, and you are sent to another screen display of text, then a key press to get back to the menu.

WHAT DO YOU SEE?

The last puzzle in lines 840-990 is just a graphic display that you have to decipher.

Since the graphic is built along simple lines, the characters all have a common look, so I simplified the character definition process by splitting them into variables. Line 890 concerns characters 104-107 and line 900 characters 108-111. Doesn't it look better than two 64-character long strings? All the sub in line 990 has to do is put the variables in the right order. More characters are defined in line 910, as well as a line made up of eight spaces.

Lines 920-940 display the graphic. Careful! All the characters are lowercase letters, so it is an "l", not a "1" (one), that you need to type. When the display is done, the picture appears as if by magic when we color the characters (they were colored black on black in line 880). The GOSUB 1020 waits for you to press a key. When you do, you get the answer to the puzzle, another key press, and a return to the main menu.

Have fun, but don't strain your eyes!

VISUAL PERCEPTION

```
100 REM ** VISUAL PERCEPTION
 ** by L. Dorais / Ottawa UG
 / Sept. 1990 !109
110 DATA dummy !140
120 DIM A(5),B(5):: GOTO 140
 :: C,CC,D,K,R,RR,S,SC,X,Y,A
$,B$,C$,D$ :: CALL CHAR :: C
ALL HCHAR :: CALL COLOR !212
130 CALL SPRITE :: CALL DELS
PRITE :: CALL MAGNIFY :: CAL
L LOCATE :: !@P- !062
140 CALL SCR(8):: DISPLAY AT
(3,6):"VISUAL PERCEPTION": :
 : : :"  1- COLOR CONTRAST":
 :"  2- WHICH IS THE LONGEST
?" !074
150 DISPLAY AT(12,3):"3- SAY
 THE COLOR": :"  4- WHAT DO
YOU SEE?": : :"  5- END" !06
6
160 GOSUB 1020 :: IF K<49 OR
 K>53 THEN 160 !166
170 IF K=53 THEN END ELSE ON
 K-48 GOTO 190,510,670,850 !
206
180 !!131
190 ! ** color contrast ** !
056
200 CALL SCR(15):: DISPLAY A
T(1,8):"COLOR CONTRAST": :"I
F YOU LOOK LONG ENOUGH AT  A
 SQUARE ON A CONTRASTING" !0
01
210 DISPLAY AT(5,1):"COLOR,
AND THEN IF IT DIS-  APPEARS
, YOUR EYE WILL KEEP A 'MEMO
RY' OF IT." !196
220 DISPLAY AT(9,1):"IN THIS
 EXERCISE, YOU ARE   ASKED T
O CHOSE A CONTRAST,  THEN TH
E SCREEN WILL SHOW   A SQUAR
E SET ON CONTRASTING" !073
230 DISPLAY AT(13,1):"COLOR.
": :"LOOK AT IT LONG ENOUGH,
 NOT LESS THAN A MINUTE. THE
N    PRESS ANY KEY: THE SQUA
RE" !097
240 DISPLAY AT(18,1):"WILL D
ISAPPEAR, AND YOU    SHOULD
 SEE ITS 'MEMORY'.": :"WHEN
YOU HAVE ENOUGH, PRESS A KEY
 AGAIN." !112
250 GOSUB 1010 :: CALL CLEAR
 :: A$="FFFFFFFFFFFFFFFF" ::
 CALL CHAR(97,A$)!060
260 DATA 16,14,9,3,11,14,2,1
6,12,5 !238
270 RESTORE 260 :: FOR X=1 T
O 5 :: READ A(X),B(X):: CALL
 COLOR(9+X,A(X),B(X)):: Y=8*
X+96 !029
280 CALL CHAR(Y,"",Y+1,"0000
3C3C3C3C",Y+2,A$,Y+3,"FFFFC3
C3C3C3FFFF"):: NEXT X !087
290 GOSUB 380 :: GOSUB 470 :
: CC=SC :: GOSUB 460 !113
300 GOSUB 480 :: IF K=89 THE
N 290 !023
310 DISPLAY AT(10,1):"NOW, W
HAT HAPPENS IF THE   SQUARE
 DOES NOT DISAPPEAR,  BUT TH
E COLORS ARE INVERTED?" !098
320 DISPLAY AT(15,1):"DO YOU
 WANT TO TRY? (Y/N)" :: GOSU
B 490 :: IF K=78 THEN 140 !2
37
330 GOSUB 380 :: GOSUB 470 :
: D=SC :: SC=CC :: CC=D :: G
OSUB 460 :: GOSUB 480 :: IF
K=89 THEN 330 !086
340 DISPLAY AT(5,1):"IF YOU
LOOKED AT THE FIRST  SQUARE
LONG ENOUGH, THEN    ITS 'ME
MORY' WILL MIX WITH  THE NEW
 COLOR, AND THE COLOR" !066
350 DISPLAY AT(9,1):"OF BOTH
 THE SQUARE AND THE  BACKGRO
UND SHOULD BE SOME   MIXTURE
 OF BOTH..." !096
360 DISPLAY AT(15,1):"NOW, A
RE YOU SURE YOU LOOKEDAT THE
 FIRST SQUARE FOR MORETHAN O
NE MINUTE???" :: GOSUB 1010
:: GOTO 140 !186
370 ! subs !096
380 CALL CLEAR :: FOR X=1 TO
 5 :: Y=8*X+96 :: R=3 :: C=6
*X-3 :: D=64 !238
390 GOSUB 450 :: Y=Y+2 :: R=
11 :: D=69 :: GOSUB 450 :: N
EXT X !065
400 DISPLAY AT(22,6):"PRESS
A KEY FOR":"    CHOICE OF CO
NTRAST" !104
410 GOSUB 1020 :: IF K<65 OR
 K>74 THEN 410 ELSE K=K-64 !
109
420 IF K<6 THEN CC=A(K):: SC
=B(K)ELSE CC=B(K-5):: SC=A(K
```

(See Page 9)

VISUAL PERCEPTION—

(Continued from Page 8)

```
-5)!088
430 CALL CLEAR :: GOSUB 460
!112
440 FOR X=10 TO 15 :: DISPLA
Y AT(X,12):"aaaaaa" :: NEXT
X :: RETURN !169
450 FOR K=R TO R+2 :: CALL H
CHAR(K,C,Y,3):: NEXT K :: CA
LL HCHAR(R+1,C+1,Y+1):: CALL
 HCHAR(R+4,C+1,D+X):: RETURN
 !162
460 CALL SCREEN(SC):: CALL C
OLOR(9,CC,1):: RETURN !248
470 FOR D=1 TO 500 :: NEXT D
 :: GOSUB 1020 :: RETURN !20
8
480 GOSUB 470 :: CALL SCR(15
):: DISPLAY AT(5,3)BEEP:"ANO
THER CONTRAST? (Y/N)" !106
490 GOSUB 1020 :: IF (K<>78
AND K<>89)THEN 490 ELSE RETU
RN !242
500 !!131
510 REM ** which is the long
est? ** !109
520 CALL SCR(11):: A$=RPT$("
a",16):: B$=RPT$(" ",8)!121
530 CALL CHAR(97,"FF",98,"00
020408102040 80",99,"FF804020
10080402",100,"0040201008040
201",101,"FF01020408102040")
!024
540 CALL CHAR(102,"004020100
8040201",103,"00010204081020
40",104,"00020408102040 80",1
05,"00804020100 80402")!025
550 R=2 :: RR=20 :: C=2 :: C
C=9 :: CALL COLOR(9,2,1,10,2
,1):: GOSUB 640 !035
560 DISPLAY AT(10,6)BEEP:"WH
ICH IS THE LONGEST?": :B$&"1
- LINE AT TOP":B$&"2- LINE A
T BOTTOM":B$&"3- NEITHER" !0
00
570 GOSUB 1020 :: IF K<49 OR
 K>51 THEN 570 !064
580 FOR X=10 TO 14 :: DISPLA
Y AT(X,1):"" :: NEXT X !031
590 DISPLAY AT(R,1):"":"" ::
 DISPLAY AT(RR,1):"":"" !192
600 R=R+2 :: RR=RR-2 :: C=C+
1 :: CC=CC-1 :: GOSUB 640 ::
 IF C<6 THEN 590 !016
610 DISPLAY AT(9,6):A$&"aa"
:: IF K=51 THEN C$="RIGHT!"
ELSE C$="WRONG!" !177
620 DISPLAY AT(19,12):C$: :"
BOTH ARE EXACTLY THE SAME
LENGTH..." :: GOSUB 1010 ::
GOTO 140 !251
630 ! sub !237
640 DISPLAY AT(R,C):TAB(C);"
b"&B$&B$&"d":TAB(C);"c"&A$&"
e" !232
650 DISPLAY AT(RR,CC):TAB(CC
);"f"&B$&B$&"  h":TAB(CC);"g
"&A$&"aai" :: RETURN !035
660 !!131
670 ! ** say the color ** !1
20
680 CALL SCR(4):: CALL MAGNI
FY(2)!027
690 DISPLAY AT(1,9):"SAY THE
 COLOR...": : :"IN THIS EXER
CISE, YOU WILL  SEE THE FOLL
OWING WORDS:" !075
700 DISPLAY AT(8,4):"RED   B
LU   GRN   YEL": : :"TRY TO
SAY THE COLOR USED   TO WRIT
E THE WORD..." !091
710 DISPLAY AT(14,1):"BEFORE
 YOU READ THE WORD!": : :"SE
EMS EASY?  JUST TRY...": :
:"AND PRESS ANY KEY WHEN  YO
U HAVE ENOUGH!" !207
720 DATA 66,76,85,71,82,78,8
9,69,76,82,69,68,5,3,12,7 !2
02
730 RESTORE 720 :: FOR X=1 T
O 12 :: READ CC :: CALL SPRI
TE(#X,CC,1,193,1):: NEXT X :
: GOSUB 1010 !053
740 RANDOMIZE :: CALL SCR(2)
:: FOR X=1 TO 4 :: READ A(X)
:: Y=X :: GOSUB 810 :: NEXT
X !026
750 CALL KEY(0,K,S):: IF S=0
 THEN 760 ELSE CALL DELSPRIT
E(ALL):: CALL SCR(4):: GOTO
770 !068
760 X=INT(4*RND)+1 :: Y=INT(
4*RND)+1 :: GOSUB 810 :: GOT
O 750 !139
770 DISPLAY AT(5,1):"SO, HOW
 WAS IT???": : :"IT IS NOT T
HAT EASY FOR THE BRAIN TO FO
RGET THE WORD ANDCONCENTRATE
 ON THE COLOR..." !137
780 DISPLAY AT(13,1):"THIS I
S BECAUSE YOUR BRAIN  HAS GI
VEN THE WORD BEFORE   YOUR C
ONSCIOUS THOUGHT CAN  RESPON
D." !225
790 GOSUB 1010 :: GOTO 140 !
162
800 ! subs !096
810 CC=A(X):: SC=3*Y-2 :: R=
INT(176*RND)+1 :: C=INT(208*
RND)+1 :: CALL COLOR(#SC,CC,
#SC+1,CC,#SC+2,CC)!019
820 GOSUB 830 :: FOR D=1 TO
200 :: NEXT D :: R=193 :: GO
SUB 830 :: RETURN !028
830 CALL LOCATE(#SC,R,C,#SC+
1,R,C+16,#SC+2,R,C+32):: RET
URN !003
840 !!131
850 ! ** what do you see? **
 !075
860 CALL SCR(10):: DISPLAY A
T(1,7):"WHAT DO YOU SEE?": :
 : : :"  WHEN YOU PRESS A KE
Y,":"  YOU WILL SEE A PICTUR
E." !013
870 DISPLAY AT(9,3):"LOOK AT
 IT FOR A WHILE.": :"  WHEN
YOU THINK YOU KNOW":"  WHAT
IT REPRESENTS,": :"  PRESS A
 KEY AGAIN..." !173
880 GOSUB 1010 :: CALL SCR(2
):: CALL COLOR(9,2,2,10,2,2)
!239
890 A$="0103070F" :: B$="FF7
F3F1F" :: C$="1F3F7FFF" :: D
$="0F070301" :: X=104 :: GOS
UB 990 !051
900 A$="80C0E0F0" :: B$="FFF
EFCF8" :: C$="F8FCFEFF" :: D
$="F0E0C080" :: X=108 :: GOS
UB 990 !203
910 CALL CHAR(97,"",99,"FFFF
FFFFFFFFFFFF"):: B$=RPT$(" "
,8)!017
920 DISPLAY AT(4,9):"laaaaaa
aaaah":B$&"coaaaaaaaak":B$&"
claaaaaaaah":B$&"cclaaaaaah"
:B$&"ccmaaaaaai" !175
930 DISPLAY AT(9,9):"cclaaaa
aah":B$&"ccclaaaah":B$&"cccm
aaaai":B$&"ccclaaaah":B$&"cc
cnaaaaj" !178
940 DISPLAY AT(14,9):"cccmaa
aai":B$&"ccclaaaah":B$&"cccm
```

(See Page 10)

VISUAL PERCEPTION—

(Continued from Page 9)

```
aaaai":B$&"ccmaaaaaai":B$&"c
caaaaaaaa":B$&"cmaaaaaaaai":
B$&"caaaaaaaaaa" !229
950 CALL COLOR(9,2,16,10,2,1
6):: GOSUB 1020 :: CALL SCR(
10)!149
960 DISPLAY AT(5,1):"IF YOU
SAW...": : :"   A VASE,": :"
   OR TWO PROFILES,": : :"YO
U ARE EQUALLY RIGHT..." !231
970 GOSUB 1010 :: GOTO 140 !
162
980 ! sub !237
990 CALL CHAR(X,A$&C$&B$&D$&
B$&C$&A$&D$):: RETURN !069
1000 !@P+ ** global subs **
!148
1010 DISPLAY AT(24,7):"PRESS
 ANY KEY..." !150
1020 CALL KEY(0,K,S):: IF S=
0 THEN 1020 ELSE RETURN !076
1030 SUB SCR(X):: CALL CLEAR
 :: CALL SCREEN(X):: SUBEND
!008
```

NEWSBYTES

Vendor info sought for list

Gary Cox of the Mid-South Users Group is putting together the group's annual list of vendors of TI99/4A and Geneve products.

The list is scheduled for publication in the group's newsletter and posting on its BBS.

Cox says he is currently trying to put together a new accurate listing which includes all TI vendors' addresses, voice phone numbers and BBS numbers (if applicable).

He asks that any vendor who would like to be on the list send a post card, business card, letter or catalog to Mid-South (Memphis) TI User Group, Attn: Gary Cox, P.O. Box 38522, Germantown, TN 38183-0522. Cox prefers a catalog or flyer so he can see what products are carried.

VAST makes changes

The VAlley of the Sun TI99ers (VAST) has a new address: VAST User Group, P.O. Box 37725, Phoenix, AZ 85069. Its new BBS number is (602) 789-0012.

Ralph E. Rees, president and newsletter editor for the group, notes that the group will celebrate its 10th anniversary in November and still has two founding members in the club, Walt Brown and Dan Schell.

TI-Meeting set for Germany

The TI-Club Goettingen is scheduled to host the 9th International TI-Meeting Oct. 14-16 in Goettingen (Rosdorf).

Location is Kirchl. Gemeindehaus Rosdorf, Kirchgasse2, D-37124 Rosdorf, Germany. This is the civic center for Goettingen-Rosdorf.

Setup begins at 5 p.m. Oct. 14 and the meeting opens at 10 a.m. the following two days. Beverages and warm snacks will be available, according to the organizers.

Persons arriving by car leave the Autobahn A7 (Hannover/Kassel) at the Rasthof (Service Area) Göttingen. After continuing about 2 km, you arrive at the village of Rosdorf. Drive to the first major intersection (Mengershäuser Weg/Obere Strasse). Turn left at this intersection into Obere Strasse. After about 200 m., turn right at the next small intersection into Kirchgasse, and you arrive at the Gemeindehaus after driving straight ahead another 200 m.

Persons arriving by train should go to the bus station slightly to the right of the train station and take Bus 19, which leaves directly for Rosdorf every hour. Get out at the Obere Strasse bus stop and go back to the last intersection. Turn right into Kirchgasse, walk 200 m. and you are there.

The route will be marked, according to organizers.

For registration and information, contact Jörg Kirst, Mengershäuser Weg 5, D-37124 Rosdorf, Germany, tel. 0551/781153; Reinhard Obuch, Keplerstr. 5, D-37085 Göttingen, Germany, tel. 0551/46405; or Hans-Hartmut Kortry, Grüner Weg 10, D-37181 Hardegsen, tel. 05505/1470.

THE ART OF ASSEMBLY — PART 38

Trying the impossible

By BRUCE HARRISON

It's September 1993 as we write this, and we are in the middle of an endless project. For some time now, it's been the opinion of many skilled TI programmers that it would be impossible to make a compiler for TI Extended BASIC. We think that they've been wrong, and are trying to prove it. The only way to prove the experts wrong in such matters is, of course, to do the very thing that was declared impossible.

THE COMPILER PROBLEM

We have used compilers for BASIC programs on PC computers, and found that there's one problem that those compilers have in common: The size of the program we end up with is huge compared to what we started with. In a typical case, one may start with a program of 4,000 bytes, perform the compiling and linking steps, and find the .EXE file takes over 30,000 bytes. In the most extreme example, we tried just the simplest possible kind of BASIC program, with only one line, like this:

10 PRINT "Hello"

When compiled and linked, this simple program became a 29,000-byte monster. (As we recall, the original one line BASIC program took all of about 10 bytes.) How, you ask, can this happen? The answer lies in the approach taken to the problem of making the compiler, and in the nature of what a "program" in BASIC really is.

WHAT IS THE PROGRAM?

From the beginning of BASIC, there's a misconception that's created in our minds, to the effect that the collection of BASIC instructions which we write and save to the disk is a program for the computer. It's NOT! What's called a BASIC program is really just a collection of data. That data is used by a program called the *BASIC Interpreter* to make the computer do things. In the PC case, the BASIC Interpreter is a program called BASIC.EXE, of about 78,000 bytes, which must be loaded into the computer's memory before any BASIC "program" can be loaded or run. The interpreter typically includes a large number of routines in machine code, which are used in ways determined by the content of the BASIC "program" that it's "running." What's really running is the interpreter itself, which looks at the contents of each statement in this data, determines what routines and with what parameters need to be executed, and executes those routines.

On the TI, the BASIC Interpreter does not need to be loaded from any disk, but resides in the computer itself, or in the cartridge for Extended BASIC. The TI's case is complicated by the fact that there are really two interpretation steps required. First, the data which we call the BASIC program is "translated" into a series of GPL instructions, then those are used by the GPL Interpreter. This two-stage process may have been needed to keep the memory requirements within bounds, as the GPL code is very compact, and thus allowed even the "console" BASIC to be a

(See Page 12)

```
* SIDEBAR 38
* COMPILER INPUT AND OUTPUT
* (DOES NOT INCLUDE SUPPORTING FILES)
*
* FIRST, THE ORIGINAL XB PROGRAM
* (LISTED IN 28 COLUMNS)
*
10 FOR I=1 TO 30
20 IF I<10 THEN PRINT "I<10"
;I ELSE IF I<20 THEN PRINT "
I=>10";I ELSE PRINT "I>19";I
30 NEXT I
*
* THIS WAS SAVED WITH MERGE OPTION
* AS DSK4.IFTEMER
* COMPILER THEN PRODUCED THE FOLLOWING
*
*
* FIRST IS THE "SHELL" XB PROGRAM IFTE/M
* PRODUCED BY THE COMPILER (MERGE FORMAT)
*
1 CALL INIT
10 GOTO 100
11 I
100 CALL LINK("MAIN")
101 !@P-
32767 CALL LINK("BACK")
*
* SECOND OUTPUT IS THE ASSEMBLY SOURCE FILE
*
* ASSEMBLY SOURCE FILE
* HARRISON XB COMPILER
* DERIVED FROM:
* DSK4.IFTEMER
       COPY    "DSK4.STDOPN"          COPY IN THE "STANDARD
OPEN" FILE
L10    BL      @SETCL      SET VALUE OF CURRENT XB LINE
       DATA    10          AT 10
       BL      @FORSET     SET UP A FOR-NEXT LOOP
       DATA    1           FROM 1
       DATA    30          THROUGH 30
       DATA    1           STEP 1
       DATA    IV0         VARIABLE I (INT VARIABLE 0)
       DATA    >0000       ALL PARAMETERS ARE JUST NUMBERS
LM0    DATA    0,0         RESERVE WORDS FOR LIMIT AND STEP
FR0
L20    BL      @SETCL      LABEL FR0 IS START OF LOOP
       DATA 20
       CLR     @TRUFLG     CLEAR "TRUTH FLAG" FOR IF-THEN
       BL      @IVTFP      TRANSFER INTEGER VARIABLE TO
FLOATING POINT
       DATA    IV0         USING VARIABLE IV0 (I)
       BL      @TOGI       USE GPL INTERPRETER
       DATA    FI0         ON FAKE IF #0
       MOV     @TRUFLG,R4  MOVE THE TRUTH FLAG
       JEQ     EL0         IF IT'S ZERO, JUMP TO ELSE #0
       BL      @GETSC      GET A STRING CONSTANT
       DATA    SC0         STRING CONSTANT #0
       BL      @PRNV       PRINT THAT
       DATA    180         WITH PENDING PRINT (;)
```

THE ART OF ASSEMBLY—

(Continued from Page 11)

very rich language. We could "second guess" that idea, but there's nothing we can do about it now. Just remember that in the case of the TI, the program that's actually running when we're in BASIC or Extended BASIC is the GPL Interpreter.

In the PC compilers we've used, there are two possible ways to "link" the output program. The first is to use what's called a *Run-time Library* for execution. This limits the final product to a series of calls to another set of routines, which the computer loads into memory along with the .EXE program. (In this method, the Run-time library file must be kept on the same disk as the compiled program.)

The second method for compiling is to create a standalone .EXE program, in which case the routines from a linking library are combined directly into the .EXE program itself, so no separate file is needed to run the resulting program. In either case, there is a huge amount of memory required, so that the routines that emulate similar ones from the Interpreter will be in memory when needed.

WHY IMPOSSIBLE?

Taking an approach like that done on the PCs to making a compiler for the TI's BASICs is clearly impossible, because the required machine code to emulate what the interpreter provides would make even short BASIC "programs" become too large to fit in memory when compiled. The answer, if there ever is one, will be to make the compiler work in conjunction with what's already built into the computer. TI did its level best to make that "solution" itself impossible, by keeping all the inner workings of the BASIC and GPL interpreters secret.

At this stage (September 1993) the fundamental framework of the compiler is looking like this: The "source" XB program will be saved in MERGE format. The compiler will load from Extended BASIC. It will read the MERGE format file and produce three output files. The first will be a "shell" XB program in merge format. This shell will allow the compiled program to present itself to the computer as XB.

The second output file will be an assembly source file. The third output file will be an auxiliary data file used with the assembly source file. A special loader provided by Harry Wilhelm will put together the mergeable "shell" and the assembled object file into what will look like an XB program. The compiler will be able to perform all of its steps without the need for an E/A cartridge.

We are making progress toward our goal, and have sent out some "samples" for testing to various people in the community. It's too early to say whether we'll succeed totally, but we have already made FOR-NEXT, PRINT, IF-THEN-ELSE, HCHAR, VCHAR, and CALL KEY into assembly routines, so that "demo" programs can perform as compiled programs under XB.

Along the way, we've found out some interesting tidbits of information. Just the other day, for example, we discovered a way to fool Extended BASIC into thinking that we're running a line number from the original program. This is used with the ERROR function and with the BREAK to allow XB to report just the way

(See Page 13)

```
        BL      @PRNIV      PRINT INTEGER VARIABLE
        DATA    IV0         NUMBER 0
        DATA     0          WITH NO PENDING PRINT
        B       @L30        GOTO LINE 30
EL0
        CLR     @TRUFLG     CLEAR TRUTH FLAG
        BL      @IVTFP      TRANSFER IV TO FP
        DATA    IV0         INT VARIABLE #0
        BL      @TOGI       USE GPL INTERPRETER
        DATA    FI1         FOR FAKE IF #1
        MOV     @TRUFLG,R4  CHECK THE TRUTH FLAG
        JEQ     EL1         IF NOT TRUE, GO TO ELSE #1
        BL      @GETSC      GET STRING CONTANT
        DATA    SC1         NUMBER 1
        BL      @PRNV       PRINT THAT
        DATA    180         WITH PENDING PRINT (;)
        BL      @PRNIV      PRINT INTEGER VARIABLE
        DATA    IV0         NUMBER 0 (I)
        DATA    0           WITH NO PENDING PRINT
        B       @L30        GOTO LINE 30
EL1
        BL      @GETSC      GET STRING CONSTANT
        DATA    SC2         NUMBER 2
        BL      @PRNV       PRINT THAT
        DATA    180         WITH PENDING PRINT (;)
        BL      @PRNIV      PRINT INTEGER VARIABLE
        DATA    IV0         IV #0
        DATA    0           WITH NO PENDING PRINT
L30     BL      @SETCL      SET LINE INDICATOR
        DATA    30          LINE 30
        BL      @NXTIP      USE "NEXT" SUBROUTINE
        DATA    LM0         FOR DATA AT LM#0
        DATA    IV0         USING VARIABLE IV0
        B       @FR0        IF WITHIN LIMIT, BACK TO LABEL
FR0
        BL      @TOGI       ELSE TO GPL INTERPRETER
        DATA FEND             WITH FAKE "END" LINE (ENDS PRO-
GRAM)
        COPY    "DSK4.STDSUB"   COPY IN STANDARD SUBROUTINES
        COPY    "DSK4.FORNEXT"   COPY IN FOR-NEXT ROUTINES
        COPY    "DSK4.PRINT"   COPY IN PRINT ROUTINE
        COPY    "DSK4.STDDAT"   COPY STANDARD DATA SECTION
        COPY    "DSK4.IFTE/A"   COPY AUXILIARY DATA FILE
VARTBL  BYTE    0,0,1       XB VARIABLES TABLE (0,0 BECOMES
ADDRESS)
        TEXT    'I'
ENDTBL  EQU     $
* EACH VARIABLE IS LISTED WITH TWO BYTES RESERVED FOR
* ITS XB ADDRESS, THEN LENGTH OF ITS NAME, THEN THE NAME
        EVEN                ENSURE AN EVEN ADDRESS
IVTBL   EQU     $
IV0     DATA    0,2         0 BECOMES "I", 2 IS CROSS-REFER-
ENCE TO VARTBL
ENDIV   EQU     $
        END
*
*
*
* LAST IS FILE IFTE/A, AUXILIARY DATA
* FI0 AND FI1 ARE FAKE IF'S
* SC0, SC1, AND SC2 ARE STRING CONSTANTS
* CONTAINING "I<10", "I=>10" AND "I>19"
*
FI0     BYTE    130,132,73,191,200,2,49,48
* ABOVE IS TOKENIZED FOR :: IF I<10
        BYTE    >B0,>9D,>C8,4,76,73,78,75
        BYTE    >B7,>C7,4,84,82,85,69,>B6,>81,>C9,>7F,>FF
* ABOVE IS TOKENIZED FOR THEN CALL LINK("TRUE") ELSE 32767
SC0     BYTE    4,73,60,49,48
FI1     BYTE    130,132,73,191,200,2,50,48
        BYTE    >B0,>9D,>C8,4,76,73,78,75
        BYTE    >B7,>C7,4,84,82,85,69,>B6,>81,>C9,>7F,>FF
SC1     BYTE    5,73,61,62,49,48
SC2     BYTE    4,73,62,49,57
```

THE ART OF ASSEMBLY—

(Continued from Page 12)

it would if the original program were running. That way, if the user finds the compiled program stopping with an error, he'll know where in the source XB program to look for that error.

By the way, we have made the compiler so that Function-4 will "break" the program, and so that CONtinue will work just as it does in XB programs, picking up just where it left off when Function-4 was struck.

REAL SOURCE CODE

Today's sidebar shows a short XB program, then the source code generated from that file by the compiler. No, this hasn't been "fudged" in any way, but we've added annotation so you'll be able to understand what's happening.

At some places in the source code, you'll see lines that say BL @TOGI. This means that the compiler will turn over the next operation to the GPL Interpreter to perform. What GPL is to perform is indicated by the DATA following the BL. In most cases that's an FLXX, where XX is the number of the FL label to be used. FL is just a mnemonic for Fake Line.

This trick was not our invention, but was passed along by Harry Wilhelm. The fake line starts with the token for :: (130), then contains the tokenized form of the original program line, and ends with the tokens for :: GOTO 32767. Line 32767 of the mergeable "shell" program simply says CALL LINK("BACK"), and that returns control to the Assembly part of the compiled program right after the DATA for the TOGI line.

In the compiler's present state of development, there are many XB statements that the compiler can't handle, and when it fails to find the statement's opening token among its own routines, it branches to a section of code that generates this Fake Line and the BL and DATA for the main code section of the output source file. Thus the compiler can handle what's not included by letting XB handle it.

For IF-THEN, it gets a bit more complicated, in that we first clear a flag word at TRUFLG, then BL @TOGI, but with an FI label (for Fake If) in the DATA line. The FI line starts with ::, and contains everything in the IF clause up to but not including the THEN token. The FI line continues with THEN CALL LINK("TRUE") ELSE 32767. (All in tokenized form, of course) This means that if what's in the IF clause is true, control will return to the Assembly code at label TRUE, and this will set TRUFLG to one, so that the Assembly code can determine what to do next.

If there's no ELSE, the compiler will generate an EL label at the appropriate place to handle what's to be done if the IF is not true. The compiler also checks what's after THEN, to see if there's a GOTO to skip past the ELSE part, and it inserts a GOTO if needed and as appropriate. (see B @L30 in the sidebar)

The compiler generates labels with some idea of being consistent and mnemonic. For example, each new line of the source XB file creates a label that starts with L, followed by the line number. This way, one can very easily find the part of the assembly source file that was derived from each line of the source file.

VARIABLES

Handling the variables was an early challenge. Since we knew at the outset that some functions would forever be handled by the "Fake Line" process, it was imperative that our way of handling the variables be compatible with the way XB handles them. At the same time, we wanted some processes, like FOR-NEXT loop control, to be handled as integers only, so that speed could be improved. In essence, we wanted to "have our cake and eat it too".

That's exactly what we've done in this compiler. All variables used in the original program are listed in the shell XB merge file, so that XB can perform its normal pre-scan to reserve space for them and build its symbol table in VDP ram. In the assembly source code, we make two tables for the variables. VARTBL includes all variables used in the original program, both string and numeric. Our own integer variables have their own table just after the table of XB's variables. Each entry in the Integer table has two words, where the first is reserved for the value of that variable, and the second is the cross-reference to the position of the corresponding variable in the main table.

When the compiled program is running, our assembly part in the file STDOPN looks up all the variables' addresses in VDP RAM, and puts those into our VARTBL (where the 0,0 is in each variable entry) so that we can get any variable's current state from XB, and we can pass back values set in the integer variables when we need to. The compiler determines where in our assembly part this needs doing, and BL's to either FPTIV or IVTFP as appropriate, with DATA indicating which integer variable is to be passed to or from.

MODULARITY

The assembly support routines like FOR-NEXT, KEY, HVCHAR and so on, are contained in separate assembly source files, so they can be incorporated by COPY directives only if needed by the particular program. When we are at the point that we consider this product "finished", both the compiler itself and all its source code will be released to public domain, so that others may write additional support modules and integrate them as desired.

AN UPDATE

Last month we showed a way to use a CALL INIT within an assembly program, and explained that this messed up our return to E/A. As in many such cases, there's a "brute force" solution to this problem. If one saves the entire contents of CPU RAM Pad (256 bytes starting at >8300) before the INIT call, and then restores that before ending the program, as well as setting the word at >2030 to >061C and saves and restores the GROM address, the return to E/A will work correctly. Yes, that's quite a lot of "bother", just to prove a point, but it's just another symptom of the "secrecy syndrome" which affects everything we try to do on this little machine.

CHAPTER 11

Back in part 22, we wrote about what we called "The Business End." There's an old joke in the business world that goes: "We lose 20 cents on each item we sell, but we make it up in volume." That's no joke when it happens in real life. Our company did actually make a small profit in 1992, but that profit was nothing compared to the hundreds of hours that went into the making of the

(See Page 14)

THE ART OF ASSEMBLY—

(Continued from Page 13)

products being offered for sale. A hard decision had to be made, then, as to what the "business" would do in the year 1993. The decision was that most of our commercial business would cease operation at the end of '93. The "profits" for '92, measured in percentage, were actually pretty good, coming in at somewhere around 20 percent of sales (before taxes), but when that "bottom line" in dollars is less than one-quarter of one month's annuity from our Federal Retirement fund, it's simply not worth the endless hours spent in developing new products. If a product takes 200 hours to develop, then sells six copies at $10 "profit" each, that's an effective "salary" of only 30 cents per hour. Worse yet, that 30 cents per hour is taxed as income at about 40 percent between the IRS, the state, and the county, so the "after tax" yield is only 18 cents per hour. Our Maryland prison system pays higher hourly wages to those who make license plates. President Clinto has promised to lower this after-tax yield for us in 1993. It just wouldn't do for us to become rich from this business.

We will still offer some services, in the form of our custom program assistance for MIDI-Master users, and for those who need customized modules for use with other programs, but our standard product catalog is gone. We will continue programming, but what we develop will be aimed at the public domain or "Freeware" market. That is, it will be made available through user groups and such outlets for just whatever their copying fee amounts to.

This way, the users will get the benefit of our expertise, and we won't have to struggle with the question of "profits" that must be shared with the IRS, the Maryland State Comptroller, and the County of Prince Georges.

Extended BASIC

Mail merge envelope printer

By JERRY KEISLER

I ran across an article in MICROpendium the other day that got me thinking. It was a request for a program that will read a mail list file made with TI-Writer and print envelopes with both a return address and those addresses while ignoring the asterisks and numbers. I wrote a simple program and printed 82 envelopes from a mail merge file (same as mail list above) in about 15 minutes. I had to hand-feed the envelopes, as I have no tractor-feed envelopes. I also like to do things the cheap way and tractor-feed envelopes cost more than normal envelopes.

I considered the program for MICROpendium, but the way I write programs they only do one thing and all the variables have to be entered by listing the program.

Three days later, good thing it was the July 4th weekend, the program was completed. The program, now complete with internal HELP, will:

1. Read six styles of address anywhere in TI-Writer or FunnelWeb mail merge files.
2. Allow for hand-feed or tractor-feed of envelopes.
3. Print both small and large envelopes.
4. Allow viewing of the raw mail merge file to determine type of address setup.
5. Display the formatted address ready for print before printing it.
6. Allow selective printing of envelopes so you can choose the ones you want to print.
7. Allow the saving of all default values you normally use to the program.
8. Allow changing of defaults from the running program.
9. Remove blank lines from the address.

The hard part was getting the program to adjust to the various mail merge types and reading them from the indicated line in the file. Instructions are contained in the program, so you will not have to keep up with a document file.

Sample Address File

```
1 Tom Jones
2 2314 Anystreet
3 Anytown, State 12345
4 Ralph Friendly
*
1 Jane Smith
2 11 Anylane
3 Anycity, State 54321
4 Ted Smilley
*
1 Firstname Lastname
2 Street
3 City, State Zip
4 Account Supervisor
*
```

TIW-ENV

```
100 !SAVE DSK1.TIW-ENV !029
110 !                          !
120 !    TIW-ENV               !
130 ! by Jerry Keisler         !
140 !   July 1994              !
150 !                          !
160 ! prints address from      !
170 ! mail merge files         !
180 ! and return address       !
190 ! to envelopes.            !
200 !                          !
210 ! -- default variables     !
219
220 RA$(1)="JERRY KEISLER" !
107
230 RA$(2)="2221 COLLEGE DR"
 !075
240 RA$(3)="PARIS, TX 75460"
 !081
250 RA$(4)="" ! Return Addre
ss 4 lines max !164
260 FL=1 ! first line in add
ress part of mail merge file
  !135
270 FN$="DSK1.MAILLIST" !Fil
```

(See Page 15)

TIW-ENV—

(Continued from Page 14)

```
eName of mail merge file. !0
68
280 PN$="PIO" ! Printer Name
 !090
290 SZ$="L" !SiZe (L)egal or
 (S)mall envelope !224
300 FD$="H" ! envelope FeeD
is either (H)and or (T)racto
r !203
310 TF=7 !Tractor line Feed
default. LEGAL=18+TF. SMALL=
15+TF. there are 6 lines per
 inch. !014
320 TY=1 !TYpe of mail merge
 address layout. !068
330 PE1$=CHR$(27)&"8" !turns
 off paper out detector. !07
2
340 PE2$=CHR$(27)&"9" !turns
 on paper out detector. !236
350 GOTO 480 :: CALL KEY ::
B$(),FL$,Q$="" :: EBL,I,J,K,
S,LC,MBL,PA,TA=O :: !@P- !fa
st scan !159
360 ! -- other variables !02
4
370 ! B$() = address !189
380 ! EBL = number of End Bl
ank Lines from end of addres
see to top of next envelope,
 or eject. !009
390 ! FL$ = 2 character stri
ng of FL. !097
400 ! I,J = for next variabl
es !222
410 ! K,S call key variables
 !107
420 ! LC = Line Count is num
ber of lines to read in addr
ess part of mail merge file.
 !158
430 ! MBL = number of Middle
 Blank Lines from end of ret
urn address to top of addres
see. !090
440 ! PA 1=print all, 0=sele
ct print !206
450 ! Q$ = temp string varia
ble. !150
460 ! TA = spaces to tab ove
 before printing address. !
188
470 ! -- start screen !235
480 CALL KEY(3,K,S)!190
490 DISPLAY AT(1,5)ERASE ALL
:"ENVELOPE ADDRESSER": :"Pri
nts 4 line return address":"
and 4 line addressee." !187
500 DISPLAY AT(6,1):"return
address" :: FOR I=1 TO 4 ::
DISPLAY AT(I+6,1):SEG$(RA$(I
),1,28):: NEXT I !043
510 DISPLAY AT(6,22):"HELP N
" !248
520 DISPLAY AT(12,1):"env si
ze: (S)mall (L)egal ";SZ$ !0
50
530 DISPLAY AT(13,4):"feed:
(T)ractor (H)and ";FD$ !252
540 DISPLAY AT(14,5):"mail m
erge type (1-6)";TY !139
550 DISPLAY AT(15,5):"first
mail merge line";FL !154
560 DISPLAY AT(18,1):"OPEN F
ILE ";FN$ !099
570 DISPLAY AT(24,1):"IS THI
S CORRECT? N" :: ACCEPT AT(2
4,18)SIZE(-1)VALIDATE("YN")B
EEP:Q$ :: IF Q$="Y" THEN 760
 !170
580 DISPLAY AT(24,1):"LIST P
ROGRAM FOR DEFAULTS " !133
590 ! -- input from screen !
193
600 !  - help !185
610 ACCEPT AT(6,27)SIZE(-1)V
ALIDATE("YN")BEEP:Q$ :: IF Q
$="Y" THEN 1400 !061
620 !  - return address !182
630 DISPLAY AT(7,1):"ACCEPT
CURRENT ADDRESS Y" :: ACCEPT
 AT(7,24)SIZE(-1)VALIDATE("Y
N")BEEP:Q$ !063
640 IF Q$="Y" THEN DISPLAY A
T(7,1):SEG$(RA$(1),1,28):: G
OTO 670 ELSE DISPLAY AT(7,1)
:"" !250
650 FOR I=1 TO 4 :: ACCEPT A
T(6+I,1)SIZE(-28)BEEP:RA$(I)
:: NEXT I !051
660 !  - size !203
670 ACCEPT AT(12,27)SIZE(-1)
VALIDATE("LS")BEEP:SZ$ !111
680 !  - feed !164
690 ACCEPT AT(13,27)SIZE(-1)
VALIDATE("HT")BEEP:FD$ !074
700 !  - type !210
710 ACCEPT AT(14,27)SIZE(-1)
VALIDATE("123456")BEEP:Q$ ::
 TY=VAL(Q$)!088
720 ACCEPT AT(15,27)SIZE(-2)
VALIDATE(DIGIT)BEEP:FL :: IF
 FL<10 THEN FL$=STR$(FL)&" "
 ELSE FL$=STR$(FL)!164
730 !  - mail merge filename
 !068
740 ACCEPT AT(18,11)SIZE(-15
)BEEP:FN$ :: GOTO 570 !197
750 ! -- set variables !066
760 IF SZ$="L" THEN TA=40 ::
 MBL=10 ELSE TA=30 :: MBL=7
!186
770 IF FD$="H" THEN EBL=10 E
LSE EBL=TF !024
780 RESTORE 790 :: FOR I=1 T
O TY :: READ LC :: NEXT I !2
36
790 DATA 3,4,4,5,5,6 !003
800 ! -- do what? !003
810 DISPLAY AT(21,1):"(V)iew
 ";SEG$(FN$,6,LEN(FN$)):"(S)
elective printing":"(P)rint
all":"(E)nd" !175
820 IF FD$="H" THEN DISPLAY
AT(23,1):"" !007
830 CALL KEY(3,K,S):: IF K=8
6 THEN 1310 ELSE IF K=83 THE
N PA=0 ELSE IF K=80 THEN PA=
1 ELSE IF K=69 THEN END ELSE
 830 !034
840 ! -- open files !002
850 DISPLAY AT(21,1):"":"":"
":"" !201
860 OPEN #1:FN$ !003
870 OPEN #2:PN$ :: IF FD$="H
" THEN PRINT #2:PE1$;!039
880 ! -- input address !083
890 FOR I=4 TO 6 :: B$(I)=""
 :: NEXT I !196
900 LINPUT #1:B$(1)!035
910 DISPLAY AT(24,1):"" !088
920 IF EOF(1)=1 THEN 1730 !1
74
930 IF SEG$(B$(1),1,2)<>FL$
THEN 890 !028
940 ! -- remove left number
and space and null blank lin
es. !093
950 IF LEN(B$(1))>2 THEN B$(
1)=SEG$(B$(1),3,LEN(B$(1))-2
)ELSE B$(1)="" !156
960 FOR I=2 TO LC !206
970 LINPUT #1:B$(I)!114
```

(See Page 16)

TIW-ENV—

(Continued from Page 15)

```
980 IF EOF(1)=1 THEN 1730 !1
74
990 IF LEN(B$(I))>2 THEN B$(
I)=SEG$(B$(I),3,LEN(B$(I))-2
)ELSE B$(I)="" !039
1000 NEXT I !223
1010 IF TY<3 THEN 1100 !181
1020 ! -- remove cr's from B
$(1) and B$(2) !114
1030 FOR I=1 TO 2 :: IF POS(
B$(I),CHR$(13),1)THEN B$(I)=
SEG$(B$(I),1,LEN(B$(I))-1)!2
15
1040 NEXT I !223
1050 IF TY>4 THEN 1080 !163
1060 ! -- build name line an
d move address to B$(1) thru
 B$(4) !223
1070 B$(1)=B$(1)&" "&B$(2)::
 FOR I=2 TO 4 :: B$(I)=B$(I+
1):: NEXT I :: GOTO 1100 !24
1
1080 B$(1)=B$(1)&" "&B$(2)&"
 "&B$(3):: FOR I=2 TO 4 :: B
$(I)=B$(I+2):: NEXT I !248
1090 ! -- remove blank lines
 !014
1100 FOR I=2 TO 3 !059
1110 IF B$(I)<"!" THEN B$(I)
=SEG$(B$(I),2,LEN(B$)-1):: I
F B$(I)<>"" THEN 1110 ELSE B
$(I)=B$(I+1):: B$(I+1)="" !0
67
1120 IF (TY=1)+(TY=3)+(TY=5)
THEN I=5 !055
1130 NEXT I !223
1140 ! -- view formatted add
ress !196
1150 FOR I=1 TO 4 :: DISPLAY
 AT(19+I,1):SEG$(B$(I),1,28)
:: NEXT I !178
1160 IF (PA-(FD$="T"))=2 THE
N DISPLAY AT(24,1):"(R)eturn
" :: CALL KEY(3,K,S):: IF K=
82 THEN 1730 ELSE 1210 !159
1170 DISPLAY AT(24,1):"(P)ri
nt (S)kip (R)eturn" !198
1180 CALL KEY(3,K,S):: IF K=
83 THEN 890 ELSE IF K=82 THE
N 1730 ELSE IF K<>80 THEN 11
80 !149
1190 ! -- print envelope !20
0
1200 !  - print return addre
ss !003
1210 DISPLAY AT(24,1):"" !08
8
1220 FOR I=1 TO 4 :: PRINT #
2:RA$(I):: NEXT I !212
1230 !  - insert blank lines
 !008
1240 FOR I=1 TO MBL :: PRINT
 #2:"" :: NEXT I !012
1250 !  - print addressee !0
13
1260 FOR I=1 TO 4 :: PRINT #
2:TAB(TA);B$(I):: NEXT I !05
3
1270 !  - end blank lines !1
70
1280 FOR I=1 TO EBL :: PRINT
 #2:"" :: NEXT I !004
1290 GOTO 890 !204
1300 ! -- view file as is !1
36
1310 OPEN #1:FN$ !003
1320 DISPLAY AT(1,1)ERASE AL
L:"first 28 col of ";SEG$(FN
$,6,LEN(FN$))!066
1330 FOR I=3 TO 22 !110
1340 LINPUT #1:B$(1)!035
1350 IF EOF(1)=1 THEN I=23 !
083
1360 DISPLAY AT(I,1):SEG$(B$
(1),1,28):: NEXT I !177
1370 IF EOF(1)=1 THEN DISPLA
Y AT(24,1):"EOF press enter"
 :: CALL KEY(3,K,S):: IF K<>
13 THEN 1370 ELSE CLOSE #1 :
: GOTO 490 !211
1380 DISPLAY AT(24,1):"(C)on
tinue or (R)eturn" :: CALL K
EY(3,K,S):: IF K=67 THEN 132
0 ELSE IF K=82 THEN CLOSE #1
 :: GOTO 490 ELSE 1380 !026
1390 ! -- help !198
1400 RESTORE 1480 !042
1410 FOR J=1 TO 20 !107
1420 DISPLAY AT(1,1)ERASE AL
L:"ENVELOPE ADDRESSER  H E L
 P" !252
1430 FOR I=3 TO 22 :: READ Q
$ :: IF Q$="XX" THEN I=50 EL
SE DISPLAY AT(I,1):Q$ !052
1440 IF Q$="END" THEN I=50 :
: J=50 !247
1450 NEXT I :: DISPLAY AT(24
,3):"(C)ontinue or (R)eturn"
 !089
1460 CALL KEY(3,K,S):: IF K=
82 THEN 490 ELSE IF K<>67 TH
EN 1460 !157
1470 NEXT J :: GOTO 490 !156
1480 DATA ,ENVELOPE ADDRESSE
R,,WRITTEN,,July 1994,,by,,J
erry Keisler,2221 college dr
,paris tx 75460,,for the pub
lic domain,XX !027
1490 DATA RETURN ADDRESS,,al
though only 28 columns,,of r
eturn address show on,,the s
creen - you can put,,longer
lines in the default,!053
1500 DATA values of RA$() by
 listing,,the program and sa
ving it,,with the new defaul
t values.,XX !176
1510 DATA ENVELOPE SIZE,,ent
er "S" for small,,envelopes
about 6 1/2 in.,,by 3 2/3 in
.,,,,enter "L" for legal,,en
velopes.,XX !253
1520 DATA FEED,,printer feed
 can be either,,by tractor o
r by hand.,,use (S)elective
printing,,when using (H)and.
,XX !034
1530 DATA MAIL MERGE TYPE 1,
,1 name,2 address,3 city st
zip,XX !239
1540 DATA MAIL MERGE TYPE 2,
,1 name,2 address,3 address,
4 city st zip,!052
1550 DATA OR,,1 name,2 addre
ss,3 city st zip,4 blank,,OR
,,1 name,2 address,3 blank,4
 city st zip !107
1560 DATA MAIL MERGE TYPE 3,
,1 first name,2 last name,3
address,4 city st zip,XX !14
2
1570 DATA MAIL MERGE TYPE 4,
,1 first name,2 last name,3
address,4 address,5 city st
zip,OR !063
1580 DATA 1 first name,2 las
t name,3 address,4 city st z
ip,5 blank,OR,1 first name,2
 last name,3 address,4 blank
,5 city st zip,XX !157
1590 DATA MAIL MERGE TYPE 5,
,1 title,2 first name,3 last
 name,4 address,5 city st zi
```

(See Page 17)

TIW-ENV—

(Continued from Page 16)

```
p,XX !137
1600 DATA MAIL MERGE TYPE 6,
,1 title,2 first name,3 last
 name,4 address,5 address,6
city st zip,,OR,!161
1610 DATA 1 title,2 first na
me,3 last name,4 address,5 c
ity st zip,6 blank,,OR,XX !1
45
1620 DATA MAIL MERGE TYPE 6
cont,,1 title,2 first name,3
 last name,4 address,5 blank
,6 city st zip,XX !067
1630 DATA FIRST MAIL MERGE L
INE,,the line number in the
mail,merge file that is the
first,line of the address.,,
this is normally line number
 !196
1640 DATA (1) one but can be
 any line,in the mail merge
file.,,all addresses must st
art,with the same line numbe
r !085
1650 DATA and must be listed
 in,consecutive order from t
hat,line number.,!231
1660 DATA the mail merge fil
es may,contain other data ab
ove,and below the address fi
le.,XX !109
1670 DATA PRINTER CODES,,PE1
$ and PE2$ turn paper out,,d
etector ON and OFF.,,this al
lows the printing,,and eject
ing of envelopes,!050
1680 DATA using hand feed.,,
you may have to change PE1$,
,and PE2$ for your printer.,
,they can be found by,,listi
ng the program.,XX !060
1690 DATA (V)iew shows the f
irst 28,columns of a mail me
rge file,,(S)elective printi
ng stops,the printer between
 !198
1700 DATA envelopes. this al
lows,Skipping addresses and h
and,feeding of envelopes.,It
 also shows the formatted !1
31
1710 DATA addressee printout
before,printing.,,(P)rint al
l prints all,addresses with
out stopping.,,(E)nd ends pr
ogram.,,END !084
1720 ! -- close all !140
1730 CLOSE #1 :: IF FD$="H"
THEN PRINT #2:PE2$ :: CLOSE
#2 :: GOTO 490 ELSE CLOSE #2
 :: GOTO 490 !168
```

234COL

Extended BASIC program takes guesswork out of multiple columns

By FRED LAYTON

234COL is a program to help you print out D/V80 files in multiple columns. The program was adapted from Jim Peterson's popular "Printall" program. His sorting and column printing routines are used in this adaptation.

My idea was to revise the original PRINTALL with just one menu and a minimum of options. Its best use is putting long DOCS in three columns of condensed print. Column width is limited to 40 characters.

The program can also be used to print out an Extended BASIC program, in four columns, which has been LISTed to disk then saved with 28 character width. A short program also included with this article will convert your listing.

Printer parameters may be changed in line 190. Lines 290-320 preset your left margin (LM) and space between columns (CS).

Your last page does not do a formfeed. But you, of course, can print out multiple pages. Ordinarily you will use the defaults on the screen. Your range of column characters is between 28 and 40.

CAUTION: Be sure you reformat to the number of columns you specify under "Width of Column" (ie, 40). See line 430.

The program is small, only 12 sectors, but has a prescan so that it RUNs fast. If you select two columns, there is an Elite print option with or without double-strike.

An easy procedure to follow is to LOAD your file into TI-Writer. If it is a straight ASCII file with no printer code and no dot commands (".") and it is greater than 40 columns with carriage returns (c/r) then reformat each paragraph to 40 columns. Or try Bruce Harrison's "Reformat" program listed on page 25 of the June 1994 MICROpendium.

HINT: While in TI-Writer, do a FCTN-0 at the end of your file and take note of the total number of lines. If you plan to print out three columns, divide this number by three to give the "Lines per page?" to enter in the program (60 is default).

Be sure to print your file (PF) to disk while using TI-Writer and preface the "DSKn..." with a "C" to strip out any c/r's and that last line of gibberish which TI-W prints at the end of your file.

28COLIST

28COLIST will convert an XB program you have LISTed to disk to 28 columns, just like you see it on the screen when you have the program LOADed and do a LIST. You then have the option to directly print out the listing or SAVE it to disk in D/V80 format for future use. There is also an option to set left margin to zero or a margin of 6 to center your listing in a 40-column file for a newsletter.

If you have any comments, contact the author directly in Oakland, Calif., at 510-530-8335. He is a member of the San Francisco 99ers.

234COL

```
100 !234COL !250
```

(See Page 18)

234COL—

(Continued from Page 17)

```
110 DIM M$(400)!199
120 GOTO 170 !249
130 T,K,A,B,CS,CW,F,IP,J,KY,
LM,LP,LT,NC,P,RM,SET,SL,ST !
060
140 D$,DS$,EL$,EV$,F$,LT$,M$
(),P$,S$,T$,Y$ !129
150 CALL KEY :: CALL COLOR :
: CALL SCREEN :: CALL SOUND
!222
160 !@P- !064
170 CALL KEY(3,KY,ST):: ON W
ARNING NEXT !196
180 FOR SET=0 TO 14 :: CALL
COLOR(SET,2,8):: NEXT SET ::
 CALL SCREEN(5)!081
190 DISPLAY AT(2,2)ERASE ALL
:"<< 234COL40/ELITE/COND.>>"
:RPT$("~",28):" DON'T FORGET
-40 COLS MAX!":RPT$("~",28):
: P$="PIO.LF" !181
200 DISPLAY AT(7,5):"Number
of columns: 3": :"      Lin
es per page? 60" :: DISPLAY
AT(11,2):"Last page even col
umns? Y" !066
210 DISPLAY AT(13,7):"Width
of column: 40" !208
220 ACCEPT AT(7,24)SIZE(-1)V
ALIDATE("234"):NC :: EL$="N"
 !023
230 ACCEPT AT(9,24)SIZE(-2)V
ALIDATE(DIGIT):LP :: IF LP>6
4 THEN 230 !149
240 ACCEPT AT(11,26)SIZE(-1)
VALIDATE("YN"):EV$ :: IF EV$
="" THEN EV$="Y" :: GOTO 200
 !166
250 ACCEPT AT(13,24)SIZE(-2)
VALIDATE(DIGIT):CW :: IF CW>
40 OR CW<28 THEN 250 !018
260 IF NC<>4 THEN 270 :: IF
CW>28 THEN CW=28 :: DISPLAY
AT(15,1):"Corrected to <28>
col limit" :: DISPLAY AT(13,
24):"28" :: GOTO 310 !159
270 IF NC<>2 THEN 300 :: DIS
PLAY AT(15,5):"Elite print?
(Y/N) Y" :: ACCEPT AT(15,24)
VALIDATE("YN")SIZE(-1):EL$ !
085
280 IF EL$="N" THEN 290 :: D
ISPLAY AT(15,3):"Double-stri
ke? (Y/N) N" :: ACCEPT AT(15
,24)VALIDATE("YN")SIZE(-1):D
S$ !158
290 CS=6 :: LM=8 :: GOTO 320
 !063
300 CS=4 :: LM=3 :: GOTO 320
 !056
310 CS=4 :: LM=8 !043
320 S$=RPT$(" ",CS):: LT$=RP
T$(" ",LM)!058
330 DISPLAY AT(20,8):"Printe
r on? N" :: ACCEPT AT(20,20)
BEEP VALIDATE("YN")SIZE(-1):
D$ :: IF D$<>"Y" THEN DISPLA
Y AT(15,1):"" :: GOTO 220 !0
65
340 IF EL$="Y" THEN 350 :: O
PEN #1:P$,VARIABLE 132 :: PR
INT #1:CHR$(27);"@";CHR$(27)
;"2";CHR$(15);:: GOTO 360 !0
33
350 OPEN #1:P$,VARIABLE 96 :
: PRINT #1:CHR$(27);"@";CHR$
(27);"2";CHR$(27);"M";:: IF
DS$="Y" THEN PRINT #1:CHR$(2
7);"G";!158
360 DISPLAY AT(17,1):RPT$("-
",28):" FILEname to be print
ed.": :"  Filename: DSK" ::
ACCEPT AT(20,16)SIZE(-12)BEE
P:F$ :: F$="DSK"&F$ !156
370 B=0 :: T=1 :: K=LP*NC !0
02
380 ON ERROR 390 :: OPEN #2:
F$,INPUT :: DISPLAY AT(21,4)
:"Reading: ";F$ :: ON ERROR
STOP :: GOTO 400 !222
390 CALL SOUND(1000,110,0,-4
,0):: DISPLAY AT(21,1):"Chec
k file->"&F$ :: CLOSE #1 ::
GOSUB 600 :: GOTO 190 !169
400 FOR IP=1 TO K :: LINPUT
#2:M$(IP):: DISPLAY AT(23,12
):T :: IF LEN(M$(IP))=0 THEN
 450 !116
410 IF POS(M$(IP),CHR$(13),1
)<>0 THEN M$(IP)=SEG$(M$(IP)
,1,LEN(M$(IP))-1)!000
420 IF LEN(M$(IP))<=CW THEN
440 !134
430 DISPLAY AT(12,1)ERASE AL
L:"TOO many chars wide in fi
le":"columns. Must be";CW;"o
r less!" :: CLOSE #1 :: GOSU
B 600 :: GOTO 190 !172
440 IF ASC(M$(IP))=128 THEN
M$(IP)=RPT$(" ",CW)!208
450 M$(IP)=M$(IP)&RPT$(" ",C
W-LEN(M$(IP)))!098
460 GOSUB 580 :: IF B=1 THEN
 490 :: T=T+1 !157
470 NEXT IP !047
480 IP=IP-1 !172
490 IF EV$="Y" AND IP<K THEN
 LP=INT(IP/NC)+1 !111
500 FOR J=1 TO LP :: ON NC-1
 GOSUB 530,540,550 :: NEXT J
 !073
510 IF B=1 THEN 570 ELSE IF
EOF(2)THEN CLOSE #2 :: GOTO
570 !191
520 GOSUB 560 :: PRINT #1:CH
R$(12):: GOTO 400 !030
530 PRINT #1:LT$&M$(J)&S$&M$
(J+LP)&CHR$(10):: RETURN !13
6
540 PRINT #1:LT$&M$(J)&S$&M$
(J+LP)&S$&M$(J+LP*2)&CHR$(10
):: RETURN !178
550 PRINT #1:LT$&M$(J)&S$&M$
(J+LP)&S$&M$(J+LP*2)&S$&M$(J
+LP*3)&CHR$(10):: RETURN !22
1
560 FOR J=1 TO K :: M$(J)=""
 :: NEXT J :: RETURN !037
570 DISPLAY AT(23,4):"Try ag
ain? (Y/N) N" :: ACCEPT AT(2
3,21)VALIDATE("YN")SIZE(-1):
Y$ :: IF Y$="N" THEN STOP EL
SE CLOSE #1 :: GOTO 190 !184
580 IF EOF(2)<>1 THEN RETURN
 ELSE CLOSE #2 :: B=1 :: RET
URN !006
590 CLOSE #2 :: B=1 :: RETUR
N !030
600 DISPLAY AT(24,1)BEEP:"Pr
ess any key to continue" !19
7
610 CALL KEY(3,KY,ST):: IF S
T=0 THEN 610 ELSE DISPLAY AT
(24,1):"" :: RETURN !143
620 !END !090
```

28COLIST

```
5 ! 28COLIST !219
10 DISPLAY AT(4,1)ERASE ALL:
"1st LIST your program to
disk then RUN 28COLIST."
:"   ex:LIST""DSK1.LIST80"""
```

(See Page 19)

234COL—

(Continued from Page 18)

```
 !024
12 DISPLAY AT(8,4):"Left mar
gin (0 or 6)? 0":RPT$("-",28
):: ACCEPT AT(8,26)SIZE(-1)V
ALIDATE("06"):LM !055
15 DISPLAY AT(11,5):"INPUT F
ILEname?":"    ex:DSKn.LIST8
0": :"    DSK1": :"    OUTPU
T FILEname?":"    ex:PIO or
DSKn.LIST28": :"    DSK" !24
3
17 ACCEPT AT(14,8)SIZE(-12):
F$ :: IF LEN(F$)<3 THEN 17 :
: ACCEPT AT(19,5)SIZE(-15):P
$ :: F$="DSK"&F$ !148
20 OPEN #1:F$,INPUT :: OPEN
#3:P$,OUTPUT :: ON ERROR 50
!164
25 C=28 :: LINPUT #1:A$ :: I
F LEN(A$)<80 THEN 40 !177
30 LINPUT #1:B$ :: IF VAL(SE
G$(A$,1,POS(A$," ",2)))<VAL(
SEG$(B$,1,POS(B$," ",2)))THE
N F=1 :: GOTO 40 !180
35 A$=A$&B$ :: IF LEN(B$)>=8
0 THEN 30 !155
40 A=LEN(A$):: L=A/C+.99 ::
FOR I=0 TO L :: PRINT #3:TAB
(LM);SEG$(A$,1+I*C,C):: NEXT
 I :: IF EOF(1)*(F=0)THEN 55
 !005
45 IF F=1 THEN F=0 :: A$=""
:: GOTO 35 ELSE 25 !242
50 ON ERROR 50 :: RETURN 35
!049
55 CLOSE #1 :: CLOSE #3 :: D
ISPLAY AT(23,4):"Do another?
 (Y/N) N" :: ACCEPT AT(23,22
)SIZE(-1)VALIDATE("YN"):Y$ :
: IF Y$="Y" THEN 5 !237
```


Western offers hardware hackers kit

Western Horizon Technologies has released a hardware hackers prototyping kit, which Don O'Neil of the company says provides all necessary components to make a general purpose interface between a user's "home brewed" circuits and the TI 99/4A.

O'Neil says the kit includes all parts necessary to build a fully buffered interface, CRU and memory decode compatible with the Geneve and schematics of that and other applications.

Also provided with the kit is a set of XB links designed to provide control of CRU bits on projects.

Each chip in the kit comes with its appropriate two or three level wire wrap circuit.

The kit includes one 74LS125; one 74LS138; three 74LS244; one 74LS245; one 74LS520; one custom WHT Decode GAL; one eight-position DIP switch; one 1 78MC205 1.5 amp 5V regulator and heat sink; one 60-pin TI Leading Edge connector; one 47 ohm resistor; one 3904 transistor; one Red LED; one 4.5-inch x 8.5-inch perfboard; two 30-pin ZIP WW Strips (for Leading Edge connector); one set of main circuit schematics with P-Box pinout; three sets of optional circuits schematics and parts requirements; one diskette with utility programs; three mounting bolts and nuts for the Leading Edge connector and regulator; and a spool of wire wrap wire.

The kit sells for $39.95 with no warranty and no technical support available.

For further information or to order, contact Western Horizon Technologies, 10225 Jean Ellen Dr., Gilroy, CA 95020,

1994 TI FAIRS

MAY

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

OCTOBER

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

NOVEMBER

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

1995 TI FAIRS

FEBRUARY

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

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

Graphics

Pix Pro and graphics conversions

This article appeared in the newsletter of the Ozark 99ers User Group, Springfield, Missouri.—Ed.

By TOM BEERSMAN

Let's say you have a picture you have drawn or gotten off a BBS. You happen to show it to a friend. He asks if he can have a copy of it. Unfortunately, it was drawn in TI-Artist or TI-Artist+ and the only art program he has is GRAPHX. How can he view it?

Don't fret — Pix Pro comes to the rescue. With Pix Pro, which was written by Jim Resiss, you can load pictures saved in a variety of popular formats, and it allows you to convert to another format. Pix Pro has eight load formats and six save formats. In essence, it does the job of 40 individual conversion programs.

This is what the Save Menu looks like:

1. Artist (TI-Artist picture format)
2. GRAPHX (GRAPHX format)
3. Instance (TI-Artist instance format)
4. Page Pro (Page Pro format)
5. Picasso (Picasso format)
6. Pix (Pix format)
7. Pix 128 (?)

If you load in a picture file larger than the screen (Picasso or Page Pro), a number of keys become active that let you view a portion of the picture. When the picture is larger than the screen, the screen acts as a sort of "window" on the picture. Certain keys, which will be familiar to users of TI-Writer, are used for "moving" this window to different parts of the picture.

FCTN-5 to move the window to the right. When you are all the way over to the far right-hand side of the picture, FCTN-5 will take you to the left-hand side. In otherwords, it will wrap-around.

Press FCTN-3 to move the window to the left. When the window is displaying the left side of the picture, pressing this key again will move the window to the right-hand side.

Press FCTN-4 to move the window down. Like the key above, when you are at the bottom, pressing FCTN-4 again will take you to the top of the picture.

Press FCTN-6 to move the window towards the top of the picture. Pressing this key when the window is displaying the top of the picture will move the window to the bottom of it.

In addition to the ability to shift the win-

(See Page 21)

REVIEW

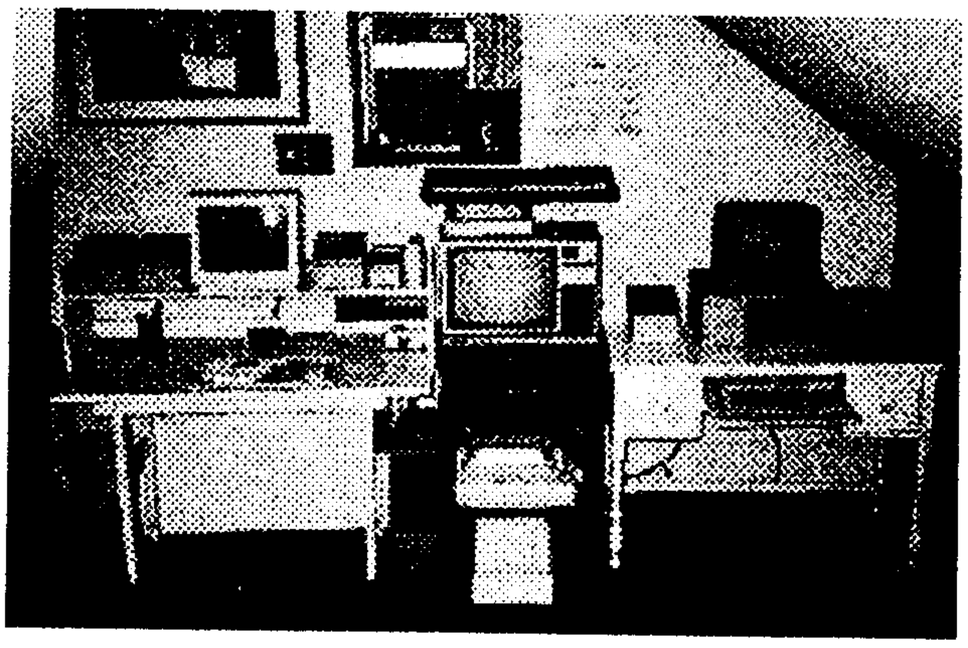
Distributor: Asgard Software, Harry Brashear, 27753 Main St., Newfain, NY 14108; 703-491-1267.

SYSTEM OF THE MONTH

Dual system great for programming

This month's System of the Month is owned by Tony Knerr, of Downingtown, Pennsylvania.

The main system (left in the photo) is a TI99/4A with a TI RS232, CorComp RAMdisk, Horizon RAMdisk, CorComp Triple Tech, P-GRAM Plus, Myarc floppy disk controller and Myarc Hard and Floppy Disk Controller. The floppy controller controls two 5.25-inch, 360K floppy drives and the HFDC controls a 5.25-inch 720K floppy, 3.5-inch 1.44-megabyte floppy and two 20-megabtye hard drives. The hard drives are installed in the Peripheral Expansion Box with an external power supply. The four floppy drives are above and right of the keyboard. Peripheral items include a 1200 baud Atari modem, Casio MT540 Synthesizer for use with Midi-Master, Rave 99 101-key keyboard, Magnavox 14-inch color monitor and Epson Apex 80 dot-matrix printer. The TI console is hidden behind the backboard of the desk, along with the power supplies for the drives. The switch box on the right front table leg near the floor-standing PEB turns it all on and off.



The second system (rightin the photo) is also a TI that includes a TI 32K card, TI RS232, Rave speech card, P-GRAM Plus, CorComp disk controller and a TI monitor. There are two 5.25-inch, 360K floppy drives and a Myarc Personality Card with 20-megabyte hard drive. The hard drive can format only 15 megabytes.

Both computers are connected via RS232 cable and are able to share the printer through a switchbox.

"This setup is excellent for programming, as I can run a time-consuming utility or assembly on one while working on the other," Knerr writes. "Some of my work includes P-GRAM Utilities v2, a completely rewritten disk manager for the personality card that allows hard drive formatting up to 15 megabytes, many Extended BASIC music programs, as well as Midi-Master music files. I also host meetings of the Chester County TI Users Group and am a member of the Philadelphia Area TI Users Group."

Unusually, while most programmers work in assembly or Extended BASIC, Knerr says he is available to write programs for anyone with a GRAM device who needs a special purpose GPL program. His CompuServe ID is 72070,573.

PIX PRO—

(Continued from Page 20)

dow on a picture in large jumps, there are a number of keys that give you finer control over the portion of the picture displayed. Here they are:

FCTN-E moves the window up eight pixels.

FCTN-X moves the window down eight pixels.

FCTN-S moves the window left eight pixels.

FCTN-D moves the window right eight pixels.

Being able to move the window over a specific part of the picture is very important. Some of the save formats available are limited in size to what you can display on the screen — GRAPHX and TI-Artist picture formats, for example. When you go to save the picture, the area that you are displaying in the window will be the part that is saved, if you opt for one of those formats.

The windowing functions also complement the clipping function. You will often have to move the window over to the part of the picture containing what you want to clip before clipping, since the clipping function is confined to what you see on the screen.

Being able to move the window over a specific part of the picture is very important.

When you are finished with this program, press FCTN= to quit back to the master title screen, or the MDOS command line.

Okay, let's say you have many picture/instance files of your own and others you have collected over the years, but you don't have enough disk space. Pix Pro offers a way to keep your pictures and save some space, too. To do so, load in the picture file then pick option No. 6 from the menu. You can either leave the filename the same or rename it if you like. One feature I like is if you forget to save part of a picture as an instance or run out of time, no sweat. Once you have loaded your picture file in Pix Pro, it will display the picture on the screen. Press C (to clip). At this point a little arrow should appear on the screen. Use the arrow keys to move it to the upper left-hand corner of the part of the picture you wish to clip and press Enter. The arrow will change into a box. Use the arrow keys to surround the area you wish to save and press Enter again. It will then display only what was in the box. Next press Enter and pick the save format you want from the menu. Or, press FCTN-9 to abort.

Note that you cannot abort the clip function after you select it, surround and hit Enter.

Asgard Memory System

A new day for the TI99/4A

By BRUCE HARRISON

This review was written prior to Asgard Software quitting the TI business. The review is published for those who may purchase one on the second hand market or in the event that a third-party supports it.—Ed.

Let's start with your author's admission of guilt. Readers of my regular column will readily accuse me of being strongly biased against the addition of various hardware gadgets to the TI-99/4A. Guilty as charged! The folks at Asgard may be worried about such a biased author reviewing this product, but they need not worry. This product is good!

Lately, we've become all-too accustomed to having new and great-sounding products announced, touted widely, and then dropped without ever getting to the stage of "production," let alone being placed in our P-Boxes. Not so the Asgard Memory System, which has had a remarkably short period elapse between our first knowledge and the reality of a product. Thanks for this must go to the eagerness of Jim Krych, the backing of Chris Bobbitt and Harry Brashear, and the talents of programming geniuses Joe Delekto and Art Green. Thanks to these folks, there's a real chance that TI users will find a light at the end of the ten-year tunnel.

REVIEW

REPORT CARD
Performance ..... **A**
Ease of Use ..... **A**
Documentation ..... **A**
Value* ..... **?**
Final Grade ..... **A**

Cost: $119.95
Manufacturer: Asgard Software, 1423 Flagship Dr. Woodbridge, VA 22192 (703) 491-1267
Requirements: TI-99/4A Console, P-Box, Monitor, DSDD disk drive.
Readers should know that the AMS is currently not being produced.—Ed.

WHAT IS THIS THING?

To start with, it's not just a memory, but a system that can make the TI into a world-class all-purpose computing machine. Physically, it's a half-height card that goes into the P-Box in place of the 32K card. The only compatibility problem that we know about is that it can't co-exist with either the Cor-Comp or Myarc "RAMdisk" systems, since both of those must supplant the 32K memory, and two things can't occupy the same memory address space at the same time. More modern RAMdisks, such as the Horizon 3000 and 4000 series, can coexist peacefully with the new system, as we proved to our own satisfaction by placing the card in our own P-Box. (Our Horizon 3000 does not have RAMBO, so we can't be certain of compatibil-

(See Page 22)

ASGARD MEMORY SYSTEM—

(Continued from Pag 21)

ity with that feature, but we don't think that would cause any problem.) The System part is a set of software, which includes a new menu driver that allows the use of the larger memory for assembly programs. Programs designed and developed for the system will be able to perform truly wonderful feats. For those of us who like to make our own programs, there are the Macro Assembler and Linker that come with the system, which were reviewed last month. Thus we can "home-brew" programs well over the 32K limit that we've had to live with before. The base size for the memory is 128K, which allows the "system software" and programs of 70K or so to be resident at the same time.

QUICK TESTS

Our first little test, after placing this card in the P-Box, supplanting our TI 32K card, was to simply load up and run our little CALIB test. This test checks the speed at which the machine is running in a manner which reflects the memory-access time as well as the CPU speed. For a "normal" TI with the TI 32K card in place, this program puts the number 199 on the screen twice, once just to show the "norm," and the second reflecting actual measured time for the current configuration. For example, on a Geneve this second number may be 48, indicating a much faster operating speed. Our concern here, mainly, was that the speed with this new card be not too different from "normal," so that timing-sensitive operations would behave as expected. The AMS card gave us the hoped-for result, with 199 being displayed twice on the screen. This means that programs you ran on your TI before will behave exactly as you'd expect.

IN THE SYSTEM

Next, we used the program ABOOT (included in the package) to get ourselves into the "system." We were curious about whether the normal type of Option-5 program file would work the same in the AMS environment. We tried selection A (Load and Run) from the AMS menu, and tried DSK1.FW (Funnelweb). As far as we could tell, Funnelweb worked perfectly in all respects. It also worked with the card in place operating outside the system. Then we remembered that this was an old version of F'web, not the latest edition. The old version happened to be closer at hand.

We pulled out our working copy of Funnelweb V4.4, and tried that with AMS's system. No go! Funnelweb itself would load up, but none of the normal Funnelweb programs (Text Edit, for example) would load. Why? One of the differences between the old version and the new one is the use of our boot-track process in the newer one. This might be the problem, so we tried our own word processor under the AMS system, and this went bonkers! We know exactly why this is so. Art Green's DSRLNK routine, unlike just about any other DSRLNK, does not leave the data about the CRU address and device name pointer in CPU RAM Pad. (Most DSRLNK routines leave the CRU address in >83D0 and a pointer which can be used to find the device name at >83D2.)

Thus, the use of Art's DSRLNK in the system has deprived us of information we thought could be depended upon, and caused some otherwise good software not to work. In the docs for Art Green's Linker, he mentions that his DSRLNK does not leave anything behind in RAM Pad, as if this were an advantage. It's not, if the software we're loading depends on finding out what disk it was loaded from. Tony McGovern's Funnelweb is robust enough to gracefully report errors when trying to load its program files, and didn't crash the system like our own word processor did, but still we wish Art Green would make his DSRLNK leave a trail like the others do. Some of us use that information, Art! What possible advantage is there in not providing that data?

We called Joe Delekto, and he confirmed that the system software uses Art Green's DSRLNK, as we suspected. We strongly recommend that this be fixed, not just for us, but for others who may be designing new software for use with AMS. Of course either Funnelweb or our own WP can be run using the card without the AMS system software, and everything will work correctly.

During our rather lengthy conversation with Joe, he pointed out a few facts we hadn't yet discovered. Very skilled assembly programmers can use routines included on one of the disks that are provided to "home brew" software that takes advantage of the card's memory without using the AMS system software. One author reportedly has come up with an assembly routine for use with Extended BASIC and the AMS card. Using this routine, one can load more than one XB program into the card's pages, then run the "resident" XB programs selectively. Making this kind of use of the memory will require some study, but basically the job consists of writing through a CRU process into the registers that control paging of the memory, so that different parts of the large memory can be "mapped" into the normal high memory addresses.

HERITAGE

One rather interesting doc supplied with the system gives the background of how this new product relates to the TI-99/8. The technology base and architecture of this new memory card are rooted in the TI 99/8, which of course was never put into production. In effect, what Asgard has done fulfills the promise that was to have been the 99/8. It will take some time for this new product to get established, so early purchasers may need patience to wait for the programmers to take full advantage of the new capability. There's no guarantee that this wait will be worth it, but at least there's a real product being offered, not "vaporware", and that product is firmly rooted in TI's original concepts, not in some half-baked idea.

THE DOCS

As we said in our review of the Assembler and Linker that are included with this package, the docs are voluminous, but clearly written and worth reading. Those who are not familiar with assembly programming will want to skip over a lot of the material, but for people who plan to write software in really big amounts, a careful reading of the docs is a must. The Linker docs are particularly important for the programmer, as the rules for making full use of the memory are somewhat complex. Once the programmer understands these rules, the Linker will make construction of large programs easier, but lots of studying will be required to get "up to

(See Page 22)

ASGARD MEMORY SYSTEM—

(Continued from Page 22)

speed" with the system.

EASE OF USE

This topic can be looked at in several ways, so let's begin with the situation for the ordinary non-programmer user. For normal everyday use of the programs in your library, nothing changes. The card will behave as if it's just an ordinary 32K memory expansion, and programs will run just the same as always.

As programs designed for the expanded memory become available, things will get a bit more complicated. To run programs that are designed within the system environment, you'll need to load up ABOOT from E/A Option 5 first, then run the programs from that. Some programmers will choose to work outside the system and still make use of the extra memory, and these programs will most likely be runnable from E/A without the ABOOT system software. In either case, the use of programs will not be significantly more difficult.

For those of us doing assembly programming, things will be different, as there's a new set of rules overlaid on the usual problems of writing and debugging programs. For that matter, just debugging presents a whole new range of things that can go wrong. Maybe the next product developed for this system should be an advanced debugger. So far as we know, there's no existing tool that would allow probing the depths of an 80K program on the TI.

The learning curve for programmers will be steeper for those of us who have done assembly programming on the PC, where we can make our "modules" in any length from two bytes to 64K bytes, and can assign segment boundaries in any manner we want. Thinking in 4K increments will take some getting used to.

THE GRIPES

Right from the start, there was the fact that the distributed disk containing all the system software and docs was DSDD. Many of us in the community still use the TI disk controller, and can't use DSDD disks. We recommend that the distribution disks be provided as DSSD. This point is made a bit more galling by the fact that the docs state that the distribution is DSSD. Maybe that one extra disk would push the cost of shipping up by a quarter or so, but that's small compared to the irritation that DSDD can cause.

When ABOOT loads up, it reads a "script" D/V80 file to make its menu on the screen. There are several items listed on this menu that are not included in the package. We were able to edit the script file so that only included items showed up on the menu, but we think that editing should have been done before the disks were distributed.

There is a "stay resident" capability provided by the system software, so that programs can be loaded and kept just a keystroke away. That worked as long as we were still in the system, but using the re-entry program ASHOE rendered the resident programs unavailable. Delekto told us that this should work, and thought that perhaps we had the wrong version of ASHOE. All distributed copies should have the correct versions of the programs.

The card, as we said at the beginning, is half-height. This makes it more difficult to install in the P-Box, especially for those with aging fingers, like yours truly. Once it's in there, it worked correctly, but the particular model we received has no activity light. Sometimes that light is the only clue we have that something is happening, particularly on some XB programs. We are told that the light has been added to all new production units, so this gripe should go away.

SUMMARY

This is a very good product, with the potential to open a whole new "world" to the TI-99/4A user. The card we tested was the AMS with 128K memory. This model can also be ordered with 512K memory, or can be upgraded from 128 to 512. The AMS-2, as we understand it, will be able to go to 1 megabyte capacity, and the AEMS into the hinterlands up to 16 megabytes, all within the original framework as intended for the TI-99/8. That's an incredible capability.

As our report card indicates, the product gets straight A's, but we've made the Value grade a question mark. The true value of this product will only be realized by having programs to take advantage of the memory capacity now offered. For the sake of all our readers, we hope that one day you'll be running mega-programs on your TI-99/4As. This package is not super-cheap, but it's far less expensive than a PC or Mac, and if the software becomes available, this may be the most valuable product ever to hit the TI market.

BUGS & BYTES

Exec generates 'steam'

Exec 2.0 by Tim Tesch has been posted on some boards. The Delphi posting generated a response by Barry Boone, author of the original Exec.

Though admitting he probably wouldn't have minded had he been asked regarding the enhancements of his Geneve utility, Boone says in his post that he is "a bit steamed" about not being contacted. However, Boone and Tesch have made it up, according to later messages. Tesch says he never got a reply to an Internet message, and Boone surmises that the Internet lost the post, as it does sometimes.

The new version reportedly takes care of the keyscan routine when running option 5 out of MDOS. The joystick reversal of other versions is fixed. The new version of Exec also reportedly allows users to load programs from the 6000 cartridge area without having TI-mode on and has an added option for automatically loading ROMpage so the user can run MDM5 from MDOS.

One reported bug in V.2 is a reversal of verify. Unless the user types "VERIFY ON" from the command line, the user will not be notified if the disk he is writing to has a bad sector.

MICRO-REVIEWS

PC99 is close to perfect but works best with fast PC

By CHARLES GOOD

There is nothing very "micro" about this month's column. I am devoting the entire review to a single piece of very important software. If you want your important software reviewed, send it to me at P.O. Box 647, Venedocia OH 45894. My evening phone is 419-667-3131 and my internet address is now cgood@lima.ohio-state.edu.

Let's be honest with ourselves. The TI community no longer exists in its own isolated little world. Many TI computer users also use other types of computers. In particular, many TI users also have a DOS (IBM-compatible) computer at home. In addition to myself, TI users who own a home DOS computer include Berry Traver, Tony McGovern, Bruce Harrison, Bill Gaskill and the vast majority of TI user group officers attending the most recent Lima MUG Conference.

Let us also recognize the fact that many previous TI users have sold their TIs and replaced them with DOS computers. Well, it is now possible to have your cake and eat it too. At least two software products exist which allow you to run 99/4A software on DOS computers. I am reviewing one this month and hope to review the other in the near future.

These TI emulators represent a whole new category of software only dreamed of a few years ago. A DOS computer can now be made to behave exactly like a 99/4A. Why would one want to do this? The answer is that some TI software is really good or offers unique features, such as the Funnelweb word processor whose multilingual capabilities are unknown to me in any DOS word processor. Like policemen, you can't always find an expanded TI system when and where you need one. These days DOS systems are much more common than 99/4A systems.

I feel that emulators will enhance, rather than decrease, interest in the 99/4A. My father-in-law, for example, learned computing on a 99/4A but left the TI community 7 years ago when he got his first 8088 DOS computer. He now has a 486 something or other on which he occasionally plays his favorite TI games and uses label printing software written in TI Extended BASIC. Due to an emulator, my father in law has returned this year to the TI community.

PC99 by CaDD Electronics

This commercial 99/4A emulator is technically very close to perfect. Minimum recommended requirements are a 386 computer with VGA graphics, 640K memory, and a hard drive. You can also special order a version that will run on a 286 DOS computer with the above configuration. Almost all features of a 99/4A system are emulated and all 99/4A software and modules, apparently without exception, will run correctly. The emulated system includes three DSSD "drives," PIO, RS232, joystick(s), and 1 channel sound through the PC speaker. If you don't have joysticks on your DOS computer, joystick movement can be simulated from the keyboard. Speech is not emulated. Applications programmed for speech or more than one sound channel run normally but without speaking or enhanced sound. I am reviewing PC99 release 2A. Full 3 channel TI sound emulation with a DOS sound card is being worked on for a future release.

The PC99 package includes the emulator itself, the Extended BASIC, Editor/Assembler and Tombstone City modules, 99/4A <—> PC transfer software, and an amazingly complete set of utilities. You can purchase at a modest extra cost DOS files to emulate any module TI ever made for the 99/4A, as well as files to make PC99 emulate a 99/4 (without the "A"). TI is paid a royalty on each console operating system and module sold.

Software allows you to transfer whole TI disks to PC99 format either using a TI system directly cabled to the DOS machine, or indirectly without cabling by using PC Transfer. PC Transfer (not included when you purchase PC99) is software that runs on a TI system with a double-sided disk controller and allows you to move TI files from one TI drive to a 360K DOS disk you put in a second TI drive. You then take the DOS disk from the TI, put it in your DOS computer, and convert the files on this disk to PC99 format. I have done so successfully and find the procedure lengthy and confusing, requiring lots of user intervention. Using a cable to link the two computer systems makes the procedure much easier.

Transferring a DSSD disk between cabled computers takes just a few minutes and requires almost no user intervention once the transfer starts. In addition to whole TI disks, you can also transfer GRAM files to PC99 to run as emulated modules. To do this you need a GRAM device to make GRAM files of your TI module collection. If you don't have a GRAM device, such as GRAM Kracker, or a particular module, module files runnable from PC99 can be purchased from CaDD. File transfers can go in both directions. Any TI software created on PC99 can be sent over to a real TI system, either cabled or not cabled.

PC99 emulates whole TI disks, not individual TI disk files. You have 3 "drives" on line when running the PC99, each with either a SSSD or DSSD TI disk represented by a single DOS file. The large size of these DOS files that emulate TI disks makes it difficult to fit PC99 onto one 3.5-inch disk and run it directly from the disk, but this can be done if you include only one module on the disk and leave out the docs and PC99's configuration utility. Normally you would install PC99 onto a hard disk. OLD, SAVE, and other disk operations from within PC99 just modify these TI "disk" PC files. Because these files exactly emulate TI disks, emulated TI software correctly reads TI "disk" directories. A large assortment of DOS utilities are provided to manipulate the emulated TI disks. You can get a TI type directory

(See Page 25)

MICRO-REVIEWS—

(Continued from Page 24)

from DOS, extract modify and reinsert single TI files to and from the emulated disks, etc. The method PC99 uses to emulate TI disks works very well.

All 40-column 99/4A software and transferred modules I have tried work perfectly running under PC99 release 2A. I know of no exceptions, except for the lack of speech and full TI sound. Because only 16K of VDP is emulated you can't run 80-column or Geneve specific software from PC99.

There is, unfortunately, one potentially very big problem with PC99's software emulation. Execution speed of TI software running from PC99 is extremely slow. I guess this the price to be paid for "perfect" emulation. On my 386DX/40 TI software running under PC99 seems to just crawl along. When running the Funnelweb v5.01 editor, maximum typing speed is about 60 characters (not words) per minute. As with a real 99/4A there is no keyboard buffer, so you can't type faster than the speed your letters appear on screen. Using a TI word processor running from PC99 just isn't practical on my DOS machine, at least not yet. I have been given a beta version of release 2B to play with and it is perhaps 20 percent faster than 2A because of speeded up CPU operations. This speed increase still isn't enough to allow me to use Funnelweb's word processor on my 386.

The speed of PC99 is in part determined by the DOS computer's central processor, and I am told that on a 486DX2/66 PC99 release 2A will drive the Funnelweb word processor at acceptable speed. I can't personally verify this. If you have a fast 486 or Pentium DOS computer, then PC99 release 2A's speed may not now be a problem. From what I have seen, future releases of PC99 will certainly be faster than 2A.

The only advantage of PC99's slow speed is with games. Because I can react quickly and the game can't, I get fantastic scores. I have no trouble leaving the Tombstone City town and killing all the bad guys with PC99. On a real TI I always get zapped soon after I try to leave town.

PC99 has an excellent assembly language memory debugger. Any kind of 99/4A memory manipulation is possible. Because the PC99 debugger doesn't occupy any part of the memory reserved for the TI, the debugger can do tricks that are not possible with any debugger operating from a real 99/4A. An even more enhanced debugger screen display is in the works for a future PC99 release.

An appropriate feature of any professional software product for which you pay a professional price is a comprehensive (on disk) manual backed up by technical support. You get this support with PC99, either by phone or U.S. mail. Registered owners can phone (not a toll free number) CaDD evenings and weekends and speak to one of the PC99 authors. If the phone line is not attended you can leave a message on the answering machine and your call will be returned. You might need this sort of help the first time you try transferring your TI software to PC99, particularly if you are using computers that aren't cabled together. CaDD also offers to convert TI software to PC99 format for you if cabling a DOS and 99/4A computer is not practical. You send them your TI disks and you get your software back in the mail in PC99 format. There is a nominal charge for conversion, starting at $1 for a single disk. The more disks you send the cheaper per disk it gets.

The most important question that should be asked by those considering purchasing PC99 is, "Will 99/4A software emulation be too slow on my particular DOS computer?"

The most important question that should be asked by those considering purchasing PC99 is, "Will 99/4A software emulation be too slow on MY particular DOS computer?" You need to be able to test drive PC99 on your computer to answer this question, and hopefully you will soon be able to do so for free. At my suggestion CaDD is asking TI for permission to distribute a free crippled version of PC99 for evaluation purposes on a "you send CaDD a high density 3.5 inch disk and a postage paid return mailer" basis. They would then return your disk with a full speed but limited feature "cripple ware" version of PC99 and some 99/4A software in PC99 format for you to speed check on your machine. By the time you read this review this trial version may be available. Write CaDD for details. PC99 release 2A costs.$147 to new purchasers. If you have already purchased an earlier release, the cost to upgrade is the difference between what you originally paid and the current price. It costs $7 to upgrade from release 2 to release 2A. CaDD's address is at 81 Prescott Rd., Raymond NH 03077.

ADDITIONAL COMMENTS ABOUT EMULATORS

Get a TI-DOS serial cable! Without the ability to transfer your own important TI software to a DOS computer 99/4A emulators are little more than expensive toys that allow you to play around with the few pieces of TI software that come with the emulator. File transfers via cable are easy. You can't just run out to Wal-Mart and buy a serial cable. You have to make one or have somebody make one for you. That's because the TI RS232 port is wired a bit differently than everybody else's RS232, and there are two different sized connectors for COM ports on DOS computers. The PC99 documentation gives pin in/out data for the needed cable. I had a cable made to my specification (specified cable length and DOS COM port) and tested on an emulator by L.L. Conner Enterprise, 1521 Ferry St., Lafayette IN 47904. You can phone voice at 317-742-8146 for a price quote.

The question of distribution of copyrighted TI products needs to be discussed. The PC99 people have a license from TI to sell the code of the 99/4A operating system and all official TI 99/4A modules. TI is paid a royalty on such sales. Such a license is probably not difficult to get these days, since O.P.A. (Gary Bowser) also has licensed the 99/4A operating system. As of this writing (mid July) the other TI emulator some of you have heard about does not have a license from TI to distribute code contained within 99/4A consoles or cartridges. TI complained about such distribution, and the other emulator has been temporarily withdrawn from the marketplace.

As I understand things, software and computer code patent and copyright protection boils down to this — software owners can make for themselves or pay

(See Page 26)

MICRO-REVIEWS—

(Continued from Page 25)

someone else to make as many backup copies of their software as they want. But each legally owned piece of software or computer operating system code can only be run on one machine at a time. If you own two 99/4A consoles (even a broken console) then you have the right to run two copies of the console operating system simultaneously, and one of these can be on a DOS computer. If you own an Extended BASIC module, then you have the right to run XB on a DOS computer. For members of the TI community, making a DOS computer behave like a 99/4A and run 99/4A software is probably not a violation of TI's patent and copyright protection.

Videotape explores Internet

A videotape whose subject is explaining use of the Internet step by step has been released. The Internet: Your Lession in Navigating the Information Super Highway; a How-To Guide to Mining Its Treasures includes a brief history, a demonstration in the use of modem and connectivity, a tutorial on UNIX commands, E Mail, Gopher, Archie, Telenet and FTP. The video also discusses the future of Internet.

The 70-minute videotape is available for $179 (including public performance rights) or $79 (home viewing rights only) from the Educational Reform Group, 76 Glenview, Wilton, CT 06897, (203) 834-0144.

USER NOTES

Understanding Extended BASIC error messages

The following was written by Earl Raguse and has appeared in several user group newsletters.

When I program, I constantly get error messages. I suppose this is because I am constantly making errors. There is no stigma attached to making errors while programming. Afterall, to err is human. The real problem is deciphering what is meant by the often cryptic messages, and what to do about them.

The Extended BASIC manual has four pages of error messages (pages 212-215) and sometimes even more cryptic explanations of them. There is a whole set of file errors not discussed here. I am going to discuss only the errors I run into frequently.

Among the most frequent are those beginning with "BAD." They include ARGUMENT, LINE NUMBER, SUBSCRIPT and VALUE.

For ARGUMENT, the usual error is an unacceptable value in a SOUND statement. Frequency may be less than 110, attenuation may not be >30, and duration may not be zero or greater or less than +/-4250, and the noise specification must be from -1 to -8.

The CALL COLOR and SCREEN subprograms insist on values within the specified ranges of 1-16.

BAD LINE NUMBER is an indication of a line number called from within a program that the interpreter could not find.

A BAD SUBSCRIPT is invariably a zero value. However, if you somehow get a value in excess of 32767, you would also get this message. Make sure that the variable you are using for a subscript is not zero, which it will be if you have not in some way placed a value in it. One of the most frequent causes of zero values in my programs is an inadvertent skipping over of a set of lines because of some IF THEN logic, or an ill-advised GOTO statement.

BAD VALUE just means that you have supplied an illegal value to a function. This can be because of many things — you just have to review the statements leading to the error line. Again, zero is an unacceptable value for many functions.

COMMAND ILLEGAL IN A PROGRAM is fairly simple. The only cure is to

(See Page 27)

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

USER NOTES

(Continued from Page 26)

comment out that statement. Reserved words, such as SAVE, LIST, MERGE, OLD and NEW fall into this category.

FOR NEXT NESTING simply means that the interpreter could not find a NEXT for each FOR during program execution. Now, if you look at the listing and see no error, i.e. you have a FOR for each NEXT and vice versa, then look for a GOTO or IF-THEN-ELSE statement that causes the program to jump into or out of the middle of a FOR-NEXT loop. That way the program does not see the FOR or NEXT. The former, however, will give a NEXT WITHOUT FOR message.

IMPROPERLY USED NAME is most often caused by the improper, or not at all, dimensioning of a subscripted variable. There are many other easons listed in the XB manual, but the above is the most likely.

ONLY LEGAL IN A PROGRAM is the opposite of COMMAND ILLEGAL IN A PROGRAM. This applies to GOTO, GOSUB, INPUT, ON RETURN, SUB, SUBEXIT, SUBED, DEF, etc. They can be used only within a program. There is no sidetracking this issue.

RETURN WITHOUT GOSUB is a command error message. It is caused most frequently by a faulty GOTO or IF-THEN-ELSE taking the program into the middle of a subroutine. There are difficult to find, and I find the best way is to verify every GOTO and IF-THEN-ELSE statement until I find the one that goes where it shouldn't. I know of no shortcuts for this, except to be careful in the first place. This error most frequently happens when one moves or copies a statement and forgets to change line number references.

STRING-NUMBER MISMATCH means you are trying to assign a number to a string variable, or a string to a number variable. This is usually easily checked. The solution usually is to use VAL or CHR$ to convert the item to suit the variable that you want to assign to it.

SUBPROGRAM NOT FOUND is a message I frequently get, because I call one of my favorite subprograms and then forget to merge it into the program. The solution is easy — just merge the subprogram in. The other possibility is that you have a spelling error. Check on that.

Troubleshooting is an art, at least I find it so. I get better at it, but still sometimes I forget how I found the error the last time and have to fiddle for hours trying to resolve it.

Lining up decimals

The following item was written by Col Christensen and appeared in Bug-Bytes, the newsletter of the TI Brisbane Users Group in Australia.

I didn't get a chance to try these yet. Want to print a column of values correctly aligned to the decimal point? The key in this little program is the variable PLACE, which indicates the position of the decimal point in a number.

```
100 FOR I=1 TO 10 :: READ A(I):: NEXT I
110 FOR I=1 TO 10 :: PLACE=POS(STR$(A(I)),".",I):: IF PLACE=0 THEN PLACE=LEN(STR$(A(I)))+1
120 PRINT TAB(15-PLACE);A(I)
130 NEXT I
140 DATA 101.57,250,3.12,35789,250.35
```

Like to type in your program using a screen color different from cyan? Type this in the command mode:

(See Page 28)

USER NOTES

(Continued from Page 27)

```
FOR I=0 TO 14 :: CALL COLOR(
I,16,9):: NEXT I :: CALL SCR
EEN(9):: ACCEPT AT(1,1):A$
```

When you have pressed Enter and the cursor flashes at the top of the screen, press FCTN-4. Try some other color if red is not your favorite. Be careful as you type your program, as any error detected by the interpreter will reset the colors to black on cyan. This is an incentive to type precisely.

Castnines teaches different way to add numbers quickly

The following program was written by the late Jim Peterson. It's an educational program that teaches how to quickly add numbers without actually adding up the totals. All documentation is on-screen.

CASTNINES

```
2 DIM S(24)!122
10 GOTO 100 !179
11 SET,T,X,Q,X$,J,SUM,DIF,SU
M$,A,B,K,ST,C,D,TP,R@,XX,M$,
DIF1,DIF2,ZZ,DIFZ,DIF3,DIFX,
TT,F,N !027
30 CALL CLEAR :: CALL COLOR
:: CALL SCREEN :: CALL CHAR
:: CALL VCHAR :: CALL KEY ::
 CALL SOUND !154
40 !@P- !064
100 CALL CLEAR !209
110 ! CASTING OUT NINES prog
rammed by Jim Peterson 11/83
, XBasic version 7/85 !021
120 ! COPYRIGHT 1983 Tigercu
b Software, 156 Collingwood
Ave., Columbus Ohio 43213 !1
02
130 ! REPRODUCTION PROHIBITE
D. DELETION OF COPYRIGHT NOT
ICE PROHIBITED. !149
140 FOR SET=2 TO 13 !016
150 CALL COLOR(SET,2,16)!212
160 NEXT SET :: CALL SCREEN(
5):: CALL CHAR(128,"00"):: C
ALL CHAR(94,"001800FF0018"):
: CALL CHAR(92,"3C4299A1A199
423C"):: CALL VCHAR(1,31,32,
96)!046
170 CALL COLOR(1,2,11):: CAL
L CHAR(32,"003844443C040830"
)!076
180 DISPLAY AT(10,5):"ÄCASTI
NGÄOUTÄNINESÄ": : :"TCX-1119
"&CHR$(128)&"\"&CHR$(128)&"T
igercubÄSoftware": : : : : :
 : : :"PressÄanyÄkey" :: G
OSUB 910 :: CALL COLOR(1,2,1
6)!055
190 CALL SCREEN(16):: CALL C
HAR(32,"00"):: CALL CHAR(128
,"00000000FF"):: GOSUB 200 :
: GOTO 220 !253
200 DISPLAY AT(1,13):"729848
":TAB(10);"X  908375":TAB(12
);"ÄÄÄÄÄÄ":TAB(11);729848*5:
TAB(10);729848*7:TAB(9);7298
48*3 !097
210 DISPLAY AT(7,8):729848*8
:TAB(9);"000000":TAB(6);7298
48*9:TAB(6);"ÄÄÄÄÄÄÄÄÄÄÄÄÄÄ";T
AB(7);"662975677000" :: RETU
RN !116
220 DISPLAY AT(14,1):"IS THI
S CORRECT?": : :"YOU ARE ABO
UT TO LEARN AN": :"EASY WAY
TO CHECK!": : :"Press any ke
y" :: GOSUB 910 !123
230 DISPLAY AT(4,1):"
It is called": :"    CASTIN
G OUT NINES": :" - but that
sounds too ": :"      old-f
ashioned.": :!245
240 DISPLAY AT(12,1):" Let's
 call it Throwing out": :"9'
s and Matching What's Left":
 :"Over." :: DISPLAY AT(18,1
):"  Here's how we do it - "
: : :"Press any key" :: GOSU
B 910 !247
250 GOSUB 270 !095
260 GOTO 290 !114
270 DISPLAY AT(2,1):"First,
here's how to throw":"the 9'
s out of a number." !033280
DISPLAY AT(6,1):"1. Add toge
ther the digits":"   in the
number": :"2. Divide the tot
al by 9": :"3. See how much
is left over" :: RETURN !007
290 DISPLAY AT(20,8):"Press
any key" :: GOSUB 910 !169
300 !DISPLAY AT(3,1):"EXAMPL
ES:": : !218
310 DISPLAY AT(3,1):"EXAMPLE
S -": :"2031  2+0+3+1 = 6":
:"6 ^ 9 = 0 with 6 left over
": :"4563  4+5+6+3 = 18": :"
18 ^ 9 = 2 with 0 left over"
 !150
320 DISPLAY AT(13,1):"2562
2+5+6+2 = 15": :"15 ^ 9 = 1
with 6 left over" !186
330 DISPLAY AT(17,1):" We do
n't care how many":"times 9
goes into the num-":"ber, we
 just want to know":"how muc
h is left over.": :"Press an
y key" :: GOSUB 910 !147
340 DISPLAY AT(12,1)ERASE AL
L:"  Now you try it!" :: RAN
DOMIZE !230
350 T=T+1 :: IF T=8 THEN 460
 !128
360 X=INT(RND*8000+2000):: D
ISPLAY AT(14,12):X: :" How m
uch is left over?": :" " ::
ACCEPT AT(16,25)VALIDATE(DIG
IT)SIZE(1):Q :: X$=STR$(X)::
 FOR J=1 TO LEN(X$):: SUM=SU
M+VAL(SEG$(X$,J,1))!191
370 NEXT J :: DIF=SUM-INT(SU
M/9)*9 :: IF Q<>DIF THEN 390
 !015
380 DISPLAY AT(18,7):"That's
 right!" :: DISPLAY AT(14,1)
:" ": :" " :: SUM=0 :: GOSUB
 930 :: GOTO 350 !025
390 DISPLAY AT(4,4)ERASE ALL
:"No, that's not right" :: G
OSUB 280 :: DISPLAY AT(13,12
):X :: SUM$=SEG$(X$,1,1):: A
=INT(800*RND+100):: B=INT(80
0*RND+100):: SUM=VAL(SUM$)::
 FOR J=2 TO LEN(X$):: SUM$=S
EG$(X$,J,1)!157
400 !SUM=SUM+VAL(SUM$) !034
410 DISPLAY AT(13+J,1):SUM;"
+";VAL(SUM$);"=";!SUM !047
420 SUM=SUM+VAL(SUM$):: DISP
LAY AT(13+J,12):SUM !013
430 NEXT J :: DISPLAY AT(20,
1):SUM;"^";9;"=";INT(SUM/9);
"with";SUM-INT(SUM/9)*9;"lef
t over": :"Try again - press
 any key" !089
440 CALL KEY(0,K,ST):: IF ST
```

(See Page 29)

USER NOTES

(Continued from Page 28)

```
=0 THEN 440 !165
450 CALL CLEAR :: SUM=0 :: G
OTO 360 !057
460 DISPLAY AT(1,1)ERASE ALL
:" OK, You've got it.":"Now,
 here's how we use it to":"c
heck multiplication." !226
470 DISPLAY AT(4,1):" Throw
out the 9's from the":"multi
plicand - that's the":"numbe
r on top, the one that" !121
480 DISPLAY AT(7,1):"is gett
ing itself multiplied":" - a
nd see what's left over.":"
Throw out the 9's from the"
!214
490 DISPLAY AT(10,1):"multip
lier - that's the one":"that
's doing the multiplying":"
- and see what's left over."
 !091
500 DISPLAY AT(13,1):" Multi
ply those two left-":"overs
together, throw out":"the 9'
s and see what's left." !197
510 DISPLAY AT(16,1):" Now t
hrow out the 9's in":"the pr
oduct - that's the":"answer
to the problem - and":"see w
hat's left over." !057
520 DISPLAY AT(20,1):" If th
e two leftovers match,":"the
 answer is correct!": :"Pres
s any key" !059
530 CALL KEY(0,K,ST):: IF ST
=0 THEN 530 !000
540 CALL CLEAR :: DISPLAY AT
(1,1):"EXAMPLE:":TAB(10);371
6:TAB(10);2438:TAB(10);"ÄÄÄÄ
":TAB(9);29728:TAB(8);11148:
TAB(7);14864:TAB(7);7432:TAB
(7);"ÄÄÄ" !059
550 DISPLAY AT(10,7):9059608
 :: DISPLAY AT(12,1):"3+7+1+
6=17  17^9=1 and 8":"left ov
er.":"2+4+3+8=17  17^9=1 and
 8":"left over." !109
560 DISPLAY AT(16,1):"8x8=64
   64^9=7 and 1 left":"over.
" !177
570 DISPLAY AT(18,1):"9+0+5+
9+6+0+8=37":"37^9=4 and 1 le
ft over": :"1 matches 1 - th
e answer is correct!":"
   Press any key" !176
580 CALL KEY(0,K,ST):: IF ST
=0 THEN 580 !050
590 CALL CLEAR :: DISPLAY AT
(10,1)ERASE ALL:"Now you try
 it!" !222
600 A=INT(8000*RND+1000):: B
=INT(8000*RND+1000):: C=A*B
:: X=INT(2*RND):: IF X=1 THE
N 620 !032
610 D=C+INT(50*RND+5):: GOTO
 630 !019
620 D=C !069
630 DISPLAY AT(12,1):A;"X";B
;"=";D: :"Is that correct?":
 :"Answer Y(es) or N(o)" !23
```

(See Page 30)

USER NOTES

(Continued from Page 29)

```
9
640 CALL KEY(3,K,ST):: IF ST
<1 THEN 640 !115
650 IF (K<>78)*(K<>89)THEN 6
40 !036
660 IF ((X=1)*(K=89))+((X=0)
*(K=78))THEN 870 !224
670 DISPLAY AT(1,1)ERASE ALL
:"no, that's not right" :: D
ISPLAY AT(3,5):STR$(A)&"x"&S
TR$(B)&"="&STR$(D):: TP=A ::
 R@=5 :: GOSUB 680 :: GOTO 7
10 !229
680 FOR J=1 TO LEN(STR$(TP))
:: XX=VAL(SEG$(STR$(TP),J,1)
):: M$=M$&STR$(XX)&"+" :: SU
M=SUM+XX !151
690 NEXT J :: M$=SEG$(M$,1,L
EN(M$)-1)&"="&STR$(SUM):: DI
SPLAY AT(R@,1):M$ :: DIF=SUM
-INT(SUM/9)*9 :: DISPLAY AT(
R@+2,1):STR$(SUM)&"^9="&STR$
(INT(SUM/9))&" with "&STR$(D
IF)&" left over" !115
700 SUM=0 :: M$="" :: RETURN
 !046
710 DIF1=DIF :: TP=B :: R@=9
 :: GOSUB 680 :: DIF2=DIF ::
 ZZ=DIF1*DIF2 :: DIFZ=ZZ-INT
(ZZ/9)*9 !065
720 DISPLAY AT(13,1):STR$(DI
F1)&"x"&STR$(DIF2)&"="&STR$(
ZZ): :STR$(ZZ)&"^9="&STR$(IN
T(ZZ/9))&" with "&STR$(DIFZ)
&" left over " :: TP=D :: R@
=17 :: GOSUB 680 !098
730 DIF3=DIF :: SUM=0 :: DIF
X=DIF3-INT(DIF3/9)*9 :: IF D
IFX<>DIFZ THEN 750 !011
740 DISPLAY AT(21,1):STR$(DI
FX)&" matches "&STR$(DIFZ):"
 Answer is correct" :: GOTO
760 !210
750 DISPLAY AT(21,1):STR$(DI
FX)&" does not match "&STR$(
DIFZ):"Answer is not correct
" !198
760 DISPLAY AT(24,1):"
Press any key" :: GOSUB 910
:: GOTO 600 !151
770 CALL CLEAR :: DISPLAY AT
(2,1):"Now here are a couple
 of":"shortcuts.": :"You can
 throw out the 9's":"before
you add up the":"digits." !2
53
780 DISPLAY AT(9,1):"3295":
:"3+2+9+5=19^9=2 with 1":"le
ft over." !176
790 DISPLAY AT(14,1):"3295":
 :"throw out the 9": :"32 5
  3+2+5=10^9=1 with 1":"left
 over.": :"The leftover is t
he same!" !106
800 DISPLAY AT(23,1):"Press
any key" :: CALL KEY(0,K,ST)
:: IF ST<1 THEN 800 !213810
DISPLAY AT(4,1)ERASE ALL:"Yo
u can also throw out com-":"
binations that add up to 9":
 :" 54827": :"5+4+8+2+7=26^9
=2" !172
820 DISPLAY AT(11,1):"with 8
 left over.": :"54827": :"5+
4=9 so throw it out.": :"2+7
=9 so throw it out.": :"That
 still leaves 8!" :: DISPLAY
 AT(23,1):"Press any key" ::
 GOSUB 910 !128
830 DISPLAY AT(7,1):"Here's
another -": :" If the leftov
er from the":"multiplicand i
s 0, you don't" !121
840 DISPLAY AT(11,1):"have t
o figure the multi-":"plier,
 because 0 times any-":"thin
g is 0." !168
850 DISPLAY AT(15,1):"A mill
ion times zero is": :TAB(12)
;"ZILCH": :TAB(13);"!!": : :
 : :"Press any key" :: GOSUB
 910 :: GOTO 950 !196
860 GOTO 860 !174
870 DISPLAY AT(20,1):"That's
 right!": : :: GOSUB 930 !20
4
880 TT=TT+1 !201
890 IF TT=6 THEN 770 !103
900 GOTO 600 !169
910 CALL KEY(0,K,ST):: IF ST
<1 THEN 910 !127
920 CALL CLEAR :: RETURN !21
9
930 F=INT(600*RND+110):: FOR
 N=1 TO 12 :: CALL SOUND(-99
9,F*1.059463094^N,0)!234940
NEXT N :: CALL SOUND(-1,4000
0,0):: RETURN !172
950 DISPLAY AT(1,1):" Now, h
ere's how to find the":"erro
r in the problem." :: GOSUB
960 :: GOTO 970 !052
960 DISPLAY AT(3,1):TAB(10);
3462:TAB(10);1792:TAB(10);"Ä
ÄÄÄ":TAB(10);6924:TAB(8);311
58:TAB(7);24244:TAB(7);3462:
TAB(7);"ÄÄÄÄÄÄÄÄ":TAB(7);6204
904 :: RETURN !036
970 DISPLAY AT(14,1):"3462 3
+6=9, throw it out -":"4+2=6
 leftover":"1792, throw out
the 9, 7+2=":"9, throw it ou
t, leftover 1":"and 6 x 1 =
6" !113
980 DISPLAY AT(19,1):"620490
4, throw out the 9,":"6+2+4+
4=16^9=1 with 7 left":"over.
  7 doesn't match 6," :: DIS
PLAY AT(22,1):"something is
wrong!": :"So - Press any ke
y" !054
990 CALL KEY(0,K,ST):: IF ST
<1 THEN 990 !207
1000 !GOSUB 1850 !241
1010 DISPLAY AT(14,1):"6924
is the product of the":"2 (i
n 1792) x 3462.":"The leftov
er of 3462 was 6," !219
1020 DISPLAY AT(17,1):"so 2x
6=12 and 12^9=1 with 3":"lef
t over. The leftover of":"69
24 should also be 3 - ":"692
4, throw out the 9," !220
1030 DISPLAY AT(21,1):"6+2+4
=12, 12^9=1 with 3 left":"ov
er. 3 matches 3, so that":"l
ine is correct.":"Press any
key" !178
1040 CALL KEY(0,K,ST):: IF S
T<1 THEN 1040 !002
1050 !GOSUB 1850 !241
1060 DISPLAY AT(14,1):"31158
 is the product of 9":"(in 1
792) x 3462, which had":"a l
eftover of 6. 9x6=54 and":"5
4^9=6 with 0 left over." !03
5
1070 DISPLAY AT(18,1):"31158
, throw out 8+1=9,":"throw o
ut 3+1+5=9, also":"leaves 0,
 so that line is":"OK.": : :
"Press any key" !193
1080 CALL KEY(0,K,ST):: IF S
```

(See Paeg 31)

USER NOTES

(Continued from Page 30)

```
T<1 THEN 1080 !042
1090 !GOSUB 1850 !241
1100 DISPLAY AT(14,1):"7 in
1792 times the leftover":"of
 6 from 3462 equals 42.":"42
^9=4 with 6 left over. ":"24
244, 2+4+2+4+4=16 and" !076
1110 DISPLAY AT(18,1):"16^9=
1 with 7 left over.":"7 does
n't match 6, so the":"error
is in this line.": : : :"Pre
ss any key" :: GOSUB 910 !25
5
1120 DISPLAY AT(10,1)ERASE A
LL:" And finally, here's how
 to":"check a long division
prob-":"lem." :: DISPLAY AT(
23,1):"Press any key" :: GOS
UB 910 :: CALL CHAR(129,"000
0000F08080808")!161
1130 CALL CHAR(130,"08080808
08080808"):: DISPLAY AT(1,8)
:"969 969 - quotient":TAB(4)
 "ÄÄÄÄÄÄÄÄ":"728Ç705776 70577
 - dividend" !093
1140 DISPLAY AT(4,5):"6552":
TAB(5);"ÄÄÄÄ   728 - divisor
":TAB(6);"5057":TAB(6);"4368
  344 - remainder" !006
1150 DISPLAY AT(8,6):"ÄÄÄÄ":
TAB(7);"6896":TAB(7);"6552":
TAB(7);"ÄÄÄÄ":TAB(8);"344" !
240
1160 DISPLAY AT(14,1):"Subtr
act remainder from div-":"id
end": :"705776-344=705432":
:"Throw out 9's": :"7+5+4+3+
2=21": :"21^9=2, with 3 left
 over" !054
1170 DISPLAY AT(24,1):"Press
 any key" :: CALL KEY(0,K,ST
):: IF ST<1 THEN 1170 !073
1180 DISPLAY AT(14,1):"Throw
 out 9's of quotient": :"9+6
+9=24": :"24^9=2, 6 left ove
r": :"Throw out 9's of divis
or": :"7+2+8=17 17^9=1, 8 le
ft over": :!024
1190 DISPLAY AT(24,1):"Press
 any key" :: CALL KEY(0,K,ST
):: IF ST<1 THEN 1190 !093
1200 DISPLAY AT(14,1):"Multi
ply leftover of":"quotient b
y leftover of":"divisor": :"
6x8=48": :"48^9=5, 3 left ov
er, matches" !044
1210 DISPLAY AT(21,1):"divid
end leftover. Correct.": : :
!171
1220 DISPLAY AT(24,1):"Press
 any key" :: CALL KEY(0,K,ST
):: IF ST<1 THEN 1220 !123
1230 CALL CLEAR :: GOSUB 200
 :: DISPLAY AT(20,1):"SO NOW
, IS THIS CORRECT?" :: DISPL
AY AT(23,12):"END" !166
1240 GOTO 1240 !043
```

CLASSIFIEDS

FOR SALE

FOR SALE

TI HexBus CC-40 computers ($40); HexBus Printer/Plotters ($40); HexBux RS232 ($20); all 3 for $75. All in mint condition. CC-40 software also available. Call Tom at 703-455-9276.

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