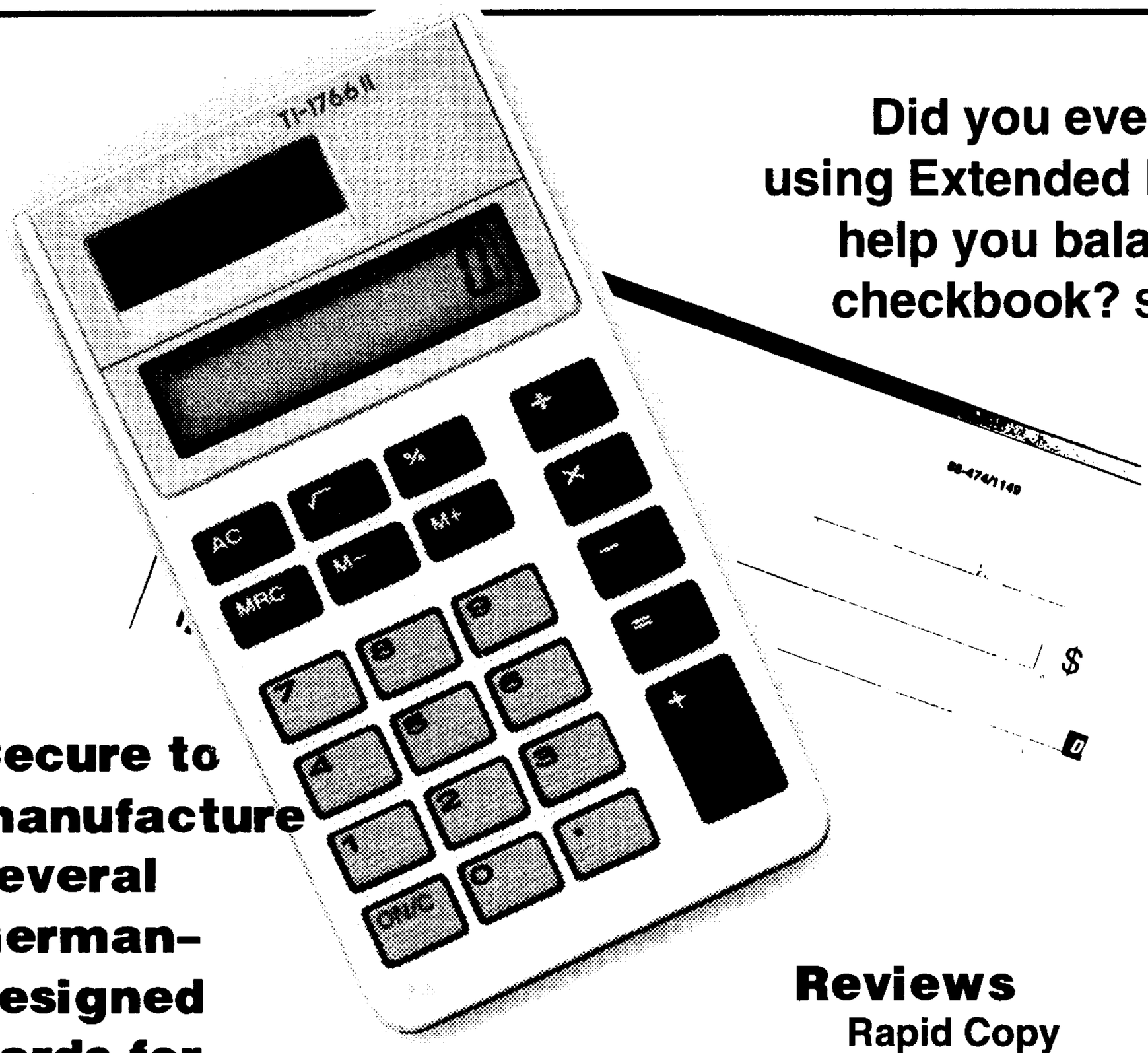
Covering the TI99/4A and the Myarc 9640

MICROpendium

Volume 14 Number 2 March/April 1997 $6



Did you ever think of using Extended BASIC to help you balance your checkbook? See Page 22

Tlers gather in the U.S., United Kingdom and Germany

Cecure to manufacture several German-designed cards for U.S. distribution

Reviews

Rapid Copy

JP Drawing

Addatex Game Software

Programs

CONTENTS

MICROpendium

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

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

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

All editions of MICROpendium are mailed from the Round Rock (Texas) Post Office.
Mailing address: P.O. Box 1343, Round Rock, TX 78680.
Telephone & FAX: (512) 255-1512
Delphi TI NET: MICROpendium
Internet E-mail: jkoloen@io.com
Home page: http://www.io.com/~jkoloen/
John Koloen .....Publisher
Larua Burns .....Editor

*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.

COMPETITION COMPUTER 800-471-1600

(Nationwide & Canada ORDERS ONLY)

350 MARCELLA WAY MILLBRAE, CALIFORNIA 94030

COMPETITION COMPUTERS HAS HARDWARE!!!

PRINTERS FOR YOUR TI:
DOT MATRIX,TRACTOR FEED
TI TEXT AND GRAPHICS COMPATIBLE
PARALLEL INTERFACE,WITH 2 RIBBONS $35+SHIPPING

SAME AS ABOVE BUT WITH COLOR PRINTING $70
SAME AS ABOVE BUT WITH SERIAL INTERFACE $70

PRINTER FOR TI IEEE CARD GPIB INTERFACE $70

SHIELDED TI PIO CABLES $20
UNSHIELDED TI PIO CABLES $10

FREE PRINTERS(YOU ONLY PAY SHIPPING) WE HAVE
SEVERAL QUME DAISYWHEEL SERIAL INTERFACE PRINTERS
WIDE CARRIAGE WORKHORSES.EXTRA RIBBONS $1 EACH

HARDDRIVES FOR YOUR TI:
OLD MFM DRIVES FOR YOUR MYARC HFDC NO WARRENTY $20

NEED A NEW DRIVE? BETTER BITE THE BULLET AND BUY
A WHT SCSI CARD AND DRIVE.

COMPLETE SCSI HARD DRIVE SYSTEMS START AT $230
AND INCLUDE EXTERNAL DRIVE, CASE, AND WHT SCSI CARD

43MB SEAGATE 5.25HH SCSI HARDDRIVES $40
OTHER SCSI DRIVES AVAILABLE CALL FOR LATEST LIST
ALL OUR SCSI DRIVES COME TI FORMATTED

NEED A HARDDRIVE EXTERNAL CASE AND POWER SUPPLY $30

MOUNT YOUR 3.5"HD IN A 5.25 BAY WITH OUR KIT ONLY$5

WERE YOU GIFTED A HARD DRIVE BUT CAN'T FIGURE OUT
WHAT TO DO WITH IT, SEND US A COMPLETE DESCRIPTION
INCLUDING MANUFACTURER, MODEL # AND SIZE(HALFHEIGHT
OR FULL HEIGHT AND WE WILL IDENTIFY IT FOR YOU AND
SUPPLY COMPLETE CONFIGURATION AND CABLING INFO FOR $5

FLOPPY DRIVES:
DUAL 5.25 DSDD DISK DRIVE KIT $55
DUAL 3.5 DSDD DISK DRIVE KIT $55(NEW LOW PRICE)

IDENTIFICATION SERVICE FOR THAT FLOPPY DISK DRIVE YOU
FOUND: SEND MANUFACTURER,MODEL NUMBER AND SIZE HH/FH
ALONG WITH LARGE SASE ENVELOPE AND $2

NEW PRODUCT FOR YOUR YOUR MYARC GENEVE 9640!!

HAS YOUR GENEVE BEEN MOUSELESS OR HAVE YOU BEEN A
SLAVE TO SERIAL MOUSE DRIVERS?
DESPAIR NO MORE! COMPETITION COMPUTERS GENEVE MOUSE
IS HERE! A DIRECT REPLACEMENT FOR YOUR GENEVE MOUSE
IT NEEDS NO SPECIAL ADAPTORS OR SOFTWARE. ONLY $25

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

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

TI-99/4A™ CD ROM

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

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

CARTRIDGE SALE! $19.95 EACH!
ANTEATER CHICKEN COOP
CONSOLE WRITER,DISK FIXER
FROG STICKERS HEN PECKED
KING OF THE CASTLE
PRINCESS AND THE FROG
ROMOX DEMO ROTOR RAIDERS
SPEED READING A(TEENS+ADULTS)
SPEED READING B(FOR CHILDREN)
SUPER DUPER TOPPER
TYPO-MAN TYPO II
AMBULANCE DRIVING DEMON
HENHOUSE RABBIT TRAIL
SCHNOZ-OLA ST. NICK
VIDEO VEGAS TYPEWRITER 99
TRIS TRIS 2
PAINT'N PRINT A(GP100,CP700)
PAINT'N PRINT B(GP550,OKIDA)
PAINT'N PRINT C(IBM,GEMINI)
HOMEWORK HELPER+
DATABASE SORT DATABASE ENTRY

QUALITY 99 SOFTWARE TITTLES:
ANY TITTLE FOR ONLY $9.95

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

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

COMMENTS

It's TI fair season

The fall is when the Chicago Faire takes place, but lately it seems that spring is fair season in the TI world, with the roving Fest West and the Ohio Multi User Group conference coming up in this country, and events in the United Kingdom and Germany on the way in Europe.

We are expecting to hear about new stuff introduced at some of these fairs. Who would have thought it about a computer which hasn't been manufactured in so long?

But no hard feelings toward Texas Instruments. There are some movements going forward toward holding a TI fair in Lubbock, Texas, home of the 4/A, andwhere TI still maintains a presence. Even the powers that be at TI have expressed some interest in this event. Stay tuned.

CARDS FROM GERMANY

Of course, one of the wonders of this little orphan is that even "between" fairs you hear about new stuff coming out. Elsewhere in this issue, you can read about the new cards from Germany which will be distributed in the United States by Cecure Electronics of Wisconsin. How many other orphan systems can you upgrade the way you can ours?

—JK

BUGS AND BYTES

There's a bug in Multiplan

The following article was taken from the Spirit of 99 newsletter, the newsletter of the Central Ohio Ninety-Niners Inc.. We are unsure who wrote it.

Just as I had used TI-Writer or Funnelweb to do my assignment writing with no problems, I thought that it would be appropriate to use Multiplan to do all of the calculations to the assignments.

I set one spreadsheet up with the answers to the first assignment and then duplicated it for each additional assignment, since each assignment had different numbers.

When I went to mark the assignment, I found that there were some mistakes in the final answers. I double checked the students' work through my calculator and found that Multiplan had made some mistakes. At first I thought that it was because I had made a mistake entering a formula. I checked it. It wasn't the formula. It was Multiplan.

There is a bug in Multiplan. You may not have noticed it before. I have used Microsoft Multiplan for quite a while and had not come across this bug.

Here is a printout of the file to show where the bug is:

```
1.OE-O1      3.2E-O1      1.OE-O1
2.OE-O1      4.5E-O1      2.0E-O1
3.OE-O1      5.5E-01      3.OE-O1
9.OE-02      3.0E+00      9.0E+OO error
1.OE-02      1.OE+OO      1.OE+OO error
2.0E-02      1.4E+00      2.0E+OO error
1.0E-03      3.2E-02      1.0E-03
1.OE-04      1.OE-O1      1.OE-02 error
1.OE-05      3.2E-03      1.OE-05
1.OE-06      1.OE-02      1.OE-04 error
1.OE-07      3.2E-04      1.OE-07
1.OE-08      1.OE-03      1.OE-06 error
1.0E-O9      3.2E-05      1.0E-O9
1.OE-10      1.OE-04      1.OE-08 error
1.OE-11      3.2E-06      1.OE-11
1.OE-12      1.OE-05      1.OE-10 error
1.OE-13      3.2E-07      1.OE-13
```

As you can see, the SQRT (square root) function messes up when it does the square root of even negative powers of X. If X=0.01, it gives 1 instead of 0.1 as an answer. It even messes up negative powers of numbers. There is no problem with the odd negative powers, only the even ones.

Now, you probably would not use such small numbers in your spreadsheets, so you may not have come across this bug before. As far as I can determine, it is a flaw in the program GROMs that operate on either the TI or Geneve version. Both give the same mistakes.

This was not the only problem that I had. I also had problems with the SQRT function when I was using really tiny numbers, like 10 to the -13 power. So, maybe — well, not maybe — there is a problem with the SQRT algorithm. Which makes you wonder if there are other problems, too. No one ever said that Microsoft made perfect programs. Just look at Windows and Windows 95.

I also tried using Multiplan on my Apple IIGS. It worked just fine.

I also tried the original Multiplan program for the TI. It also had the same error.

FEEDBACK

.RAM-Karte note

A small note on the GRAM-Karte from Michael Becker: It is a well-done piece of hardware and software, but there is no chance to speed up the TI multiplied by two with a 16-bit RAM, nor can you have eight times the speed with his GRAM device.

His card has the option to read the information found in the GRAM simulation without the waitstates generated by the trusty old TI. So remember, most time is spent in the GPL-Interpreter itself.

Christian Kunze
Hamburg, Germany

HFDC advice: change cables first

I had problems with my Hard and Floppy Disk Controller and replaced a huge amount of parts till I found out in the end that the hard drive cables seemed to be defective. I don't know which one, the wide or the narrow one; I replaced both of them. and now after a long journey of trial and error everything works perfectly again.

My advice to other users: change cables first if problems occur. I read this advice before from another user, but shame to me, I ignored it, or better, I couldn't believe that the solution is that simple. I would have saved some money and a lot of time and sleepless nights. So: change the cables! They are cheap and easy to change and if it doesn't help you didn't waste too much.

Kurt Rasowisch
Vienna, Austria

Farewell from PUG

As you know by now, the PUG is no more. A valiant life of over 16 years (1981-1996) was marked by the resilience and determination of the club members. We would not let the TI "die," and valiantly worked to keep the Pittsburgh Users Group alive and well.

Most of the years the PUG was in existence, the leadership was in the capable hands of Gary Taylor. From the "Night of the Black Death," when this elected librarian found himself to be the only officer left standing, until the club's demise in May of 1996, Gary was either the leader or right behind the leader in making certain all possible stones had been turned to get the most for the most member.

The final meeting of the PUG saw a somewhat comic and very touching scene. No one wanted "the end" to come, and we turned the event into a Chinese Auction and pizza party. The pizza was — well, as my son would say, there is no such thing as bad pizza. The Chinese Auction was — well, read on.

At a Chinese Auction, everyone purchases (or in this case, was handed) a set number of tickets. Then, all the items to be

(See Page 6)

FEEDBACK

(Continued from Page 5)

auctioned are displayed, and each person decides how many tickets they want to put up for any particular item. There were a few things that members wanted, but the most often heard comment was "donation" when a winning number was drawn. You see, Mickey Cendrowski had located a local organization that was thrilled to have TIs and I would have to guess she left the meeting with over half of the items donated to the kids. It was really thrilling to know that these would not be collecting dust in some closet or basement, but would live on and light up some young lives once again.

So, this letter is being composed on my TI, which stands ready to work for me any time. And this letter stands as one reason the TI was, is and always will be a great computer.

Sure — the new computers are fancier, more compact and faster. *But* I would never try to get a quarter back out of the cool air slots such as I did when a teacher of computer at St. Norbert School. My laptop stopped working. Taking it apart and fixing it as I do my TI was unthinkable! And where else could —

A child learn about programming by writing programs within a few days? When I taught at St. Norbert School, third graders could write programs with color, sound, words. In a local competition, a first grader won a first place ribbon for her program.

An adult be able to stop a program, list it and change the program as desired? The new computers have technical support to fix problems, but not to help you customize your program.

Anyone be helped into the computer age with ease using such wonderful sha ware as FunnelWeb, 1000 Words (written by a PUG member), Midi Interface, Telco?

The list of fun programs is endless as well. From TI Sings to action and adventure games — the TI has been and will continue to be fun, entertaining and educational.

I thank all of you. And I hope that faraway friends I never met such as Earl Raguse, Ken Gilliland and others know how much their efforts have meant to all of us.

Stay in touch, if only on line.

Susan J Harper
Clarksburg, West Virginia

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

Cecure to manufacture German-designed cards in US

Cecure Electronics Inc. has completed negotiations with the System 99 User Group (SNUG) of Mannheim, Germany, to manufacture and sell a variety of hardware products in the U.S.

The first three products are cards for use in the TI Peripheral Expansion Box. They are:

- High Speed GPL-II (HSGPL-II)
- Second Generation CPU (SGCPU)
- Enhanced Video Processor Card (EVPC).

These cards were demonstrated by Gerd Weissmann of Kaiserslautern, Germany, a member of SNUG, at the 14th annual Chicago TI Faire, held in Evanston, Illinois, last November.Members of SNUG are Michael Becker, Walter Beez, Harald Glaab, Roland Meier, Gerd Weismann, and Juergen Stelter.

The cards were designed in Germany by Michael Becker, a member of SNUG. The cards were originally marketed to TI users in Europe.

The 80-column menu for the HSGPL-II card is now available in English for all registered owners. A copy may be obtained, at no additional charge, by sending proof of purchase to Cecure Electronics. Include a formatted SSSD disk along with a postage paid return disk mailer. A 40-column English version is being developed for users who do not have an 80-column card and want to use the HSGPL-II by itself.

Pricing for all three cards will be determined by the total number of orders received. However, Cecure says that the U.S. price will be less than the price in Europe.

Pricing for all three cards will be determined by the total number of orders received. However, Cecure says that the U.S. price will be less than the price in Europe. User groups will be given volume discounts for purchases for their members.

Also, if there is enough interest, kits that include only the gate array and printed circuit boards will be offered.

For Additional information include a complete description of your TI system, your mailing address, day and evening phone numbers, the best time to call and the name of the user group you belong to in your communication to: John F. Walden, Marketing; Cecure Electronics Inc.; P.O. BOX 2; Muskego WI 53150-0002; e-mail CECURE.ELECTRONICS.John.Walden@juno.com.; fax: 414 422-9889.

German-designed cards show big promise

The following is a compilation of information about the SNUG-designed cards taken from the Internet. Most of the information was written by Michael Becker, designer of the cards.

DISK CONTROLLER CARD

The BwG-Disk-Controller (nickname: means "Bester-wo-gibt") is fully CorComp-compatible. We made this card because the German Atronic controller was discontinued in the late 1980s. It is based on a WD1773 controller and works with a maximum of four 360KB (1440 sectors) floppy disk drives. The Corcomp version of PC-Transfer works fine with the BwG.

Since Triple-Tech was not available in Germany, the BwG has an internal real-time clock. The BwG uses a high-quality printed circuit board with gold-plated contacts and is fully buffered. The Corcomp k controller is full unbuffered so every misplacement in the box will destroy the PALs or the TMS9901 on it.

Gerd Weissmann showed the German PEB cards at the Chicago TI Faire last November. One the cards, the Enhanced Video Processor Card (EVPC), is an 80-column card. It produces 32,000 colors. It requires a Geneve/Amiga-compatible monitor, such as the no longer produced Magnavox 8CM515 to see all the colors. Weissmann showed highly detailed GIF pictures of earth globes and a picture of a hamster with each hair showing. Using the same picture he switched back and forth between 256 colors (the resolution of all other 80-column cards for the TI and the Geneve's resolution) and the EVPC's maximum resolution. The difference was dramatic. This card works with a regular TI99/4A system, or with the high speed CPU/GRAM system (SGCPU/HSGPL-II) that was also demonstrated.

SECOND GENERATION CPU

The Second Generation CPU/High eed GPL-II system consists of two cards that put the entire 99/4A inside the Peripheral Expansion Box Weissmann used a 99/4A keyboard (only) cabled to the SGCPU. The card now also works with an AT-style keyboard. The keyboard plugs into a jack on the card. The GRAM card (HSGPL-II), which must be used with the fast CPU card, has lots of GRAM memory. There are 16 banks of GRAM, and most of these banks are in flash EPROMs so they are very stable yet can be reprogrammed. Some of the GRAM memory is in RAM so that users can make alterations to GRAM modules before loading them into a flash EPROM.

These are expensive cards with unique features. The price of the 80-column card was 380DM ($1US=1.5DM).

The SGCPU is 100 percent TI-compatible with up to 1MB of AEMS-compatible RAM. All software written for AEMS will run on the SGCPU.

The reason the prices are so high are as follows: I have to pay nearly $150 for very good, gold-plated boards. We have only small quantities (30 to 50 pieces). Higher volumes should reduce the price.

HSGPL GROM CARD

The HSGPL is a GROM card (built with EEPROM) which supports all the 16 GPL banks supported by the TI99/4A operating system (for module library). The first two banks can be mapped with GRAM (built with RAM) for everyday loading, because you should only program EEPROM up to 1,000 times.

My HSGPL is the only GPL card in the world which supports original MBX modules with this special ROM banking. The only banking I do not support is the Databiotics banking, because it has more than 32KB of ROM. The maximum I can support is four banks of 8K. The HSGPL can be used with the original TI99/4A console — you don't need the SGCPU. However, if you use the SGCPU, you need the HSGPL and any 80-column card, such as the Dijit AVDP or EVPC (our card for the PEB). Both of these cards may be used with the TI99/4A console.

QUESTIONS AND ANSWERS

The questions asked here were submitted to Becker over the Internet by David Ormond.

Q: Is this a 9938 or 9958 device?

A: This is a 9938 device with a standard VGA color palette. This device is used on most older VGA cards and has a built-in color DAC.

Q: Are other 80-column devices (excluding the Geneve) completely compatible with regular 9918 operation which is stock in the 99/4A? And are they more-or-less completely compatible with each other running typical 80-column applications like Funnelweb. What is the story with the EVPC?

A: Fully compatible. Many of users use Funnelweb and it works fine.

Q: So the HSGPL card has 16 times 64K GROM (which is really EEPROM), which includes the console GROMs. So you can't use a HSGPL with a stock console. And 16 banks gets you the ability to load up to 13 cartridges? Are these reachable through the stock GPL menu?

A: Remove the GROMs from the sockets of the stock console. That's all you have to modify. Sixteen banks means all the GPL Read/Write addresses which are supported by the original TI99 operating system in ROM 0 ("Review Module Library"). With the new GROM0 from Winfried Winkler you can select forward and backward. The Review Module Library works only forward. You can program in the HSGPL as many modules as your memory allows. You can combine some modules with free GROMs and if GROMs do not overlap.

Q: About the ROM-6. This is the >6000 through >7FFE cartridge space, right? Which is 4K. So what is "16 banks, each banked by 4"? You switch between entire 4K ROM banks by doing a dummy write to address >6000 (to switch in bank 0), >6002 (for bank 1), >6004 (for bank 2), or >6006 (for bank 3). And this is all EEPROM versus RAM, so we can't run Supercart-type applications?

A: The range from >6000 trough >7FFF

(See Page 8)

CARDS —

(Continued from Page 7

is 8K. TI banks only the upper half, but we bank the whole 8K like Atari. This is the same if the lower half is programmed twice. The lower two GPL banks (9800, 9804) may be changed with GRAM/RAM banks which every size of the HSGPL card has. The RAM may be switched alone to emulate the Mini-Memory Module. The reason we have added two GRAM/RAM banks to the card is that many programmers of TI modules didn't' use the design rules from TI and made some GROM reads with direct addressing. Maybe with a read from >9800. If you have programmed such a module into a higher bank like >9810, then your system will hang up. In some modules we have exchanged such bugs with an indexed read with Register 13, but if the contents of R13 is destroyed by the program, this method will not work. So you can transfer the whole bank into both GRAM/RAM banks and switch over these banks to both base banks. This may be done with the transfer routine of the HSGPL program or simply with a BASIC CALL BANK(x). This will make the copy for you and the switch over. After a reboot (Quit or Bye) you have the standard screen without Review Module Library and your module is now in the base GROM. The trick of both base banks is: the console ROM 0 has a bug. Sometimes it will look into the next GPL address and has to find the same GROM0. Normally it looks for a difference between base 0 and 1. If a difference exists, the text Review Module Library will be shown. But sometimes GROM0 is read from base 1 (I do not know why) and so you have to have two GRAM/RAM banks. All other cards do not need this feature, because their bases are not fully decoded and their contents will be seen on all (or more) bases.

Q: Do you have multiple DSRs, which take up multiple available CRU locations, or a single DSR which you somehow switch in up to 512K of pages? Is the DSR location is selectable via a DIP switch anywhere from CRU >1000 to >1F00?

A: The DSR is very simple. It looks for an external GROM/ROM. If it finds a external GROM/ROM, then the module space (GROM3-GROM7 and the ROM6) will be disabled in the card. So you can use the HSGPL program to read-out your modules. Only GROM0-2 are enabled and you have to remove them from your console. The CRU is fixed on >1B00. This is the CRU location, which TI reserved for the expansion of the GPL interpreter. The ROM0 switches sometimes to that CRU location and looks for an interpreter code. We have reserved this area in the DSR flash for future expansion. Because this CRU address is used by the operating system of TI99, no other card (except the HSGPL card) has to use this address. If the GPL interpreter finds any code in a DSR ROM (>4020 and other) on CRU >1B00, your system may hang. Be careful to switch any other card to this location. We have made a correct return on these critical DSR-locations.

> My HSGPL is the only GPL card in the world which supports original MBX modules.... The only banking I do not support is the Databiotics banking, because it has more than 32KB of ROM.

Q: And there are really three ways to switch 4K banks in ROM-6, the XBASIC method (described above), the Atari method (uses two banks via dummy writes to >7000 and >7002), and the MBX method (whatever that is). But not the DataBioTics Super Space II CRU switch method?

A: The MBX method is a semaphore method to switch the ROM bank. You write the bank-number directly to location >6FFE. In any MBX module you will find a RAM of 256 bytes. This is decoded ROM location >6C00 trough >6FFF. So you can read back the bank number from this RAM, because it will be written in parallel to the switching register. Nice feature, but incompatible with all other banking methods. This is the reason, other programs will not correctly read out the ROM banks of MBX modules. You have to write 00, 01, 02 or 03 to the banking register. I cannot support the DataBiotics method, because this uses a maximum of eight banks. I switch 8K blocks and so I need 64K ROM or RAM. But I have only 32K (8K by 4). The CRU switching method was not the problem. I have tried to implement this, but I do not have the memory size. Sorry, but I want to support only original modules (or licensed and sold by TI like MBX). Atari was similar and no problem.

Q: The HSGPL card comes with software (or is it built-in firmware) that allows you to program the GROM and ROM-6 Flash EEPROMs and dump cartridges/ disk files/GROM/ROM, including MB. cartridges.?

A: Both. The CALLs are built-in, the HSGPL configuration program is built-in and on an extra disk (E/A5 file). A DSR loader program is included which allows you to reprogram your DSR with future versions. All memory is Flash EEPROM, which are reprogrammed in the system by your TI.

Q: Does this use the TMS9900 or 9995, or something else? What clock rate? The main memory is on the 16-bit bus with no wait states? So it runs full speed?

A: TMS9900. I have not produced a second Geneve. This is a fully compatible TI99/4. The clock-rate is 12MHz. The main ,memory runs in full 16-bit speed, but may be switched to the original speed by a CALL MEM8, or with your own assembler routines (a simple CRU bit). The scratch pad is extended to the full size from >8000 trough >83FF, but some programs do not like this feature (Mini-Memory has a bug, write to a 83xx location but read back from 82xx, I think a mistake from the source-code).

(See Page 8)

CARDS —

(Continued from Page 8)

Q: How do you alter the memory size? AEMS is 128K (one chip), 256K (two chips), 512K (one expensive module), or 1MB (two expensive modules). How does SGCPU do it? Does everything come in twos, since you use 16-bit width?

A: Because all memory is in 16-bit, you need two chips. Each of 32K (16K is not available), 128K or 512KB. So you have 32K, 256K or 1MB.

Q: And when you say "AEMS compatible," that means the same arrangement as the Asgard/SW99ers AMS, with 4K banks allocatable anywhere in the >2000->3FFE and >A000->FFFE ranges, with the two control bits at CRU >1E00 and >1E02 and the >'612 mapper registers at >4000 through >401E (or whatever)?

A: Yes! But without a real '612 (too expensive) I have done it it in another way. But it is fully compatible for all software.

Q: Is there a means of forcing wait states and dropping the clock rate so it runs in a 4A compatibility mode?

A: You may switch off the 16-bit speed.The clock-rate is then the same as the TI994A.

Q: Do the ROM-6 provisions of SGCPU conflict with those of the HSGPL?

A: No. If you enable the internal ROM-6 of SGCPU, the CPU does not look outside itself for that location.

Q: Are the console ROMs EEPROM or NVRAM, such that we can change the interrupt vectors and XOP tables?

A: The "console ROMs" are 27C256 EPROMs which you can reprogram with your own content. It was not possible to use EEPROMs here, because you cannot reprogram and use your operating system at the same time.

Q: So right now you connect a TI keyboard via a long ribbon cable to the SGCPU. The other option is a standard IBM-style keyboard interface (that hopefully is similar to the WHT AT keyboard interface (which accepts any keyboard) rather than the Geneve (which accepts only obsolete keyboards)), and when available, connects right into the SGCPU. Or would you need a newer model of SGCPU with the AT keyboard interface?

A: The keyboard interface uses a single chip controller on the SGCPU. This part must be added in its socket after the program is finished. That's all. It will accept any MF2-type PC keyboard, I think Windows-95 keyboards, too.

Q: A processor bus expansion connector, IDC ribbon cable style?

A: Yes, 10-pin IDC with a special feature — if you enable this port by external decoding logic, you can disable every memory location inside the SGCPU and use this location externally. For example, you can use the last free bytes of ROM0 for a 16-bit port by enabling the ENA16 signal on that connector on that address. Then a fetch to this address space will go outside the SGCPU in 16-bit. It is nearly the same as the upper half connector of IBM-PCs, which is the difference between XT and AT-style connectors. If you use it, all actions will be in 16-bit, but if you do not use it, all will be in 8-bit with wait-states.

READER-TO-READER

Using LLIST with a serial interface

This comes from Martin Zeddies, Hauptstrasse 26, D-38446, Wolfsburg Germany; phone/fax:+49-5363-71125.

For a long time I have used my Geneve to write programs under MY-BASIC. At the moment I work with version 3.0 and found a problem with the MY-BASIC's printer interface.

A bug appears when you try to use the LLIST command with a printer that works with an RS-232 serial interface. If you try to list a program with LLIST nothing will happen. Only the LED on the RS-232 card will flash.

I'd like to find a solution to this problem. Here is what I have tried so far:

I checked the four Basic files with a disk editor and found only one place in the complete program where PIO appears. In the Basic that I use you can find the definition of the printer three sectors before the end of the first Basic file. That is the 62nd sector of the entire BASIC program. The name starts at byte 72 in this sector.

I changed the name at this place to PRN from PIO (that is the same length and so there is no need to change the length byte) and hoped that my Geneve would use the internal PRN interface to print out the program listing using the LLIST command.

Unfortunately, what I got was I/O-Error 07 as I tried to print out a program listing. Maybe there is another user who can tell me where else the printer is defined. I think there must be a place for an interface name of 40 characters, because that is the standard length of MDOS filenames and device names.

I think if you have a parallel printer in your system you never see the problem when you use MY-BASIC.

If you have a computer problem, software or hardware, that's been keeping you up at night, send it here and we'll pass it on to our TI/Geneve savvy readers.—Ed.

TI FAIR ROUNDUP

TIers gather in U.S., U.K., and Germany

TI99/4A users in various locales will be able to see products for their computer and network with their fellow hobbyists at fairs worldwide.

Fest West and a German TI fair are set for April, and the Annual Group Meeting for United Kingdom TI users and the Multi Users Group Conference in Lima, Ohio, for May.

FEST WEST '97

Fest West is scheduled for Saturday April 5 at the San Jose Civic Auditorium across from the Convention Center in downtown San Jose, California.

In past years, the Fest has been held in Salt Lake City, Phoenix, Anaheim, Tucson, San Diego, Las Vegas and Los Angeles. For the very first time, the Fest will be held in the heart of Silicon Valley, organizers point out.

As the event is centrally located in the heart of downtown San Jose., many attractions are within just a few blocks walking distance, such as the Tech Museum of Innovation, the Children's Discovery Museum, the San Jose Museum of Art, American Musical Theater of San Jose, San Jose Repertory Theater and American Museum of Quilts & Textiles.

Hotels in the downtown area include: Best Western Inn, 455 South Second St., (408)-298-3500; Townhouse Motel, 475 South Second St., (408)-295-5558; Fairmont Hotel, 170 South Market St., (408)-998-1900; Hilton Hotel, 300 Almaden Blvd., (408)-287-2100 (adjacent to the convention center); Holiday Inn, 282 Almaden Blvd., (408)-998-0400 (adjacent to the Civic Auditorium).

The first two hotels listed have low room rates, according to organizers.

Transportation to and from the airport is available via local hotel shuttle buses, taxi, bus and airport super-shuttle. Transportation to and from Amtrak at the Tasman station is available via local light rail. The light rail system goes nearly anywhere within downtown San Jose, Fest organizers say, and can be taken to the Cal-

As the event is centrally located in the heart of downtown San Jose., many attractions are within just a few blocks walking distance, such as the Tech Museum of Innovation, the Children's Discovery Museum, and the San Jose Museum of Art.

train station to make a trip to San Francisco.

Adult admission is $5 for all day; children under 16 are free with a paid adult. The fair includes opportunities to win prizes including a Western Horizons Technology SCSI Controller, WHT AT Keyboard interface, Super AMS, Horizon RAMdisk, TI CD-ROM and software.

For a Fest West '97 Vacation Package, write or call: Fest West '97, c/o South Bay TI Users Group, 3297 Woody Lane, San Jose, CA 95132, (408) 934-0352, or on the World Wide Web go to "http://www.sonyx.com/wht/ti/" and select Fest West 97 from that page.

TI-FAIRE SOUTHWEST GERMANY

The TI-Faire of the regional TI user groups in Southwest-Germany will be held April 26, according to a recent Internet post. The location is Heppenheim/Bergstrasse, the place is Hotel Starkenburger-Hof near the railway-station of Heppenheim. Those interested in coming to this fair are asked to contact Volker Papst, Heppenheim, Phone: +49-6252-2903 or Michael Becker, Mannheim, Phone: +49-621-722735 For overnight stays in the hotel, prices are bed and breakfast, single person: 68 DM; bed only: 58 DM.

UK ANNUAL GROUP MEETING

The TI Users Group U.K. has scheduled its Annual Group Meeting for 10 a.m.-5 p.m. May 10 at the Princess Anne St. Johns Ambulance Training Centre, Trinity Street, Derby, England.

Eight 2400 baud modems will be awarded to the first members who attend, according to the TI*MES, the group's newsletter.

For further information, contact Trevor Stevens, 249 Southwell Rd. East, Rainworth, Notts., NG21 0BN, UK, telephone 01623 793077, or BBS 01623 491282 (Friday 7 p.m. to Sunday 10 p.m. only, times are United Kingdom times).

MUG CONFERENCE

The Multi Users Group (MUG) Conference is scheduled for May 23-24 in Reed Hall on the campus of Ohio State University at Lima.

The event is free to vendors and participants.

The Jim Peterson Memorial Awards will be presented at the conference. Nominations for the four awards in community service, Geneve, hardware and software, respectively are being handled by Jim Krych. Anyone can nominate a recipient by writing Krych at 3969 Clague Rd., North Olmsted, OH 44070 or e-mail him at ab453@cleveland.Freenet.Edu..

Seminar speakers scheduled so far are Bob Carmany, "A Program That Writes Programs with Token Codes," and Ron Markus, "The Prostick II Joystick and Texaments Products."

For further information, contact Charles Good, P.O. Box 647, Venedocia, OH 45894, phone (419) 667-3131, e-mail good.6@osu.edu or the Web site at www.bright.net/~cgood/mug1997.html.

Good notes, "As requests for table space and seminar speakers are received the MUG Conference Web site is updated almost immediately."

THE ART OF ASSEMBLY — PART 63

The c99/assembly connection

By BRUCE HARRISON

And now for something completely different.... With apologies to all Monty Python fans, our topic for this edition is all new. We're going into the mysterious world of C programming, and making some things happen that C normally won't allow. First, our thanks to Clint Pulley, for designing c99, and to Vern Jensen, for inspiring us to create some new libraries for use by the C programmers in our midst. Thanks also to Tom Shorock, whose disk "O Say Can You C" provided some early insight into our experimental C programming efforts, and to Charles Kirkwood, who also provided some examples of C programming. No, your assembly author is not turning into a C programmer, but if one is going to make libraries of routines to be called from C, one must have at least a little knowledge of that language in order to test the routines.

This all started with a letter from Vern Jensen. He's doing C programs, and some of what he's trying to do is outside the normal realm of C. For example, he wanted to do some tricks with scrolling that simply can't be done outside of the enhanced graphics (aka half-bit map) mode. For those who don't C, the normal mode for C programs is *text* mode, with white on dark blue as its color scheme. That works nicely for many purposes, but for games of any kind, one needs the graphics modes. In Vern's case, we were sure from the outset that he'd need at least the enhanced graphics mode, and he'd also need some "tools" for operating in that mode, including sprite capability.

CROSSING THE BRIDGE

As our readers will remember, we just recently crossed the bridge to enhanced graphics with subroutines for use in assembly programs. Those get invoked by BLWP vectors, which is very handy indeed for assembly work, but isn't at all useful with C. Thus we took our original HBSUB/S file, printed out Clint Pulley's Manual for c99, and set to work.

In the originals, the parameters were passed as DATA/BYTE statements after the BLWP instruction, so our getting of parameters involved moving words or bytes from the location pointed to by R14. In the C case, R14 is a stack pointer, and our parameters get placed on C's stack before our routine is called. Thus we use R14 as a base to get our parameters.

For example, if there's just one parameter, we MOV @2(R14),R0 to get our parameter. If there are more, we pull them from the stack with MOV @4(R14),R1, etc. That turned out to be easy to handle, provided we remember that the parameters are on the stack in inverse order, so the last one is at two bytes above R14, the next to last at four bytes above R14, and so on.

In our original assembly BLWP routines, we return to the main program by an RTWP instruction. In the case of C, we return to the C program by B *R13, since R13 is where C puts the return address when it branches to our code. In our own code, we can use registers 0 through 7 with complete freedom, since C uses them only for temporary storage. If we have a value to pass back to a C variable, that goes into R8, and C places it in the variable for us upon return. Otherwise, we leave R8 through R15 alone! C has those registers "spoken for." Since most of our routines already were using the lower numbered registers, the limitation of R0 through R7 was no problem.

THE C SIDE OF THE BRIDGE

Each of our routines gets called from C by just a simple statement. Let's say, for example, that we wanted to put a string of characters on the half-bit map screen using our routine HBSTR. We do it from C like this:

```
hbstr(row,col,"This goes on the screen at Row, Col");
```

Note that lowercase is used, and that there's a semicolon at the end of the statement. Row and Col may be either numbers or variables. The string is passed to us as an address, and is actually stored by C as an ASCIZ string.

"What's THAT?", you ask. It means that the characters in the string's content start at the address passed us by C, and the end of the string is marked by a byte of 0. Thus our HBSTR routine has to check each character before putting it on screen, and if it finds a 0, then the string is finished. There's one more trick up C's sleeve in these ASCIZ strings. If the quoted string contains a \n, C compiles that to character 10 (>A), and this indicates that we're to start a new line on the screen. In our routine, then, whenever a character 10 is found, we AI R0,32 and then ANDI R0,>FFE0, which puts our writing address at the start of the next half bit-map screen row.

THE EXTRA GOODIES

In our usual fashion, we added a whole library of routines to put sprites on the half-bit map screen, complete with automatic motion, and fleshed out the "normal" stuff with a scroll screen routine, a clear screen routine, and so forth. There's a third library for taking inputs from the half bit-map screen. The main half bit-map library also includes routines to take you smoothly back to text mode, or into the normal graphics mode from enhanced graphics, so you can mode-switch gracefully within a C program.

As time goes on, we're always adding new things to the libraries. For example, we added a "rainbow" capability to take maximum advantage of enhanced graphics mode. Thus you can colorize one or more characters with eight different color schemes, one for each row of the character. There's a simpler routine called HCOLOR, which uses only a single foreground and background color, but which can color a number of successive characters on just one call.

DIFFERENT CONVENTIONS

In our normal assembly work, everything is "zero-based." Color codes run from 0 through 15, rows from 0 through 23, columns from 0 through 31, and so on. In C parlance, colors run from 1 through 16, rows from 1 through 24, and columns from 1 through 32, etc. The routines that we designed for use with C follow the C conventions in these matters, with one exception. The

(See Page 12)

THE ART OF ASSEMBLY —

(Continued from Page 11)

numbers for sprites run from 0 through 31. Thus, except for this case, the average C programmer should feel quite at home with our routines. Of course he or she will need to get used to dealing with the idea of having the enhanced graphics mode itself to deal with, plus the burden of dealing with sprites and such, all of which are alien to the run of the mill C program.

As we did for assembly programmers, we've given the C programmer working in enhanced graphics mode a complete set of character definitions for the sprites, separate from the character definitions for ordinary screen printing. There are routines for doing a "CALL CHAR" type operation for both the normal and sprite characters. These use the conventional hex character string just as you'd use in Basic, except that each call can have no more than 16 hex characters, thus one can define only one character per call.

THE CHOICES ARE YOURS

There are three libraries available. They're called CHBSUB/O, CHBINP/O,and CHBSPR/O. The second and third ones cannot be used alone, but must be used along with CHBSUB/O. If your enhanced graphics work doesn't need sprites or inputs from the screen, then you can use just CHBSUB/O without the second or third ones. Please note that these subroutine libraries are designed for use only with C programs, and can't be used from ordinary assembly programs. That's so because they depend entirely on the C method of parameter passing and the C method of executing a return to the main program. Fortunately for us, Clint Pulley made the whole process very simple by providing for stacking parameters and returning values to the C program.

FOR WHATEVER THEY'RE WORTH

Here are our opinions about c99. Clint Pulley did a fantastic job of designing a system to implement this language on the TI-99/4A. Ordinarily, we'd consider this kind of thing impossible, but Clint did it and made a very easy and natural interface to the assembly that's the machine's natural language. His one-pass compiler is easy, quick, and user-friendly. Our only gripe, and this is true of any compiled language, is that it takes a lot of memory space to do even a little program. We place the same gripe against our own Extended BASIC compiler, where small programs become quite large when compiled.

We have found two areas of incompatibility between Clint's C system and our own enhanced TI. These involve our use of Horizon RAMdisks and Horizon's P-GRAM card. The C compiler itself will not run when any of our three Horizon RAMdisks is turned on. It loads, but then lights up one of the RAMdisk activity lights and locks the computer up. We talked to Clint about this, and while he's aware of it, he's got no solution.

The second area of incompatibility is a bit more serious, as it involves both the compiler and Option-5 programs generated from C source. Upon startup, the C-derived option-5 programs check for the presence of the Editor/Assembler utilities in low memory. If those are already present, all is well. If the E/A utilities are not loaded, the C support tries to load them directly from GROM, and that's where the problem exhibits itself. In the "normal" case where an E/A module is present, the utilities are there in the >7000 block of GROM, and the C program finds and loads them perfectly well. If, however, one has a P-GRAM setup like ours, Extended BASIC is in the GROM space up through the >D000 block, and E/A is in the >E000 and >F000 blocks. Thus if we load up a C-derived Option-5 program in our normal configuration, we get a system lockup as soon as it tries to load the utilities.

For our own purposes, we use a version of our own "sandwich" for converting the C object file(s) into Option-5, and since the sandwich loads the E/A utilities from memory using the SLAST memory space, everything works okay even with our P-GRAM setup.

For our own purposes, we use a version of our own "sandwich" for converting the C object file(s) into Option-5, and since the sandwich loads the E/A utilities from memory using the SLAST memory space, everything works okay even with our P-GRAM setup. Of course, this latter problem can also be solved by simply putting our E/A cartridge in the GROM port, and then our P-GRAM effectively removes itself from the system.

As we almost always do, we've made the disk containing these C utilities for operation in enhanced graphics (or half bit-map) available as public domain software through the Lima Users' Group library. As always, the disk contains demos that show off most of the features, a set of instructions for using the libraries, and such. This disk will not be of any use to those who are not into C programming. Advance copies have gone to some of our friends who are programming in C. (You know who you are.) See the listing for a list of routines contained in the library.—Ed.

That's quite a lot, isn't it. Where we've used # in the sprite cases, that would be a number from 0-31, corresponding to the full range of sprites available from assembly code. The levels for magnification are 1-4 to correspond with XB numbering, where 1 is normal size, 2 is double size, and 3 and 4 are 4-character sprites. Positions are in dot-row and dot-column coordinates for sprites. For other things, Row and Col numbers are in the graphics mode ranges 1-24 and 1-32, respectively. (These correspond to the HCHAR, VCHAR, GCHAR columns in BASIC/XB parlance.)

c99 programmers should note that all variables with the exception of the string variables used in ACCSTR and HBSTR must be

(See Page 13)

THE ART OF ASSEMBLY —

(Continued from Page 12)

of the INT type. The string variables used with ACCSTR and HBSTR must be a CHAR type array. HBSTR will actually work with either a CHAR array or a quoted string, as c99 passes us the starting address in either case. ACCSTR puts the string into the named array in ASCIZ format, as required by c99.

There's no sidebar this issue, as the source code for these libraries would take up half an issue by themselves. We invite others besides Vern Jensen to ask for help with things related to interfacing between C and assembly. Perhaps we'll turn out a special routine just for you! Next issue we'll try to surprise you again with an off-the-wall topic like this.

c99 library contents

Here's a list of the routines contained in these libraries, with a short description of what each does.

Contents of CHBSUB/O

CALL FROM C	DESCRIPTION
sethb()	Sets computer to Half Bit Map
setcc(fc,bc)	Sets colors for all Half Bit Map screen characters to fc foreground and bc background.
setgm()	Sets computer to Graphics mode
settm()	Sets back to Text Mode (C's default)
hbstr(row,col,"Quoted String")	Displays the stuff between quotes at row, col. (Note that col must be 32 or less)
hbchr(row,col,ascii)	Displays a single char at row, col
hbcls()	Clears the half-bit screen
hbclf(row,col,rpts)	Clears a part of the screen starting at row, col, and extending rpts locations horizontally
hcolor(char,foregrnd,backgnd,repeats)	Sets color scheme for one or more characters
hrbow(char,repeats,fc,bc,fc,bc...)	Sets a multi-color scheme for one or more characters
calchr(char,"pattern string")	Sets the definition of one character to the contents of the string. (like CALL CHAR)
hbscrl()	Scrolls the half-bit screen up one row
hbvchr(row,col,char,repeats)	Similar to BASIC VCHAR hbhchr(row,col,char,repeats)Similar to BASIC HCHAR hbdint(row,col,integer)Displays an integer numeric value or variable on the screen at row,col

Contents of CHBINP/O:

CALL FROM C	DESCRIPTION
x=accnum(row,col,clrsig)	Accepts an integer number from the screen at row, col. The clrsig indicates whether or not to clear the field before taking input.
accstr(row,col,maxlen,clrsig,buffer)	Accepts a string of maxlen at row, col, places that in buffer. As above, clrsig indicates clearing of the field, or not.
k=hbgkf(row,col)	Accepts a single keystroke with flashing cursor, then echoes the key to the screen at row, col.
k=hbgk(row,col)	Same as above, but no cursor.
hbclf(row,col,rpts)	Clears a part of the screen starting at row, col, and extending rpts characters.

Contents of CHBSPR/O, for sprites:

CALL FROM C	DESCRIPTION
sprite(#,ypos,xpos,char,color,yvel,xvel)	Puts a sprite on the screen at ypos, xpos, with Char character, Color color, and velocities Yvel, Xvel.
motion(#,yvel,xvel)	Changes motion of sprite # to yvel,xvel y=yposit(#)Sets y to the sprite #'s y position
x=xposit(#)	Sets x to the sprite #'s x position
locspr(#,ypos,xpos)	Sets location of sprite # to ypos, xpos pattrn(#,char)Sets character value sprite # to char
color(#,color)	Sets color of Sprite # to color
magnif(maglev)	Magnifies all sprites to maglev (1-4)
revmo(#)	Reverses motion of sprite #
k=coinc(a,b)	k will be one if sprites a and b are in coincidence, 0 if they're not
delspr(#)	Deletes sprite # and all higher numbers
rev1(char)	Reverses a sprite character pattern
rev4(char)	Reverses a 4-character sprite pattern
sprchr(char,"pattern string")	Defines pattern for a sprite character to contents of string (similar to CALL CHAR)

EXTENDED BASIC

Creating the domino effect

The following Extended BASIC program, called The Domino Factor, was written by M.C. Sumner and originally distributed by CSI Software.

Using the arrow keys, steer the Dominex through the maze without hitting walls or tumblers. The Dominex leaves a trail of dominoes that must also be avoided. When you hit something the dominoes fall and you are awarded one point for each domino. Catch the thumpers for extra points.

DOMINEX

```
100 REM *DOMINEX* !034
105 DMX=3 :: DIR=2 :: PY=12
:: PX=16 !236
106 CALL CHAR(42,"FFFFFFFFFF
FFFFFF")!014
110 CALL MAGNIFY(3)!224
120 CALL CHAR(140,"3E3E363E2
43E3E3E")!125
130 CALL COLOR(14,2,15)!022
140 CALL CHAR(64,"3C429D9191
9D423C")!056
150 CALL CHAR(112,"03070F080
8080808080808080808080FFEFEF
E0E0E0E0E0E0E0E0E0E0E0E0CF8"
)!060
151 CALL CHAR(96,"0000007E7E
")!201
152 CALL CHAR(97,"0018181818
1818")!140
153 CALL CHAR(98,"00040E1C38
7020")!152
154 CALL CHAR(99,"002070381C
0E0400")!251
155 CALL CHAR(100,"7E7E5A7E2
4183C7E")!133
156 CALL CHAR(101,"0000007E5
A7E667E")!081
157 CALL CHAR(102,"00665A3C1
8182418")!053
158 CALL CHAR(103,"00665A3C1
8244281")!051
160 CALL CHAR(116,"0F1F3F202
0202020202020202020203F00F0F0F
0707070707070707070707060C0")!
197
161 CALL CHAR(104,"0000387C9
2BAFEAA")!126
162 CALL CHAR(105,"387CBA92F
EAA286C")!162
170 CALL CHAR(120,"0F1F3F202
0202020202020202020203F0000E0E0E
0E0E0E0E0E0E0E0E0E0C080")!21
0
180 CALL CHAR(124,"7EFE86868
6868686868686FC")!086
190 CALL CHAR(128,"7CFC8C8C8
C8C8C8C8CF8")!198
200 CALL CHAR(132,"000000010
30F0B3E26F694949C9090F000000
08080")!175
210 CALL CHAR(95,"00FF")!150
220 CALL SCREEN(15):: CALL C
LEAR !027
230 CALL HCHAR(7,1,95,32)!18
3
240 CALL SPRITE(#1,112,2,160
,10,#2,112,2,150,18,#3,112,2
,140,26)!034
250 CALL SPRITE(#4,116,2,130
,36,#5,116,2,120,43)!204
260 CALL SPRITE(#6,120,2,110
,49,#7,120,2,100,55)!201
270 CALL SPRITE(#8,124,2,93,
63,#9,124,2,86,68)!138
280 CALL SPRITE(#10,128,2,80
,73,#11,128,2,73,77)!223
290 CALL SPRITE(#12,132,2,61
,81)!101
300 DISPLAY AT(1,1):"CSI PRE
SENTS:" !220
310 DISPLAY AT(8,17):"THE" !
069
320 FOR A=1 TO 108 !154
330 READ Y,X !251
335 CALL SOUND(-10,-5,2)!110
340 CALL HCHAR(Y,X,140)!180
350 NEXT A !215
360 FOR A=1 TO 20 :: CALL SO
UND(-200,440+(20*A),5):: NEX
T A !225
365 FOR A=1 TO 20 :: CALL SO
UND(-200,840-(20*A),5):: NEX
T A !230
370 CALL CLEAR !209
380 DISPLAY AT(1,1):"THE DOM
INO FACTOR 1.0" !140
390 DISPLAY AT(24,1):"@1982
M.C. SUMNER" !131
400 DISPLAY AT(12,12):"PRESS
:" !085
410 DISPLAY AT(14,14):"1) ST
ART GAME" !219
420 DISPLAY AT(16,14):"2) RE
VIEW RULES" !149
430 CALL KEY(0,K,S):: IF S=0
THEN 430 !243
435 IF K=50 THEN 540 !083
437 IF K=49 THEN 720 ELSE 43
0 !008
440 DATA 12,11,12,12,12,16,1
2,19,12,21,12,23,12,25,12,27
,12,30 !182
450 DATA 13,11,13,13,13,15,1
3,17,13,19,13,20,13,21,13,23
,13,25,13,27,13,29,13,31 !01
5
460 DATA 14,11,14,13,14,15,
4,17,14,19,14,21,14,23,14,2
,14,27,14,29,14,31 !090
470 DATA 15,11,15,13,15,15,1
5,17,15,19,15,21,15,23,15,25
,15,26,15,27,15,29,15,31 !04
5
480 DATA 16,11,16,12,16,16,1
6,19,16,21,16,23,16,25,16,27
,16,30 !218
490 DATA 19,9,19,10,19,11,19
,13,19,14,19,15,19,17,19,18,
19,19,19,21,19,22,19,23 !033
500 DATA 19,26,19,29,19,30,1
9,31,20,9,20,13,20,15,20,17,
20,22,20,25,20,27,20,29 !243
510 DATA 20,31,21,9,21,10,21
,13,21,14,21,15,21,17,21,22,
21,25,21,27,21,29,21,30 !209
520 DATA 22,9,22,13,22,15,22
,17,22,22,22,25,22,27,22,29,
22,30 !154
530 DATA 23,9,23,13,23,15,23
,17,23,18,23,19,23,22,23,26,
23,29,23,31,0,0 !198
540 REM-INSTRUCTIONS !124
545 A=0 !248
550 CALL CLEAR :: CALL DELSP
RITE(ALL):: CALL MAGNIFY(1)!
```

(See Page 15)

DOMINEX —

(Continued from Page 14)

```
038
560 DISPLAY AT(1,1):" THE
DOMINO FACTOR RULES" !237
570 DISPLAY AT(24,1):"@1982
M.C. SUMNER" !131
580 DISPLAY AT(5,5):"USING T
HE ARROW KEYS," !249
590 DISPLAY AT(6,1):"STEER T
HE DOMINEX " !017
600 DISPLAY AT(7,1):"THROUGH
THE MAZE WITHOUT" !003
610 DISPLAY AT(8,1):"HITTING
WALLS OR TUMBLERS." !122
620 DISPLAY AT(10,5):"THE DO
MINEX LEAVES A " !226
630 DISPLAY AT(11,1):"TRAIL
OF DOMINOES WHICH YOU" !206
640 DISPLAY AT(12,1):"MUST A
LSO AVOID." !190
650 DISPLAY AT(14,5):"WHEN Y
OU HIT SOMETHING," !192
660 DISPLAY AT(15,1):"THE DO
MINOES FALL AND YOU " !064
670 DISPLAY AT(16,1):"ARE AW
RDED 1 POINT FOR EACHDOMINO
 CATCH THE THUMPERS FOR EX
TRA POINTS." !203
680 DISPLAY AT(22,5):"PRESS
ANY KEY TO BEGIN" !086
690 CALL SPRITE(#1,100,5,40,
175,#2,102,2,56,220,#3,104,6
,128,240)!100
700 A=A+1 :: IF A=2 THEN A=0
!163
705 CALL PATTERN(#1,100+A)::
CALL PATTERN(#2,102+A,#3,10
4+A)!068
710 CALL KEY(0,K,S):: IF S=0
THEN 700 !002
720 CALL MAGNIFY(1):: CALL C
LEAR :: CALL DELSPRITE(ALL)!
038
725 DISPLAY AT(12,10):"SELEC
T:" !135
727 DISPLAY AT(14,12):"1) TU
MBLER" :: DISPLAY AT(16,12):
"2) TUMBLERS" !043
728 CALL SPRITE(#1,102,7,104
,194,#2,102,7,120,194,#3,102
,7,120,204)!202
730 CALL KEY(0,K,S):: CALL P
ATT :: IF S=0 THEN 730 !068
735 IF K=49 THEN T=1 ELSE IF
K=50 THEN T=2 ELSE 730 !160
740 CALL MAGNIFY(1):: CALL D
ELSPRITE(ALL):: CALL MAKEMAZ
E :: GOTO 3050 !211
745 ODIR=0 :: DIR=2 :: MY=0
:: MX=1 :: DT=97 !207
750 REM-MAIN TREE !060
760 GOSUB 1000 :: GOSUB 1000
:: CALL PATC !011
770 CALL THUNK :: GOSUB 1000
:: GOSUB 1000 :: CALL PATC
!129
780 CALL TUMBLE(T):: GOTO 76
0 !192
1000 REM-KEY AND MOVE !250
1010 DP=DP+1 :: IF DP=2 THEN
DP=0 !239
1020 CALL PATTERN(#1,100+DP)
:: CALL SOUND(-100,131,10,13
2,10,133,10,-4,9):: CALL COI
NC(ALL,C):: IF C<0 THEN GOSU
B 3500 !176
1030 CALL KEY(0,K,S):: IF S=
0 THEN RETURN !003
1040 IF K=69 THEN DIR=1 :: M
Y=-1 :: MX=0 :: DT=96 :: GOT
O 1060 !151
1050 IF K=68 THEN DIR=2 :: M
Y=0 :: MX=1 :: DT=97 :: GOTO
1060 !214
1055 IF K=88 THEN DIR=3 :: M
Y=1 :: MX=0 :: DT=96 :: GOTO
1060 !216
1057 IF K=83 THEN DIR=4 :: M
Y=0 :: MX=-1 :: DT=97 !158
1060 IF ODIR=0 THEN 1110 ELS
E IF DIR<>ODIR THEN 1125 !17
5
1110 CALL GCHAR(PY+MY,PX+MX,
G):: IF G>32 THEN 2000 ELSE
CALL LOCATE(#1,(PY+MY)*8-8,(
PX+MX)*8-8)!007
1120 CALL HCHAR(PY,PX,DT)::
ODIR=DIR :: PX=PX+MX :: PY=P
Y+MY :: RETURN !210
1125 ON DIR GOTO 1130,1160,1
190,1220 !169
1130 IF ODIR=2 THEN DT=98 ::
MY=-1 :: MX=0 :: GOTO 1110
!087
1140 IF ODIR=4 THEN DT=99 ::
MY=-1 :: MX=0 :: GOTO 1110
!090
1150 IF ODIR=3 THEN DIR=3 ::
GOTO 2000 !093
1160 IF ODIR=1 THEN DT=98 ::
MY=0 :: MX=1 :: GOTO 1110 !
148
1170 IF ODIR=3 THEN DT=99 ::
MY=0 :: MX=1 :: GOTO 1110 !
151
1180 IF ODIR=4 THEN DIR=4 ::
PX=PX-1 :: GOTO 2000 !171
1190 IF ODIR=2 THEN DT=99 ::
MY=1 :: MX=0 :: GOTO 1110 !
150
1200 IF ODIR=4 THEN DT=98 ::
MY=1 :: MX=0 :: GOTO 1110 !
151
1210 IF ODIR=1 THEN DIR=1 ::
GOTO 2000 !089
1220 IF ODIR=1 THEN DT=99 ::
MY=0 :: MX=-1 :: GOTO 1110
!087
1230 IF ODIR=3 THEN DT=98 ::
MY=0 :: MX=-1 :: GOTO 1110
!088
1240 IF ODIR=2 THEN DIR=2 ::
PX=PX+1 :: GOTO 2000 !166
2000 REM-DOMINOES FALL !100
2005 PY=PY-MY :: PX=PX-MX !1
11
2010 CALL DELSPRITE(ALL):: C
ALL SOUND(100,110,1,111,1,11
2,1,-8,1)!014
2020 IF DIR<>1 THEN 2080 !12
5
2030 CALL GCHAR(PY,PX,G):: I
F G<>96 THEN 2050 !194
2035 CALL SUBPOINTS(SSCORE)!
113
2040 CALL HCHAR(PY,PX,32)::
CALL SOUND(10,-3,5):: PY=PY+
1 :: GOTO 2030 !101
2050 IF G=98 THEN CALL HCHAR
(PY,PX,32):: CALL SOUND(10,-
5,5):: PX=PX-1 :: GOTO 2090
!023
2060 IF G=99 THEN CALL HCHAR
(PY,PX,32):: CALL SOUND(10,-
5,5):: PX=PX+1 :: GOTO 2220
!153
2070 IF G<96 THEN 3000 !255
2080 IF DIR<>2 THEN 2140 !18
6
2090 CALL GCHAR(PY,PX,G):: I
F G<>97 THEN 2110 !255
2095 CALL SUBPOINTS(SSCORE)!
113
2100 CALL HCHAR(PY,PX,32)::
```

(See Page 16)

DOMINEX —

(Continued from Page 15)

```
CALL SOUND(10,-3,5):: PX=PX-
1 :: GOTO 2090 !161
2110 IF G=98 THEN CALL HCHAR
(PY,PX,32):: CALL SOUND(10,-
5,5):: PY=PY+1 :: GOTO 2030
!219
2120 IF G=99 THEN CALL HCHAR
(PY,PX,32):: CALL SOUND(10,-
5,5):: PY=PY-1 :: GOTO 2150
!086
2130 IF G<96 THEN 3000 !255
2140 IF DIR<>3 THEN 2210 !00
1
2150 CALL GCHAR(PY,PX,G):: I
F G<>96 THEN 2170 !058
2155 CALL SUBPOINTS(SSCORE)!
113
2160 CALL HCHAR(PY,PX,32)::
CALL SOUND(10,-3,5):: PY=PY-
1 :: GOTO 2150 !223
2170 CALL HCHAR(PY,PX,32)::
CALL SOUND(10,-5,5)!084
2180 IF G=98 THEN PX=PX+1 ::
GOTO 2220 !194
2190 IF G=99 THEN PX=PX-1 ::
GOTO 2090 !066
2200 IF G<96 THEN 3000 !255
2210 IF DIR<>4 THEN 2020 !06
7
2220 CALL GCHAR(PY,PX,G):: I
F G<>97 THEN 2240 !129
2225 CALL SUBPOINTS(SSCORE)!
113
2230 CALL HCHAR(PY,PX,32)::
CALL SOUND(10,-3,0):: PX=PX+
1 :: GOTO 2220 !029
2240 CALL HCHAR(PY,PX,32)::
CALL SOUND(-10,-5,0)!017
2250 IF G=99 THEN PY=PY+1 ::
GOTO 2030 !006
2260 IF G=98 THEN PY=PY-1 ::
GOTO 2150 !127
2270 IF G<96 THEN 3000 !255
3000 REM *TOTAL SCORE* !046
3010 DISPLAY AT(8,12)SIZE(6)
:"DOMINO" :: DISPLAY AT(9,12
)SIZE(6):"FACTOR" !122
3015 CALL HCHAR(11,16,42,2)!
222
3020 SCORE=SCORE+SSCORE+BSCO
RE :: DISPLAY AT(22,15)SIZE(
13):"SCORE:";SCORE !069
3025 BSCORE=0 !117
3030 IF SSCORE>150 THEN DMX=
DMX+1 :: CALL BMUSIC !031
3040 IF SSCORE<100 THEN DMX=
DMX-1 :: CALL LMUSIC ELSE CA
LL SMUSIC !228
3042 SSCORE=0 !134
3045 IF DMX<1 THEN 4000 !078
3050 DISPLAY AT(22,1)SIZE(14
)BEEP:"DOMINEXUS:";DMX !104
3060 CALL SPRITE(#1,100,3,88
,120,#2,102,7,32,112):: IF T
<2 THEN 3070 ELSE CALL SPRIT
E(#3,102,7,32,156)!068
3070 CALL SPRITE(#4,104,5,10
4,40,#5,104,5,104,224,#6,104
,5,24,28,#7,104,5,24,224)!16
6
3080 THK(4)=1 :: THK(5)=1 ::
THK(6)=1 :: THK(7)=1 :: PY=
12 :: PX=16 !223
3090 GOTO 745 !058
3500 CALL COINC(#1,#2,8,C)::
IF C<0 THEN 2000 !224
3510 CALL COINC(#1,#3,8,C)::
IF C<0 THEN 2000 !225
3520 CALL COINC(#1,#4,8,C)::
IF C<0 THEN BSCORE=BSCORE+2
5 :: CALL DELSPRITE(#4)!113
3530 CALL COINC(#1,#5,8,C)::
IF C<0 THEN BSCORE=BSCORE+2
5 :: CALL DELSPRITE(#5)!115
3540 CALL COINC(#1,#6,8,C)::
IF C<0 THEN BSCORE=BSCORE+2
5 :: CALL DELSPRITE(#6)!117
3550 CALL COINC(#1,#7,8,C)::
IF C<0 THEN BSCORE=BSCORE+2
5 :: CALL DELSPRITE(#7)!119
3560 DISPLAY AT(9,12)SIZE(6)
:BSCORE !171
3570 RETURN !136
4000 REM-END OF GAME !141
4001 DISPLAY AT(4,10)SIZE(10
):"GAME OVER" !121
4005 IF SCORE>HSCORE THEN HS
CORE=SCORE !050
4010 DISPLAY AT(8,12)SIZE(6)
:" HIGH" :: DISPLAY AT(9,12)
SIZE(6):"SCORE-" !221
4020 DISPLAY AT(12,12)SIZE(7
):HSCORE !221
4030 DISPLAY AT(19,1):" PR
S 1 FOR GAME,2 TO EXIT" !049
4040 CALL KEY(0,K,S):: CALL
COLOR(2,RND*12+2,1):: IF S=0
THEN 4040 !209
4045 CALL COLOR(2,2,1)!172
4050 IF K=49 THEN SCORE=0 ::
DMX=3 :: GOTO 720 !111
4060 IF K=50 THEN 4070 ELSE
4040 !075
4070 CALL CLEAR :: END !222
5000 SUB MAKEMAZE !188
5010 CALL CLEAR !209
5020 CALL HCHAR(1,3,42,28)::
CALL HCHAR(18,3,42,28)!027
5030 CALL VCHAR(1,3,42,18)::
CALL VCHAR(1,30,42,18)!045
5040 CALL HCHAR(4,6,42,3)::
CALL HCHAR(4,11,42,12)!220
5050 CALL HCHAR(4,25,42,3)!1
76
5060 CALL HCHAR(4,14,42,6)::
CALL HCHAR(15,14,42,6):: CA
LL HCHAR(14,16,42,2)!1225070
CALL HCHAR(15,6,42,6):: CAL
L HCHAR(15,22,42,6)!025
5090 CALL HCHAR(12,3,42,3)::
CALL HCHAR(12,28,42,3)!016
5100 CALL HCHAR(7,14,42,6)::
CALL HCHAR(10,14,42,6):: CA
LL HCHAR(11,16,42,2)!1175110
CALL HCHAR(9,6,42,3):: CALL
HCHAR(9,25,42,3)!186
5120 CALL VCHAR(7,6.42,3)::
CALL VCHAR(9,8,42,4):: CALL
VCHAR(9,11,42,4)!154
5130 CALL VCHAR(7,27,42,3)::
CALL VCHAR(9,25,42,4):: CAL
L VCHAR(9,22,42,4)!076
5140 DISPLAY AT(8,12)SIZE(6)
:"DOMINO" !127
5145 CALL VCHAR(7,6,42,3)!14
3
5147 DISPLAY AT(24,1):"@1982
M.C. SUMNER" !131
5150 DISPLAY AT(9,12)SIZE(6)
:"FACTOR" :: SUBEND !163
6000 SUB PATC !149
6010 A=A+1 :: IF A=2 THEN A=
0 !163
6020 CALL PATTERN(#2,102+A,
```

(See Page 17)

DOMINEX —

(Continued from Page 16)

```
3,102+A,#4,104+A):: CALL PAT
TERN(#5,104+A,#6,104+A,#7,10
4+A)!149
6030 SUBEND !168
7000 SUB TUMBLE(T)!249
7010 CALL POSITION(#1,Y1,X1,
#2,Y2,X2,#3,Y3,X3)!014
7020 IF Y1<Y2 AND X1<X2 THEN
CALL MOTION(#2,-2,-2):: GO
TO 7060 !075
7030 IF Y1<Y2 AND X1>X2 THEN
CALL MOTION(#2,-2,3):: GOTO
7055 !214
7040 IF Y1>Y2 AND X1<X2 THEN
CALL MOTION(#2,2,-2)!090
7050 IF Y1>Y2 AND X1>X2 THEN
CALL MOTION(#2,2,2)!153
7055 IF T<2 THEN SUBEXIT !23
3
7060 IF Y1<Y3 AND X1<X3 THEN
CALL MOTION(#3,-2,-2):: SUB
EXIT !071
7070 IF Y1<Y3 AND X1>X3 THEN
CALL MOTION(#3,-2,2):: SUBE
XIT !134
7080 IF Y1>Y3 AND X1<X3 THEN
CALL MOTION(#3,2,-2):: SUBE
XIT !134
7090 IF Y1>Y3 AND X1>X3 THEN
CALL MOTION(#3,2,2)!156
7100 SUBEND !168
8000 SUB SUBPOINTS(SSCORE)!1
17
8010 SSCORE=SSCORE+1 :: DISP
LAY AT(8,12)SIZE(6):SSCORE !
084
8030 SUBEND !168
9000 SUB THUNK !248
9010 SUBEND !168
10000 SUB PATT !166
10010 A=A+1 :: IF A=2 THEN A
=0 !163
10020 CALL PATTERN(#1,102+A,
#2,102+A,#3,102+A)!126
10030 SUBEND !168
10040 SUB BMUSIC !050
10060 CALL SOUND(200,262,5):
: CALL SOUND(200,330,5):: CA
LL SOUND(200,294,5):: CALL S
OUND(200,349,5):: CALL SOUND
(200,330,5)!169
10070 CALL SOUND(200,392,5):
: CALL SOUND(200,349,5):: CA
LL SOUND(200,440,5):: CALL S
OUND(200,392,5):: CALL SOUND
(200,494,5)!185
10080 CALL SOUND(200,523,5):
: CALL SOUND(200,523,5):: CA
LL SOUND(200,262,5):: CALL S
OUND(500,262,5)!161
10090 SUBEND !168
10100 SUB LMUSIC !060
10110 FOR A=1 TO 20 :: CALL
SOUND(-200,400-(12*A),3):: N
EXT A !221
10120 SUBEND !168
10130 SUB SMUSIC !067
10140 FOR A=1 TO 20 :: CALL
SOUND(-200,400+(12*A),4):: N
EXT A !221
10150 SUBEND !168
```

Hardware maintenance

A little Teflon, an eraser are all it takes to clean a disk drive

By MICHAEL O'DOWD

The following article was originally published in the newsletter of the 9T9 user group.—Ed.

I decided it was time to clean one of my old single-sided drives to see if would it perform any better. However, I did not want to use one of the commercially available disk cleaners.

One of my extra drives is in a box that includes a power supply, so I removed the four case screws and the bottom screws which were holding the disk drive in the box. Knowing how one can forget where a part goes when it comes time to put things together again, I made a sketch and notes of everything I did, no matter how small.

All those resistors and chips looked very formidable so, taking the bull by the tail, I removed the power plug and put a dot with a marker pen on the male and female parts of the plug . The four pins are marked at the factory, but a magnifying glass is required to see the numbers properly. The dot saves time and when replacing parts. I also removed the 34-pin cable from the edge connector — the other end goes to the disk controller card.

With the disk drive removed from the box, I removed the connections plugged into the rear of the board. There are several of them and I sketched and marked them as well. Then the two screws holding the board were removed and the board slid out from under two tags. I gazed at the unknown.

Remembering my sea-going days when I cleaned gyro compass parts with carbon tetrachloride (dangerous stuff), I decided to use alcohol. I cleaned the edge connectors with a soft rubber eraser and set the board on aluminum foil to protect it from static. Using the alcohol, I cleaned around the worm-gear mechanism, being careful not to disturb the head. I used cotton swabs and changed them frequently and cleaned the head with a new swab dipped in the alcohol. Some Teflon lubricant was applied on moving parts and I blew and sucked out dirt with a vacuum and put everything back together. To my surprise the drive functioned on all cylinders.

As you can see, the procedure is not difficult though a bit more thorough than what can be done with one of the commercial drive cleaning kits.

Handy chart keeps track of TML commands

Here is a handy chart created by Rick Kellogg to help users of The Missing Link to keep track of commands. The version on the MICROpendium disk is in the form of a text file that can be output to a printer in such a way that it can be printed to a single sheet and quarter-folded so that it can be placed in the TML disk jacket.

Index of missing link commands

COMMANDS	PARAMETER NOTES
SCREEN COMMANDS (Pages 5-6)	
CALL LINK("CLEAR")	CLEARs entire screen to bkgnd color
CALL LINK("COLOR",F,B)	F=Foreground; B=Background color
PEN COMMANDS (Pages 6-7)	
CALL LINK("PENHUE",F,B)	Changes the PEN color (1 to 16)
CALL LINK("PD") *PEN DOWN*	PENDOWN — Turn ON pixels contacted
CALL LINK("PR") *PEN REVERSE*	PENREVERSE — INVERT pixels contacted
CALL LINK("PE") *PEN ERASE*	PENERASE — Turn OFF pixels contacted
CALL LINK("PU") *PEN UP*	PENUP — Pass over pixels — NO CHANGE
WINDOW COMMANDS (Pages 7-8)	
CALL LINK("WINDOW",R1,C1,R2,C2,1)	Row 0-193; Col 0-241; 1=Frame
CALL LINK("REVWIN")	1st=Graphic OUTSIDE; 2nd=Reset normal
CALL LINK("FILL",R1,C1,R2,C2)	Use with PENHUE and PD / PR / PE
TEXT COMMANDS (Pages 8-12)	
CALL LINK("PRINT",R,C,$/N,$V)	$=String; N=Number; $V=String Variable
CALL LINK("INPUT",R,C,$/N,L,P$)	$/N->Above L=Length; P$=Prompt String
CALL LINK("CHAR",A,H$)	A=ASCII Code; H$=Hexadecimal String
CALL LINK("CHSIZE",W,H)	W=Width (1-8); H=Height (1-12)
CALL LINK("FORMAT",FC,N1,N2)	FC=Format Code; N1=xxx; N2=xxx
CARTESION COMMANDS (Pages 12-13,18)	
CALL LINK("PIXEL",R,C,F,B)	Places a pixel on the screen
CALL LINK("LINE",R1,C1,R2,C2,F,B)	Draws a LINE between the two points
CALL LINK("CIRCLE",R,C,RAD,SC,F,B)	R,C=Center; RAD=Radius; SC=Suppression
CALL LINK("BOX",R1,C1,R2,C2,F,B)	Draws a BOX between R1,C1 and R2,C2
TURTLE COMMANDS (Page 13)	
CALL LINK("TURN",ANGLE)	+ANGLE=Right; -ANGLE=Left
CALL LINK("FWD",DIST,ANGLE,1)	+ - DIST=in pixels; 1=Return Start
CALL LINK("PUTPEN",R,C,ANGLE)	Pickup and Move R=0-192; C=0-240
CALL LINK("GETPEN",R,C,ANGLE)	Returns current position in R,C,A
SPRITE COMMANDS (Pages 14-16)	
CALL LINK("CHAR",A,H$)	A=ASCII Code; H$=Hexadecimal String
CALL LINK("SPRITE",#,A,CC,R,C,RV,CV)	A=ASCII; CC=Color; RV=R Vel; CV=C Vel
CALL LINK("DELSPR",#)	(0-32) If #=0 then DELETE all SPRITES
CALL LINK("FREEZE")	Turn OFF all automatic SPRITE motion
CALL LINK("THAW")	Turn ON all automatic SPRITE motion
CALL LINK("SPRPOS",#,R,C)	Retrieve Upper Left corner values
CALL LINK("DSTNCE",#1,#2,NV)	DISTANCE between 2 SPRITES — put in NV
CALL LINK("DSTNCE",#,R,C,NV)	NV=Square of Upper Left Corner and R,C
PERIPHERAL COMMANDS (Pages 16-17)	
CALL LINK("LOADP",DSK#.xxxxxxxx,1)	TI-ARTIST Picture 1=_P only NO _C
CALL LINK("SAVEP",DSK#.xxxxxxxx,1)	TI-ARTIST Picture 1=_P only NO _C
CALL LINK("DUMP")	Single Density EPSON <FCTN><CTRL>

Begining c99 — Part 3

The if-else statement

By VERN JENSEN

In the last two articles in this series, I've covered a lot of things very briefly, in order to get you up and running with the c99 compiler as quickly as possible. Now that you know how to type in and compile a program, and are familiar with some of the similarities between c99 and Extended BASIC, we'll start taking things a little slower, examining each part of the language in detail.

But before we get started, I'd like to make available to you a new disk that I recently acquired. When I made my c99 Starter Kit available, a few people sent along disks containing programs they had written along with their $5 payment for the c99 disk set. I got two games that way, as well as a special bonus: a disk containing several TI-BASIC programs that had been converted to c99. It turns out that these programs, for the most part, had been converted to c99 with a special compiler the author had written that compiles TI-BASIC programs into c99 code. The compiler is not finished, but the author sent me a copy anyway, and I'm very impressed with it. He's currently working on improving the compiler, which includes things such as adding XB support.

Even though I can't distribute the compiler (since it isn't finished), I can distribute the disk containing the TI-BASIC programs that have been converted to c99 code. The programs range from a simple demo (just a few lines) to a full-blown mine sweeper-type game. I think this disk could be quite helpful to those new to c99, since it provides several examples of c99 programs that actually compile, as well as Extended BASIC source code that you can print out and compare with the c99 code. And those who already know c99 still might find the disk interesting. (I know I did.) In any case, the disk can be ordered from me for $3. (I'd normally charge less for just one disk, but I've decided to order 50 floppy disk mailers for this purpose, since the manila envelopes I used to ship the c99 Disk Kit wouldn't be sturdy enough to ship a single floppy disk, and charging $3 will help me to cover that expense.)

USING THE c99 COMPILER WITH A RAMDISK

Recently someone asked me if I happened to know what files need to be copied from the Funnelweb disk included with my c99 Starters Kit to a RAMdisk in order to use the c99 compiler with the Funnelweb program already set up on the person's RAMdisk. The only files you should need to copy are CC and CD (the c99 compiler), and CL (the C-Loader). Configure your copy of Funnelweb to load CC as a TI-Writer Program File, and CL as a Load/Run file. That's it!

GETTING STARTED

One of the first things you have to learn about any language is how to make decisions based upon certain conditions. This is accomplished with if-else statements. The syntax of an if-else statement in c99 is

```
if (expression)
   statement1;
else
   statement2;
```

One of the first things you have to learn about any language is how to make decisions based upon certain conditions. This is accomplished with if-else statements.

If expression is true, statement1 is executed; if it is false, statement2 is executed. Here's an example of a very basic if-else statement:

```
if (value > 99)
   value = 99;
else
   value = 1;
```

This statement sets value to 99 if it was above 99, and sets value to 1 if value was not above 99. The same code in Extended BASIC would look like this:

```
IF VALUE>99 THEN VALUE=99 ELSE VALUE=1
```

The main difference between the XB code and c99 code is that in c99, the expression (value > 99) is contained within parentheses, and each statement ends with a semicolon. If statements do not need to have an "else"; that part is optional, just like in Extended BASIC. In addition, you can add "else if"s to a statement, also just like you can in Extended BASIC:

```
if (value > 99)
   value = 99;
else if (value < 0)
   value = 0;
else if (value == a)
   value = 0;
else
   a = 1;
```

You can add as many "else if" statements to the chain as you want, ending it with an "else" statement at your option. By now you may be wondering if it is possible to execute more than one statement if a particular condition is true or false. The answer is yes. As we've mentioned before very briefly, you can "group"

(See Page 20)

BEGINNING C99 —

(Continued from Page 19)

statements together with braces. Here's an example:

```
if (a > b)
{
   a = b;
   b = 0;
   z = 1;
}
else if (b > a)
{
   b = a;
   a = 0;
   z = 2;
}
```

By using open and close braces, we were able to group three statements together, so the first group will be executed if a is larger than b, or the second group if b is larger than a. You can group together as few or as many statements as you like. In fact, you can even put zero statements within the braces, and the program will still compile just fine. This can be useful if you're going to put something there later, but want to work on other things first.

EXPRESSIONS

One interesting characteristic of the if statement is that it simply tests the numeric value of the expression. An expression such as "a > b" will have a value of 1 (true) if a is larger than b, or a value of 0 (false) if a is not larger than b. Interestingly enough, you can assign the result of an expression to a variable, so a statement like "n = a>b;"will set n to 1 if a is larger than b, or 0 if it is not.

This means that you could put variables or even numbers in an if statement, instead of an expression, since an if statement simply checks to see whether the value contained between the parentheses is true (non-zero). Here's an example of an if statement that uses a variable, instead of an expression:

```
if (n)
   a = n;
```

This will set a to n if n is non-zero. I'm not sure if it is well known, but the same thing can also be done in Extended BASIC: IF N THEN A=N;

However, I don't think Extended BASIC lets you assign the result of an expression to a variable, such as "n = a>b"; in c99 you have the flexibility to do this. You might wonder how this could be useful, but there are cases every now and then. One example would be if you wanted to make a complex (and time-consuming) comparison, and save the results of that evaluation in a variable for use in more than one subsequent if statement.

OPERATORS

So far I have used only a few types of operators in my if statements. By now you may be wondering if c99 can do everything that Extended BASIC can. Rest assured; the C language has a whole suite of operators that can be used in your flow control statements. Here is a list of them:

==	(Equal to)
>	(Greater than)
<	(Less than)
>=	(Greater than or equal to)
<=	(Less than or equal to)
!	(Not - turns a true value false and vice versa)
!=	(Not equal to; same as <> in XB)
||	(Or - FCTN-A on the keyboard)
&&	(And - shift-7 on the keyboard)

Most of those are pretty self-explanatory. The last two, the OR and AND operators, are similar to the OR and AND statements of Extended BASIC. For instance, the following statement will decrement k by 32 if k is larger than 96 and smaller than 123:

```
if (k>96 && k<123)
   k = k-32;
```

This would be the same as the following Extended BASIC code:

```
IF K>96 AND K<123 THEN K=K-32
```

I used this statement in Virus Attack after reading the keyboard to convert lower case ASCII values to upper case ASCII values, so the keys would report the same values regardless of whether the Alpha Lock key was down.

Make sure to use two symbols when using AND and OR operators! The statement "if (a>1 && b>1)" is quite different from "if (a>1 & b>1)" I won't get into the & and | operators right now. They are bitwise operators, and will probably be covered in a future article. (Bitwise operators are used to perform operations on a number in binary format, such as shifting all the bits in a number left one digit.)

Take note that the "equal to" operator ("==") is also made up of two characters. Using only one — such as in the statement "if (c = 5)" — would assign 5 to c and then execute the statement since c would be non-zero, instead of comparing c with 5 and executing the statement if the two are the same. You see, in C, the code "c = 5" *always* means "assign 5 to c," regardless of whether that code is used in an if statement or not, just like the code "c == 5" always means "compare c with 5," regardless of whether it is used in an if statement or not. Using two equal signs instead of one when comparing values is probably one of the hardest habits to break when starting to write c99 code.

The "!" — or logical negation operator — turns a non-zero operand into 0, and a zero operand into 1. So a statement such as

```
if (!value)
   statement;
```

would execute the statement if value is *not* true (that is, if value is equal to 0). The ! operator can also be used with expressions, such as

```
if ( !(a > b && c == 1) )
   statement;
```

This first evaluates the expression contained within the parentheses, and if this expression is *not* true, then the statement will be executed, since the ! operator turns a false value into a true value. However, the most common use of the ! operator is simply to de-

(See Page 21)

BEGINNING C99 —

(Continued from Page 20)

termine if a variable is zero. You may wonder why you wouldn't simply use "if (value == 0)" instead. The answer is that by using the ! operator, you can sometimes more clearly indicate the action of the if statement.

For instance, you might have a variable that keeps track of whether sounds are to be played during a game, called something like playSounds. You would assign 1 (true) to the variable if sounds are to be played during the game, or 0 (false) if they are not. Then during the game you can easily write two kinds of if statements:

```
1) if (playSounds)
      statement;
2) if (!playSounds)
      statement;
```

The first statement could be used to actually play the sounds if that option is turned on. The second statement could be used if you wanted to perform some other action when the sounds are not turned on. This statement is read as "If not playSounds, do statement." You can see how using the ! operator in this case would be a bit clearer than writing "if (playSounds == 0)".

ORDER OF EVALUATION

If statements are evaluated from left to right, and evaluation is terminated as soon as the outcome of the expression is known. For instance, in the statement

```
if (a > 99 || a < 0)
   statement;
```

The "a" is first compared with 99, and if a is larger than 99, then the statement is executed immediately, without a being compared with 0. This means that you can optimize your if statements by placing the most likely case at the beginning of the statement, where it will be evaluated first.

You can use parentheses to group expressions together. For instance, the following code executes the statement if playSounds is true (non-zero) and volume is either 1 or 2:

```
if (playSounds && (volume == 1 || volume ==
2))
  statement;
```

If the parentheses weren't used, the statement would be executed if playSounds was true and volume was equal to 1, or if volume was equal to 2. This means that if volume was equal to 2, the statement would be executed, regardless of whether playSounds was true or not. Putting in the parentheses avoids this problem. Another way of writing the example above would be

```
if (playSounds)
   if (volume == 1 || volume == 2)
      statement;
```

NESTED IF-ELSE STATEMENTS

If statements can be nested. When they are, it can sometimes be confusing which else statements match which if statements. The rule here is that each else statement matches the closest if statement that does not already have an else statement associated with it. For example, in the code below, the first else statement is associated with the inner if, and the next else statement is associated with the outer if, which I have shown by indentation:

```
if (value == 1)
   if (a == 65)
      b = 1;
   else
      b = 2;
else
   b = 3;
```

However, you must remember that the C compiler ignores indentation. A statement such as the one below would not do what you expect, if you want the else statement to be executed when the first if statement is false:

```
if (value == 1)
   if (a == 65)
      b = 1;
else
   b = 3;
```

In order to achieve the desired association, braces may be used, as in the following example:

```
if (value == 1)
{
   if (a == 65)
      b = 1;
}
else
   b = 3;
```

Now the else statement will be associated with the first if statement. As a general rule, it's a good idea to use braces with your nested if statements whenever necessary to maintain clarity, even if the braces aren't needed to make the code operate as desired.

Next installment we'll take a look at the other flow control statements, such as the for, while, do while, and switch statements. These will be easy to learn now that we've covered the if statement. Then in future articles we'll take a look at functions, pointers, arrays, definitions, and others fundamental parts of the C language. Then we'll use all these things to write a complete game in c99. After that, I'll offer some tips on debugging c99 programs, programming practices that will help you avoid bugs in the first place, as well as any special interest topics I can come up with.

CONTACT INFORMATION

If you have any questions or comments, feel free to e-mail me at Jensen@lafn.org, or write to me at Vern L. Jensen, 910 Linda Vista Ave., Pasadena, CA 91103.

XBASIC program does the math when balancing your checkbook

By W. LEONARD TAFFS
Southwest 99ers

The purpose of this program is to provide a printout simulation of a two-column checkbook register as a means of verifying one's checkbook arithmetic. Line by line comparison between printout and checkbook can locate any problem areas. When the entire checkbook has been entered, the printout might be valuable if the check register is lost.

Version 3 is an improved version of CHECKPRVR2 with more options:

• A "Help" menu, callable from the main entry screen;
• Previous entries are kept on screen (they can be cleared);
• Two format options for D/V80 output files are provided;
• Printer can be flagged on/off within the program;
• Transaction number/output file/printer flags are displayed;
• Check/deposit entries with input protocol errors are refused.

A brief extract of the simulated register shows that abbreviations have to be used for the Payee/Deposit and transaction explanation columns to keep printed lines within a 96-column limit. This article shows 70 characters of the print-out line format (See Fig. 1).

The first column (TRNS) is the transaction number, automatically provided by the computer. Two entries are shown, a check entry and a deposit. Note the "D" as first character in Deposit and the minus sign in AMT.

The TOT column is a running checkbook balance.

The program is relatively simple and easy to use. Every check and deposit entry uses the same seven input prompts. The first five are for Check number (or Deposit number), Date, Payee, Amount, and Code/Notes. The sixth prompt asks you to verify your entries with Y/N, and the seventh prompt "Make Note? (0/1)" (distinguished by a telephone-chirping sound) is a CALL KEY line that serves three important functions:

1. It halts the program giving you time to review the entry you just entered and printed or saved to disk;
2. You can enter a memo about the entry which will be printed and/or saved to disk as the next record;
3. You can access the CALL KEY options available in this program. Each prompt allows a limited number of input characters (See Fig. 2).

The program uses "0/1" protocol for user response unless prompted differently — a 1 meaning yes and zero meaning no.

Throughout this program you are safe in entering uppercase letters for commands, even if some critical CALL KEYs have been programmed to accept lowercase as well. Both checks and deposits are entered using the same succession of prompts. Payee, in the case of deposits, can be used to identify the source of the deposit. When entering deposits, you must use the letter "D" as the first character of its number at the check number: prompt. The deposit amount must be prefixed with a minus sign. A warning appears between the payee and amount prompts to this effect once the I.D. source of the deposit is entered at the payee prompt. An attempt to enter a deposit without these directions will be refused. "Error in Entry" appears in same warning spot and you are returned to redo.

The amount column accepts up to nine characters. Two of these must be used for the minus sign and decimal point. The maximum figures the program can process must not exceed -99999.99 and any entry can only use seven actual digits. Check amounts are processed as positive amounts and deposit amounts are processed as negative. User must supply any needed minus signs. If you need to have this program deal with larger amounts, it can be done by changing the right program lines.

RUNNING THE PROGRAM

The opening screen presents the first input prompt at foot of the screen

(See Page 23)

Fig. 2

```
^^^^^^^^^^^^ CHECK #: ^ Limited to 4 characters.
^^^^^^^^^^^^ DATE: ^^^^^^^ " ^^ "^ 4 ^^^^ "
^^^^^^^^^^^^ PAYEE: ^^^^^^ " ^^ "^ 6 ^^^^ "
^^^^^^^^^^^^ AMT: ^^^^^^^^ " ^^ "^ 9 ^^^^ " (- sign, decimal point and
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ no more than 7 digits)
^^^^^^^^^^^^ CODE/NOTES: ^ " ^^ "^ 24 ^^^ "
```

Fig. 1

```
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ Reg. P.17
=========================================================================
|TRNS | CHK# | DATE |PAY/DEP |^^ AMOUNT^ |^^ TOTAL^^ |CODE/EXPLANATION
___________________________________________
|^ 34 | 1884 | 0127 | USP^^^ |^^^^ 16.60 |^^^ -16.60 |053 POSTAGE/AGO
___________________________________________
```

CHECKBOOK REGISTER —

(Continued from Page 22)

(See Fig. 3).

For now, bypass the first four prompts: Enter zero for printer prompt, press Enter, press Enter at the date prompt and enter zero and press Enter at next two prompts (Correction Factor and Forw'd Bal.). A screen display of a few command options that can be entered at the check number prompt appears. Press Enter to go on to the main checkbook entries screen (See Fig. 4).

PRINTING THE HELP MENU

Before making any check/deposit entries, try entering "S" at the check number prompt. If you entered a lowercase "s" by mistake, press Enter through all remaining entry prompts, enter "N" for no at the bottom line prompt to return you to check number prompt. Entering "S" here (and pressing Enter) brings you to the help menu. If your printer is handy, turn it on and press "P" to get a printout of the help screen. You don't have to set the program's printer flag for this. Having a copy will save you having to go to the help menu at every turn while getting used to the program. If you do not have a printer, the help screen is shown at the end of this article.

Fig. 3

```
^^^^^^^^^^^^^^^^^^^ | ^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ | Set Draft 96 for Printer Use|
^^^^^^^^^^^^^^^^^^^ | ^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ | USE PRINTER? (0=NO 1=YES)^^ |
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^^^^^^^^ (OPENING SCREEN)
```

Fig. 4

```
^^^^^^^^^^^^^^^^^^^ ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^ | ^^ CHECK REGISTER PROVER^^^ |
^^^^^^^^^^^^^^^^^^^ | ^^^^^^^^^^^ V.3 ^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ | By W.Leonard Taffs, SW99ers |
^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ |^ Use "-" sign for Deposits^ |
^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ | CHECK #:^^^^^ Last#:^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ | DATE:^^^^^^^^ OF 0^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ | PAYEE:^^^^^^^^ TOT 0^^^^^^ |
^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ | AMT:^^^^^^^^^^ T+CF 0^^^^^^ |
^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ | CODE/NOTES: ^^^^^^^^^^^^|^^ |
^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ | Code^^^^^^^^^^^C.F.0 ^^^^^^ |
^^^^^^^^^^^^^^^^^^^ | Rpt.P. 0 ^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ | Reg.P. 0 ^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^ | TR 0 ^^^^^^^^^^^^^^^ PR 0 ^ |
^^^^^^^^^^^^^^^^^^^ | CK^ DP ^^^^^^^^^^^^^ PCT 0^ |
^^^^^^^^^^^^^^^^^^^ ^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^^^^^^^^ (MAIN WORKING SCREEN)
```

CALL KEY SUMMARY

Use "P" to Reset Printer Flag
Use "A" to advance main Report Page Number
Use "C" to adjust Correction Factor
Use "N" to advance Check Register Page Number
Use "O" to open Output File
Use "S" to view commands
* Use "E" to clear the screen (not shown in help screen).

The acronym "PACNOSE" can remind you of the CALL KEY command letters. One additional letter "M" is only used at the check number prompt to jump to the Make Note line which is a CALL KEY program line prompt. You will arrive at this after every check or deposit entry has been made and you press "Y" to approve the entries. No cursor will be blinking at this prompt. You'll always know you're at this prompt when you hear the short telephone-like chirping sound.

NOTE: If you have not jumped from the check number to the Make Note line (using the "M" command), then the variables of any resident entries on screen at the time are not cleared.

Press Enter to return from the help screen to the main entry screen and try some of the commands. Enter "M" as a check number entry to take you to the CALL KEY command line (Make Note). The keys "A" and "N" will not do anything yet if the printer is not activated. (See "Paging" at the end of this article.)

Make a check entry now. Press "Y" to accept your entries, press Enter, then enter zero for no at Make Note prompt and you are returned for the next entry. Note that your previous entries still appear on the screen. Any time you wish to clear your entries from screen, erase your check number entry, enter the letter "M," press Enter and enter "E" at Make Note prompt. You are returned to the entry screen and will see that the five inputs have been cleared. This may work slightly differently if the printer is active. In such case nothing may appear to happen when you enter "E." It has now become a double-key instance. Having pressed "E," now enter zero and press Enter. This should clear your screen display.

Now enter a deposit, prefacing the deposit number with the letter "D" (at the check number prompt) and inserting a minus sign before the deposit amount, as mentioned previously. If done correctly, the total balance should now reflect a change due to this deposit. Enter the "C" command. You will be prompted "Adjust CF" (Correction Factor). Enter a figure and press Enter.

Look at CF on the screen to the right of the Code/Notes line. What you just entered is added to

(See Page 24)

CHECKBOOK REGISTER —

(Continued from Page 24)

this total. Look above the CF To the right of the amount line is "T+CF." This shows the running total (TOT) adjusted to a true balance — the TOTal plus the Correction Factor.

After experimenting with a few entries, end the program by entering "END" as the check number. A summary appears on screen of the total number of checks and deposits entered, total amounts of checks and deposits, correction factor if any, and the total balance as adjusted after any correction factor. All this is printed if the printer was activated, and totals only saved to a file if an outfile had been opened. You can return to the screen by pressing "R," or any other key to end the program. "R" returns you to the entry screen with the last entry still displayed.

ADDITIONAL PROMPTS

If you elect to activate the printer at program startup, additional prompts will appear. After printer and date prompts, "Use Register Paging? (0/1)" appears. If you enter 1, you are asked "Starting Register Page?" Once a number is entered, the printer must be online. "Want Pg Header? (0=No 1=Yes)" is next — 1 for yes brings up a header prompt to enter a page heading. After entering your header text, the "enter full page" prompt waits for your input. Once a page is entered, the next Open Output File prompt gives you the option to open a D/V80 file. You are prompted for a filename and disk (if 1 was entered). Your filename and disk are displayed with "O.K.? (0/1/2)."

Possible responses are:

0 Chance to re-enter filename and drive number.

1 O.K. to go on to next output file prompt.

2 Escape back to main working screen.

If you enter 1, you are asked:

0) OutFile Same as PRNTR? (outfile record in printer format).

1) Plain Format (no column lines).

Entering a zero at this prompt will send an 80-character columned line to the output file, formatted similarly to the printer format. A 1 will send entry information to an output file in this format (See Fig. 5).

Correction factor and the forward balance prompts come next. After entering these (0 for factor if none), you are returned to the main screen to start your entries where you will see your outfile disk and name displayed at the bottom of the screen. Now all CALL KEY letters are active. Your starting pages should appear in the legends near the bottom of your screen, flags should reflect OF 1 and PR 1.

If you want to advance to your next report page before you have reached 80 lines, enter "A" at the Make Note CALL KEY line. The printer will draw a double line to mark the end of the page, and a prompt "Adjust For Next Page + PETC" waits for you to advance your paper to the top of the next form. After doing so, press Enter and the printer will print the header on your new page.

If you are ready to start the next checkbook register page, e. "N" at the CALL KEY point and the printer will print a new check register heading incrementing its page number by one. This is what allows you to separate your register pages in the report. Watching the PCT counter on your entry screen will give you an idea of how far down you are getting on the report page (80 being the limit).

CORRECTION FACTOR

Correction factor is entered during program startup or by selecting "C" as CALL KEY input at Make Note at any time while making entries. This adjusts the running balance total to agree with your checkbook balance. The reason this option is included is that once a register error is discovered, the running balance will be at variance with the check register balance. This makes comparisons of screen or printout much more difficult. If you have 18 register pages which include only a single mistake or so, in making comparisons you have to carry the difference between balances in your head, once a discrepancy is found. Being able to insert a correction factor for checkbook mistakes as you go along makes comparison much more practical. At the end of your entries, the program allows for this correction factor in summary totals. The CF shown near the lower right portion of your screen shows the sum of any correction factors you input.

The forward balance is entered as a positive number. If printer is activated at program startup, even if you opt for no page numbering, header, etc., this forward balance is printed in the rep header.

The printer reminder screen tells you to set your printer for draft 96 to accomodate 96 characters per page line or your print lines will spill over. The program sets the printer for 80 lines per page.

SCREEN RETENTION OF PREVIOUS ENTRIES

Once a check or deposit has been entered and you have approved it, the input prompt information will still remain on screen as you begin to make the next entry. Version 3 of this program will return you to the first prompt again if you have made either of these errors:

1. Attempted to make a deposit but omitted the "D" as the first character in the check number prompt, or you failed to include a minus sign in front of deposit amount.

2. When entering a check after having just previously entered a deposit and failed to delete or type over a resident minus sign still on screen. The warning to use a "-" sign, seen when making deposits, will be replaced by "Error in Entry." At any time you want to skip entries to the CALL KEY line, stepping through prompts may record your on-screen data a second time. Use "M" routine from the check number prompt to avoid this.

If you have approved an entry only to discover you approved a mistake, this is a time to use the correction factor option. At the next check number prompt, backspace or use FCTN-3 to clear it and enter "M." At the CALL KEY, enter "C" to get your correction factor prompt. Enter the amount of your err with a minus sign if the error was in a check, or

(See Page 25)

Fig. 5

```
^^^^^^^^^^^( CHK#|DATE|P/Dp| Amount |Code/Expl....)<- &not printed
^^^^^^^^^^^^ 0000 0000 0000^ 00000.00 000 TEXT.....(Output Record)
```

CHECKBOOK REGISTER —

(See Page 15)

with no sign if the amount was a deposit. Then you can redo the correct entry.

The Code/Notes prompt: allows inputting a maximum of 24 characters in this entry as code and transaction description. If you created some output files for later merging, you might want to use codes at the start of the Code/Notes entry. The PRINT USING statements in this program limit the number of characters allowable in this prompt. The "!" character in column 25 of the Code/Notes prompt show you the boundary you must remain within or the printout will print all asterisks for this column. Reprogramming for longer lines would require condensed print, which my wife says is too hard to read.

This program treats checks as positive amounts and deposits as negative amounts because the user is most likely to enter far more checks than deposits. This saves much shifting for the "-" key. the "D" first character in the deposit number column and the "-" sign of the deposit amount are used by the computer to distinguish check entries from deposit entries, calculate valid running balances, and tally the total number of checks and deposits and their amounts. A positive total (running balance) indicates you have not overdrawn your account — a negative balance indicates you have.

Fig. 6

```
^^^^^^^^^^^^^^^^^^^^ | Code^^^^^^^^^^^C.F.0 ^^^^^^ |
^^^^^^^^^^^^^^^^^^^^ | Rpt.P. 0 ^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^^ | Reg.P. 0 ^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^^ | TR 0 ^^^^^^^^^^^^^^^^^PR 0^^ |
^^^^^^^^^^^^^^^^^^^^ | CK^ DP ^^^^^^^^^^^^ PCT 0^ |
^^^^^^^^^^^^^^^^^^^^ ----------------------
^^^^^^^^^^^^^^^^^^^^^^^^ (From Main Working Screen)
```

END OF FILE FORMAT

The last two lines of your output file will contain the "}" and "~" characters as their first visible character. These are entered by the computer in case files are to be merged and sorted. Don't remove them if you plan any sorting of these files.

When using Make Note, enter zero to go on to next entry. Pressing Enter will work also. If you enter a one when the printer has not been activated, you will be returned to the entry menu. But if the printer is activated, the prompt "Enter Notes" will appear. This option is included to allow you to enter any comment you wish while entering transactions. This will be printed after the last entry you printed and to the output file if open. Should you press Enter to skip entering a memo, your printer may still want to space down another line. (This 0/1 option is the only CALL KEY option!)

COUNTER AND FLAG DISPLAYS

Look at the bottom of your screen, below input prompts. You will see (See Fig. 6).

Code displays the first three characters of the explanation you entered (to view codes if user entered them). To the right on this line is the total of any correction factors entered.

"Rpt" is "Report" page — the number of your 8 1/2 x 11 printer page.

Reg.P. is the page of your check register.

TR is the abbreviation for transaction. The computer automatically counts your entries. On the right side of this line is PR, a printer flag to let you know whether you opted for printer or not. If it shows a zero, printer is not active; a one means it has been activated.

CK and DP show the last check number and last deposit number you entered. Do not rely on "Last#:" (appears to the right of the check number input) for this, as it may or may not show the last check/deposit number made. Last# will show whatever was last entered — which may have been a command having been entered there. This command will replace what had been shown previously. Always use the bottom screen line to find what your last check or deposit number was.

To the far right of this bottom line is PCT, which is the printer line counter — active only when the printer is in use. This counter is useful in letting you know when you are getting close to 80

(See Page 26)

HELP SCREEN

```
^^^^^^^^^^^^^^^^^^^^ .-------------------------------.
^^^^^^^^^^^^^^^^^^^^ |ALL WORD COMMANDS ENTERED AT|
^^^^^^^^^^^^^^^^^^^^ |^^^ CHECK#: INPUT PROMPT:^^ |
^^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^^ |Enter "END" to End Program^ |
^^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^^ |Enter "BACK" to Re-Run prg ^|
^^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^^ |Enter "XXX" to Clear Var's ^|
^^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^^ |Enter "M" to get to the CALL|
^^^^^^^^^^^^^^^^^^^^ |^^^^^^ KEY Line ^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^^ |============================|
^^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^^ |CALL KEY COMMANDS: Note: ^^ |
^^^^^^^^^^^^^^^^^^^^ |Enter all CALL KEY Letters^ |
^^^^^^^^^^^^^^^^^^^^ |at the "Make Note?" prompt^ |
^^^^^^^^^^^^^^^^^^^^ |^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |
^^^^^^^^^^^^^^^^^^^^ |Use "P" to Re-Set PRNTR Flg |
^^^^^^^^^^^^^^^^^^^^ |Use "A" to advance main pg. |
^^^^^^^^^^^^^^^^^^^^ |Use "C" to adjust Corr/Fact |
^^^^^^^^^^^^^^^^^^^^ |Use "N" to advance Reg. Pg. |
^^^^^^^^^^^^^^^^^^^^ |Use "O" to Open Output file |
^^^^^^^^^^^^^^^^^^^^ |Use "S" to view this screen.|
^^^^^^^^^^^^^^^^^^^^ |Press <ANY> key to Return...|
^^^^^^^^^^^^^^^^^^^^ |-------------------------------|
^^^^^^^^^^^^^^^^^^^^^^^^ CHECKPRVR2 HELP SCREEN
```

CHECKBOOK REGISTER —

(Continued from Page 25)

lines, if you want to judge whether to advance to the next register page or report page. The page is advanced automatically after 80 lines, otherwise.

To the right of the payee and amount lines there appears TOT and T+CF. These show the running totals. TOT is the running balance total without consideration of any correction factor. T+CF shows the true balance when correction factor adjustments are considered.

OF on the right side of screen on the date prompt line is a flag to show if you have opened an output file. If so, it displays 1 and your output file name should appear on bottom line of your screen if not temporarily blanked by other prompts. The output file is closed automatically when you end program entries. After opening a file and being returned to the entries screen, you will see to the right of the date line the flag, OF, displaying a 1 after it, reminding you an output file is open.

PAGING

Starting page numbers are set at program startup. Use "A" to advance the number of full report page. Use "N" to increment the register page number. Both letters are active if the printer is active. Both these numbering systems increment by one each time you use these letter commands. Note that by advancing the report page number you are expected to move your printer paper to the top of the form for the next new page. The program halts to inform you of this. Register paging can be incremented at any time — watching the PCT flag on your entry screen will give you an idea of how you are doing. A register page does not have to be completed by the end of any report page as they will just be continued on the next report page, but you may wish to keep register pages from being split up this way. The register page number will not increment until the program gets the "N" order from you.

One important caution is that 0/1 input *requires* a 0 or 1 input by the user. Don't rest your finger on the Enter key. As you probably know, enough such pressings of Enter at such a prompt will crash your program. This program does not have protection against this.

I would appreciate it if anyone experiences difficulty using this program. Anyone is welcome to distribute this program as long as no fee is made for it and credit is given to the SouthWest 99ers, Tucson, Arizona.

Taffs can be reached at 4124 E.First St., Tucson, Az., 85711; email: wltakts@@azstarnet.com; phone: 520-795-4148.

CHECKPRVR2

```
1 REM [CHECKPRVR2] 2-25-97
by W.Leonard Taffs,SW99ers
!026
2 !!131
3 ! MAIN PAG'G (ADV) 690/750
REGISTER PAGING (PQ) 680
Use "E" to clear prompts
!068
4 !!131
5 ! FWD BAL 230
OPEN OUTPUT FILE #3 1060
!040
6 !!131
7 ! AUTHOR's Address:
W. Leonard Taffs
4124 E. First Street
Tucson, Az. 85711
(Tel. 520-795-4148) !0
77
8 !!131
9 ! CONTROL CALL KEY L620
FLAGS: PR=PRINTER
RP=REG.PAGING
CF=CORR.FACTOR
TOT=RUNNING BAL.
!147
10 !!131
100 GOTO 170 !249
110 ACF,ADV,AM,AM2,AM2$,AM3,
AM3$,AMT,AMT$,AMU,AMU$,AMX,A
N$,C1$,C2$,C3$,C4$,C5$,C6$,C
CD$,CD$,CF,CH,CHK,CK,CK$,COR
R$,CT !028
120 DATE$,DIF,DP,DPS,DPS$,DP
T,DSC$,DT$,E$,FN$,FW$,FWD,FW
D2,FX,HD,HDR$,K,K$,L,NRP,NT$
,OF,OF$,OK,OO,OOF !224
130 P,PCT,PG,PP,PP$,PQ,PR,PS
,PY,PY$,PYA,PYA$,PYB,RP,S,S1
$,S2$,S3$,S4$,S5$,S6$,S7$,SA
,SCK$,SDP$,SET$,TAMT,TOT,TOT
3 !178
140 W1$,W2$,W3$,W4$,W5$,W6$,
W7$,X$,YN$,Z1,Z1$,Z2$,Z3$,Z4
$,Z5$,Z6$ !065
150 CALL CLEAR :: CALL COLOR
:: CALL KEY :: CALL SCREEN
:: CALL SOUND !049
160 !@P- !064
170 CALL CLEAR :: GOSUB 1520
:: DISPLAY AT(3,1):"CHECK R
EGISTER PRINT PROGRAM": :"By
W. Leonard Taffs, SW99ers":
: :: DISPLAY AT(4,13):"V.3"
!121
180 DISPLAY AT(9,1):"This pr
ogram allows you to enter
checks and deposits from yo
ur check register andprints
out in format similar" !011
190 DISPLAY AT(13,1):"to a r
egister balance column" :: D
ISPLAY AT(15,1):"This will h
elp you prove if your regist
er arithmetic is correct." !
244
200 DISPLAY AT(21,1):"Set Dr
aft 96 for Printer use" :: E
$="|" !002
210 W1$="|TRNS" :: W2$="| CH
K#" :: W3$="| DATE" :: W4$="
|PAY/DEP" :: W5$="| AMOUNT
" :: W6$="| TOTAL " :: W
7$="| CODE/EXPLANATION" !210
220 INPUT "USE PRINTER? (0=N
O 1=YES) ":PR :: IF PR THEN
OPEN #1:"PIO",VARIABLE 96 !1
36
230 PRINT :: INPUT "ENTER DA
TE (Optional) ":DATE$ :: PRI
NT :: IF PR THEN INPUT "Use
Register Paging? (0/1) ":RP
:: IF RP THEN PP$="Reg. P."
!082
240 IF RP THEN PRINT :: INPU
T "Starting Register Page? "
:PQ :: PRINT :"Have Printer
Set On-Line...." !060
250 IF RP THEN OPEN #2:"PIO"
```

(See Page 27)

CHECKPRVR2 —

(Continued from Page 26)

```
:: PRINT #2:CHR$(27);CHR$(4
8):: PCT=PCT+1 !011
260 IF RP THEN CLOSE #2 !110
270 IF PR THEN PRINT :: INPU
T "WANT PG HEADER? (0=NO 1=Y
ES)":HD :: IF HD THEN PRINT
:: INPUT "HEADER: ":HDR$ ::
PRINT !054
280 FW$="Forw'd Bal." :: IF
HD THEN INPUT "ENTER FULL PA
GE #: ":PG :: PRINT :: INPUT
"OPEN OUTPUT FILE? (0/1) ":
OOF :: IF OOF THEN GOSUB 122
0 !187
290 PRINT :"CORRECTION Facto
r: " :: ACCEPT AT(23,20)VALI
DATE("0123456789.-")SIZE(-9)
:CF :: TOT3=TOT3+CF !107
300 PRINT :: INPUT "Forw'd B
al. (+) : $ ":FWD :: FWD2=FW
D :: TOT=FWD :: TOT3=TOT3+FW
D :: CALL CLEAR :: !088
310 CALL CLEAR :: IF PR THEN
DISPLAY AT(15,1):"SURE DRAF
T IS SET FOR 96?" :: CALL KE
Y(0,K,S):: IF S<1 THEN 310 :
: PRINT : :"PRINTING HEADER.
...." :: GOSUB 1100 :: CALL
CLEAR !248
320 DISPLAY AT(3,1):"A Runni
ng Balance will show to the
Right of AMOUNT InputPrompt.
------" !172
330 DISPLAY AT(8,1):"TO CLEA
R All FIGURES at any time,
Enter ""XXX"" at the CHECK
Input Prompt.      -----
" !049
340 DISPLAY AT(14,1):"TO Re-
START program at any time,
Enter ""BACK"" at the CHEC
K # Input Prompt. ----
---" !124
350 DISPLAY AT(20,1):"To TER
MINATE program, ENTER":""EN
D"" at CHECK Input Prompt.
-----" :: DISPLAY AT(
24,1):"Press <ANY> Key to Co
ntinue." !118
360 CALL KEY(0,K,S):: IF S<1
 THEN 360 !175
370 CALL CLEAR :: DISPLAY AT
(2,4):"CHECK REGISTER PROVER
":"   V.3":"By W.Le
onard Taffs, SW99ers": :" Us
e ""-"" sign for Deposits" !
127
380 DISPLAY AT(8,1):"CHECK #
: ";Z1$;"   Last#:";CK$ ::
DISPLAY AT(10,1):"DATE:
";Z2$ :: DISPLAY AT(12,1):"P
AYEE: ";Z3$;TAB(16);"TOT "
;TOT !009
390 DISPLAY AT(14,1):"AMT: "
;Z4$;TAB(16);"T+CF";TOT3 ::
DISPLAY AT(10,15):"OF";OF !1
56
400 DISPLAY AT(16,1):"CODE/N
OTES: |":Z5$ !00
3
410 DISPLAY AT(19,1):"Code "
;SET$;TAB(16);"C.F.";CF:"Rpt
.P.";PG:"Reg.P.";PQ:"TR";CT;
TAB(22);"PR";PR:"CK ";Z1$;"
DP ";SDP$;TAB(22);"PCT";PCT
!027
420 IF OF THEN DISPLAY AT(24
,1):"outfile: ";OF$ !233
430 ACCEPT AT(8,10)SIZE(-4):
CK$ :: IF CK$="" THEN 440 EL
SE Z1=ASC(CK$):: IF (Z1>=48)
*(Z1<=57)THEN Z1$=CK$ ELSE I
F Z1=68 THEN Z6$=CK$ !219
440 IF CK$="BACK" THEN CALL
CLEAR :: DISPLAY AT(12,5):"R
e-RUNNING Program!" :: RUN !
136
450 IF PR THEN IF CK$="XXX"
THEN AMT,CK,CHK,DPS,FWD,PR,T
OT=0 :: CLOSE #1 :: CALL CLE
AR :: GOTO 220 ELSE 460 !012
460 IF PR=0 THEN IF CK$="XXX
" THEN AMT,CK,CHK,DPS,FWD,PR
,TOT=0 :: CALL CLEAR :: GOTO
220 !147
470 IF CK$="END" OR CK$="end
" THEN 930 !222
480 IF CK$="M" OR CK$="m" TH
EN 750 ELSE IF CK$="S" THEN
GOSUB 1360 !182
490 ACCEPT AT(10,10)SIZE(-4)
:DT$ :: Z2$=DT$ !062
500 ACCEPT AT(12,10)SIZE(6):
PY$ :: PYA$="      " :: PY=L
EN(PY$):: PYA=LEN(PYA$):: PY
B=PYA-PY :: PY$=PY$&SEG$(PYA
$,1,PYB):: Z3$=PY$ !160
510 IF SEG$(CK$,1,1)="D" THE
N DISPLAY AT(13,3):"Use (-)
sign!" !113
520 ACCEPT AT(14,6)VALIDATE(
"0123456789+-.")SIZE(-9):AMT
$ :: FX=POS(AMT$,".",1):: AM
3=LEN(AMT$):: AM3$=SEG$(AMT$
,FX+1,AM3)!068
530 IF POS(AM3$,".",1)THEN 5
20 !136
540 Z4$=AMT$ :: IF (SEG$(CK$
,1,1)<>"D")*(SEG$(Z4$,1,1)="
-")+(SEG$(CK$,1,1)="D")*(SEG
$(AMT$,1,1)<>"-")THEN DISPLA
Y AT(13,3):"Error in Entry!"
:: GOTO 430 !118
545 Z4$=AMT$ :: IF AMT$="" T
HEN 550 ELSE AMT=VAL(AMT$)!0
22
550 ACCEPT AT(17,1)SIZE(28):
NT$ :: Z5$=NT$ :: DISPLAY AT
(23,1):"O.K.? Press 'Y'-Yes
'N'-No" !192
560 IF NT$<>"" THEN CCD$=SEG
$(NT$,1,3):: IF (ASC(CCD$)<4
8)+(ASC(CCD$)>57)THEN SET$=S
ET$ ELSE SET$=SEG$(CCD$,1,3)
!185
570 ACCEPT AT(23,28)VALIDATE
("NnYy")SIZE(-1):YN$ :: IF Y
N$<>"Y" AND YN$<>"y" THEN 37
0 :: DISPLAY AT(23,1):RPT$("
",28)!065
580 PP=PP+1 :: CT=CT+1 :: TO
T=TOT-AMT :: TOT3=TOT3-AMT :
: IF SEG$(STR$(AMT),1,1)<>"-
" THEN CK=CK+1 :: CHK=CHK-AM
T !004
590 IF SEG$(STR$(AMT),1,1)="
-" THEN DP=DP+1 :: DPS=DPS+A
MT !002
600 IMAGE "### ### #### ####
######.## ######.## #######
############ ##############
######" !135
610 ! IMAGE=79 CHARS !068
620 IMAGE ### # ### # #### #
#### # ###### # ######.## #
######.## # ##############
############ !183
630 IMAGE # ### # #### # ###
# # ###### # ######.## # ###
###.## # ##################
########## !042
640 ! 505=IMAGE FOR OF$ !189
650 ! IMAGE=86 CHARS !066
```

(See Page 28)

CHECKPRVR2 —

(Continued from Page 27)

```
660 IF RP THEN IF PCT>=80 TH
EN IF PR THEN PG=PG+1 :: PRI
NT #1: : : : : : : : :: GO
SUB 1100 !178
670 IF RP THEN 690 !083
680 IF PR THEN PRINT #1,USIN
G 600:X$,CT,CK$,DT$,AMT,TOT,
PY$,NT$ :: PCT=PCT+1 :: GOTO
750 !061
690 IF RP THEN IF PR THEN PR
INT #1,USING 620:X$,E$,CT,E$
,CK$,E$,DT$,E$,PY$,E$,AMT,E$
,TOT,E$,NT$ !097
700 IF SA THEN 730 !109
710 IF OF THEN PRINT #3,USIN
G 630:E$,CT,E$,CK$,E$,DT$,E$
,PY$,E$,AMT,E$,TOT,E$,SEG$(N
T$,1,23)&CHR$(13)!088
720 IF OF THEN PRINT #3:RPT$
("-",79)&CHR$(13):: DISPLAY
AT(24,1):"Record saved to Ou
tput File" !071
730 IF SA THEN GOSUB 1290 ::
IF OF THEN PRINT #3:AN$ !11
1
740 IF RP THEN IF PR THEN PR
INT #1:TAB(4);RPT$("-",81)::
PCT=PCT+2 !001
750 CALL SOUND(50,1600,0)::
CALL SOUND(50,1400,0):: CALL
SOUND(50,1600,0):: DISPLAY
AT(24,1):"Make Note? (0/1) "
;TOT !175
760 CALL KEY(0,K,S):: IF S<1
THEN 760 !064
770 IF (K=69)+(K=101)THEN Z1
$="" :: Z2$="" :: Z3$="" ::
Z4$="" :: Z5$="" :: Z6$="" !
Clear Screen !082
780 IF (K=80)+(K=112)THEN GO
SUB 1570 ! Re-Set Print Flag
!100
790 IF (K=83)+(K=115)THEN GO
SUB 1360 :: GOTO 370 ! View
Commands/Call Key Presses !2
34
800 IF (K=79)+(K=111)THEN OO
F=1 :: GOSUB 1210 ! Open Out
Put File !011
810 IF (OF=1)+(PR=1)+(RP=1)T
HEN IF K=48 THEN 910 ! Make
Note=0 !015
820 IF (K=67)+(K=99)THEN INP
UT "Adjust C.F.: ":ACF :: TO
T3=TOT3+ACF :: CF=CF+ACF ::
CALL CLEAR :: GOTO 370 !150
830 IF RP THEN IF (K=78)+(K=
110)THEN NRP=1 :: PQ=PQ+1 !0
50
840 IF (K=65)+(K=97)THEN ADV
=1 ! if A set main page !164
850 IF RP THEN IF (ADV=1)+(N
RP=1)THEN PRINT #1:TAB(4);RP
T$("=",81):: PCT=PCT+1 :: IF
ADV THEN 900 !037
860 IF NRP THEN IF PR THEN P
RINT #1:TAB(75);PP$;PQ:TAB(4
);RPT$("-",81):: NRP=0 :: PC
T=PCT+2 :: GOTO 910 !251
870 IF (OF=1)+(PR=1)+(RP=1)T
HEN IF K=49 THEN PRINT :: IN
PUT "Enter Notes: ":CORR
$ ELSE 760 !078
880 IF (OF=0)*(RP=0)*(PR=1)T
HEN PRINT #1:TAB(10);CORR$ !
234
885 IF PR=1 THEN PRINT #1:TA
B(10);CORR$ !120
890 IF OF THEN IF CORR$<>""
THEN PRINT #3:TAB(5);"Note:
"&CORR$&CHR$(13):: DISPLAY A
T(24,1):"Note Saved to Outpu
t File..." !119
900 IF RP THEN IF PR THEN IF
ADV THEN PRINT :: INPUT "AD
JUST FOR NEXT PAGE + PETC.":
K$ :: PG=PG+1 :: PQ=PQ+1 ::
GOSUB 1100 ! Page Header !18
2
910 CALL CLEAR :: IF SEG$(CK
$,1,1)<>"D" THEN SCK$=CK$ EL
SE SDP$=CK$ :: GOTO 370 !024
920 GOTO 370 !194
930 REM ** END/CONT PROG **
!052
940 CALL CLEAR :: PRINT :"BE
GINNING BAL: $";FWD: :" EN
DING BAL: $";TOT:"ADD
C.F. ";CF:" CPU BAL: $
";TOT3 !146
950 IF SEG$(STR$(DPS),1,1)="
-" THEN DPT=LEN(STR$(DPS)):
DPS$=SEG$(STR$(DPS),2,DPT)E
LSE DPS$=STR$(DPS)!077
960 PRINT :"You Had:": :CK;"
Chrgs Total'g $";CHK: :DP;"C
rdts Total'g $ ";DPS$:"
---------" !135
970 DIF=DPS-CHK :: IF PR THE
N PRINT #1: :TAB(4);"There
ere:";CK;"Chrgs Tot $ ";CHK
;" and";DP;"Crdts Tot $ ";D
PS$;" Diff: $ ";DPS-CHK: :
!050
980 IF PR THEN PRINT #1:TAB(
17);"Begin Bal. was:    $";
FWD: :TAB(17);"End Bal. was
:     $";TOT:TAB(17);"Corre
ct Factor:  $";CF !082
990 IF PR THEN PRINT #1:TAB(
17);RPT$("-",26):TAB(17);"Ad
justed For'd Bal: $";TOT3: :
!112
1000 IF PR THEN PRINT #1:TAB
(17);RPT$("=",41):TAB(17);"P
rintout ";DATE$;" Diff + C.
F. = $";DPS-CHK-CF:TAB(17);R
PT$("=",41)!222
1010 PRINT :"OutFile: ";OF$
:: IF PR THEN IF OF THEN PRI
NT #1:TAB(17);"Output File [
";OF$;"] has been Created."
!076
1020 PRINT :RPT$("=",27):: I
F OF THEN PRINT #3:CHR$(125
&" "&STR$(CK)&" Chgs "&STR$(
CHK)&" "&STR$(DP)&" Crdts "&
DPS$&" =$"&STR$(TOT)&"(Fwd.B
al= "&STR$(FWD)&") +C.F. "&S
TR$(CF)!160
1030 DISPLAY AT(22,13):"Diff
:";DPS-CHK :: PRINT "
Diff + CF:";DPS-CHK+CF:RPT$(
"=",27)!093
1040 PRINT :" You have opt
ed to END!": :"USE ""R"" to
RETURN (Variableswill NOT be
cleared!) or":"ANY other ke
y to END." !012
1050 IF OF THEN PRINT #3:"~
End of File "&OF$&" "&DATE$
:: CLOSE #3 :: OOF,OF=0 !221
1060 CALL KEY(0,K,S):: IF S<
1 THEN 1060 !110
1070 IF (K=82)+(K=114)THEN C
ALL CLEAR :: GOTO 370 :: ELS
E 1080 !147
1080 PRINT :RPT$("=",28):"IT
WOULD BE NICE TO KNOW IF":
:" YOU FOUND THIS PROGRAM"
: :"USEFUL. Thanks for usin
it!": :"My Address is liste
d Line 7." !094
```

(See Page 29)

CHECKPRVR2 —

(Continued from Page 28)

```
1090 END !139
1100 REM ** COLUMN HEADING *
* !128
1110 PCT,PP,PS=0 !071
1120 IF HD THEN IF PR THEN P
RINT #1:TAB(25);HDR$&" "&DAT
E$;TAB(70);"SUMMARY Pg. ";PG
: : :: PCT=PCT+2 !111
1130 IF PR THEN PRINT #1:TAB
(64);FW$;TOT: : :: PCT=PCT+2
!015
1140 IF RP THEN IF PR THEN P
RINT #1:TAB(75);PP$;PQ :: AM
=1 :: PCT=PCT+1 !241
1150 IMAGE ### ##### ######
###### ######## ##########
########## ################
### !006
1160 IF RP THEN IF PR THEN P
RINT #1:TAB(4);RPT$("=",81)!
071
1170 IF RP THEN IF PR THEN P
RINT #1,USING 1150:X$,W1$,W2
$,W3$,W4$,W5$,W6$,W7$ !009
1180 IF RP THEN IF PR THEN P
RINT #1:TAB(4);RPT$("-",81)!
055
1190 CH=1 :: ADV=0 :: IF PR
THEN PCT=PCT+3 !248
1200 RETURN !136
1210 REM ** OPEN FILE OPT **
!039
1220 CALL CLEAR :: IF OOF TH
EN OO=1 :: GOTO 1230 ELSE RE
TURN !109
1230 PRINT :: INPUT "ENTER F
ilename: ":FN$ :: PRINT :: I
NPUT "To Disk: ":DSC$ :: OF$
="DSK"&DSC$&"."&FN$ :: PRINT
:OF$,"O.K.? (0/1/2) ": :!05
6
1240 INPUT " ":OK :: IF OK=1
THEN 1250 ELSE IF OK=0 THEN
1230 ELSE IF OK=2 THEN FN$,
DSC$,OF$="" :: IF OK=2 THEN
OF,OOF=0 :: RETURN !121
1250 PRINT :: DISPLAY AT(22,
1):"0) OutFile Same as PRNTR
? 1) (or) Plain Format " :
: ACCEPT AT(23,27):SA :: CAL
L CLEAR !098
1260 IF OK=1 THEN OPEN #3:OF
$,OUTPUT :: OF=1 :: DISPLAY
AT(24,1):"OutFile has been o
pened!" :: IF OOF THEN 750 E
LSE RETURN !221
1270 IF OOF THEN RETURN ELSE
370 !093
1280 RETURN !136
1290 ! REM ** JUSTIFY **
For OutFile option 2
!039
1300 AMT=LEN(AMT$):: P=POS(A
MT$,".",1):: IF P=0 THEN AMT
$=AMT$&".00" :: GOTO 1330 !1
85
1310 IF P THEN IF SEG$(AMT$,
AMT,1)="." THEN AMT$=AMT$&"0
0" :: GOTO 1330 !172
1320 IF P THEN AM2$=SEG$(AMT
$,P,AMT):: AM2=LEN(AM2$):: I
F AM2=2 THEN AMT$=AMT$&"0" !
216
1330 AMT=LEN(AMT$):: AMU$="
" :: AMU=LEN(AMU$)::
AMX=AMU-AMT :: AMT$=SEG$(AM
U$,1,AMX)&AMT$ !206
1340 AN$=CK$&" "&DT$&" "&SEG
$(PY$,1,4)&AMT$&" "&NT$&CHR$
(13)!003
1350 RETURN !136
1360 REM ** VIEW COMMANDS **
!079
1370 S1$="ALL WORD COMMANDS
ENTERED AT CHECK#: INPUT
PROMPT:" :: S2$="Enter ""END
"" to End Program" :: S3$="E
nter ""M"" to get to the CAL
L     KEY Line" !064
1380 S4$="Enter ""BACK"" to
Re-Run prg" :: S5$="Enter ""
XXX"" to Clear Var's" !134
1390 S6$="CALL KEY COMMANDS:
Note:" :: S7$="Enter all CA
LL KEY Letters at the ""Mak
e Note?"" prompt" !164
1400 C1$="Use ""C"" to adjus
t Corr/Fact" :: C2$="Use ""N
"" to advance Reg Pg." :: C3
$="Use ""O"" to Open Output
file" :: C4$="Use ""S"" to v
iew this screen." !214
1410 C5$="Use ""A"" to advan
ce main pg." :: CALL CLEAR :
: DISPLAY AT(1,1):S1$: :S2$:
```

(See Page 30)

1997 TI FAIRS

APRIL

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

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

MAY

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

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

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.

CHECKPRVR2 —

(Continued from Page 29)

```
:S4$: :S5$: :S3$:RPT$("=",2
8): :S6$:S7$ !128
1420 C6$="Use ""P"" to Re-Se
t PRNTR Flg" :: DISPLAY AT(1
8,1):C6$:C5$:C1$:C2$:C3$:C4$
!126
1430 DISPLAY AT(24,1):"Press
<ANY> key to Return..." !15
3
1440 CALL KEY(0,K,S):: IF S<
1 THEN 1440 :: IF (K=80)+(K=
112)THEN 1460 ELSE 1450 !090
1450 GOTO 370 ELSE RETURN !2
03
1460 OPEN #4:"PIO" :: PRINT
#4:TAB(15);"HELP SCREEN FOR
CHECKPRVR2 (V.3) PROGRAM":TA
B(15);RPT$("=",40): :TAB(10)
;S1$: :TAB(10);S2$: :!200
1470 PRINT #4:TAB(10);S4$: :
TAB(10);S5$: :!006
1480 PRINT #4:TAB(10);S3$: :
TAB(10);RPT$("=",28): :TAB(1
0);S6$:TAB(10);S7$: :TAB(10)
;C6$:TAB(10);C5$:TAB(10);C1$
:TAB(10);C2$ !070
1490 PRINT #4:TAB(10);C3$:TA
B(10);C4$:TAB(10);"Use ""E""
to Clear Entry Screen":TAB(
10);RPT$("=",29)!112
1500 CLOSE #4 :: GOTO 370 !2
22
1510 REM [CALL/BLUE] !229
1520 CALL SCREEN(5)!150
1530 FOR L=0 TO 14 !111
1540 CALL COLOR(L,16,1)!051
1550 NEXT L !226
1560 RETURN !136
1570 REM ** RESET PR FLAG **
!033
1580 DISPLAY AT(23,1):"Print
er Flag: ";PR :: DISPLAY AT(
24,1):"Enter 1 to Set 0 to
Cancel" !255
1590 CALL KEY(0,K,S):: IF S<
1 THEN 1590 !130
1600 IF K=49 THEN IF PR=0 TH
EN OPEN #1:"PIO",VARIABLE 96
:: PR=1 :: GOTO 1620 ELSE 7
50 !030
1610 IF K=48 THEN IF PR=1 TH
EN CLOSE #1 :: PR=0 ELSE PR=
0 !077
1620 DISPLAY AT(23,1):RPT$("
",56):: GOTO 750 !111
1630 RETURN !136
```

System of the month

A bedroom computer

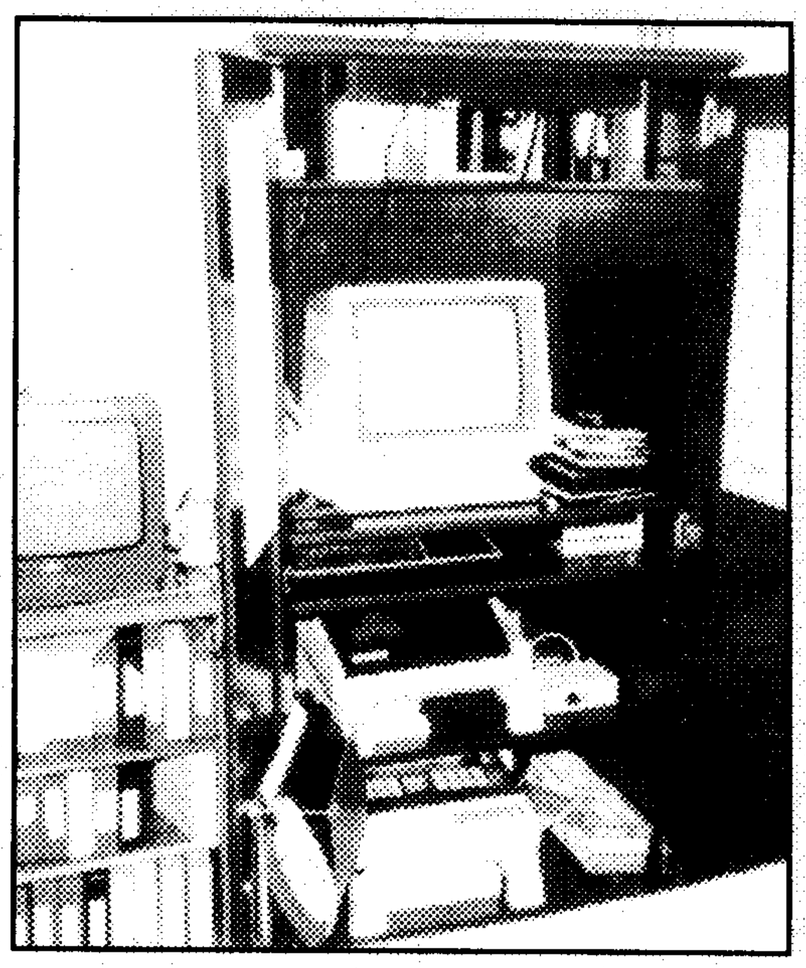
This system comes to us from Richard R. Hay of West Columbia, South Carolina. He writes:

I recently moved into a two-bedroom apartment, where there was no room for my TI99/4A except in the bedroom. Because there was not enough floor space for a normal computer system, I was able to obtain a computer cabinet which was configured to hold everything stacked vertically.

The monitor is on a fixed shelf. The next lower shelf slides out and carries the console, a Cor-Comp CC900 Micro Expansion unit and an outboard disk drive assembly with two half-height disk drives.

An SG-10 printer, loaded with mailing list labels, is mounted on the next lower shelf, and an NX-1040 printer, loaded with 8 1/2-inch paper, on the lowest sliding shelf. Beside the SG-10 is a switching assembly for selecting which printer is to be used.

Boxes of disks are carried on the top, fixed shelf and in a small bookcase at the left. There is a bulletin board on the inside of each door, as well as hangers for a small number of files on each.



If you've got a TI or Geneve system that you're proud of or is unique, describe it to us and send a photo. We'll share it with other TI/Geneve owners.—Ed.

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

By JOHN PARKEN

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

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

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

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

Back to what I started to talk about. Any Extended BASIC program like TIPS or DL99 can be listed to a disk file and then searched with TI-Writer for occurrences of CHR$(27);CHR$(65);CHR$(n), which is the line spacing of N/72 of an inch. In most 9-pin printers there are five vertical line spacing commands (see Fig. 1).

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

Fig. 1

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

The program I am looking at is Ron Wolcott's TIPSSHOW, line 260, which includes the following printer command: CHR$(27);CHR$(65);CHR(8). This equates to 1/72 of an inch equals 0.01388 times 8, which equals 0.1111104 of an inch linefeed. In the 24-pin printer, 5/360 of an inch equals 1/72 of an inch. There eight times that would equal 40/360 of an inch which, when divided out, is accurate to five places. This rewrites the line to CHR$(27);CHR$(43);CHR$(40). After making this change, save the program to DSKx.TIPSSHOW. Make sure you use a working copy of the program and not the original. This program does a great job of cataloging the TIPS disk on my Canon inkjet.

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

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

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

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

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

New software developed in Germany

Recent software developments from Germany were noted on the TI listserver. The information comes from Oliver Arnold (oliver.arnold@gedos.de). He can also be reached at Pestalozzistr. 15; 80469 Muenchen; Germany.

HP-MGR is a user-friendly print manager for Hewlett Packard printers. This program lets users print up to eight TI-Artist pictures on a single sheet of paper. It is menu driven with a variety of functions, including disk directory, show TI-Artist picture, print double-wide, etc. It's written in c99.

Risk is a strategy game for two to six players. It is written in assembly language and uses full bitmap graphics.

A batch program that provides speech support for Infocom adventures. It requires Winfried Winkler's new speech card, RAM expansion, and a speech synthesizer.

And here's some more information about the Teletext card that's under development. An external device, it is connected to the RS232 port. It's other end is connected to a CVBS signal from a TV or other source, such as a satellite receiver. The software is c99 and assembly language. It controls the decoder chip via the RS232 port.

Menu-driven software is used to control the card, which allows users to dump pages with graphics to a printer or save them to other devices in D/V80 format, among other things. With an 80-column card, users can store 176 pages in memory. The card requires an expanded TI or Geneve system. It also supports the Becker, Mechatronic, TIm and EGI 80-column cards.

Getting the most from your RS232 cards

By RON WARFIELD

This article was originally published in the British Columbia News newsletter.—Ed.

I have noticed more and more people wanting to connect extra peripherals to their RS232 cards. As it stands, the TI RS232 supports two serial and one parallel port. To use the extra serial port you need a "Y" cable and use RS232/2 as the address.

Well, now you have an extra printer, a mouse, and maybe you want to connect a direct cable to your "other machine." Where are you going to plug in all these toys?

This modification is for TI RS232 cards only. Some CorComp cards come with a switch to do the same thing.

I will explain how to add an extra RS232 card to the Peripheral Expansion Box so you can have RS232/1, RS232/2, RS232/3, RS232/4, PIO/1, and PIO/2.

This modification is for TI RS232 cards only. Some CorComp cards come with a switch to do the same thing.

Open the second RS232 card shell by removing the screws or popping the plastic shell. Place the card face up with the LED to your left and the card edge connections towards you.

About1 3/4 inches in from the left side and 2 1/2 inches down from the top you will see an integrated circuit with the number 74LS02N on it. Marked on the board will be U15. At the bottom of this chip you will see a resistor marked R5 (brown, black, brown). It will have a black mark around it printed on the board. Right below this resistor you will see a metal plate strip with a soldered-over hole at each end. The markings on the board are PTH1.

Carefully unsolder the resistor and move it down to the empty holes directly below it. You will have to remove the solder in the holes or, if you have three hands, you can hold the resistor to the holes and solder iron to the holes and when the solder melts the resistor will go into the hole. Good luck with this. Use a small iron and be careful. I will not be held responsible for damage you may cause. When you are finished, replace the clam shell around the card.

When this is done, place the modified card alongside the first card in the PEB and you will have four serial ports and two parallel ports.

To address these ports, you will have to change some of your programs to get the port you want — RS232/1, RS232/2, etc. You will also need two "Y" cables for the serial ports.

I have modified the second RS232 card in my computer and have my main printer on PIO/1, a label printer on PIO/2, a mouse on RS232/1, my Amiga hard-wired to RS232/2, my TI Pro hard-wired to RS232/3, and Telco (modem) on RS232/4. This is great now because I can transfer files between the Pro and Amiga to the TI, print everything to the main printer on PIO/1 and print labels to PIO/2.

With a couple of A-B-C switches I share the modem and printer with the two other computers. If you can, you might as well make playing with your toys as easy as possible.

MICRO-REVIEWS

Rapid Copy, JP Drawing, Addatex Software

By CHARLES GOOD

RAPID COPY
by Barry Boone and Texaments

The person who copies disks from my user group's library for our out-of-area member has been trying to get me to use Rapid Copy for a couple of years. "Charlie," he said, "I purchased Rapid Copy at a computer show a few years ago and it is the best disk copier I have seen. I use it all the time." At the 1996 Multi User Group Conference I met the program's author, Barry Boone, in person and decided to give Rapid Copy a try. I purchased a copy from Ramcharged Computers, the current owner of all Texaments products. To make a long story short, Rapid Copy is now the primary whole disk copy software on my 99/4A system, replacing Rediskit, my previous favorite disk copy program.

Rapid Copy comes on a SSSD disk with three versions all on the same disk for TI, CorComp and Myarc floppy disk controllers. You can load it as DSK1.LOAD from Extended BASIC, in which case the software will detect which kind of floppy controller you have and automatically load the correct version of Rapid Copy. You can also load the appropriate version as an Editor/Assembler Option 5 file, which is what I do off a RAMdisk in my 99/4A system.

From the main menu you can copy disks, format disks, catalog disks and run the Myarc or CorComp disk manager if you are using a Myarc or CorComp floppy disk controller. The first two are really handy features and are two of the reasons I now prefer Rapid Copy. The third option is not really significant because there are better disk managers out there, such as Funnelweb's Disk Review or DSKU. When formatting disks with Rapid Copy you can select single- or double-sided and (with a non-TI controller) single or double density. You can also select the interface. I just accept the default interface value. If you are using a Myarc floppy disk controller you can also select 16 or 18 sectors per track for double density. The disk catalog option is really handy! Without exiting Rapid Copy one can find out what files are on the target disk before these files are erased in the copy process. You can catalog any floppy drive, but not RAMdisks.

The copy process is easy and flexible. Before copying you can use the main menu to chose verification on/off. The default is off. You are prompted for the drive numbers (1-4) that contain the master and target disks. You can use the same drive number for both master and target disks if necessary and are then prompted to insert each disk as needed. If you have a Myarc RAMdisk Rapid Copy will recognize it as drive 5. Other types of RAMdisks are not supported, and, no, you can't copy to multiple drives. Only one target disk at a time is allowed. Too bad. If you are using a TI controller you are also prompted whether or not you want to format the target disk. With other types of controllers the target disk is always automatically reformatted whether or not it has previously been formatted. Press the "any key" and the copy process begins. First the target disk is formatted, then 5 (DD disks) or 10 (SD disks) tracks are read into memory and then written to the target disk. An on-screen display shows the current track being formatted/read/written. I did some timed tests to check on copy speed. Using a CorComp controller and copying an almost full DSSD disk, Rediskit (which like Rapid Copy always formats the target disk) took 1 minute 18 seconds. Rapid Copy with verification off took 1 minute 24 seconds. Rapid Copy with verification on took 1 minute 39 seconds. With a TI controller these times will be slightly longer.

Verification is an important option in Rapid Copy and one of the reasons I now prefer Rapid Copy to Rediskit. Rediskit has some error traping and will sometimes abort when it finds a disk error, which is good. But rarely Rediskit fails to detect a disk error and makes a bad copy without telling you. So far this has never happened to me with Rapid Copy if I have verification "on." If Rapid Copy detects an error it doesn't just abort the copy operation, it reports an error and asks you if you want to continue anyway and ignore the error. Sometimes ignoring is a good idea, particularly if the error is in the master disk.

The two features I like most about Rapid Copy are its ability to catalog target and master disks from within the program and its ability to verify and make sure you have a good copy. Rapid Copy isn't quite as fast as Rediskit, but the above two features make the small amount of additional copy time worthwhile to me. Rapid Copy is commercial. You can get it with a hard copy user guide for $9.95 from Ramcharged Computers.

JP DRAWING
by Jean-Pierre Morin

This is a very full-featured drawing program that can be run from either Extended BASIC or EA5. It is public domain, and everyone should take a look at it if only to see the demo. When you first boot JP Drawing you are asked if you want to run the demo. If not you are put in drawing mode with the cursor in the center. If you run the demo you can grab a cup of coffee and relax, because the demo goes on and on for well over five minutes. Music plays which almost never repeats itself while image after image flashes onto the screen, moves around and changes into other images. The demo is really marvelous!

This demo runs from a script file which feeds instructions into the program. These script files can be very long, many minutes in length, and can contain instructions for anything the program is capable of drawing. There are two ways to create and save such a script file. From within drawing mode you can enter a command that says, "Save everything I do starting now as a script for playback later." Later, in draw

(See Page 34)

MICROREVIEWS—

(Continued from Page 33)

ing mode, you can enter another command to terminate the creation of this script. There is also an editor which you access from drawing mode with Fctn-Clear (Fctn-4). From this editor you can type in a long sequence of drawing commands, edit and save the list and then run the script.

JP Drawing has a long list of capabilities. You can draw on a full or split screen in bit-map mode using either joysticks or eight direction arrow keys. There are seven different brush sizes, including some that produce a ribbon effect (sometines broad, sometimes narrow) as the brush is moved. There are two different kinds of screen dumps, including a fast small dump that is rotated 90 degrees to the right and a slower full-page-width dump.

As is common in most drawing programs, you can mark the screen at one point, move the cursor to another place and draw something based on these two points. These include straight line, circle and rectangle. You can also do something between two defined points that you can't do with other drawing programs. You can draw an arc! You can specify the radius of the circle, the number of degrees of curvature of the arc, and whether the arc curves left or right between your two defined points.

Other commands include changing cursor and background colors from a 15-color tablet, fill an area with color and put text on the screen. Your text can be curved in an arc or even in a circle if you want. There is a command that advances x pixels and rotates y degrees with each successive letter you type. This is one example of the logo-like commands that are part of JP Drawing.

There are a number of predefined shapes which you can draw with a single keypress. These include star, house, polygone, flower and tree. Some of these shapes have to be loaded into memory from the editor and overwrite the program's background music. You can also define your own shapes with logo-like commands that include foreword, back, right and left x number of pixels. You can also set the angle of motion. The predefined or user-defined shapes can be drawn repeatedly on screen rotating the angle and advancing x pixels with a single command that includes all these variables: number of repetitions, angle of rotation and number of pixels to advance. This results in logo-like recursive shapes that can be quite intricate.

You can save and later reload up to seven screens of graphics. JP Graphics is written in TI Forth and when you save a screen you are saving its information into one of the TI Forth screens or blocks on disk. You can only save and load to and from the JP Drawing disk and the resulting screen saves do not appear on a disk directory if you catalog the disk. The saved screens have no names other than the "01" through "07" names that can only be recognized by the program. These screen saves cannot be imported into any other TI software.

In conclusion, JP Drawing is fun to play with and will do some unique things such as drawing arcs and logo-like recursive patterns. The demo is lots of fun to watch, executes very rapidly, like an assembly language program and illustrates the potential JP Graphics has for creating animation. The inability of the program to export or import graphics to and from other software limits its real world practical uses. It was not written to be "practical." It was written to allow users to draw color graphics in the TI Forth environment without having prior knowledge of TI Forth. This was a fun project for the author and users will also find it fun. The documentation ends with "Have a good time!"

Send me $1 and I will send you JP Drawing on a DSSD disk.

ADDATEX SOFTWARE
by Malcolm Adams

This is so old that it is new. In fact, this software is so old, and so new, that until a couple of months ago I didn't know any of it existed. In 1983 and 1984 a fellow named Malcolm Adams sold 99/4A software in the UK under the Addatex name. The software consisted of various arcade action, logic and education games and was sold on cassettes for £7 each, equal to about $13 US in 1983. The software was written in TI BASIC or Extended BASIC and some of the software gives you the option of using the Speech Editor or TEII cartridge to access speech. These programs were written for the legions of European users who had console-only systems. I never heard of any of these programs. They were not reviewed or advertised in *99er Magazine* and are not mentioned in Mike Wright's 99/4A CYC. The software was, however, reviewed in a number of British computer magazines back in 1983 and 1984.

Very recently Malcolm Adams has established an internet web site (http://ourworld.compuserve.com/homepages/addatex/) dedicated to the 99/4A and his Addatex software. This web site is what makes the old Addatex software so new, to me at least. In my opinion this is the most visually appealing of the numerous 99/4A-related web sites on the internet. Part of the display includes a television set that contains the 99/4A title screen. Elsewhere on the web site you see this television with changing screen shots of various Addatex programs. You can view screen shots and user instructions for all the programs and you can also view the reviews of these programs published in British computer magazines of the day including *Games Computing*, *Software Today*, *Personal Computer News* and *Home Computing Weekly*. The reviews are, of course, highly complementary. The software is given four or five stars out of five. The web site says that the 99/4A, "was by far the best home computer for the domestic market place." and invites users to, "Enjoy this nostalgia trip with software from Addatex and capture the feel of the first real home computer."

One way you get to "capture the feel" of the 99/4A on the internet is to listen to a sound file of a speech synthesizer equipped console introducing itself. The second way you can "capture the feel" is by downloading from the web site onto your IBM compatible PC a file that contains a working version the V9T9 TI emulator complete with all the Addatex software. The V9T9 emulator was originally shareware but it and its source code have been placed in the public domain by the

(See Page 35)

MICROREVIEWS —

(Continued from Page 34)

V9T9 author. Just unpack the downloaded zip file to the root directory of your C drive and the necessary subdirectories with their respective files will be automatically created. You need pkunzip or winzip software to unpack the zip file. Once unziped everything is now ready to run. This is DOS software that will run on even the slowest 386 with VGA graphics. A very nice icon showing the 99/4A title screen is included in case you want to run the software from the Windows 95 or 3.1 desktop, but you don't have to use either windows 3.1 or windows 95 to get the emulator to work. When you start the emulator you get your choice of TI's demonstration cartridge or Addatex software. At the next screen you see, on your PC's monitor, the 99/4A title screen. When you press the "any key" to start you are given your choice of TI BASIC and either the demonstration cartridge or the Addatex programs. If you select Addatex programs you get a menu that lets you select any of programs. The last item on the menu lets you read, from within the emulator, the 1983/84 published software reviews and some software user guides. All the programs also have their own on line instructions.

The software includes 8 games in TI Basic or Extended Basic, all very professionally done. All are colorful and most are accompanied by music and on line instructions. Some include an option for speech, although the emulator provided does not support speech More complete versions of this emulator obtainable elsewhere on the internet do support speech. All these programs are certainly better than the average commercial written-in-basic program of the 1983 time period. The following paragraphs give short descriptions of each Addatex game.

ROBOPODS is an aliens from space game. A defined number (increasing with each of the five difficulty levels) of robopods appear on screen, each numbered. There are also land mines. You have to use keyboard or joysticks to move your little man under and into each robopod to deactivate it. The graphic display of your man climbing into the bottom and out of the top of each robopod is kind of cute. You have to visit the robopods in numerical order and you can't step on the mines or cross your own trail. This is a hard game, particularly because of the prohibition against crossing the trail of radioactive dust you leave behind with your footprints. Occasionally a protective mat appears on your trail that allows you, or the robopods, to move across your trail.

SECRET AGENT takes you to the days of James Bond. You have to run across the rooftop of a moving passenger train to retrieve a briefcase of secret papers at the other end of the train. You have to properly jump between train cars and avoid being knocked off the roof by stationary trackside electric wire towers as the train moves by. It's not easy. Once recovered, another briefcase appears at the other end of the train. When all the train's briefcases are recovered you find yourself next to cable cars. Briefcases are hidden in some of the cars and the agent has to jump into and out of the cable cars to recover all the briefcases. Your have a certain number of secret agents to consume, one at a time. When they have all fallen off the trains the game ends.

SPONTANEOUS REACTION is an addictive strategy game that won first prize in a UK computer magazine's programming competition. It is somewhat similar to Othello and uses two players, or one player against the computer. You can play with or without joysticks and with or without speech. The supplied emulator does not support speech, so it makes no difference which speech option you choose. If you play alone against the computer you may very well lose. The computer is a good player. The object of the game is to fill the entire board with your pieces according to the rather complex rules of the game.

GALACTIC ENCOUNTER is the only game I have ever seen that uses the same game code to provide speech from any of the following: Extended BASIC, TEII, and Speech Editor. You can also run

(See Page 36)

MICROREVIEWS —

(Continued from Page 35)

the game without speech from console BASIC. An introductory screen asks you to select from a menu which of these three modules, or no module, you are using. From the emulator you should select Extended BASIC. This is a logic game. Two players, or one player against the computer, place armies on empty board squares one per turn. Opponent's armies in adjacent squares are eliminated. The winds of fortune can change very rapidly in a single move so that a board full of your enemy's pieces can suddenly fill with your own pieces.

THE PUZZLER gives you two types of puzzles and the option to use the speech synthesizer if available. It is written in TI BASIC. "Match the Shapes" puzzles are like the game known as Concentration. You get 24 tiles and you turn them over two at a time hoping for a match. You have to match colorful detailed patterns, which is a lot harder to do than matching the picture graphics found in most concentration games. Two people play, or one plays against the computer. You have the option of making the computer a difficult or not so difficult opponent in a one person game. In "Pattern Match" puzzles you get a display of 24 complex patterns with resolutions sometimes down to one pixel. You have to find the two displayed patterns that are identical. Even the easy mode is very hard. The game is timed and if you run out of time you are shown the correct answer. These both make great one person a games and are my personal favorites of the Addatex game group.

ARITHMETIC FUN TIME is your typical elementary school +-*/ practice software complete with music and cute graphics. You solve the problems just the way you would on a piece of paper. For example, in addition and subtraction you work from the right to left as you write down the answer on the monitor.

CHECKERS is *not* a computer version of the board game of the same name. It is instead a computer version of "jump a peg." You get a pegboard with pegs all around in the peripheral two rows and all with spaces in the center vacant. You enter the coordinates of a "from" and "to" location to jump a peg and remove the jumped peg. The object of the game is to remove all except one peg and have this last peg in the center. I spent many hours playing this kind of game many years ago as a young preteenager with a real pegboard. It is a one-player game, it is hard, and there are several possible solutions. This is another of the enjoyable Addatex logic games.

The Addatex software has been placed in the public domain by its author. If you have web access on the internet I highly recommend a visit to the very well done Addatex site. If not, send me $1 and I will

(See Page 37)

USER NOTES

Assembly program copies adventure files from tape to disk

The following assembly code was posted to the Internet by Michael Zapf. It's an assembly language program used to copy adventure files from tape cassettes to disk.

The source code should be typed in and saved to a file called CSTODSK_S. Then, using the assembler, enter DSKx.CSTODSK as the object file name and R as options. When you use the program you will not be asked for a file name to copy the cassette data to. It is always CS1PRG. Before you start the program again, rename the file so that it won't be overwritten.

```
* Copy PRG files from CS1 to DSK1
* (especially useful for adventures)
*
* TI BASIC READY
* > CALL FILES(1)
* > NEW
* > CALL LOAD("DSK1.CSTODSK")
* > CALL LINK("COPY")
* Rename the disk file with Disk Manager *
* Michael Zapf 1985, 1996

DEF     COPY
REF     DSRLNK,GPLLNK,VMBW,VMBR

VDPBUF EQU  >0800
MAXLEN EQU  >3300          * should suffice
CHKVAL EQU  >A500          * arbitrarily cho-
sen

DSKPAB DATA >0600,VDPBUF,0
DLNG   DATA 0
       DATA >000B
       TEXT 'DSK1.CS1PRG'

CSPAB  DATA >0500,VDPBUF,0,MAXLEN
       DATA >6003
CSNAME TEXT 'CS1'

COPY   LI   R0,>0380
       LI   R1,CSPAB
       LI   R2,13
       BLWP @VMBW

* Fill VDP buffer with >A5
* (to determine the actual size)

       LI   R0,VDPBUF+>4000
       SWPB R0
       MOVB R0,@>8C02
       SWPB R0
       MOVB R0,@>8C02
```

(See Page 38)

MICROREVIEWS —

(Continued from Page 36)

send you the complete Addatex game package on a 3.5-inch PC disk. You of course need an IBM-compatible to play this TI software. I'm sorry, but I am just too lazy to type in all the code onto a TI disk. I will also send you the program pkunzip.exe on the same disk so that you can unzip the addatex file onto the hard drive of your PC. The unziped file gives you the V9t9 emulator, all the Addatex games, a special version of extended basic, and the TI demonstration cartridge all of which run off of the emulator.

ACCESS

Charles Good (source for JP Drawing and Addatex games); P.O. Box 647, Venedocia, OH 45894. Phone 419-667-3131. Preferred internet email good.6@osu.edu

Ramcharged Computers (source for Rapid Copy); 6467 E. Vancey Dr., Brookpark, OH 44142. Phone 216-243-1244

NEWSBYTES

Competition produces mouse for Geneve

Competition Computer has produced a Geneve mouse, at $25 plus $3 shipping. According to Kyle Crichton of the company, it is a replacement mouse for the Geneve that is completely transparent for it.

"It is a true bus mouse that will work with Geneve software," he says. "It doesn't need anything else."

Crichton says the company will be shipping after Fest West. For further information, contact Competition Computer, 350 Marcella Way, Millbrae, CA 94030, or (415) 697-1108.

Virgil Thomason, J. Lynn Nidiffer die

Virgil Thomason, former co-president of TI-CHIPS in Ohio, died Feb. 21. He had been active at club meetings and served as greeter at the 1996 Multi User Group Conference.

J. Lynn Nidiffer, age 67, died Feb. 6. Funeral services were Feb. 10 at Hillcrest Mausoleum in Dallas. A member of the Dallas TI Home Computer Group since 1982, he had served as the club's representative and alternate to the Computer Council of Dallas, adventure software librarian and newsletter editor.

USER NOTES

(Continued from Page 37)

```
        LI    R1,CHKVAL
        LI    R2,MAXLEN
CLLP    MOVB  R1,@>8C00
        DEC   R2
        JNE   CLLP

* Prepare for GPLLNK

        LI    R0,CSNAME
        LI    R1,>834A
        MOV   *R0+,*R1+
        MOVB  *R0,*R1
        LI    R0,>0003
        MOV   R0,@>8354
        LI    R0,>038D
        MOV   R0,@>8356
        CLR   @>83D0
        LI    R0,>0800
        MOVB  R0,@>836D
        SWPB  R0
        MOVB  R0,@>837C

        BLWP  @GPLLNK
        DATA  >003D

* Find out how long the file was.

        LI    R0,VDPBUF
        SWPB  R0
        MOVB  R0,@>8C02
        SWPB  R0
        MOVB  R0,@>8C02
        CLR   R1
        LI    R2,MAXLEN
        MOV   R0,R3

LSLP    MOVB  @>8800,R1
        CI    R1,CHKVAL
        JEQ   $+4
        MOV   R0,R3
        INC   R0
        DEC   R2
        JNE   LSLP

        AI    R3,-VDPBUF
        INC   R3

* Result is in R3; should be a multiple
* of >40 (only with cassette files).
* Let's hope the file didn't end with
* >A5 (the CHKVAL).
* We round up to a >40 multiple so that
* this case is effectively worked
* around.

        AI    R3,63
        ANDI  R3,>FFC0

        MOV   R3,@DLNG

* Set up the PAB for saving.

        LI    R0,>0380
        LI    R1,DSKPAB
        LI    R2,21
        BLWP  @VMBW

        LI    R0,>0389
        MOV   R0,@>8356
        CLR   @>837C

* Here we go.

        BLWP  @DSRLNK
        DATA  8

RT

END
```

Easy VCR connection

This item was written by Andy Frueh of the Lima User Group.

You can hook up your TI to a VCR using a standard 300 ohm to 75 ohm TV antenna adapter or a composite monitor cable. Adapters are found with almost all home video game systems or at electronics supply stores. It has a cable TV male connector and two screw terminals. The male plug goes into the "cable in" jack of the VCR. The screw terminals go to the modulator's "to TV" wire. You can then hook the VCR to a stereo's "aux in" jack using a standard audio/video cable.

Turning printers into typewriters

The following article appeared in the newsletter of the West Penn 99ers. Ed Machonis is the author.—Ed.

The QB-99ers newsletter published the following article that may be of interest to others. I am always looking for short cuts.

There are often times when we just want to type a short note or letter and, rather than load in a full-blown word processor, we settle for writing it out with such low-tech implements as a pen or pencil.

It is very easy to turn your printer into an electric typewriter. Four lines of BASIC code will do it:

```
1 OPEN #1:"PIO"
2 INPUT A$
3 PRINT #1:A$
4 GOTO 2
```

(See Page 39)

USER NOTES

(Continued from Page 38)

This program enables the user to type a line of text, edit it as desired, and then print it by hitting the Enter key.

Whenever a line of text is to be indented or contains a comma, that line must begin and end with a quotation mark. The quotes will not be printed, nor will they be counted in width of the line of text. To skip a line, just hit Enter.

This program allows sending printer codes directly to an Epson RX-80 printer provided they are in the same form as in the previously described RX-80 program. For example, CTRL-period [CHR$(27)], Shift-E, Enter sends the printer control code for emphasized type. Other codes, of course, may be sent in the same manner.

By adding a few more lines, the program can be more useful. We can require an input as to the maximum line width to be printed and use this information to set equal right and left margins. A check has been added to ensure that the maximum line width is not exceeded and that it includes a prompt to display what an overly long line can be shortened to. User instructions have also been added. The expanded 10 line BASIC program looks like this:

```
1 PRINT "TO INDENT TEXT OR T
O USE A COMMA, BEGIN & END
THAT LINEWITH QUOTATION MARK
S." ::
2 INPUT "PRESS ENTER TO SKIP
A LINE. HOW WIDE?(80 CHARAC
TERS MAX)":WIDTH
3 MARGIN=INT((80-WIDTH)/2)
4 OPEN #1:"PIO"
5 INPUT "INPUT LINE A LINE O
F TEXT: ":TEXT$
6 IF LEN(TEXT$)>WIDTH THEN 7
ELSE 9
7 PRINT :"LINE TOO LONG! SHO
RTEN TO" :: WIDTH;"CHARACTER
S MAX." :: (SEG$,1,WIDTH)::
8 GOTO 5
9 PRINT #1:TAB(MARGIN);TEXT$
10 GOTO 5
```

When typing notes, etc., where it is desirable to start printing at column 1, input a line width of 80 and monitor the line width on the screen.

A simple way to use this program for correspondence is to use a line width of 56. This will fill exactly two lines of the TI screen. Right margin justification can be accomplished by inserting spaces between words until the second line of text is completely filled.

The OPEN statement in line 4 should be changed as required for the particular printer in use. The line width feature is designed for a Pica character set. Line 3 can be changed to accomodate Elite or condensed type styles.

XBASIC bug?

This comes from Martin Zeddies, of Wolfsburg, Germany. It's an Extended BASIC program that seems to indicate an error in Extended BASIC. Give it a try. Is it a bug in XBASIC?

```
100 CALL CLEAR
110 ! By the great dane. Thi
s concerns TI-EXT.-Basic Ver
sion 110
120 PRINT "This program SHOU
LD print ": :"P=1 , but due
to an error": :"in the exte
nded Basic it": :"prints P=3
.": : :
130 PRINT "to avoid an error
": :"simply add a meaningles
s": :"statement like X=X aft
er the": :"GOSUB and before
the ELSE .": : :
140 D=0
150 IF D=0 THEN GOSUB 170 EL
SE Y=5 :: P=3
160 GOTO 180
170 P=1 :: RETURN
180 PRINT "P = "&STR$(P)
190 CALL KEY(0,K,S):: IF S=0
THEN 190
```

TI XBASIC error

The first several lines of this program says it all. It comes from Martin Zeddies of Wolfsburg, Germany.

```
100 CALL CLEAR
110 ! By the great dane. Thi
s concerns TI-EXT.-Basic Ver
sion 110
120 PRINT "This program SHOU
LD print ": :"P=1 , but due
to an error": :"in the exte
nded Basic it": :"prints P=3
.": : :
130 PRINT "to avoid an error
": :"simply add a meaningles
s": :"statement like X=X aft
er the": :"GOSUB and before
the ELSE .": : :
140 D=0
150 IF D=0 THEN GOSUB 170 EL
SE Y=5 :: P=3
160 GOTO 180
170 P=1 :: RETURN
180 PRINT "P = "&STR$(P)
190 CALL KEY(0,K,S):: IF S=0
THEN 190
```

CLASSIFIED

FOR SALE

GENEVE 9640

With keyboard, MY-Word, MY-Art, Advanced BASIC, About 50+ disks full of games and Geneve programs worth $100+. Buy it all for $175. Roger, 317 664-6001. v14n2

FOR SALE

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

WANTED TO BUY

WANTED

Copy of the terminal program "Triad" Richard R. Hay, 2604 Pine Lake Dr., W. Columbia, SC 29169 v14n2

Devoted to the TI99/4A since 1984

Subscription Fees

❑ 6 issues, USA, **$35**

❑ 6 issues, Mexico, **$40.25**

❑ 6 issues, Canada **$42.50**

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

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

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

Address Changes

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

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

Check/MO ❑ ❑ ❑ (check one)

Card No. _____

Expiration Date _____

(Minimum credit card order is $9)

Signature _____

(Required on credit card orders.)

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

Name _____

Address _____

City _____

State _____ **ZIP**_____

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

Disks, Etc.

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

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

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

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

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

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

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

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

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

GENEVE DISKS (SSSD unless specified)

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

❑ **GPL 1.5** $4.00

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

❑ **Myarc BASIC 3.0**.....$4.00

❑ **MY-Word V1.21** $4.00

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

GENEVE PUBLIC DOMAIN DISKS

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

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

PERIODICAL

A 1/98 T
CHARLES GOOD
P.O. BOX 647
VENEDOCIA OH 45894