Subscription Fees

❑ 6 issues USA $35
❑ 6 issues Canada/Mexico $42.50
❑ 6 issues other countries surface mail
___ Surface mail $40
___ Air mail $52

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 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 ❑ ❑ ❑

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.

PERIODICALS

Micropendium
P.O. Box 1343
Round Rock, TX 78680

Covering the TI99/4A and Geneve home computers

MICROpendium

Volume 14 Number 6 November/December 1997 $6

PC99 users gain access to Tigercub PD disks

Tim Tesch takes over Myarc repairs/upgrades

Got a project?
Keep the momentum going

Chicago TI Faire report

Definitive site for virus info

BRUCE HARRISON ON A/L SOUND

HELP FOR MODULE TRANSFERS USING V9T9

Reviews of Invoice Management and Font Designer

Programs in this issue

Random Symmetrical Character Generator
Load Maker in XB
Sunrise-Sunset in XB
Sound List Processing in A/L
Gift Exchange in XB

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.

Foreign subscriptions are $42.50 (Canada and Mexico); $40 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

Internet E-mail: jkoloen@earthlink.net

Home page: http://www.earthlink.net/~jkoloen/

John Koloen ..... Publisher
Laura Burns ..... Editor

The SouthWest Ninety Niners User Group Proudly Announces

FEST WEST '98 - LUBBOCK

Where: Sheraton Four Points Hotel and Texas Instruments, Inc., Lubbock, Texas
When: February 14, 1998 from 8:00 A.M. until ??:?? P.M.

THE FIRST TI COMPUTER FAIR TO BE HELD AT THE BIRTHPLACE OF THE TI-99/4A!

TEXAS INSTRUMENTS — FEST WEST '98 - LUBBOCK

A "TI Experience" will be held on the morning of February 14th at the Texas Instruments Facility in Lubbock where the TI-99/4A Home Computer was manufactured!

Be in Lubbock, Texas for the first ever TI-99/4A Homecoming! Meet fellow TI99ers from all over the country and world, including some from our past.
This will be a true ONCE IN A LIFETIME experience!

For more information, contact ...
The Official Fest West '98 Web Site: http://personal.riverusers.com/~twills/fw98.html
The SouthWest Niners U.G. Web Site: http://www.theriver.com/Public/sw99ug/
Tom Wills - (520) 886-2460 - Twills@TheRiver.Com or Mike Doane - (520) 298-3835 - MDoanc@TheRiver.Com

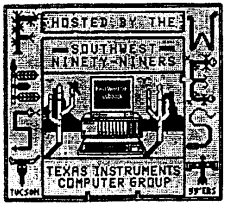
HOTELS:	*RATES*	*TELEPHONE*	*REQUEST RATE FOR*
Sheraton Four Points	$54 for 1 or 2 persons; $10 each Addl. Person	(806) 747-0171	Texas Instruments Computer Fair
Koko Inn	$44 for up to 4 people	(800) 782-3254	Fest West '98

Fest West '98 - Lubbock Schedule
2/13/98 - 4:00 PM until 9:00 PM - Pre-registration in the Sheraton Four Points Hotel Hospitality Room.
2/14/98 - 3:00 AM until Noon - TI Experience at TI facility. NOTE: You must be there before 8:30 AM!
1:30 PM until 6:00 PM - Vendor portion of FW98 in the Sheraton Four Points Hotel's Santa Fe Ballroom.
1:00 PM until ??:?? - Hospitality Room at Sheraton Four Points Hotel.

Door Prizes, Raffle, Speakers, Demos, TI-99/4A Personalities, and more!

Don't miss the TI Event of a lifetime! Be in Lubbock on February 14th, 1998 for Fest West '98 - Lubbock!

Admission only $4.00 per person. Children under 15 free when accompanied by a paying adult.



Texas Instruments Logo used with permission of Texas Instruments, Inc.

COMMENTS

PC99 gets Tigercub PD programs

The prolific Charles Good has done it again. I sometimes wonder how anything in the TI community gets done unless Charlie Good is involved. His latest effort is going to please anyone who uses PC99, the great TI99/4A emulator that runs on PCs. Using PC99 transfer utilities, he's converted all of the Tigercub public domain disks from TI format to PC99 format. We're not talking about a weekend's worth of work here. We're talking 583 disks requiring more than 149 megabytes of hard drive space on a PC.

Charlie then passed the converted disks to Mike Wright of CaDD Electronics, which produces PC99. Wright is now offering to provide users access to this huge library of TI software. Wright says it is possible to compress the 149Mb down to 37.4Mb using lharc. (An advantage of lharc is that users can extract any single disk file without having to decompress the entire lharc, or LZH, file.)

If you're interested in obtaining the file, you can send Wright a PC-formatted Zip disk with return mailer and postage (Mike Wright, 45 Centerville Dr., Salem NH 03079-2674). Or, you can e-mail him at mjmw@xyvision.com and he will send instructions on accessing an FTP site where you can download it. Or, if you have a Web or FTP site, he will upload the file to your site.

Wright has also redone the Tigercub public domain catalog into Adobe Acrobat format. This can be included with the Tigercub files. Just ask for it when you contact him. And let's not forget that the Tigercub Public Domain archive is the legacy of Jim Peterson.

WE THOUGHT NOTEWORTHY WAS NOTEWORTHY

Yeah, I know, we printed a program in the last issue that we printed in 1993. Even if we knew about it, we couldn't have changed the program. We got into a big rush to make a mailing deadline and the only way to make it was to put MICROpendium out as fast as we could. Which, given our new format and production process, is not very fast. In fact, it's extremely labor-intensive. You've heard of the lost weekend. Well this is the lost fortnight. That's about how long it takes from the time we print the first copy to the time we get the last overseas copy into the mail. That's because we printed out each sheet on a laser printer. They have to go through twice, once for each side. Then we have to collate the copies, staple them, fold them, seal them with tape, label them, sort them for the post office, and get them to the post office. We underestimated the amount of time it would take to do all of this, which is why we were hard up against a deadline and couldn't have replaced the Noteworthy program even if we wanted to.

Despite all of this, MICROpendium got out a little later than usual. Starting with this edition, we hope, it will get out on schedule.

—JK

1997 CHICAGO TI FAIRE

Seminars keep visitors busy at 15th Chicago fair

Story and photos
By CHARLES GOOD

The 15th annual TI International Faire was an all-day event sponsored by the Chicago User Group. Think of that for a moment, 15 years! I have been attending annually since 1985. This year's event was at the Evanston Public Library and was organized by Hal Shanafield. In addition to the exhibit room, there was a separate seminar room with more or less non-stop seminars. Several user groups and dealers had display/sales tables.

Hal Shanafield hands out awards after the faire.

In his seminar, Bruce Harrison showed some of his new commercial and public domain products. He had a Yamaha keyboard and impressed the audience by playing music using his revised MidiMaster 99 software. There are AMS enhanced, and non-AMS versions of this software. It now allows up to 16 tracks, with individual volume control, and has many other improvements over the original MidiMaster. Some musical pieces Bruce played were original Midi arrangements by his partner Dolores P. Werths. Bruce also showed an improved AMS slideshow, which can store 84 pictures in memory if you have a 1meg AMS card.

Berry Harmsen talks about TI users in Europe.

REPORT FROM EUROPE

Berry Harmsen of the Dutch Users Group talked about what is happening in the TI world in Europe. He reported that there is now a European SCSI card which is faster than the SCSI card sold in this country. Shanafield has been given the documentation for this product, in German, and offers to pass copies on to others.

Berry talked about several cards made by Michael Becker and friends, some of which will be sold in this country by Cecure Electronics. These cards include a DSDD controller with clock, a PEB card that takes the place of the console and uses an AT keyboard, an 80-column card with

Continued on page 6

1997 CHICAGO TI FAIRE

Continued from page 5

more than 256K colors, and a GRAM Kracker-like card called the High Speed GPL card.

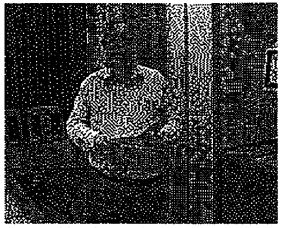
Berry reports that there are still three active TI user groups in Europe and that the German and English groups are growing. The 1998 European TI show will be in England.

PC99 UPGRADE SHOWN

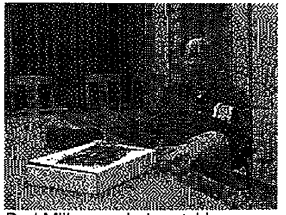
Mike Wright wowed his audience by showing the latest capabilities of PC99. He showed emulation of the Myarc 512K memory expansion card running Myarc Extended BASIC II. He also used the latest AMS diagnostic software to"test" a 1 meg AMS card emulated under PC99.

In my opinion, the ability to emulate a 1 meg AMS card will become important in the future. That is because Rich Gilbertson will soon make available a version of Extended BASIC that can make full use of the AMS card's additional memory for assembly programming.

Mike reports that the Jim Peterson public domain software library of over 550 TI disks has been converted into PC99 format and is available from him and others for free.



Bruce Harrison showed an enhanced MidiMaster.



Bud Mills says he's catching up on his RAMdisk business.

Mike also showed a cross-indexed version of all Jim Peterson's Tips from the Tigercub in Adobe Acrobat format viewable on any PC or Mac. Most or all of the items described in this report will be included in the next upgrade of PC99.

Bud Mills gave an upbeat talk about what is happening with Bud Mills Services. He said that almost all repair work on his products is finished and those who have been waiting should get their PGRAMs or RAMdisks in the mail soon. He reports that some PGRAM cards will trash a module on power down and has worked out a hardware solution to the problem which he will publish soon.

He also said he plans another production run of Horizon 4000 boards with a slight redesign to reduce the possibility of shorting out the card on the back of the PEB. Memory prices for Horizon RAMdisks have come way down, Bud says.. He should soon be able to sell a

Continued on page 31

FEEDBACK

Blast from the past

Upon opening my mailbox I was quite surprised to discover MICROpendium in the new format. At first I thought it might, perhaps, be something extra, like a listing of fairware programs or something along that line. After taking a closer look at it and realizing it was a new format for the magazine I immediately sat down and started to look through it.

As far as I am concerned I cannot find anything wrong with the format but was a little disappointed to discover the program called NOTEWORTHY, as to me it was a "waste" of space due to the fact you had published this same program in the March 1994 issue.

I shall continue to subscribe to MICROpendium as long as you continue to publish it. I still look forward to its coming with as much enthusiasm now as I had years(?) ago.

Donald E. Hunt
Battle Creek, Michigan

Think positive

I was very surprised when I saw your new format — very nice.

The heavier quality of paper and the print on the extremely pure white is much easier to read, especially forus old TIers.

If anybody out there has any negative responses, don't worry, they don't count.

Ed Mandich
Treasurer, West Penn 99ers
East McKeesport, Pennsylvania

EXTENDED BASIC

Random symmetrical character generator

By JIM PETERSON

This month the old Tigercub would like to share with you his methods of creating random symmetrical redefined characters. I doubt that I'm the first one who ever thought of this, but I've never seen it in anyone else's programs and I don't think it can be done in BASIC on any computer other than the TI.

In its basic form it goes like this:

```
100 DIM A$(16)
110 DATA 00,18,24,3C,42,5A,66,
7E,81,99,
A5,BD,C3,DB,E7,FF
120 FOR J=1 TO 16
130 READ A$(J)
140 NEXT J
150 FOR L=1 TO 4
160 RANDOMIZE
170 X=INT(16*RND+1)
180 B$=B$&A$(X)
190 C$=A$(X)&C$
200 NEXT L
210 CALL CHAR(65,B$&C$)
```

Now, no one should ever use a routine without understanding it, because you won't be able to debug it and you won't be able to modify it. So, let us go through this.

If you take a good look at the chart on page 109 of your Beginner's BASIC, or page II-77 of the Users Reference Guide you will see that those pairs of hexadecimal numbers in line 110 represent rows of bits which are mirror images of each other. Therefore, if we assemble a character from these pairs, it will have left-right symmetry. So, lines 100-140 read these pairs into an array Then, lines 120-200 go through a loop four times, each time picking one of these pairs at random, by randomly picking a subscript number between 1 and 16.

The top half of the hex code of our redefined character is built up by adding these pairs to the end of string B$, which starts out as being blank. For instance, if on the four loops the random values generated for X are 1, 2, 3, and 4, the pairs selected are 00, 18, 24 and 3C, and B$ is successively built up as 00, 0018, 001824, and finally 0018243C At the same time C$ is built up with the same pairs in reverse order as 00, 1800, 241800, 3C241800. Finally, line 210 redefines ASCII character 65 as string B$&C$, which is "0018243C3C241800", and which is symmetrical top-to-bottom as well as left-to right.

Now that you understand how it works, let's program it a bit more efficiently.

```
100 FOR L=1 TO 4
110 RANDOMIZE
120 X$=SEG$("0018243C425A667
E8199A5BDC3DBE7FF",INT(16*RN
D+1)*2-1,2)
130 B$=B$&X$
140 C$=X$&C$
150 NEXT L
160 CALL CHAR(65,B$&C$)
```

You can cram that into one line of

EXTENDED BASIC

Extended BASIC! So, what's it good for?

Well, let's add:

```
90 FOR CH=40 TO 152 STEP 8
Change 160 to CALL CHAR(CH,B$&C$)
170 B$=""
180 C$=""
190 NEXT CH
```

Now we've redefined the first character of sets 2 through 16. Don't forget lines 170 and 180 Since B$ and C$ are formed by adding onto themselves, they must be cancelled out before we start over or they will just keep on adding onto themselves.

Next, let's give each character set a foreground color, and a different background color.

```
200 FOR SET=2 TO 16
210 X=INT(15*RND+2)
220 Y=INT(15*RND+2)
230 IF Y=X THEN 220
240 CALL COLOR(SET,X,Y)
250 NEXT SET
Now for the fun:
80 CALL CLEAR
260 CALL SCREEN(5)
270 CALL HCHAR(INT(24*RND+1)
,INT(32*RND+1),INT(15*RND+1)
*8+32,INT(10*RND+1))
280 CALL VCHAR(INT(24*RND+1)
,INT(32*RND+1),INT(15*RND+1)
*8+32,INT(10*RND+1))
290 IF INT(10*RND)<>0 THEN 2
70
300 CALL CLEAR
310 GOTO 270
```

Or, if you're in Extended BASIC, let's change:

```
90 FOR CH=40 TO 136 STEP 4
155 SP=SP+1
160 CALL CHAR(CH,RPT$(B$&C$,
4))
165 CALL SPRITE(#SP,CH,INT(1
5*RND+2),1,1,INT(10*RND+2)-I
NT(10*RND+2),INT(10*RND+2)-I
NT(10*RND+2))
185 CALL SCREEN(5)
200 FOR D=1 TO 100
210 NEXT D
220 CALL MAGNIFY(INT(4*RND+1
))
230 GOTO 200
and delete 240-310.
```

You would prefer something a bit more useful? OK, let's try a different variation of the same principle.

```
100 CALL CLEAR
110 RANDOMIZE
120 DATA TIGERCUB PRESENTS,T
HE,CHAMELEON,SCREEN BORDER,A
ND,WIPE,by Jim Peterson," ",
" TOUCH ANY KEY"
130 M$="1800665AC342DB667E18
8100995AC3A5
E78142DB24BD660
081429924007E5AC3A5C3
241800
FFDB5AFF7EFF0099188100660018
"
140 RESTORE 120
150 FOR P=1 TO 9
160 READ A$
170 PRINT TAB(15-LEN(A$)/2);
A$;" "
180 NEXT P
185 PRINT :::::
190 GOSUB 240
200 CALL KEY(0,K,ST)
```

Continued on page 10

EXTENDED BASIC

Continued from page 9

```
210 IF ST=0 THEN 200
220 GOSUB 320
230 GOTO 140
240 CALL CHAR(128,SEG$(M$,IN
T(43*RND+1)*2-1,16))
250 X=INT(15*RND+2)
260 Y=INT(15*RND+2)
270 IF Y=X THEN 260
280 CALL COLOR(13,X,Y)
290 CALL HCHAR(24,1,128,64)
300 CALL VCHAR(1,31,128,96)
310 RETURN
320 T=T+1-ABS(T=2)*2
330 ON T GOTO 340,360
340 CALL VCHAR(1,3,128,768)
350 GOTO 370
360 CALL HCHAR(1,1,128,768)
370 CALL CLEAR
380 RETURN
```

In this one M$ consists of any of those symmetrical pairs typed in at random, and we define a character which has only left/right symmetry by randomly pulling out any sequence of 16 of these. Another way to do this is:

```
100 CALL CLEAR :: RANDOMIZE
:: FOR L=1 TO 8 :: B$=B$&SEG
$("0018243C425A667E 8199A5ED
C3DBE7FF",INT(16*RND+1)*2-1,
2):: NEXT L
110 CALL CHAR(128,B$):: B$="
" :: CALL MAGNIFY(2):: CALL
SPRITE(#1,128,2,8,120)
120 CALL KEY(0,K,ST) :: IF S
T=0 THEN 120 ELSE 100
```

Now start tapping any key until you find an appropriately evil-looking alien ship or man-eating bug for your game program. If you consult that chart in Beginner's BASIC again, you will find that the first eight of those pairs do not turn on the first or last bit, therefore do not fill a print space. So, let us enter another program:

```
100 CALL CLEAR
110 FOR CH=129 TO 154
120 RANDOMIZE
130 FOR L=1 TO 3
140 X$=SEG$("0018243C425A667
E",INT(8*RND+1)*2-1,2)
150 B$=B$&X$
160 C$=X$&C$
170 NEXT L
180 CALL CHAR(CH,"00"&B$&C$)
190 B$=""
200 C$=""
210 NEXT CH
220 INPUT M$
230 GOTO 220
```

Now type any of the letters with the CTRL key held down. Your computer has a built-in Venusian alphabet!

Many other effects and variations are possible. I use this routine frequently in my Tigercub programs. It provides the infinite variety of kaleidoscopics displays in Kaleidovision, Multivision and Ten Thousand Sights. It enables me to provide completely different assortments of colorful cards to turn over in each new game of Match-a-Patch, and a new screenful of walls in each game of Getaway.

It provides the colorful characters of the Mongolian Typewriter for the little tots, and the rainbow displays that reward correct answers in Kinder-minus, and many more. My Color Programming Tutor will show you ways to make the displays even more colorful, and my Random Character Generator will give you your choice of eight different types of these characters, print out an assortment of 40 of them, display them singly and in strips and blocks and as a sprite, allow you to change their foreground and background colors, print out their hex code, and enlarge them to your choice of three sizes, even filling the screen.

EXTENDED BASIC

V9T9

Some help for module transfers and V9T9 utilities

The following compilation of materials appeared on Internet TI listserver. It was prepared by Dennis Remmer, editor of BugBytes, *the newsletter of the TI Brisbane (Australia) User Group. The original postings are edited. Responses are in italics. —Ed.*

Original posting from: Randy randyb@utilicom.com

Response from: Brian Tristam Williams POLAR@global.co.za

The following long message is a culmination of all my unsuccessful attempts over the last few weeks to get V9T9 and its utilities to work properly. I have invested a great deal of time printing and reading all of V9T9's docs as well as a lot of fruitless trial and error. To make a long story short, I'm at my wit's end and I'm going to give up for good on V9T9 and delete all the TI files I downloaded from the Internet if I don't make a breakthrough soon.

My only success with V919 6.00 (I got the one from ftp.io.org/ ... /opanit) is that I've had no problem using any of the module images, DOADs (Disks On A Disk), or ROMs that I fortunately already had on my PC so I could test things out. Overall, I'm impressed by the quality of the emulation. But my problems are many:

1. I can't get any of V9T9's utilities to work with DOADs no matter what I try (haven't tried FIADs [Files In A Directory] yet).

Neither can I, Randy, but others report success at this. The utilities are little DOS executables that perform rather simple tasks, and I'm sure that someone could rewrite them to be more tolerant. I have a 486-DX4 100Mhz with DOS 6.22 and Windows 95, by the way. Perhaps we could try an older version of DOS — I'm sure that when Ed (Schwartz) wrote these utilities he wasn't using the latest DOS.

I know the DOADs are correct because they work fine when accessed inside of V9T9. I remember hearing a few other people had the

Continued on page 12

V9T9

Continued from page 11

same trouble, but I don't recall the solution. I am using DOS 6.20 on a Pentium 75 with 32 megabytes.

For instance, the Pirate Adventure disk is PIRATE.DSK, has a volume name of PIRATE, and one file named PIRATE. Using V9T9 I have no trouble using this or any other DOAD. However, no matter what I try with any of the utilities such as TIDIR, TICHKDSK, etc., I get "no files match PIRATE.DSK, continuing" if I type "TIDIR PIRATE.DSK."

When I type "TIDIR PIRATE:" I get the volume label followed by the same error message.

When I type "TIDIR PIRATE" I get: Error on file (path)PIRATE.DSK Bad file structure."

If I use TTCHKDSK * in the DISKS (DOADS) directory, every file gives the error: "Not a v9t9-compatible file, or a damaged v9t9 file." I know the files are all valid because they work under V9T9.

Without the use of these utilities, all of the file transfers I need to do are impossible. I've tried running both from a Windows 3.1 DOS shell and pure DOS, with and without EMS memory managers, plus every uppercase, lowercase, and logical combination of image/file names I could think of.

Another problem was with TI2XMDM. Every time I used that utility such as: "TI2XDM MUNCHMNG.BIN ." the utility locked up my PC.

You can't do that! You can only use TI2XMDM on disk files — FIADs or DOADS. And since there is a problem using the utilities on DOADs you have to use a FIAD. So, TI2XMDM will work only on files with V9T9s header. I think that the best way to transfer a program such as MUNCHMNG. BIN to the TI is to leave it as-is. Use a terminal program on the TI and save it as a program file — I'm just guessing here, so I'd welcome corrections. Another thing for me to look at.

Maybe I don't have the syntax down right and that's part of the problem, but given that nothing else works, I think there may be something wrong with the whole archive. Has anyone else successfully used the utilities in 60OV9T9.ZIP from the Opanit ftp archive?

2. When I follow the instructions in TRANSFER.TXT on how to use Forth to send the "TRANS" program over to the TI, as soon as I type "TRANSFER" as instructed (or even "HELP") I get a blank screen and nothing happens. What am I doing wrong?

There's a strange one. When I first got V9T9, I managed to run the TRANSFER program, but I've lost familiarity with it. Now, I tried this, and I also just got a blank screen — I'll have to look into that.

As an alternative, I tried XMODEMing TRANS to the TI and use the resulting DIS/FIX 128 file with the CALL LOAD/LINK commands from the Receiver program on the TI. That didn't work, either.

I type FORTH.BAT, select the Forth module and get the Forth prompt (which says "type HELP for help," among other things). When I type "TRANSFER" (without the quotes of course) I get a blank screen. The same thing happens if I type "HELP." The PC hasn't locked up because I can still select other modules, exit with CTRL-BREAK, etc. However, I can't get the TRANS program over to the TI as described in TRANSFER.TXT.

So I tried transferring the TRANS file (a FIAD, I guess) in the DISK directory using ProComm and Mass Transfer. The file successfully got to the TI as a DIS/ FIX 128. I tried using the CALL LOAD and CALL LINK statements from the TI's Receiver program, which I typed in by themselves, on this TRANS file, but I got a "BAD CHARACTER IN xxx" on the line with the CALL LOAD.

TRANS isn't a DIS/FIX 128 file. It needs to be a DIS/FIX 80 file of size 64 (sectors). I also couldn't use the Forth transfer program, so I had to use Extended BASIC transfer programs that I wrote, called OBTRANS (on V9T9) and OBRECV (on the TI). The program was generic enough to transfer any DIS/FIX 80 file. I'll check it out again for you.

By the way, I looked at a byte listing of the TRANS file and I noticed it contained what appeared to be a hex dump in ASCII format. I have no idea what the Forth program does, but I suppose it converts the ASCII into binary and sends one byte at a time to the TI to be stored in a binary file usable by CALL LOAD/LINK. Maybe since the file I transferred was still ASCII that's why I got the "BAD CHARACTER" error? If I converted it to binary before transferring it, would it work then?

No — the file is an uncompressed assembly language object file (gee, I've learned something about assembly), which is a hex-dump. Leave it as an ASCII hex-dump. I think your problem is that it's supposed to be a DIS/FIX 80 file.

3. Is it possible to take module dumps (images) that work with V9T9, transfer them to the TI, use an Extended BASIC loader program, and have them work on the TI as if they were real modules? If yes, is there a generic loader program I can get that will work on all module images?

4. Some TI dealers have sold module images on disk, supposedly with permission, that work just like a real module on the TI. Are these module images in the same GROM and Bank C and D segments as V9T9 would use? Can they be transferred to V9T9, processed with XMDM2TI, and then added to MODULES.INF? Or are these not "pure" module segments so they must be used as disk files with the Extended BASIC loader provided?

How does this loader program work? Unlike V9T9 modules which have G, C, and D extensions for where the image is loaded, all of the module images on disk do not follow this format. The loader program only lists the first segment, so how does the loader program know what

Continued on page 14

V9T9

V9T9

Continued from page 13

filename comes next and where to put it? Are the smarts to this loading process in the LOAD code in each of these modules?

For example, I have the module image on disk for Treasure Island. The files on this disk have the names "TISLAND," "TISLANE," and "TISLANF." All are of the program type. This module image has a loader program in Extended BASIC which is easy to understand, except for a few mysterious numbers in CALL LOAD which I have no documentation for. What does CALL LOAD(8196,254,0) do? Or how about FOR S=1 TO 4 :: CALL LOAD(-S*6144,0) :: NEXT S ? Finally, there's a CALL LINK ("LOAD","DSKI.TISLAND") statement. Besides the CALL INIT, I can type the five statements above and have the Treasure Island module run successfully. I know that CALL LINK loads the TISLAND file somewhere into memory and executes code at the LOAD entry point. I have no idea how it knows the filenames of TISLANE and TISLANF and where in memory to load them. I don't suppose TISLAND is the GROM segment, TISLANE is the C bank, and TISLANF the D bank?

For instance, I couldn't XMODEM these three files over to the TI, run them through XMDM2TI, rename TISLANE to TISLANG.BIN, TISLANE to TISLANC.BIN, and TISLANF to TISLAND.BIN, edit MODULES.INF appropriately, and then run it in V9T9? That's what I was hoping I could do, but I somehow doubt it's that easy. I have a feeling these images have special loading code with the other filenames hard-coded inside. The files are sort of daisy-chained. If this is the case, is it easy to convert these modified images into something usable by V9T9?

One critical piece of information buried in V9T9's manual was that the GROM segment always has AA hex as its first byte. However, no indication was given on how to identify a misnamed bank C file from a bank D one. Is there a way besides trial and error?

5. If I understand right, V9T9 FIADS, DOADS, and modules do not have the 128-byte TI header, but all files transferred to the TI must have a TI header?

The files in the DOADs have exactly the same header as TI diskettes do.

FIADs have the same length header as XMODEM transfers (which you refer to as TI headers), but they differ in content. The FIAD headers contain the name of the file, and if you change the filename in DOS so that it doesn't match the header, V9T9 will return an error message. FIAD names, like the TI, can be up to 10 characters in length. If they are longer than eight characters, V9T9 will put the last two in the extension, such as "TI-ARTIS.T". If you try to use XMDM2TI on a file such as "TIWRITER.ARK," you will get an error message, as V9T9 will try to convert this 11-character filename into a TI file called TIWRITERAR. You will need to rename the file to either TIWRITER.AR or just TIWRITER.

Don't even try XMDM2TI to DOADS. As you've said, the utilities have a problem with DOADS. Use FIADs to transfer the files, then use Disk Manager to copy them to a DOAD.

Most of the files on ftp sites with either .arc or ark extensions are XMODEM transfers. These have a header containing the word TIFILES (your TI header). The header is also exactly the same length as V9T9's FIADS.

**You can see whether the file is a FIAD or and XModem transfer by opening it up in Windows 3.11's Notepad.Exe. If it is an XMODEM file, you'll see the word TIFILES at the beginning. If it is a FIAD, you'll see the name of the file in the header. Alternatively, in DOS, you can just use the DOS command TYPE to check the header. The easiest method is Windows 95'ss Quick View.*

There is a potential improvement for V9T9: I don't know why Ed decided to use a different header for his FIADs — he could have just made it so that V9T9 would simply access TIFILES-type XMODEM transfers in its FIAD directories. Then you could just download a file and stick it straight into one of V9T9s FIAD directories, without having to use XMDM2TI.

Therefore, before sending any file which works with V9T9 to the TI, I need to run it through TI2XMDM first?

Only if you're using a terminal program on the TI which uses XMODEM, and a terminal program on the PC, or in V9T9, which does the same. If you're transferring from V9T9's emulation to the TI, you don't need to do this. For instance, you can save a textfile in V9T9's TI-Writer by saving your file to a filename called RS232.BA=3D2400, and then load this file on your TI's TI-Writer by loading from the same file (I think — lemme check it out).

You can also save BASIC files from V9T9 by typing in SAVE RS232.BA=2400 and using OLD RS232.BA=2400 on your TI. Again, you wouldn't have to use the TI2XMDM program. Only use it if you're using terminal programs to do the transfers.

If I transfer converted text files to the TI, is it important to make sure the TI file type is DIS/FIX 80?

I'm not sure that TI-Writer would accept it as such. Again, I'd have to check this out.

How about a transferred module image — does it have to be of program type in order to work with CALL LOAD/LINK?

In general, I'd say my biggest problem with V9T9 and all the TI files I have is the lack of detailed explanations about the various file formats. I'm never sure when I need a TI header, a program-type, or DIS/FIX type, what type of module bank I have, or when a file off the Internet is a FIAD or ready to be XMODEMed binary file for the TI. Is there some-

Continued on page 16

V9T9

V9T9

Continued from page 15

thing that explains all of these things without getting too bogged down in the gory details? It would be enough if I could visually inspect a hex byte listing of any TI file sitting on my PC and know enough to determine the file type and format.

Whew! I think those are all of my questions. If I could get V9T9's utilities working and if I understood what I can and can't do regarding transferring module images to and from the TI, I'd be a happy camper. Right now I'm just a tired, frustrated camper.

Any help would be greatly appreciated, even if it's just the answer to one of my many questions. As Princess Leia might say, "Help me, TI-99ers, you're my only hope." Thanks.

That's it for now, but here is what's on my list to check out for you (just to remind me...

- *Try the utilities on DOADs with an older version of DOS (I don't have one at the moment).*
- *Get Forth to run again, and find out why it's crashing.*
- *Check if you can transfer a DIS/VAR 80 or DIS/FIX 80 file from TI-Writer on both machines.*
- *Try transferring MUNCHMNG.BIN to the TI.*
- *Get my OBTRANS and OBRECV files back from wherever they are again — I haven't needed them since I transferred all of my modules and disks.*

THE FOLLOW-UP

OK, here are some results to the things I had to try two days ago:

- *Try the utilities on DOADs with an older version of DOS.* I still haven't gotten my hands on an older version.
- *Get Forth to run again, and find out why it's crashing.*

I can't seem to get the TRANSFER program to operate consistently under Forth. Sometimes it loads, and sometimes it doesn't. You might find that it freezes up, and then you press CTRL-F12 to restart Forth, and you try again, and it works.

One note about the file Forth.Cnf. Initially, the lines pertaining to the RS232 access, "RS232/1 = 2,4", are preceded by "#". This means that they are ignored by Forth, and you have no RS232 access. You have to remove the # sign from the line which pertains to RS232/1, and you have to find out which COM: port on your PC that you have linked to the TI, and what IRQ it employs.

For example, if you are using COM1 for your mouse and you have COM2: linked to your TI, and your COM2 port uses IRQ 4, you need to set this line to "RS232/1=2,3". This is my situation.

Anyway, just forget about Forth. It has never worked for me. Since I know that you already have Extended BASIC in V9T9, because you downloaded the ZIP file that I put together, you might as well do the transfer IN V9T9. Do it like this (assuming that the files Util.Cnf, Utils.Bat and Utils.Inf in your V9T9 directory are still more or less the same as when you downloaded them):

V9T9

1. Make sure that the RS232/1 setting in Utils.Cnf contains the correct references to COM port and IRQ setting, then run Utils.Bat.

2. Type in Ed Swartz's program, and save it on your TI as DSK1.RECEIVE.

```
80 REM Receiver for TRANS ob
ject code from PC
300, 600, 1200, 2400, 4800,
9600
100 INPUT "Enter the baud ra
te:  *:B
110 OPEN #1:"RS232.BA="&STR$
(B)&".DA=8.PA=N",UPDATE, VAR
IABLE 81,INTERNAL
120 OPEN #2:"DSK1.TRANS",OUT
PUT,DISPLAY,FIXED 80
130 PRINT "Receiving..."
140 INPUT #1:A$
150 IF A$="" THEN 200
160 PRINT #2:A$
170 PRINT
180 PRINT #1:"O"
190 GOTO 140
200 CLOSE #2
210 CLOSE #1
220 CALL INIT
230 CALL LOAD("DSK1.TRANS")
240 CALL LINK("TRANS")
```

3. Go into Extended BASIC (or preferably Super Extended BASIC, since I know that you have it) from Utils.Bat, and type in the following program in V9T9:

```
80 REM Transmitter for TRANS
 object code to TI
90 REM Legal baud rates: 300
, 600, 1200, 2400, 4800, 960
0
100 INPUT " Enter the baud ra
te: ":B
110 OPEN #1:"RS232.BA="&STR$(
B)&".DA=8.PA=N",UPDATE,VARIAB
LE 81,INTERNAL
120 OPEN #2:"DSK3.TRANS",INPU
T,DISPLAY,FIXED 80
130 PRINT "Transmitting... "
140 IF EOF(2) THEN 200 ELSE I
NPUT #2:A$
150 PRINT #1:A$
170 PRINT
180 INPUT #1:B$
190 GOTO 140
200 CLOSE #2
210 PRINT #1
220 CLOSE #1
```

I wrote this program as closely as possible to Ed's RECEIVE program. Save it in V9T9 as DSK3.TRANSMIT.

4. Run both of the programs. Now you have the RECEIVE program running on your TI, and the TRANSMIT program running on your other "TI." Make sure that your machines are connected correctly, then enter the baud rate on both machines (9600 worked fine for me, but you can also enter 4800, 2400, 1200, 600 or 300), pressing Enter on your TI first.

5. Assuming that the TRANS file is in the DISK directory of V9T9, which was configured as DSK3. in Utils.Cnf, the programs should both start running, without any problems. I just tried this. You will see a period for every record that is transferred and saved. After a few lines of

Continued on page 18

V9T9

Continued from page 17

periods, the program will end on V9T9, and you'll have to be patient on the TI side, because it will be busy loading the TRANS program.

6. If everything is OK up until this point, you should exit V9T9 with Ctrl-Break, and then, if you're in V9T9's directory, type in "RECV." This will run the PCs receiver program under DOS. By this time your TI should have started running the program TRANS. Now follow the instructions, and you should be able to transfer all of your modules.

7. Send them to me!

• Get my OBTRANS and OBRECV files back from wherever they are.

OK, you don't need those anymore, now that you have the above two programs. I couldn't find the original two programs, which were freshly written on both machines, but the above program makes use of Ed Swartz's original RECEIVE program.

What I learned from this little adventure, which I didn't realize when I first typed in the program, is this: When I wrote my own program on both sides, I found that they did not have any way of telling the PC to stop sending information. I found that while the TI was writing the first record to disk, the PC would just keep sending, resulting in a buffer overflow on the TI's RS232, which would give you an error message at the next "INPUT #1:A$" statement. This baffled me, and at the time I learned a lot about RS232 cables. But the cabling didn't solve the problem. My eventual solution back then was to introduce a CALL SOUND delay onto the PC side, so that it would give the TI time to write the record to diskette and wait for the next record, before it was sent. This was a really ugly way of doing things, and I didn't see why it was necessary when SAVE and OLD operations via RS232 worked flawlessly.

Then, last night, I checked out Ed's program again. The program on the TI simply sends after each receive! That is, if you look at Ed's program, after every input statement, the TI sends back an "O" character to the PC, which I've made provision for on the PC, and then waits for the next record. My questions to those knowledgeable on the subject are:

Is this called "handshaking?"

Has anyone out there successfully gotten the TRANSFER program in Forth to work? Even when it does run for me, I get I/0 errors — I presume that's because of the absence of this handshaking.

• Check if you can transfer a DIS/VAR 80 or DIS/FIX 80 file, using TI-Writer on both machines.

(Response from jgroslou@nbnet.nb.ca):

Using the terminal program in Windows 3.11 on the PC and TI-Writer on the TI you can transfer to the PC at 9600 baud (19200 on the Geneve) but you can only transfer from the PC to the TI at 600 baud. Apparently this is an inherent limitation of TI-Writer.

The other settings are odd parity and 7 data bits. I assume that you will have the same limitation transferring from v9T9 using TI-Writer on both machine. The above information appeared in an article by Charles Good in the September 1995 issue of *Bits, Bytes and Pixels*, the newsletter of the Lima (Ohio) User Group.

• Try transferring MUNCHMNG.BIN to the TI.

This will have to wait — I've taken my TI apart to try and see if I can organize a composite video output.

V9T9

THE ART OF ASSEMBLY — PART 67

Sounds good to me!

BY BRUCE HARRISON

It's been a while since we've done anything about the TI sound chip. Once again, however, we've been inspired by other programmers. Vern Jensen asked about whether the sound list method that's available to Assembly programmers could be made to work from c99 programs. Yes, it can, and in creating some routines for use with c99, we also added new tricks for the Assembly programmer, and even an easier method for preparing the sound lists to use with these routines.

CONTINUOUS BACKGROUND

The first thing we wanted to do was to make it possible to have a "theme song" play continuously in the computer's interrupt cycle, while allowing the main program to continue doing its thing. Since no sound list can last forever, we set up a User Interrupt to sense when the tune is finished and start it over again. All that the main program needs to do is cycle through LIMI 2 and LIMI 0 now and then, so that sound processing can be done through the code in ROM and the User Interrupt.

WE INTERRUPT THIS INTERRUPT

Okay, once we had the theme playing all the time and the program able to do other stuff as the theme played on, the thought came to us: "What about sound effects?" The idea then was to have an effect sound list interrupt the theme for a short time, then have the theme pick up where it left off. This idea was aimed at games, where some event in the game may need to start a sound effect and perhaps visual effects as well, but then the theme should pick up at its next note. This led to another User Interrupt that would time out the sound effect and then turn control back to the theme's User Interrupt with the theme then resuming at whatever was the next note in its list.

TWO OTHER THINGS

There are two other ways to run the sound list. One should be able to just run a sound list one time through, then have it stop, but have the main program continue unhindered. Finally, one should have a way to let the program be

Continued on page 20

THE ART OF ASSEMBLY Part 67

Continued from page 19

delayed until a sound list finishes. We've put those two capabilities in as well.

THE BOUNCING BALL

No, we didn't really make a ball bounce, but we provided a "pacing" capability so that, while a sound list is playing, we can let the main program check on its progress. Thus we can have the main program perform actions that are paced by the notes in the music, yet still be able to perform other actions, like keyboard scans, during the music. While we were at that, we added a quick check so that if the music is on a one-time playing, the main program can determine whether it's finished that play or not.

LOADING THE LISTS

To make the placing of sound lists into VDP RAM quick and easy, we created a routine that will load up to 10 sound lists at one whack, and had this routine stash away the VDP locations of each in a lookup table. This way, the sound lists can be activated selectively whenever needed. The lists themselves can be either within the main program's object file or in separate object files with REFs and DEFs linking them up at load time. The calling of this routine is different for c99 and Assembly, even though the internal process is similar Let's say we want to load three lists called CHIME, CRASH, and VIV. From c99, we'd do this:

```
setsou(0,chime,crash,viv);
```

From Assembly, that would be:

```
BLWP @SETSOU
DATA 3,CHIME,CRASH,VIV
```

In either case, the lists would be loaded and referenced in the order given, so that activation would be by number, with 1 being CHIME, 2 being CRASH, and 3 being VIV. In the c99 case, the routine works backwards through the c99 stack until it finds that 0, which is how it "knows" how many lists to load. In the Assembly case, it just takes the first DATA after the BLWP as the number of lists to load. In either case, it's important that each list itself ends with a note of zero duration. Our usual practice is to make the last note in a list be a silence for all four generators. Thus the last line in our source list would look like this:

```
BYTE 4,>9F,>BF,>DF,>FF,0
```

That zero duration is important anyway because it signals the Interrupt Handler in ROM when a sound list ends. Our loading routine uses that 0 duration to determine the size of the list, so it knows how many bytes to dump into VDP RAM. The loading routine does a very quick scanning of each list, stopping when it finds 0 duration, then dumps the list into VDP by a VMBW operation. The lists are actually loaded backwards in sequence, with the last byte of the first list being at the address pointed to by >8370 when the routine is called. Each successive list is then loaded at lower addresses, until all the lists are in place. As it loads each list, the routine updates the word at >8370, and puts each list's VDP address into a lookup table in its own memory space. Later

THE ART OF ASSEMBLY Part 67

the activation routine uses that lookup table to tell the sound processor where the selected list starts.

ACTIVATING SOUND LISTS

When the lists get loaded, no sound starts. The actual playing of the lists is done by activating them. Activation requires two parameters, the first being the number of the particular list from the loading process, and the second being the type of play for that list. The number is just 1 through n, where 1 is the first one named in the loading process and n the last. The activation checks the number against the number actually loaded, and will do nothing if you're asking to play number 4 when you've only loaded 3. It also will reject 0 or any negative number.

We mentioned earlier that there are four ways to use a list. Here are the four types of play:

Type 1 — Play continuously with repeats when done.
Type 2 — Play once through on background, then stop.
Type 3 — Interrupt playing of a type 1 for one play.
Type 4 — Play once but delay main program.

In any of these cases, the call to the activation routine gets two parameters. From Assembly:

```
BLWP @STSOU
DATA NUMBER,TYPE
```

From c99:

```
stsou(number,type);
```

Let's suppose we're using number 3 sound list as a theme song that will run all the time in a game program. We'd start that in the beginning of the game by BLWP @STSOU with DATA 3,1. It would start playing, repeating itself when it finished, and so continuing "forever" in the background.

Let's then say we reached a point in the game where some event (like a sprite coincidence) occurred, and a crash sound effect was needed. We'd just BLWP @STSOU with DATA 2,3. The crash would stop the theme, play itself in the background, then the theme would resume at its next note.

One very important service is provided in the routines, called ENDSND. This not only stops any sounds that were playing, but restores the original state of >8370 and in effect removes the lists loaded into VDP RAM. It's important that your programs use BLWP @ENDSND before exiting.

There's also an emergency stop, which will stop any sound that's playing, but won't affect the lists like ENDSND. This is called STPSND, and is invoked by just BLWP @STPSND, with no parameters. This stops the current sound, but does not affect the lists in VDP RAM, nor does it restore the original value to >8370.

MAKING SOUND LISTS

This has been a terrible chore. If you're doing anything that involves musical

Continued on page 22

THE ART OF ASSEMBLY — Part 67

Continued from page 21

notes instead of just noises, it's a bear! Not only do you have to have some knowledge of music, but you have to translate music into hex code. Many of us learned in grade school that the staff with a G clef could be read by remembering two simple mnemonics: The lines starting at the bottom are E, G, B, D, and F. These can be remembered as the initials of Every Good Boy Does Fine. The spaces are F, A, C, and E. These can be remembered as spelling the word FACE. (Please don't ask about ledger lines.)

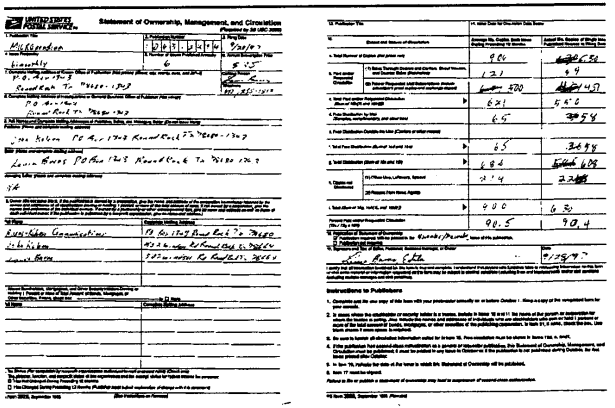
Given this, your author can plow through a piece of music, reading one note at a time. The hex part has to be looked up in that great book, the Editor/Assembler manual, also one note at a time.

Now let's take a trip down memory lane. About 10 years ago, your author devised a method whereby his partner Dolores P. Werths, who's a musician, could translate music from the printed staves into a quasi-musical notation in source files, and then by using auxiliary files of EQUates, the Assembler would generate the hex. That technique, in modified form, has been reinvented for the rest of you, so that you can write your Sound List source files in musical terms, then have the Assembler convert your musical instructions into hex codes for the sound chip.

Just for example, suppose you wanted Generator 1 to play a C of the fourth octave, at volume level 3, for a duration of one quarter note. You'd put this into the source file:

```
BYTE 3,G1+C4A,C4B,G1V+3,Q
```

Notice that there are two pieces of the note C4, a part called C4A, which gets



THE ART OF ASSEMBLY — Part 67

added to the generator number (G1), and a second part, which is a byte of its own as C4B. To indicate a volume, you tack a V onto G1, to get G1V, then add the level desired, in this case 3. The duration is just a single letter Q for quarter note. The 3 after the word BYTE is just to tell the ROM software that what follows are three bytes to be sent to the sound generator. That is, G1+C4A, one byte, C4B, a second, and G1V+3, a third. The Q is one byte, too, but that byte doesn't go to the sound generator. The ROM software uses that byte to count the duration of this note in 60ths of a second. To get the Assembler to translate what you see above into hex, we use two "COPY" files that are nothing but EQUates. The note frequency and volume equivalents are in NOTES, and the durations in TEMPO.

HOW DO I GET THIS STUFF?

Easy as a $2 apple pie. (Where in this world could you get an apple pie for $2?) There are three disks in the set, of which you'll need only two. For either c99 or Assembly programmers, you get the disk LISTMAKER. Then you add to that either the CSOUND disk or the ASMSOUND disk as appropriate to your choice of programming language. You send $1 per disk to the Lima Users' Group, c/o Dr. Charles Good, P.O. Box 647, Venedocia, OH 45894. Each of these disks contain complete instructions, demo programs written in the language of your choice, all the source files, object modules, and even some ready-made sound lists that you may find useful and amusing. We guarantee these will last longer than a $2 apple pie, too.

TODAY'S SIDEBAR

In the sidebar is the source code for the Assembly version of the sound list loading and activation subroutines. We've annotated this fairly completely, so at least our Assembly programmers should be able to follow it. If you get MICROpendium on disk, I've included the two demo programs DEMO and DEMO2 in E/A Option 5 form, and the sidebar's source code as an object module called SIDE67/O. The code as shown in the sidebar is complete, and will Assemble into an object module, ready for use. The programs DEMO and DEMO2 are the demos from our ASMSOUND disk. There are similar demo programs on the CSOUND disk, with the names SOUND and PACED.

Next month we're going back into the Bit Map, but we'll be dealing with TI-Artist Instances.

SIDEBAR 67

```
0001  *  SIDEBAR67
0002  *
0003  *  A COMPLETE MODULE
0004  *
0005  *  PUBLIC DOMAIN SOFTWARE
0006  *  SOUND LIST PROCESSING
```

Continued on page 24

THE ART OF ASSEMBLY — Part 67

Continued from page 23

```
0007  *  USER MAY LOAD UP TO 10 SOUND LISTS
0008  *  AND SELECTIVELY ACTIVATE THEM
0009  *
0010  *  by Bruce Harrison
0011  *  01 SEP 1995
0012             DEF     SETSOU,STSOU,ENDSND,STPSND
0013             DEF     PACEIT,FINEPC,SETPAC,TESTSO
0014             REF     VMBW,VSBR
0015  *
0016  * SETSOU PLACES THE SOUND LISTS IN VDP RAM
0017  * FOR ACTIVATION WHEN DESIRED
0018  * INVOKE SETSOU BY:
0019  * BLWP @SETSOU
0020  * DATA (NUMBER OF LISTS TO FOLLOW)
0021  * DATA LIST1,LIST2,LIST3,...LIST 10
0022  *
0023     SETSOU  DATA    SWS,SETSO
0024     SETSO   MOV     *R14+,R8NUMBER OF LISTS TO R8
0025             MOV     R8,@NOLST       PLACE AT NOLST
0026             MOV     @>8370,@OLD70   SAVE ADDRESS FROM >8370
0027             LI      R7,LISTBL       POINT AT TABLE OF ADDRESSES
0028     STSOU0  MOV     *R14+,R4GET NEXT CPU LIST ADDRESS
0029             MOV     R4,R1   ADDRESS IN R1
0030             CLR     R2      R2=0
0031     SETSO2  MOVB    *R4+,R6 LENGTH BYTE IN R6
0032             SRL     R6,8    MAKE A WORD
0033             A       R6,R2   ADD LENGTH TO R2
0034             INCT R2         ADD TWO FOR DURATION & LENGTH BYTES
0035             A       R6,R4   ADD LENGTH TO ADDRESS
0036             MOVB    *R4+,R5 GET DURATION BYTE
0037             JNE     SETSO2  IF NOT ZERO, REPEAT
0038     * IF THE DURATION BYTE IS NOT 0, WE'RE
0039     * NOT AT THE END OF THE SOUND LIST, SO WE
0040     * KEEP ON GOING. R4 POINTS TO LENGTH BYTE OF NEXT GROUP
0041             MOV     @>8370,R0       LAST AVAIL VDP ADDR TO R0
0042             S       R2,R0   SUBTRACT LENGTH OF SOUND LIST
0043             BLWP    @VMBW   WRITE LIST TO VDP
0044             MOV     R0,*R7+ PUT VDP ADDRESS IN TABLE
0045             DEC     R0      ONE BYTE BACK
0046             MOV     R0,@>8370       NEW LAST AVAILABLE TO >8370
0047             DEC     R8      DEC COUNT OF LISTS
0048             JGT     STSOU0  IF > ZERO, DO ANOTHER
0049     SETEX   RTWP            ELSE RETURN TO CALLER
0050     *
0051     * STSOU STARTS A SOUND THAT WILL CONTINUE OR NOT
0052     *       INVOKE BY :
0053     *       BLWP @STSOU
0054     *       DATA LIST#,TYPE
0055     *       list # = number of list (1-10)
```


THE ART OF ASSEMBLY — Part 67

```
0056     *       type = 1,2,3, or 4
0057     *              1 means continue indefinitely, with repeat
0058     *              2 means play once, then stop sound
0059     *              3 means play and then continue previous list
0060     *              4 means play and suspend other action until list
finishes
0061     STSOU   DATA    SWS,STSO
0062     STSO    MOV     *R14+,R4LIST NUMBER TO R4
0063             MOV     *R14+,R7TYPE OF PLAY TO R7
0064             CI       R7,1   COMPARE TYPE TO 1
0065             JLT     SETEX   IF LESS, EXIT
0066             CI       R7,4   COMPARE TO 4
0067             JGT     SETEX   IF >, EXIT
0068             C       R4,@NOLST       CHECK NUMBER OF LISTS LOADED
0069             JGT     SETEX   IF R4 >, EXIT
0070             DEC     R4      ZERO BASE
0071             JLT     SETEX   IF <0, EXIT
0072             SLA     R4,1    DOUBLE NUMBER IN R4
0073             LIMI 0          STOP INTERRUPTS
0074             CI      R7,3    TYPE 3?
0075             JNE     NOT3    JUMP IF NOT
0076             MOV     @>83CC,@SAVCON  ELSE SAVE PRESENT PLACE
0077     NOT3    MOV     @LISTBL(R4),R0  GET THE LIST'S ADDRESS
0078             MOV     R0,@>83CC       MOVE ADDRESS TO >83CC
0079             SOCB    @H01,@>83FD     SET VDP FLAG
0080             MOVB    @H01,@>83CE     TRIGGER SOUND PROCESSING
0081             CI      R7,4    TYPE 4?
0082             JEQ     TYPE4   IF SO, JUMP
0083             CI      R7,1    TYPE 1?
0084             JEQ     STCNT   IF SO, JUMP
0085             CI      R7,3    TYPE 3?
0086             JEQ     SET3    IF SO, JUMP
0087     CLRALL  CLR     @>83C4  NO USER INTERRUPT
0088             CLR     @SAVCON CLEAR SAVCON
0089             CLR     @RESADR CLEAR RESET ADDRESS
0090             JMP     SETEX   THEN EXIT
0091     SET3    MOV     @CK2ADR,@>83C4  IF TYPE 3, DIFFERENT INTERRUPT
0092             JMP     SETEX   THEN EXIT
0093     STCNT   MOV     @CKADR,@>83C4   PUT INTERRUPT'S ADDRESS IN PLACE
0094             MOV     R0,@RESADR      SET FOR REPEATS
0095             JMP     SETEX   RETURN TO C99
0096     TYPE4   CLR     @>83C4  CLEAR USER INTERRUPT
0097             CLR     @RESADR CLEAR RESET ADDR
0098             CLR     @SAVCON CLEAR CONTINUE ADDRESS
0099     T4R     LIMI    2       ALLOW INTS
0100             LIMI    0       STOP THEM
0101             MOVB    @>83CE,R1       CHECK COUNT BYTE
0102             JNE     T4R     IF NOT 0, REPEAT
0103             JMP     SETEX
```

Continued on page 26

THE ART OF ASSEMBLY — Part 67

Continued from page 25

```
0104    *
0105    * ENDSND MUST BE USED JUST BEFORE
0106    * EXITING YOUR PROGRAM
0107    * INVOKE ENDSND BY:
0108    *       BLWP @ENDSND
0109    *       stops all sound generation
0110    *       (unconditional)
0111    *       cancels all lists loaded
0112    *
0113    ENDSND  DATA    SWS,ENDSN
0114    ENDSN   LI      R1,SNDOFF       SILENCE SOUND STRING
0115            MOVB    *R1+,R2 LENGTH BYTE TO R4
0116            SRL     R2,8    RIGHT JUSTIFY
0117    MOVLP   MOVB    *R1+,@>8400     ONE BYTE TO SOUND CHIP
0118            DEC     R2      DEC COUNT
0119            JNE     MOVLP   IF NOT ZERO, REPEAT
0120            MOVB    @ZERO,@>83CE    CLEAR THE SOUND COUNT
0121            SZCB    @HFE,@>83FD     AND THE SOUND FLAG
0122            MOV     @OLD70,@>8370   OLD ADDRESS BACK IN >8370
0123            CLR     @NOLST  CLEAR NUMBER OF LISTS
0124            JMP     CLRALL  THEN CLEAN UP
0125    *
0126    * STPSND SIMPLY STOPS ANY SOUND LIST INSTANTLY
0127    * INVOKE BY:
0128    *               BLWP @STPSND
0129    *
0130    STPSND  DATA    SWS,STOP
0131    STOP    LI      R1,SNDOFF+1     SILENCE BYTES
0132            LI      R2,4    FOUR TO SEND
0133    SILEN   MOVB    *R1+,@>8400     SILENCE ONE
0134            DEC     R2      DEC COUNT
0135            JNE     SILEN   RPT IF NOT 0
0136            MOVB    @ZERO,@>83CE    ZERO COUNT
0137            SZCB    @HFE,@>83FD     RESET VDP FLAG BIT
0138            CLR     @>83C4  CLEAR USER INTERRUPT
0139            RTWP            RETURN
0140    *
0141    * CKS2 AND CKSND ARE USER INTERRUPTS
0142    * FOR HANDLING CONTINUING SOUND LISTS
0143    *
0144    CKS2    MOVB    @>83CE,@>83CE   SOUND STILL PLAYING?
0145            JNE     QRET    IF SO, EXIT
0146            MOV     @RESADR,@RESADR ANY PREVIOUS LIST IN PLAY?
0147            JEQ     QRET    IF NOT, EXIT
0148            MOV     @SAVCON,@>83CC  OLD LIST POSITION BACK
0149            MOV     @CKADR,@>83C4   SET OTHER USER INTERRUPT
0150            JMP     REST2   THEN JUMP AHEAD
0151    CKSND   MOVB    @>83CE,@>83CE   CHECK THE SOUND PROCESSING BYTE
0152            JNE     QRET    IF NOT ZERO, EXIT
```


THE ART OF ASSEMBLY — Part 67

```
0153    RESET   MOV     @RESADR,@>83CC  ELSE RE-SET VDP ADDRESS
0154            JEQ     QRET    IF NO RESET, STOP
0155    REST2   SOCB    @H01,@>83FD     SET VDP FLAG
0156            MOVB    @H01,@>83CE     TRIGGER SOUND PROCESSING
0157    QRET    RT              RETURN TO INTERRUPT HANDLER
0158    * PACEIT
0159    * PACES BY A RUNNING SOUND LIST
0160    * CAN BE USED TO TIME NOTE BY NOTE
0161    * OR BY 1/60 SECOND INTERVALS
0162    * PACEIT MEASURES BY NOTES
0163    * FINEPC MEASURES BY 1/60THS
0164    * SETPAC INITIALIZES STATES
0165    *
0166    * INVOKE PACEIT BY:
0167    * BLWP @PACEIT
0168    * on exit STATUS EQ IF SAME NOTE
0169    *       STATUS NE IF NEW NOTE
0170    *
0171    PACEIT  DATA    SWS,PACE
0172    PACE    MOVB    @>83CE,R1       ANYTHING IN >83CE
0173            JEQ     PACEX   IF ZERO, NO SOUNDS PLAYING, EXIT
0174            C       @>83CC,@OLDCC   COMPARE POINTER TO OLD VALUE
0175            JEQ     PACEX   IF EQUAL, SAME NOTE STILL PLAYING
0176            MOV     @>83CC,@OLDCC   ELSE NEW NOTE STARTED
0177    PACEX   STST    R15     STATUS REGISTER TO R15
0178            RTWP            THEN RETURN
0179    * FINEPC REPORTS WHETHER A 1/60 HAS PASSED
0180    * SINCE LAST CALL
0181    * INVOKE FINEPC BY:
0182    * BLWP @FINEPC
0183    * on exit STATUS EQ IF LESS THAN 1/60th ELASPSED
0184    *       STATUS NE IF 1/60TH HAS PASSED
0185    *
0186    FINEPC  DATA    SWS,FINP
0187    FINP    CB      @>83CE,@OLDCE   SAME 1/60TH INTERVAL?
0188            JEQ     PACEX   IF SO, EXIT
0189            MOVB    @>83CE,@OLDCE   ELSE REPLACE SAVED VALUE
0190            JMP     PACEX
0191    * SETPAC STARTS THE PACING PROCESS
0192    * INVOKE BY:
0193    * BLWP @SETPAC
0194    * DO THIS IMMEDIATELY AFTER STARTING SOUND LIST
0195    *
0196    SETPAC  DATA    SWS,SETPC
0197    SETPC   MOV     @>83CC,@OLDCC   GET EXISTING SOUND POINTER
0198            MOVB    @>83CE,@OLDCC   AND 60THS COUNT
0199            JMP     PACEX   THEN EXIT
0200    *
0201    *       TESTSO REPORTS WHETHER A SOUND LIST IS
```

Continued on page 28

THE ART OF ASSEMBLY — PART 67

Continued from page 27

```
0202    *       CURRENTLY RUNNING
0203    *       INVOKE BY:
0204    *       BLWP @TESTSO
0205    *       on return, STATUS EQ IF NO SOUND LIST RUNNING
0206    *                  STATUS NE IF SOUND LIST RUNNING
0207    *
0208    TESTSO  DATA    SWS,TESTS
0209    TESTS   MOVB    @>83CE,R1       MOVE THE COUNT BYTE
0210            JMP     PACEX   THEN JUMP
0211    *
0212    * DATA SECTION
0213    *
0214    OLDCE   BYTE    0       STORAGE FOR 60THS COUNT
0215    OLDCC   DATA    0       STORAGE FOR POINTER
0216    SWS     BSS     32      SOUND WORKSPACE
0217    CK2ADR  DATA    CKS2    ADDRESS OF SPECIAL INTERRUPT
0218    CKADR   DATA    CKSND   ADDRESS OF OUR INTERRUPT
0219    ZERO    BYTE    0       ZERO BYTE
0220    H01     BYTE    >01     BYTE OF 1
0221    HFE     BYTE    >FE     BYTE OF -2
0222    NOLST   DATA    0       NUMBER OF LISTS LOADED
0223    RESADR  DATA    0       ADDR FOR CONTINUOUS SOUND PLAY
0224    SAVCON  DATA    0       CONTINUE SOUND ADDR
0225    LISTBL  DATA    0,0,0,0,0,0,0,0,0,0     TABLE OF SOUND LIST
ADDRESSES
0226    SNDOFF  BYTE    >04,>9F,>BF,>DF,>FF
0227    OLD70   DATA    0
0228            END
```

Web site dispels virus rumors

Reports of computer viruses frequently appear in user group newsletters as filler items. They also get bandied about over the Internet to the extent that it's almost impossible to tell the rumors from the reality.

But there's a place on the Internet you can go to get the facts on the latest viruses. It's called the Computer Incident Advisory Capability, CIAC. It's run by the U.S. Department of Energy. You can find it on the Web at http://ciac.llnl.gov/ciac/.

This site pays serious attention to viruses and computer security and is primarily aimed at aiding government contractors and agencies. However, it's also available to the public — you tax dollars at work. If you access the site, your host name and e-mail address will automatically logged by the system. Apparently, this is done to discourage hacking.

But if you've got nothing to hide and aren't the type to be spreading around viruses or breaking into top secret computer networks, it's a great place to go to get the facts behind computer security and virus rumors.

LOAD MAKER

Load Maker is a fine example of an XBASIC auto-loader

Load Maker by Karl Romstedt is an excellent example of an Extended BASIC auto-menu maker. The program creates an easy-to-use listing of a disk's contents that is saved in merge format under the name of LOAD. Load Maker defaults to drive 1, but it is a simple matter to create a routine specifying input and target drives.

The program reads file names from drive 1 and then saves them as a merge file. You can resave it as a program file so that it will autoload the next time you use the disk.

When loaded, the LOAD program displays file names on the left side of the screen, followed by the size of the program and file type. Fifteen files are displayed. Pressing the Enter key selects a file to run. For disks that contain a large number of files, additional pages of file names by pressing the page number. If you try to load a text file, the program will play a tone and wait for another selection. If you select an assembly program, it will try to load it and crash. It wouldn't take too much time to for a user to improve the error-handling. Otherwise, it does its job very well.

LOAD MAKER

```
100 CALL CLEAR :: FOR I=0 TO
 12 :: CALL COLOR(I,2,8):: N
EXT I :: CALL SCREEN(5)!185
110 DIM A$(127),B$(127),C$(1
27),E(127),T$(5)!248
120 DISPLAY AT(3,4)BEEP:"Rom
stedt's  Load Maker" :: DISP
LAY AT(8,1):"This program cr
eates a fast,attractive disk
 load menu." !191
130 DISPLAY AT(10,1):"You wi
ll need to remake the load p
rogram if you change  the co
ntents of the target  disk."
 !101
140 DATA dis/fix,dis/var,int
/fix,int/var,program !084
150 FOR I=1 TO 5 :: READ T$(
I):: NEXT I !228
160 DISPLAY AT(20,1):"Place
the Target disk in     drive
1 and press any key." !163
170 CALL KEY(0,K,S):: IF S<1
 THEN 170 !240
180 DISPLAY AT(10,9)ERASE AL
L:"Reading data" :: DELETE "
DSK1.LOAD" !002
190 OPEN #1:"DSK1.",INPUT ,R
ELATIVE,INTERNAL :: INPUT #1
:D$,J,J,K !012
200 FOR I=1 TO 127 :: INPUT
#1:A$(I),A,E(I),S :: IF LEN(
A$(I))=0 THEN 240 !169
210 A=ABS(A):: C$(I)=T$(A)::
 IF A<>5 THEN L$=STR$(S):: C
$(I)=C$(I)&RPT$(" ",4-LEN(L$
))&L$ !236
220 FOR J=1 TO LEN(A$(I))::
```

Continued on page 30

LOAD MAKER

Continued from page 29

```
K=ASC(SEG$(A$(I),J,1)):: IF
K>63 AND K<91 THEN K=K+32 !1
71
230 B$(I)=B$(I)&CHR$(K):: NE
XT J :: N=N+1 :: NEXT I !150
240 CLOSE #1 :: DISPLAY AT(1
0,1)BEEP:"Press any key to s
ave LOAD  as a merge file."
!041
250 CALL KEY(0,K,S):: IF S<1
 THEN 250 ELSE DISPLAY AT(10
,4)ERASE ALL:"Saving DSK1.LO
AD,MERGE" !141
260 OPEN #1:"DSK1.LOAD",VARI
ABLE 163 !135
270 L=110 :: L$=CHR$(0)&CHR$
(L)&CHR$(147)&CHR$(200)&CHR$
(LEN(D$))&D$ !025
280 L$=L$&CHR$(179)&CHR$(200
)&CHR$(LEN(STR$(N)))&STR$(N)
:: FOR I=1 TO N !140
290 L1$=CHR$(200)&CHR$(LEN(A
$(I)))&A$(I)&CHR$(179)&CHR$(
200)&CHR$(LEN(B$(I)))&B$(I)&
CHR$(179)&CHR$(200)&CHR$(LEN
(STR$(E(I))))&STR$(E(I))!080
300 L1$=L1$&CHR$(179)&CHR$(2
00)&CHR$(LEN(C$(I)))&C$(I)!2
38
310 IF LEN(L$)+LEN(L1$)<162
THEN L$=L$&CHR$(179)&L1$ ELS
E L$=L$&CHR$(0):: PRINT #1:L
$ :: L=L+1 :: L$=CHR$(0)&CHR
$(L)&CHR$(147)&L1$ !217
320 NEXT I :: L$=L$&CHR$(0):
: PRINT #1:L$ !045
330 PRINT #1:"dA$127B
$127C$127E127
PRODUCT OF ROMSTEDT'S LOAD M
AKER" !085
340 PRINT #1:"M1D$N
K1NA$KB$KEKC
$KKPPN15.01S1
1S215N" !101
350 PRINT #1:"CLEAR
SCREEN5COLOR02
8128228
3284285
21562157
2158215" !165
360 PRINT #1:"COLOR9
28102811
281228131
515J24HCHAR
J212930J" !028
370 PRINT #1:"328
D$21D$SS1S
2SML$A$SL$B$S" !
023
380 PRINT #1:"KS5PG
15K1L$K13
###ESK18C$S
S" !071
390 PRINT #1:"221
arrows select, enter run
s.pagePG1ofPP1
-digit for page." !226
400 PRINT #1:"KEY0K
SS0K13
K48K50PPPGK49S
1PG151S2S114N
MS1" !226
410 PRINT #1:"M5PG
151B$MK11K
69K101MM2MS1
1MS21" !041
420 PRINT #1:"MM1MS
```


LOAD MAKER

```
2MS1" !234
430 PRINT #1:"M5PG
151A$M" !062
440 PRINT #1:"program
int/var 254C$M10
SOUND992220
" !060
450 PRINT #1:"SCREEN
8CLEARCHARSETD$
DSK1.A$MRunning D$"
 !200
460 PRINT #1:"PEEK3
1952KSPEEKK256S
65534KSS1K256S
65534" !119
470 PRINT #1:"INITS2
D$LOADS1S2M1S
2LOADS1MD$M1
MLOADS1M0" !1
28
480 PRINT #1:"DSK1.0123
456789" !010
490 PRINT #1:CHR$(255)&CHR$(
255):: CLOSE #1 !109
500 DISPLAY AT(10,1):"To fin
ish load creation,  type":
" ":"NEW":"MERGE DSK1.LOAD":
"SAVE  DSK1.LOAD" !087
```

1997 CHICAGO TI FAIRE

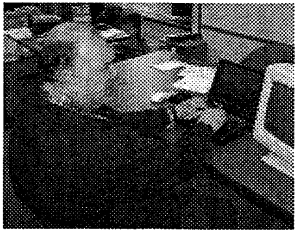
Continued from page 6

built 1 meg RAMdisk for about $180, which is the same price that the original 90K Horizon sold for.

HARMSEN, GOOD FETED

After the Evanston library closed Faire participants adjourned to a nearby grill for a delightful meal paid for by the Chicago User Group. Hal Shannafield then presented the awards to two very surprised individuals. The Grant B. Schmalgemeier memorial award is given annually to a member of the Chicago UG who has made outstanding contributions to the group. This year's recipient was Berry Harmsen in recognition of his efforts in providing news and communication between European and North American user groups. I remember Grant Schmalgemeier from some of the first Chicago Faires I attended in the mid-1980s. Grant had a table near the entrance door and tried his best to personally greet each of the hundreds of fair attendees.

The John Birdwell memorial award was given this year to me, Charles Good, in appreciation for some of the TI fairs I have organized in past years. I had no idea this was coming and am deeply appreciative of the award. John Birdwell, as many of you know, was the author of DSKU and other fine TI software and was a MICROpendium columnist.



Mike Wright shows new power of the PC99 TI emulator.

PROJECT ADVICE

You need to build momentum to overcome the distractions

BY DAVID ORMOND

This article originally appeared in the newsletter of the SouthWest Ninety-Niners.—Ed.

For all those people out there who start something and never finish it, this one's for you.

For all those guys who start a woodworking project and it languishes half-finished on the workbench for years, this one's for you.

For all the ladies who (like my wife) start sewing a dress or a cross-stitch project, and it gets swept into plastic grocery bags, onionskin patterns still pinned to bits of fabric, this one's for you.

And probably for nobody more than those who start a project and then, with the best of intentions of finishing the first, lay it aside to start a second. And a third. Ad infinitum. This one's for you.

My thing with computers is programming, making the machine do what I want. My hat is off to those who use their TI for word-processing, budgets, databases, graphical arts, and all the other classic "home" computing applications. But what really toggles my flip-flop is programming.

The thing about programming is, like any other human activity, the more you do it, the better you are at it. If you don't do enough of it, you never get very far with it. It seems like all I ever do with my TI anymore is the home computing applications, since keeping a budget with Multiplan and writing newsletter articles (especially late ones) are the most justifiable expenditures of my time on the machine. I keep feeling (and rightly, no doubt) that my time on the machine is much resented by the rest of my family. A few hours a month writing, a couple hours a week inputting spreadsheet data, these are bad enough. But many hours each day to get a user interface looking right, several entire evenings debugging an interrupt service routine, and days upon days experimenting with memory management strategies.... I just can't take the reproachful glances of my wife and children as I pound away in my little nook.

MOMENTUM IS CRUCIAL

Momentum is crucial. If you can keep a project rolling along, you can enjoy some hope of seeing it finished. But if you have to put it down for even a little while, it is hard to come back to it. The longer it lies there, the harder it is to even remember where you left off. So, with my background in software engineering, I figured I would be "smart" this time and write what I wanted my primary project to do — in a Software Requirement Specification (SRS) — so that even if I had to lay it aside, as I knew I would, it would be easier upon taking it up again to figure out what the next step needs to be.

Now, if I *really* wanted to do it right, in the MIL-STD-2167A way of doing things, I would have written out a Software Design Document (SDD) with how I would actually build the program, with what software modules, what they would be named, who would call whom, what data items were necessary, etc. Sounds like a lot of work? It is.

I *did* start a SDD, but I began to suspect that the payoff would not be worth the effort. However, in lieu of a formal SDD, and something that 2167A requires anyway, I keep a notebook on my project. The nice thing about these non-code aspects of software development is that you can work on them anywhere. You can review your requirements (is this REALLY what I want?) and doodle down your ideas of how to make things work at the breakfast table, on the bus, on your lunch hour, during slow times at work, even just before you turn out the light at night. A SRS and a Software Development Folder (SDF), aka Software Engineering Notebook, is an excellent way to keep the momentum going on a project.

DISTRACTIONS

Having solved the momentum problem, I immediately proceeded to reintroduce it by allowing myself to be distracted. One of the arguments I use on people who deride me for staying with my 20-year-old "obsolete" computer is that there are literally millions of other programmers turning out third-rate trash for the PC. If money were the only object, as the PC proponents would have use believe, I, programming on the TI, would not be any worse off than the vast majority of PC hobby programmers.

The big difference is recognition. Producing even third-rate trash for the TI, I have a vastly better chance of contributing to the TI world — and being recognized for it — than virtually any PC programmer at any level! There are so many opportunities to contribute to the TI world that it is frustrating to me that I cannot do more. So, when an opportunity comes along of hacking even limited VT-100 terminal emulation capability into the old Paul Charlton Fast-Term program, I cannot resist. Of course, it is mostly for my own use, but did I mention this is my main interest in the TI, seeing it do my things? If anyone else at all finds it useful, that's just gravy! And it really didn't take that long. So I could justify the distraction.

And that would have been bad enough except, like the pneumonia victim, one disease opens the door to others. I was in communication with Don O'Neil, who along with Rich Gilbertson and Kyle Crichton of Competition Computer designed the RXB cartridge, and we were speculating on what else you could do with the technology. The cartridge has a large EPROM and a larger programmable logic device (PLD). The PLD is set up to allow GROM programs (most TI cartridges) and ROM programs to exist on the same device.

So, I got to thinking — shucks, it isn't any great trick to program an EPROM. Why could I not put my

Continued on page 34

PROJECT ADVICE

PROJECT ADVICE

Continued from page 33

own things on this?

Now, I fully intend to buy the RXB cartridge. It's a great program, a truly extended Extended BASIC, including a SCSI manager and support for the Asgard Memory System (AMS). Besides that, the RXB cartridge is Competition's strategy to recoup the investment in the new cartridge technology. As the producer of the AMS, our user group knows what that is, don't we? Once my finances get to the point where I can manage this — my personal allowance account is seriously in the red — I will report to you on this excellent new product.

Once I get the RXB cartridge, I want to get a blank cartridge, hopefully at a much-reduced price, so I can put my own stuff on it.

I have enjoyed my DataBioTics Super Space II cartridge. For those not familiar with the "supercart," it is a cartridge that contains an Editor/Assembler GROM, with an 8K RAM wired into it as ROM (GROM and ROM can coexist in a cartridge). The Super Space II has four additional 8K RAM banks switched in via CRU commands, and it is battery-backed. The problem is that the battery-backup on my two SSII carts don't work.

What is a supercart good for? Some programs, like RAG software tools, detect the supercart memory and use it. But for me, I got hold of John Johnson's old MENU program and tweaked it to load into the cartridge space. So, as long as my computer stays on, longer if the battery backup worked, I can have the excellent MENU/BOOT program available at a key press from the TI menu screen.

As some of you Horizon RAMdisk owners may already know, MENU started life as a Horizon RAMdisk Operating System (ROS) feature. And, as most HRD owners know, the Horizon suffers terribly from having its "root RAM" corrupted. One of the early upgrades to the Horizon was to put the ROS in EPROM. This ROS would never get corrupted again. So I was thinking, it would be sweet to put MENU in EPROM in this cartridge so it would always be there, with no need to fuss over this under-designed battery backup. It wouldn't take much to put an EPROM in mine, but having it as a supercart is still useful. Plus, it's the only one I have available. The other is in my TI at work. And who knows what they would cost today, if you could find one? But if I can get a blank RXB cartridge, I can keep my SSII supercart and have a "cheap" permanent MENU.

That was the long way of saying that I allowed myself to be distracted by another project, of imposing some structure on the MENU program (which was a single blob of the worst-looking spaghetti code I've ever seen), altering it so that all of its variables are in system RAM rather than cartridge RAM, which don't need to be there after we burn it into EPROM, and fixing the configuration system to load from a DSK1 file. This way you won't be stuck with the same configuration for all time being burned into EPROM. You can even change configurations to a load a different set to programs from your DSK1 floppy.

The next variation I will try is to put the AMS ASHOE utility in there, too, which allows you to recover gracefully if your AMS program crashes. I can just dream of having MENU and ASHOE permanently available to me from the TI menu screen. Maybe put the E/A GROM contents in there, too. Maybe put some program, like a TI-Writer clone, in there as well, and load/run instantly at a keypress.

There are so many things to do with my TI. It's frustrating that I don't have the time to do as much as I would like. In some ways I am glad that I don't have more than my $10 per week allowance to spend on computer junk, or I would constantly be getting new hardware toys rather than focusing on my programming projects.

And since that RXB cartridge is still a month away in my budget plans, maybe I will be able to pick up my SDF project notebook and get my original project rolling again.

INTERNET

Using encoding/decoding programs to send TI software over the Internet

BY BRUCE RODENKIRCH

The author is a member of the Cleveland Area User Groups in whose newsletter this article appeared.—Ed.

Using e-mail on a Freenet or Internet is a great way to transmit text, but what does one do to send a program file? A BASIC or Extended BASIC file can be listed as a D/V80 file and transmitted just like an ordinary text file. The recipient must then retype it or convert it to a program file using a program designed to do this.

But what about assembly files, or archived files? The Internet will handle only text files and will filter out characters which have high or low ASCII codes.

In the non-TI world, the Unix utilities UUENCODE and UUDECODE are used to transmit all file formats. Thanks to Clint Pulley, we have a public domain program for the TI and Geneve which will transmit binary files via e-mail. TIED is a c99 program for the TI, and GED is a version for the Geneve. TIED will run on the Geneve in TI mode. TIED can be loaded using Editor/Assembler option 5.

The TIED and GED programs are available from numerous sources, including user group libraries, the

Continued on page 36

INTERNET

Continued from page 35

Internet — try ftp.premierweb.com/pub/systems/ti99/mirrors/opanit/TI994A/ — and the programs are included on the Nov./Dec. 1997 MICROpendium monthly disk.

Here is how to use TIED to send and receive files:

• Prepare a group of files you wish to transmit and archive them.

• Run TIED and enter the filename of the archived file.

TIED will encode this file and produce a D/V80 files. This file can be transmitted via e-mail and the recipient will be able to run the file through TIED which will decode it and produced an archive file which can be unarched in the usual way. Pretty neat, huh?

So far, so good. But there is a caveat. When using Telco to receive a file the logging function will be a nuisance when the file is too long. Telco cannot stop the flow of incoming data when it pauses to save the buffer to disk, which can result in some garbled text. Most Internet services will pause at the end of each screen and, if you are lucky, the log function will happen at the time the download pauses so that there is no data loss.

When a screen is garbled, however, it can be redrawn by pressing CTRL-L. The garbled screen will still be in the buffer but with some careful editing can be removed. I have found that it is easier to do if a logging function is "forced" with FCTN-L immediately before the screen is redrawn. Then, opening a new logging file with FCTN-L will start with good screens. This makes the bad lines easier to find and delete.

It is a good idea to keep the length of the TIED file to less than 40 sectors to avoid the tedious editing of garbled lines. Note the format of the TIED text — two 60-character lines followed by one 51-character line. A break in this pattern indicates an anomaly. I have received some long files from one of the FTP (File Transfer Protocol) Internet sites and they were in TIED format and required some careful editing. You will more likely use TIED in sending archived files to your TI friends via e-mail where you will have some control over file length.

TIED V1.2 will encode D/F128 and I/F128 files (archives, for example) into a D/V80 ASCII representation. It has been contributed to the public domain by Clint Pulley and can be found with documentation on library disks in the Cleveland Northcoast and TI Chips library (95092) and the Akron library (DS-1152). Another library disk includes the TIED and TIED-DOC files (96063 and DS-1228).

I hope I have not made this sound too difficult to use. The TIED program looks at the file you enter and determines whether to DECODE or UNCODE it, so you can't go wrong there. Make sure your e-mail buddies get copies of this so you can sent your favorite programs back and forth. If you use it for a text file, the embedded codes — such as carriage returns — will not be lost.

BEGINNING c99 — Part 7

All about the #include statement

BY VERN JENSEN

For a while now I've been using something in each program that I havent explained very much. You see it at the beginning of each program with the line:

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

The #include statement simply inserts the named file into your source code in the exact spot that the #include statement is encountered.This is done when the program is compiled. You can use the #include statement in many ways.When working on Virus Attack, it allowed me to break my source code up into several files, since the program was too long to fit into the Editor as a single file.The first file had several #include statements at the end that included all the other source files for Virus Attack. And when the program was compiled, the compiler saw them all as one single file.

However, a more common use of the #include statement is to allow the user to access the functions in a library, such as the GRF1 library. A library is simply a set of functions written in either c99 or assembly that have been compiled and assembled. The resulting object code can then be loaded along with your program at run time, allowing your program to use those functions. However how does the compiler know that these functions you are using in your code are going to be loaded later? The answer is in the extern statement.

When you want access to a function that is part of a library that you are going to load at run time, you must name that function with the extern statement before using it. For instance, if the only functions we were going to use in the GRF1 library were the Grf1(), Screen(), ChrDef(), and HChar() functions, we could put this statement at the beginning of our program:

```
extern Grf1(), Screen(), ChrDef(), HChar();
```

However, we would only be able to use those functions, since the C compiler has no way of knowing about a function that is part of a library unless we specifically tell it about the function, since the library is loaded after the program is compiled. Now would be a good time to boot up Funnelweb, fire up the editor, and take a look at the file GRF1;H. (It should be on your work disk, normally in drive 2.) If you do, you'll notice that GRF1;H is simply a bunch of extern statements, naming every single function in the GRF1 library So #including GRF1;H in your program is an easy way to make all the functions in GRF1 available to your program, although you still have to remember to load the GRF1 library later.

The "H" at the end of GRF1;H stands for "Header".The C standard is that

Continued on page 38

BEGINNING c99 Part 7

Continued from page 37

files containing extern statements should be called headers. This makes it clear that it's not a program, and not a library when cataloging your disk. The original c99 4.0 release had the file named as GRF1RF, the RF probably standing for "References" (extern statements are converted into assembler REF statements). I changed this to make it conform more to the C standard.

In addition to sharing functions between object modules, you can also share variables. For instance, if a certain library contains a global variable called myVar that you'd like access to, you can add this statement to your program:

```
extern myVar;
```

Note, however, that for you to have access to this variable, the library must specifically make it available to other object modules with the entry statement, described below. Notice also that there are no parentheses after myVar. If you use parentheses, the c99 compiler assumes it is a function name, and if you don't, it assumes it is a variable name.

If you want to create your own library, you can do so quite easily. Just as you use the extern statement to make external functions and variables accessible to your program, you use the entry statement to make local functions and variables accessible to other object modules. However, it is important to realize that parentheses are not used in entry statements. So to create a library containing a single function, we could write something like this:

```
/* POWER/C */

entry Power;

Power(theNum, power)
int theNum, power;
{
  int result;

  result = 1;
  while (power > 0)
  {
    result = result * theNum;
    power-;
  }

  return result;
}
```


BEGINNING c99 Part 7

Since function parameters are just copies of the values passed to the function, we can modify the parameters without fear of changing any values passed by the caller. In this case, we change the power parameter, decrementing it until it reaches 0, and multiplying the result by theNum each iteration through the loop.

Go ahead and type in the code above, saving it as POWER/C. Then compile and assemble it, saving the compiled result as POWER/S, and the assembled object code on disk 2 as POWER/O. You've just created your first library! Now type in the test program below, saving it as POWTEST/C.

```
/* POWTEST/C - FROM THE NOV/DEC */
/* ISSUE OF MICROPENDIUM       */

  /* This lets us use these external functions */
extern Power(), PutNum(), DisStr();

main()
{
  int n, row;

  row = 5;
  for (n=0; n <= 14; n++)
  {
    DisStr(row,2,"2 to the power of");
    PutNum(row,20,n,0);
    DisStr(row,23,"is");
    PutNum(row,26,Power(2,n),0);
    row++;
  }
}
```

After saving the program above, purge the contents of the editor and type in this CLOADER file, saving it as POWTEST/L.

```
DSK2.CSUP
DSK2.POWTEST/O
DSK2.POWER/O
DSK2.PUTNUM/O
```

This will load CSUP as usual as well as POWTEST/O. However, we also load our library, POWER/O, which was compiled and assembled earlier. The

Continued on page 40

BEGINNING c99 Part 7

Continued from page 39

last file, PUTNUM/O, is one of Bruce Harrison's excellent c99 utilities. You'll need to copy it from his disk (included in the c99 starter kit) onto your "work disk", which should be in drive 2. It provides the functions PutNum and DisStr, for displaying numbers and strings on the screen. Run the program PRINTINST on Harrison's disk to print the instructions for all the c99 utilities on it.

Now compile and assemble POWTEST/C, saving the results as POWTEST/S and POWTEST/O. Then run CLOADER, entering DSK2.POWTEST/L at the prompt. If everything goes as planned, you should see a screen full of the results of multiple calls to the Power function. Congratulations! You've just created your first library and used it in an actual program.

Most libraries are written in assembly language, since it's much faster and more compact if you write it in assembly than if you write it in C. However, there is nothing to stop you from writing libraries in C, as we have just done. I wrote several utilities for Virus Attack in C and compiled them into a library, although before the program was finished I ended up writing to Bruce Harrison, asking him to make faster versions of my routines. Without his help, Virus Attack wouldn't be the same as it is today. This is often how C programs are written, even on modern computers; you write the bulk of the code in C, and the time-critical portions in assembly. This allows you to produce a fast application, but without having to write everything in assembly.

Besides using the #include statement to include header files, you can also use it to include source code directly, as I've already mentioned. So instead of writing POWER/C as shown above, we could eliminate the entry statement, and include POWER/C itself in our program, rather than using the extern statement and loading POWER/O at run time. This produces a more compact result, since everything is put in a single object file, instead of two. This can be important when you write large programs, since eventually you'll run into the limits of the 99/4A's memory, and will want to do everything possible to keep your program small enough to load.

There is, however, a drawback to including source code directly into your program instead of loading it later as an object file at run time. The drawback is that the source code must be compiled every single time you compile your program, resulting in an additional delay. How much of a delay it adds depends on how long the file is, and whether it's written in C or assembly. (Assembly goes much faster, since it doesn't need to be compiled.) In fact, the RANDOM;C file on the c99 Libraries disk uses this exact method; you include RANDOM;C directly into your program, rather than loading a library at run-time. It doesn't add much of a delay to the compilation process, since the routines in RANDOM;C are written in assembly. (Load the file and see for yourself.)

By now you probably know more about the #include statement and libraries than you ever wanted to know. So I'll leave you to ponder what we've just

BEGINNING c99 Part 7

covered, while I consider what to write about next issue.

CONTACT INFORMATION

We're now on our seventh article, and I'm curious to know how many of the 20+ people who ordered the c99 Starter Kit are still on board. Do you find the articles helpful? Confusing? I'm especially interested in hearing from those who had never programmed in c99 before I started this series. Do you now find yourself able to code simple programs, or did you get lost a couple of articles back? Knowing where you are will help me as I write articles in the future.

My new e-mail address is Jensen@loop.com. And as always, you can write to me at Vern L. Jensen, 910 Linda Vista Ave., Pasadena, CA 91103.

SUNRISE/SUNSET

Use your TI to calculate time of sunrise and sunset anywhere

Do you want to know when the sun is going to rise or set in your neck of the woods? You could look in a local newspaper and wait for the local evening weather news. But if you don't have a newspaper or can't wait for the weather report, you can run the following program get an answer immediately.

SUNRISE-SUNSET was originally written by J. Campbell. It was translated for the TI by K.D. Wentzel., It does exactly what its name promises. All you have to do is answer the following prompts:

> Standard or Daylight Saving Time

> Leap Year

> Month

> Day

> Latitude/Longitude (the default is for a location in North Carolina)

Based on the information you provide, the program calculates the time of sunrise and sunset.

That's all there is to it. Now isn't that easy?

SUNRISE-SUNSET

```
10 REM***SUNRISE-SUNSET !112
20 REM***Radio Nederland !21
2
30 REM***Wereldomroep !045
40 REM***Ref:Prof.J.Campbell
   !044
50 REM ***Ref:TI99/4A versio
n By:  !189
60 REM*** K. D. Wentzel !108
70 REM ***3454 Washburn Ave.
     !124
80 REM ***Charlotte,NC 28205
    !092
90 REM *** Rev. C,  2/21/84
 !033
100 DIM G(16),Q(16),F(3),W(3
)!230
110 P=3.141592 !117
120 T$="Y" !087
130 CALL SCREEN(15)!200
140 CALL CLEAR !209
150 PRINT "   SUNRISE - SUNS
```

Continued on page 42

SUNRISE/SUNSET

Continued from page 41

```
ET" !078
160 PRINT "    ——    —
-" !127
170 PRINT :"            PROGR
AM" !162
180 PRINT "            ——
-" !016
190 PRINT : : : : : : :!143
200 Y=81 !074
210 Z=267 !130
220 REM***G(3)=SUN Pos. on
     !147
230 REM*** the 15th of the !
081
240 REM*** Month,g(3)=Jan. !
249
250 REM*** g(4)=Feb.,etc.
     !082
260 G(3)=-21.1757 !100
270 G(4)=-12.7775 !107
280 G(5)=-2.2562 !047
290 G(6)=9.6556 !124
300 G(7)=18.792 !121
310 G(8)=23.2954 !169
320 G(9)=21.5753 !168
330 G(10)=14.1515 !203
340 G(11)=3.1496 !161
350 G(12)=-8.4003 !092
360 G(13)=-18.4051 !146
370 G(14)=-23.2513 !144
380 G(15)=G(3)!082
390 G(16)=G(4)!084
400 G(1)=G(13)!078
410 G(2)=G(14)!080
420 Q(3)=31 !166
430 Q(4)=31 !167
440 Q(5)=28 !174
450 Q(6)=31 !169
460 Q(7)=30 !169
470 Q(8)=31 !171
480 Q(9)=30 !171
490 Q(10)=31 !213
500 Q(11)=31 !214
510 Q(12)=30 !214
520 Q(13)=39 !224
530 Q(14)=30 !216
540 Q(15)=Q(3)!102
550 Q(16)=Q(4)!104
560 Q(1)=Q(13)!098
570 Q(2)=Q(14)!100
580 PRINT "Local=STANDARD or
 DAYLIGHT" !109
590 PRINT "Time?  (S or D):"
;!120
600 INPUT LT$ !086
610 IF LT$="S" THEN 630 !018
620 IF LT$<>"D" THEN 580 !14
6
630 PRINT : :!006
640 PRINT " LEAP YEAR ? (Y o
z N)";!095
650 INPUT S$ !009
660 PRINT !156
670 IF S$="N" THEN 730 !036
680 IF S$="Y" THEN 700 !017
690 GOTO 640 !209
700 Q(5)=29 !175
710 Y=Y+1 !043
720 Z=Z+1 !045
730 PRINT !156
740 PRINT "INPUT MONTH (1 to
 12)";!138
750 INPUT K !221
760 IF K<1 THEN 730 !221
770 IF K>12 THEN 730 !017
780 K=INT(K)+2 !076
790 PRINT !156
800 L=K+1 !016
810 PRINT "INPUT DAY (1 to";
Q(L);")";!147
820 INPUT I !219
830 IF I<1 THEN 810 !044
840 IF I>Q(K+1)THEN 810 !247
850 PRINT !156
860 IF I>15 THEN 900 !189
870 IF I<15 THEN 920 !208
880 D=G(K)!001
890 GOTO 1090 !149
900 L=K-2 !018
910 GOTO 930 !244
```


SUNRISE/SUNSET

```
920 L=K-3 !019
930 FOR M=1 TO 3 !062
940 H=L+M !096
950 F(M)=G(H)!186
960 W(M)=Q(H+1)!144
970 NEXT M !227
980 W(1)=-1*W(1)!185
990 IF I<15 THEN 1020 !052
1000 X=I-15 !081
1010 GOTO 1030 !089
1020 X=W(2)-(15-I)!063
1030 W(3)=W(2)!061
1040 D=(X-W(3))*F(1)/W(1)!05
9
1050 D=D-(X-W(1))*F(3)/W(3)!
067
1060 D=D*X/(W(1)-W(3))!210
1070 H=W(1)*W(3)!071
1080 D=D+F(2)*(X-W(1))*(X-W(
3))/H !211
1090 D=D*P/180 !129
1100 E=COS(D)!129
1110 IF T$="N" THEN 1400 !19
8
1120 PRINT :"Want to use you
r home" !239
1130 PRINT "Coordinants for
Lat & Long" !159
1140 INPUT "Enter (Y or N):"
:HC$ !061
1150 IF HC$="Y" THEN 2420 !0
08
1160 IF HC$<>"N" THEN 1140 !
185
1170 PRINT !156
1180 PRINT "INPUT LAT.: (DEG
S,MINS.)" !153
1190 INPUT U,X !242
1200 U=P*(U+X/60)/180 !029
1210 IF HC$="Y" THEN 2470 !0
58
1220 PRINT "NORTH or SOUTH (
N or S):";!219
1230 INPUT S$ !009
1240 IF S$="N" THEN 1280 !07
7
1250 IF S$="S" THEN 1270 !07
1
1260 GOTO 1220 !023
1270 U=-1*U !231
1280 IF HC$="Y" THEN 2500 !0
88
1290 PRINT !156
1300 PRINT "INPUT LONG: (DEG
S,MINS.)" !186
1310 INPUT S,X !240
1320 S=S+X/60 !113
1330 IF HC$="Y" THEN 2530 !1
18
1340 PRINT "EAST or WEST (E
or W):";!038
1350 INPUT S$ !009
1360 IF S$="E" THEN 1400 !18
8
1370 IF S$="W" THEN 1390 !19
6
1380 GOTO 1340 !144
1390 S=-1*S !227
1400 N=COS(U+D)!160
1410 B=2*COS(U)*E !085
1420 R=B-N !098
1430 IF R=0 THEN 1680 !156
1440 R=N/R !116
1450 IF R<0 THEN 1470 !202
1460 GOTO 1650 !199
1470 A=0 !248
1480 L=K+1 !016
1490 FOR M=4 TO L !145
1500 A=A+Q(M)!012
1510 NEXT M !227
1520 A=A-Q(L)+I !022
1530 IF A<Y THEN 1600 !156
1540 IF A>Z THEN 1600 !158
1550 IF U>0 THEN 1580 !061
1560 S$="NIGHT" !123
1570 GOTO 1620 !169
1580 S$="LIGHT" !121
1590 GOTO 1620 !169
```

Continued on page 44

SUNRISE/SUNSET

Continued from page 43

```
1600 IF U>0 THEN 1560 !041
1610 GOTO 1580 !129
1620 PRINT "24 HR.";S$;"TODA
Y" !227
1630 PRINT !156
1640 GOTO 2040 !078
1650 R=SQR(R)!162
1660 T=2*ATN(R)!091
1670 GOTO 1690 !239
1680 T=P !098
1690 H=12*T/P !094
1700 R=ABS(S)!155
1710 IF S=R THEN 1740 !050
1720 M=-1 !199
1730 GOTO 1750 !043
1740 M=1 !005
1750 IF M<0 THEN 1770 !242
1760 A=24-H !057
1770 N=INT(R/15)!142
1780 B=R-15*N !085
1790 I=0 !000
1800 IF B<7.5 THEN 1830 !144
1810 I=-1 !195
1820 N=N+1 !021
1830 B=B/15 !054
1840 L=1 !004
1850 A=H-M*(I+N+B)!225
1860 R=INT(A)!141
1870 X=(A-R)*60 !203
1880 T=INT(X)!166
1890 X=X-T !132
1900 GOTO 2290 !073
1910 PRINT :"*uct/gmt*******
************* !105
1920 PRINT "SUN RISES";R;"HR
S";T;"MINS GMT" !067
1930 R1=R !147
1940 T1=T !151
1950 H=24-H !064
1960 L=0 !003
1970 GOTO 1850 !144
1980 PRINT "SUN SETS ";R;"HR
S";T;"MINS GMT" !028
1990 PRINT "*local time*****
************ !077
2000 R2=R !148
2010 T2=T !152
2020 GOSUB 2550 !079
2030 PRINT !156
2040 PRINT "MORE? (Y or N):"
;!011
2050 INPUT S$ !009
2060 IF S$="Y" THEN 2090 !13
3
2070 IF S$="N" THEN 2280 !05
6
2080 GOTO 2040 !078
2090 PRINT !156
2100 PRINT "NEW DATE? (Y or
N):";!004
2110 INPUT S$ !009
2120 IF S$="Y" THEN 2170 !21
3
2130 IF S$="N" THEN 2150 !18
2
2140 GOTO 2100 !139
2150 IF T$="N" THEN 2170 !20
3
2160 GOTO 1110 !169
2170 PRINT !156
2180 PRINT "NEW SITE? (Y or
N):";!027
2190 INPUT T$ !010
2200 IF T$="N" THEN 2250 !02
7
2210 IF T$="Y" THEN 2230 !01
8
2220 GOTO 2170 !209
2230 IF S$="N" THEN 1110 !16
2
2240 GOTO 730 !043
2250 IF S$="Y" THEN 730 !047
2260 PRINT !156
```


SUNRISE/SUNSET

```
2270 PRINT "CRAZY LOGIC THER
E, SIR!" !132
2280 END !139
2290 IF X<0.5 THEN 2340 !159
2300 T=T+1 !033
2310 IF T<60 THEN 2340 !109
2320 T=0 !011
2330 R=R+1 !029
2340 IF R<0 THEN 2390 !102
2350 IF R<24 THEN 2370 !137
2360 R=R-24 !084
2370 IF L=0 THEN 1980 !195
2380 GOTO 1910 !204
2390 R=R+24 !083
2400 GOTO 2370 !154
2410 END !139
2420 REM***HOME LAT.&LONG. !
020
2430 REM *** LAT. !135
2440 U=39 !073
2450 X=58 !077
2460 GOTO 1200 !003
2470 S$="N" !075
2480 GOTO 1240 !043
2490 REM *** LONG. !214
2500 S=83 !070
2510 X=0 !015
2520 GOTO 1320 !124
2530 S$="W" !084
2540 GOTO 1360 !164
2550 REM*** LOCAL TIME SUB-
     !137
2560 REM*** ROUTINE... !008
2570 IF LT$="S" THEN 2670 !0
18
2580 IF LT$="D" THEN 2700 !0
33
2590 R3=R1-LOCTIME !149
2600 R4=R2-LOCTIME !151
2610 IF R3<0 THEN 2730 !238
2620 IF R4<0 THEN 2750 !003
2630 PRINT "SUN RISES";R3;"H
RS";T1;"MINS ";TYPE$ !214
2640 PRINT "SUN SETS ";R4;"H
RS";T2;"MINS ";TYPE$ !177
2650 PRINT "****************
************ !023
2660 RETURN !136
2670 LOCTIME=5 !201
2680 TYPE$="EST" !218
2690 GOTO 2590 !119
2700 TYPE$="EDT" !203
2710 LOCTIME=4 !200
2720 GOTO 2590 !119
2730 R3=R3+24 !185
2740 GOTO 2620 !149
2750 R4=R4+24 !187
2760 GOTO 2630 !159
```

NEWSBYTES

PC99 to add support for RAMdisk, AMS

PC99 users will be able to take advantage of the Myarc RAMdisk and Myarc Extended BASIC II following an agreement with Lou Phillips, former president of Myarc Inc.

According to Mike Wright of CaDD Electronics, distributor of PC99, the next release of the TI99/4A emulator will include support for the Myarc 512K RAMdisk card, and the AMS card in sizes ranging from 128K to 1024K. The Myarc RAMdisk will include ROM support for XBII.

Continued on page 46

NEWSBYTES

Continued from page 45

PC99 will detect that a Myarc RAMdisk is present and will save the contents in a DOS file, restoring the contents the next time PC99 is run.

All of the above was demonstrated at the Chicago TI Faire on Nov. 8. A shipment date will be announced when everything is ready.

Wright says that Dr. Eric Bray was instrumental in securing Phillips' approval for CaDD.

For more information, contact Wright at mjmw@xyvision.com.

Eicher heads Hoosier group

The Hoosier Users Group (HUG) is keeping its BBS on-line as a resource for the TI and Geneve community. The BBS (8N1) can be reached at 317-782-9942. Users can download files on the first call.

Dan Eicher, who was recently selected as president, also has decided to continue publishing the group's newsletter, The HUGger.

The user group also has a new mailing address: Hoosier Users Group, c/o Dan Eicher, 2720 Palo Verde Ct., Indianapolis IN 46227.

Officers are: President, Dan H. Eicher (eicher@delphi.com), president; Bryant C. Pedigo (bpedigo@midlink.com), vice president and librarian; and Greg Larson, secretary/treasurer, (uffda@indy.net).

Memberships are available for an annual rate of $22. Membership includes six issues of The HUGger newsletter, access to the Hoosier Users Group bulletin board (HUG BBS) and access to HUG disk library (no copying charge). Only charge for HUG library disks is cost of mailing for those who cannot pickup disks at monthly meetings. Meetings are held the third Sunday of the month.

Harrison honored

Valeda Lieder Inc., a music company in Staten Island, New York, recently honored Bruce Harrison with a cash award of $200 for his work in the revision of Mike Maksimik's MIDI Master 99 program.

Richard Bell of the company sent a letter congratulating Harrison for giving musicians an affordable tool to pursue their passions.

Lima users cease monthly newsletter

The Lima, Ohio, 99/4A Users Group has announced that is has published its last regular newsletter, with the September 1997 issue. Charles Good, the editor, notes that the group has lost its low-cost copying facilities and most of its local membership.

The group will continue to meet monthly at the Ohio State University at Lima campus and maintain its software library. The newsletter will continue on an irregular basis.

NEWSBYTES

Tesch takes over Myarc upgrades, repairs

Tim Tesch, of S&T Software, will manage all Myarc software and hardware upgrades, repairs, and distribution, according to an agreement between Cecure Electronics Inc. and Tesch. This became effective October 22.

According to Tesch, "I spoke at length with Don Walden about Myarc's future. After some discussion, he accepted my offer to assume responsibility for all Myarc-related aspects of Cecure Electronic Inc. Don will maintain responsibility for all remaining aspects of CEI.

"I ask that all people with outstanding Myarc requests (Geneve, HFDCC, floppy controller, software, etc.) contact me. Doing so will allow me to estimate the workload, as well as evaluate those requests which may have to be denied in the interest of the customer."

E-mail correspondence for Myarc questions should go to Tesch at: "ttesch_myarc@juno.com." This address will be the focal point for all on-line inquiries. Tesch can also be reached at 1856 Dixie Rd., Port Washington, WI 53074. Equipment should not be shipped to this address.

Bach Italian Concerto available for users

Bruce Harrison introduced a new piece of MIDI-Master music at the Chicago Faire, J.S. Bach's Italian Concerto, programmed by Dolores P Werths.

The piece is available by mail order. Users ordering should specify whether they have AMS and the make and model keyboard used for MIDI music, as four different versions are available. According to Harrison, the AMS version loads all three movements into memory so they can be played from beginning to end.

The product is available for $7 from Bruce Harrison, 5705 40th Place, Hyattsville, MD 20781.

Fest West posts forms on web site

Here is an update on plans for Fest West '98:

All the forms related to Fest West ''98—Lubbock can now be found on the official Fest West '98 web page which can be accessed at http://www.theriver.com/Public/sw99ug/. Forms include the vendor booth registration, speaker registration, official announcement pages, and the release agreement for the TI experience. The forms can be outputted by users.

According to Fest West organizer Tom Wills, of the host Southwest Ninety-Niners User Group, "Texas Instruments informed me that they did not contact Bill Cosby. Now I'm baffled as to who did, and how they ended up sending the a letter to my attention. Well, now that the ice has been broken, so to speak, I'm going to pursue this a bit further to see if we could get Bill Cosby on tape. No promises, however."

Continued on page 48

NEWSBYTES

Continued from page 47

For those who are wondering about the hotels, the Sheraton Four Points hotel has a web page that can be accessed at http://www.sheraton.com/sheraton/html/Properties/FourPoint/889.html.

SW99ers have new web address

The SouthWest Ninety Niners have a new web address. It is: http://www.theriver.com/Public/sw99ug/

It was moved into the public area of the server as the Internet service provider (ISP), River Communications, has decided the web page serves a function within the community. Now it will become available to all web users who attach to River Communication's server.

The web address for the Official Fest West '98—Lubbock will remain the same. That address is: http://personal.riverusers.com/~twills/fw98.html

Anyone going to the old SW99UG web page address will be directed to the new web page.

MICROREVIEWS

Invoice Management and Font Designer

BY CHARLES GOOD

INVOICE MANAGEMENT by Larry Tippet and Asgard Software

The other day I was talking on the telephone to a well-known TIer who told me that he owns a business that uses 99/4A computers and software for all recordkeeping and correspondence. The business involves lots of writing and this person uses the Funnelweb word processor exclusively.

"The TI gives me all the computing power I need to make my business a success," I was told.

There is a lot of good public domain and commercial software written for the TI. This month I am reviewing one such software package.

Inventory Management is an 80-column database designed for the small business or user group. Its primary function is to maintain records of delivered products. The software is written in Extended BASIC using Alexander Hulpke's X80 80-column extensions. It will run on a Geneve under GPL mode or on an 80-column 99/4A and will easily run directly off of floppy disk. You can also install the software on a SCSI or HFDC (Hard and Floppy Disk Controller) hard drive or a RAMdisk.

Inventory Management will print invoices showing your company

MICROREVIEWS

name, address, phone number, the customers' name and address and an itemized list of products delivered showing the quantity, the cost of each item, tax, and total owed. The software allows the immediate printing of an invoice, or you can save your invoices to files for a once a month print session. You can also print identical invoices showing the same list of itemized goods and services to all clients in a specified disk address file. This feature can be used by groups such as TI user groups to bill the members for annual dues, or by a company that regularly provides the same services to a group of clients.

All data is saved to disk files, which can be searched and viewed at any time. From these files you can print a detailed monthly or annual report of all invoices recorded showing the total monthly billing and sales tax. Cumulative totals are kept on disk and can be printed showing year-to-date totals of billing and tax.

Mail Room, by Larry Tippet and Asgard, is one of the first software packages I ever reviewed for MICROpendium. If you have Mail Room then you can use its name/address database directly in Invoice Management without modification. The same basic data file can be used by both software packages! When you run LOAD on the Invoice Management disk you are presented with a menu asking if you want to run Mail Room or Invoice Management. Both programs can be conveniently placed on the same floppy disk or hard drive subdirectory

Invoice Management comes set up to print invoices on a particular type ("TOPS") of fanfold, tractor-feed, multicopy blank invoices that are commonly available at good office supply stores. The software prints all the data in the correct position on these preprinted invoices.

The software package also comes with on-disk invoice templates in TI-Writer formatter and in Page Pro posters format. Print out one of these templates, take it to your friendly neighborhood copying center, and make a bunch of copies. Then insert a single sheet, blank invoice into your printer every time you want to print a new invoice. Actually, you don't need to use preprinted invoices forms at all. Just use blank fanfold tractor-feed or single sheet paper. All the data is printed in a format that most folks can still understand, even without the nice preprinted boxes that divide up the area of blank preprinted invoice forms.

When you first run Invoice Management you must configure it. This has to be done only once a year, or whenever configuration defaults change. Here is where you enter a path name for data files, the name of your account data file (could be an existing Mail Room file), the printer name, current year, starting invoice number, tax rate as a decimal, default "Shipped via" text, default "Terms" text (such as "cash"), default bottom of invoice message (such as "Thank you for your order"), and an invoice header listing your company's name and address. Now you are ready to enter invoices.

Continued on page 50

MICROREVIEWS

Continued from page 49

You see on screen a full-sized, 80-column invoice in a WYSIWYG display. As you enter data the cursor jumps from one field of this invoice display to the next. You are first asked for a client code. This code puts the name and address of the client in the "To" invoice box. If you are billing a new client, type "add" for the client code. Then, as you type the new client's code (you make one up), name, address, and telephone number in the "To" field of the on-screen invoice display, the data is also added to your on-disk account file.

Next enter the invoice date within the default year, and the invoice number. If you don't, a consecutive number is automatically generated by the software. You can then enter some "Attention" text or accept the default text which reads "Accounts Payable." This gets printed as part of the address on the invoice.

Next you enter a "job number," a date shipped or accept the default invoice date, some "shipped via" text (default is "best way" but you might substitute something like "UPS" or "US Mail"), some "terms" text (default is "cash"), and a purchase order number. Any of these fields can be left blank and most have configurable defaults you may or may not decide to accept.

Now you are ready to type in the merchandise description and cost. Here you can, from within Invoice Management, access a little pop-up calculator if you need to do some quick arithmetic. Enter quantity, merchandise description, and unit cost. The software automatically figures out "amount" by multiplying unit cost and quantity and displays this number. Each invoice can have up to 12 items.

When you have entered the data for all items you are asked if sales tax is to be added. If yes, then sales tax is automatically calculated (based on the tax rate entered when configuring Invoice Management) and displayed. You can now immediately print the invoice and save it to disk, or just save it to disk.

Throughout the data entry process there is lots of error checking, and opportunity to go back and correct mistakes. The program gives you the opportunity to mark your data file "paid in full" if you want.

At the end of each invoice's printing you are asked if you want to print an envelope. Envelope printing is optional. You do all your invoices first and then, under software control, you insert envelopes into the printer and print envelopes for all your invoices. The client's address and, optionally, your business return address is printed on each envelope.

I find Invoice Management very easy to use. The program is very professional looking on-screen and prints professional-looking invoices. If data entry errors occur there are plenty of opportunities to correct such errors. It is rarely necessary to refer to the documentation.

Any store employee should have no trouble using the software. My only complaint concerns the calculation of tax. In configuring Invoice Management, the sales tax rate can only be entered as an integer percent. The software will calculate sales tax of 5 percent, 6 percent, 7 percent, etc., but not 6.5 percent. In my state, Ohio, sales tax varies from county to county and from municipality to municipality. There is a state plus county and/or municipality tax. My local sales tax rate is 6.5 percent. Invoice Management will not automatically calculate tax at 6.5 percent, and there is no way within the program of manually entering this tax prior to printing and invoice and storing its data in a file. You have to go into the program's BASIC code to make the necessary alteration, and I haven't figured out how to do this yet.

Invoice Management is commercial. Originally distributed by Asgard, you can now purchase the software from Ramcharged Computers. I think I paid $15. It comes on a DSSD disk with excellent printed documentation.

MICROREVIEWS

FONT DESIGNER
by Bruce Harrison

If you have a 24-pin dot matrix printer or a Canon Bubble Jet printer hooked up to your 99/4A or Geneve, then you will like this.

The software lets you easily design a custom font using the entire 24x36 pixel character grid. The font can be designed letter by letter and then saved to disk and later downloaded into the printer. Any printer with Epson 24-pin emulation should work. This includes many ink jet printers produced by other manufacturers besides Canon.

You can design a partial font, maybe just a few letters or characters, and the printer's normal letter-quality font will default for all the other characters.

Here is how this is done:

Select "Dump to printer" from the software's main menu. First the software sets the printer to letter-quality mode and then moves the printer's default LQ character set to the printer's download RAM.

Next, the software sends to the printer each character you have redefined, overwriting the LQ code in RAM for only that character. Thus any characters not defined in your font will print in the printer's default LQ font.

Finally, the program sends the "select download" code and your printer is ready to print in the partially or completely redefined font.

The program is easy to use. Reference to the on-disk documentation may not be necessary. Results are really good looking with the included "bold" sample font. Font Designer is public domain and comes on a SSSD disk. Send me $1 and I will mail you the disk.

ACCESS

Ramcharged Computers (source of Invoice Management), 6467 E. Vancey Dr., Brookpark OH 44141; Phone 216-243-1244

Charles Good (source of Font Designer), P.O. Box 647, Venedocia OH 45894; Phone 419-667-3131; e-mail good.6@osu.edu

USER NOTES

Software to help holiday gift exchanges

The following is by Jacques Groslouis of New Brunswick, Canada.—Ed.

Our family, like many others, pick names to give gifts at Christmas time. The following program allows the matching of up to 26 names picked randomly without repeating or matching a name with itself. The program could also be used to match up opponents in a sporting event or other competition.

The program is based on a program named RANDOMIZER by Woody Wilson which appeared as a User Note in the December, 1995 issue of MICROpendium. The main change was to a add a routine (lines 190 and 200) to ensure that a name was not matched with the same name. Since names are discarded as they are selected, this left the possibility that the last name on the list would only be left with the same name. In this case the selections are erased and the selection starts over again. Provision has also been made to print out the results and to rerun the program.

Line 310 contains the names of my children and their spouses. Change these to suit your own use. Do not change line 330 because this line is used to count the number of data items included in line 330. Since using more than 20 items will cause some items to scroll off the top of the screen a printed output should be used in these cases.

```
100 ! SAVE DSK1.XCHANGE
110 DIM B$(26,2):: CALL CLEAR ::
 CALL BLUE :: RANDOMIZE
120 PRINT TAB(11);"XCHANGE": :;T
AB(5);"By Jacques Groslouis":"
Based on a user note in":" MICRO
pendium by W.A.Wilson": : :
130 RESTORE 310 :: FOR X=0 TO 25
 :: READ B$(X,0):: NEXT X !243
140 N=26-VAL(B$(25,0)):: PRINT
:
150 A$=SEG$("ABCDEFGHIJKLMNOPQRS
TUVWXYZ",1,N)
160 FOR J=0 TO N-1
170 Y=INT(RND*LEN(A$)+1)
180 X=ASC(SEG$(A$,Y,1))-65
190 IF J=X AND J<>(N-1)THEN 170
200 IF J=X AND J=(N-1)THEN CALL
HCHAR(24-J,1,32,32*J):: GOTO 130
210 B$(J,1)=B$(X,0)
220 PRINT B$(J,0),B$(X,0)
230 A$=SEG$(A$,1,Y-1)&SEG$(A$,Y+
1,LEN(A$))
240 NEXT J
250 PRINT :
260 INPUT "Print Out (Y/N)? ":Y$
 :: IF Y$="Y" OR Y$="y" THEN GOS
UB 290
270 INPUT "     Quit (Y/N)? ":Z$
 :: IF Z$="Y" OR Z$="y" THEN END
 ELSE 130
280 STOP
290 OPEN #1:"PIO" :: FOR X=0 TO
N-1 :: PRINT #1:TAB(15);B$(X,0);
TAB(30);B$(X,1):: NEXT X :: PRINT
#1:" " :: CLOSE #1 :: RETURN
300 ! CHANGE THE FOLLOWING
```


USER NOTES

```
 LINE TO CONTAIN YOUR        OW
N DATA. MAXIMUM 26        ITEMS.
310 DATA Robert,Helen,Peter,Kim,
Joanne,Mike,Jean Paul,Lise,Colin
320 ! DO NOT EDIT THE
 FOLLOWING LINE
330 DATA 1,2,3,4,5,6,7,8,9,10,11
,12,13,14,15,16,17,18,19,20,21,2
2,23,24,25,26
340 SUB BLUE
350 ! SWITCHES DISPLAY TO WHITE
ON BLUE; JLS 7/88
360 CALL SCREEN(5):: FOR L=0 TO
14 :: CALL COLOR(L,16,1):: NEXT
L :: SUBEND
```

Continued on page 54

USER NOTES

(Continued from page 53)

Fixing glitched XBASIC programs and DV/80 files

We're not sure who the author of this item is.—Ed.

Got an adventure graphics game on disk at a club meeting. After playing through several screens the next one to load stopped with a syntax error. Listing the program showed several lines of code to be glitched. Trying to edit out the glitched code caused the screen to change from blue to red and then lock up the computer.

Not wanting to wait a month for the next meeting to exchange the disk I decided to experiment. First, I copied the disk with Jim Schroeder's Rediskit. The program on the original disk would not even load because of a bad sector.

Next, I saved the program to disk with the command LIST "DSKx.filename". This D/V80 file must then be printed to disk with the TI-Writer formatter. It will not load into the editor after listing because the file still has the glitches in it.

Next load the formatter file into the editor and delete the glitched lines and print back to disk with the command C DSKx.filename. This removes linefeed symbols put in by the formatter. If you are lucky to have a printout of the program before it got glitched it will be easy to add the missing code and then convert it back to program format with a D/V80-to-program conversion utility. In my case the next screen to loaded after this one had identical code except for a few lines that were different, so I added the lines and thus reconstructed the glitched program.

If neither of the above options is available, you could try guessing at the missing code. Of course, if you knew of someone else who bought the same program and had a modem, he could send a replacement for the bad file to you — but that is not much of a challenge.

The above procedure will also work for glitched D/V80 message files from BBSes. This is a lot easier as most of the above steps can be eliminated. Sometimes just printing the glitched file from the formatter to printer is all that is necessary, if you do not wish to save the file for latter use.

Converting archived files with PC99

Anyone who has a reasonably fast PC can get plenty of use from PC99, the TI emulator by CaDD Electronics. I found that not long after I installed the program and learned how to configure and use it, I wanted to download programs into it.

Fortunately, there are a bunch of ways to do this. You can run a terminal program, such as Telco, and download archived and other files directly into your emulated TI. Of course, you can't do this if you have an Internet service provider who doesn't support a Telnet connection.

USER NOTES

If your ISP supports only PPP, Telco and other TI terminal programs are useless.

Of course, you can always download the TI files using a PC terminal program or even a Web browser. The trick is to get them from a PC disk or hard disk into the PC99 TI format.

PC99 comes with a plethora of utilities to move files from the TI side to the PC side, and vice versa. One of these programs is called "BIN2PGM." I use this utility to convert a TI archive downloaded using a PC into an archived file that can be used by Barry Boone's Archiver in the emulated TI.

After downloading the file, which we'll call ARCFILE, here's what you do to get it into DSK2 in PC99 (for the sake of simplicity, we'll assume you copied ARCFILE into the subdirectory where BIN2PGM resides):

```
\pc99\dskutil\bin2pgm ARC
FILE ARC.X TIARC
```

This will prepare the program for use in PC99. Now you have to move it from it's place in the dskutil\ subdirectory into one of the PC99 TI "disks." This requires a second utility called "DSKIN," which also resides in the dskutil/ subdirectory. Here's the command:

```
\pc99\dskutil\dskin  \AR
C.X \pc99\dsk\dsk2
```

Now you load PC99, load the Editor/Assembler, load Archiver, and unarc the file in DSK2, which should be called TIARC.

One caveat you should know about is that most copies of PC99 have an older version of BIN2PGM that may not support archived files. An updated version is available from Mike Wright at CaDD. Wright asks that you request this updated utility only if you actually need it. If you can wait, it will be included with the next upgrade.

Those who need his program can email Wright at mjmw@xyvision.com and request it. He'll put it on his server and provide instructions for downloading. Those without Internet access may send him a 3.5-inch, PC-formatted disk with return postage and packaging (CaDD Electronics, Attn: Mike Wright, 45 Centerville Dr., Salem, NH 03079).

DISKS, BACK ISSUES

❑ Back Issues,$3.50 each to March 1996, later $6 each. List issues on separate sheet.
No price breaks on sets of back issues. Free shipping USA. Add $1, single issues to Canada/Mexico. Other foreign shipping 75 cents single issue surface, $2.80 airmail. Write for foreign shipping on multiple copies.
OUT OF STOCK: V1#1-2; V2#1

GENEVE DISKS (SSSD unless specified)
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	$7	$5
❑ Series 2	$9	$7	$5
❑ Series 3	$9	$7	$5
❑ Series 4	$9	$7	$5
❑ Series 5	$9	$7	$5
❑ Series 6	$9	$7	$5