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; 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

$_____

No sales tax on magazine subscriptions. Texas residents add 8.25% sales tax on other items, including back issues and disk subscriptions.

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 15 Number 5 September/October 1998 $6

Projects

Adding a LED

Protecting modems from surge damage

MIDI-Master

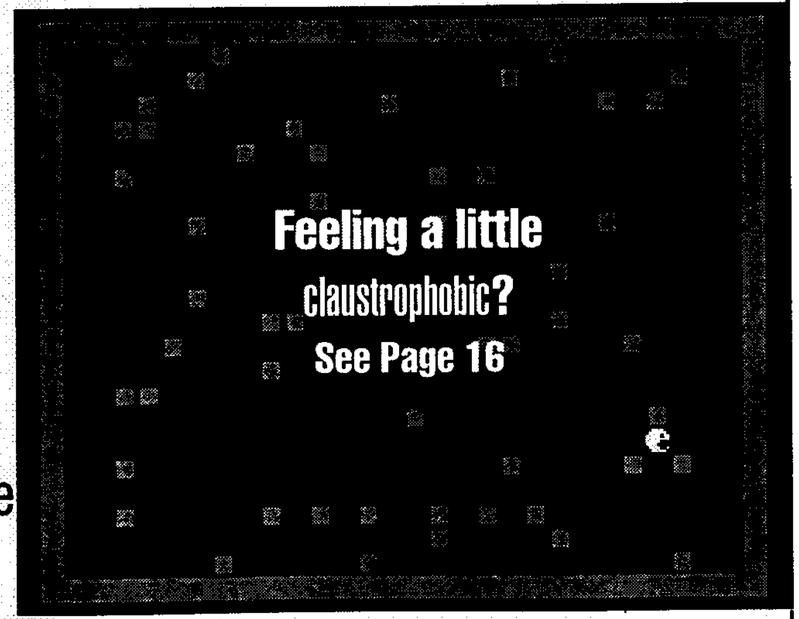
A musical adventure

Micro-reviews

Notepad80

Download File Converter

Visitors at Chicago Faire to receive free software from Europe



Extended BASIC

AMSLERTEST
Claustrophobia
FINDPRCNT
UNMERGE

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

!!CHANGE NOTICE!!

THE LOCATION HAS BEEN CHANGED

of the

16th ANNUAL CHICAGO TI INTERNATIONAL WORLD FAIRE

14 NOVEMBER 1998

The **NEW LOCATION** is

Northminster Presbyterian Church
2515 Central Park Ave.
Evanston IL 60201
Room 2

9:30 a.m. - 4:00 p.m.

Demonstrations-Seminars-User Groups
Door Prize Drawings, Free Parking,

Bus and Rapid Transportation to Door

for further information contact: **Hal Shanafield** (847)864-8644
Hotel Information Available

All attendees to the Faire will receive a disk of brand new software. This disk will be available only to Faire attendees and is not copyable.

COMMENTS

Lubbock loses TI plant

Tom Wills of the SouthWest Ninety-Niners is feeling that the group did the right thing by holding Fest West '98 at Texas Instruments' Lubbock facility this year. This site, for many years the heart of TI's consumer electronic division and the birthplace of the TI99/4A, is phasing out. TI expects to eliminate 680 jobs in Lubbock, although one-third of the employees will be offered other positions at plants in Dallas and Houston. TI is also closing the Midland, Texas, plant, whose employees will be offered jobs in Sherman, Texas. Lubbock political and civic leaders were devastated by the news. For further information, log on to http://www.lubbockonline.com.

Chicago group overcomes obstacle

It's good to see that Hal Shanafield and the Chicago user group have a Plan B in place for this year's Chicago TI Faire. The fair was originally planned for an American Legion hall. The Chicago group even advertised it in the July/August MICROpendium. But a day or two after the issue came out, it was learned that the Legion hall had already been booked, sending the user group scrambling. Hal has since found a nearby Presbyterian church with available space. And that's where the fair will be held. You can look at the article on page 5 for additional details, and mark your calendar for Nov. 14.

Those who attend this year's faire will receive a free copy of a new disk manager that operates with SCSI and other drives. Berry Harmsen will bring it over from Holland. We'll have more about it in the November/December issue. Those who miss the faire will be able to purchase the program from Berry.

Tesch finishes Geneve repairs and upgrades

Tim Tesch of S&T Software is trying to put Geneve repairs and upgrades in order and asks customers who have not received promised items to contact him.

If you are waiting for software or documentation, he says, "Let me know what you are waiting for and whether you paid any fairware fee for it. People I spoke with at Lima should contact me as well, whether they contacted me recently or not."

If you are waiting for Myarc hardware, "I have completed all repairs. If anyone else has hardware that was sent to Cecure Electronics and have yet to get it back, let me know ASAP, as I don't know how long I will be able to continue working on fixing/upgrading hardware."

If you plan to upgrade or repair equipment "requests for updates or repairs must be made via e-mail or the postal service. I will no longer accept verbal requests nor will I conduct business over the phone." This is due to the fact that he travels often and doesn't want his household "bombarded by telephone calls."

For more information, contact Tesch at ttesch@juno.com or at Tim Tesch; 1856 Dixie Road; Port Washington, WI 53074. Include your U.S. postal shipping address with all correspondence.

NEWSBYTES

Chicago Faire has new location

A new site has been named for the 16th Annual Chicago International World TI Faire Sept. 14. The new site is Room 2 at the Northminster Presbyterian Church, N2515 Central Park Ave., Evanston, Illinois, according to Hal Shanafield of the Chicago TI Users Group, the hosts for the event. A conflict prevented the Faire from being held at the American Legion Post in Evanston as previously scheduled.

Seminars will include those by Bruce Harrison, who will also introduce some new products, and Lew King, who will discuss Term 80 on the Internet. In addition, attendees will receive a free disk being brought to the Faire by Berry Harmsen of the Dutch TI users group, an "unhackable" disk with new European software. Shanafield notes that this will be available only at the event as it cannot be copied.

Following is information on hotel accommodations for TI Faire participants:

Evanston: Omni Orrington Hotel, 1710 Orrington Ave., (847) 866-8700; rates $155 single/$165 double; American Automobile Association/ American Association of Retired Persons discounts available.

Holiday Inn, 1501 Sherman Ave., (847) 965-6400; rates $134 single/ double, AAA/AARP discounts.

Morton Grove: Best Western, Morton Grove Inn , 9424 Waukegan Rd., (847) O965-6400; rates, 1 queen $49 plus tax, 1 king $55 plus tax, 1 queen double with sofa sleeper $55 plus tax, 2 double $58 plus tax; 10 percent discount for AARP, GEntertainment Book Coupon and AAA; includes continental breakfast, HBO/ cable/in-house movies; microwave/ refrigerator available upon request.

Grove Motel, 9110 Waukegan, (847) 966-0960; rates $45 single, $50 double, including tax and discount; free coffee in lobby; cable/HBO and Cinemax.

Suburban Motel, 9115 Waukegan Rd., (847) 470-0300; rates $40 single, $46 double, $46 King.

Shanafield adds that attendees can call him for information about the "ever-popular Admiral Oasis Motel, which, if not yet torn down, has colorful rooms with strategically placed mirrors available for around $30."

For further information, contact the Chicago Users Group, P.O. Box 7009, Evanston, IL 60204-7009, or call Shanafield at (847) 864-8644.

Zlotorzynski heads Will County group

Tony Zlotorzynski is the new president of the TI Users Group of Will County (Illinois). Tony Z. took office September 1, succeeding Bob Petter. Petter served as president for two years.

AMSLERTEST

Extended BASIC program tests vision

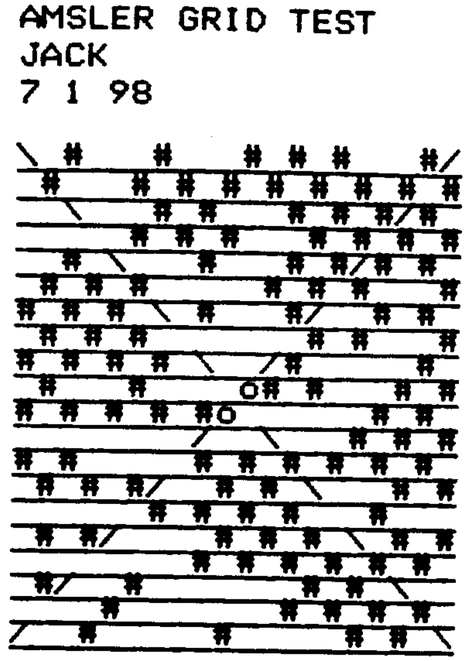
The following program, AMSLERTEST by Jack B. Cunningham, is designed as a computer-based version of a vision test called the Amsler Grid. This program, written in Extended BASIC, reproduces an Amsler Grid recording chart and outputs results to a printer. This and other tests are used by optometrists and ophthalmologists to test patients for such disorders as macular degeneration.

We caution that this is not to be used as a substitute for advice from a physician.

The program prints the grid with "lights," or white "X's", seen and those missed. The short test uses 200 lights and takes about six minutes. The long test uses 400 lights and takes about 12 minutes, with a rest period after the first 200 lights.

To use the program, sit directly in front of the center marker as it appears on the monitor (marked onscreen by two orange squares) at a distance four times the width of the screen display. Cover one eye before starting the test.

At the sound of the tone press a key every time you see a light, keeping your eye focused on the squares at all times. It is difficult to keep your eye squares and make take some time to get used to it.



AMSLERTEST

```
100 CALL CLEAR !209
110 PRINT "    AMSLER GRID T
EST 1995":"":"              B
Y":"":"       JACK CUNNINGHA
M":"":"" !167
120 PRINT "PRINTS GRID WITH
LIGHTS SEENAND THOSE MISSED.
 SHORT TEST(200 LIGHTS) 1S 6
 MINUTES, &" !135
130 PRINT "LONG TEST (400 LI
GHTS) IS 12MINUTES WITH REST
 AT 200.":"":"PRESS ANY KEY
TO CONTINUE" !194
140 CALL KEY(0,K,S)!187
150 IF S=0 THEN 140 !147
160 CALL CLEAR !209
```


AMSLERTEST

```
170 PRINT "VIEWING POSITION-
DIRECTLY INFRONT OF CENTER
MARKER AT ADISTANCE 4 TIMES
THE WIDTH ":!150
180 PRINT "OF THE SCREEN DIS
PLAY. COVERONE EYE.":":":"
AT THE SOUND OF THE TONE,WHE
N YOU ARE READY TO START" !1
02
190 PRINT "PRESS ANY KEY AND
 THEN PRESSA KEY EVERY TIME
YOU SEE A  LIGHT. KEEP YOUR
EYE ON THE CENTER!":"" !227
200 PRINT "REMEMBER—CALL
YOUR DOCTORANY TIME YOU SEE
A CHANGE!":"":"":"READY???
PRESS ANY KEY" !070
210 CALL KEY(0,K,S)!187
220 IF S=0 THEN 210 !217
230 CALL CLEAR !209
240 INPUT "NAME-":NAME$ !166
250 CALL CLEAR !209
260 INPUT "DATE MM DD YR-":D
ATE$ !214
270 CALL CLEAR !209
280 INPUT "PRINT TEST RESULT
S? - ENTER Y OR N (CAPS ONLY
)-":Q$ !234290 IF
Q$="Y")+(Q$="N")THEN 300 EL
E 280 !197300 CALL CLEAR !
9310  INPUT "E
TER 2 FOR SHORT
 TEST              4 FOR LONG
TEST-":TL !095
320 IF (TL=2)+(TL=4)THEN 330
 ELSE 310 !024
330 OCC=0 !140
340 F=0 !253
350 CALL CLEAR !209
355 CALL COLOR(2,1,1)!171
360 CALL SCREEN(5)!150
362 FOR R=5 TO 24 !123
364 CALL HCHAR(R,7,45,20)!00
7
366 NEXT R !232
368 CALL CHAR(45,"8080808080
8080FF")!021
369 CALL CHAR(46,"8080808880
8080FF")!030
370 CALL CHAR(111,"007E7E7E7
E7E7E00")!132
375 CALL CHAR(47,"8182848890
A0C0FF")!059
377 CALL CHAR(92,"80C0A09088
8482FF")!058
380 CALL COLOR(10,9,1)!227
390 CALL COLOR(12,16,1)!020
400 CALL CHAR(35,"80A8A8FCA8
FCA8FF")!154
410 CALL COLOR(1,1,1)!170
420 RANDOMIZE !149
430 DS=INT(RND*(2-1+1))+1 !1
97
440 GOSUB 1020 !079
450 CALL SOUND(100,380,15)!1
84
460 CALL SOUND(100,320,15)!1
78
470 FOR D=1 TO 10 !100
480 CALL KEY(0,K,S)!187
490 IF S<>0 THEN 520 !210
500 NEXT D !218
510 GOTO 450 !018
520 FOR W=1 TO 200 !169
530 NEXT W !237
```

Continued on page 8

AMSLERTEST

Continued from page 7

```
540 IF DS>1 THEN 1520 ELSE 1
540 !151
550 READ R C
560 IF R=1 THEN 1560 !037
570 IF R=0 THEN 1380 !111
580 IF F=0 THEN 630 !114
590 IF C/2=INT(C/2)THEN 620
!105
600 C=C+1 !255
610 GOTO 630 !199
620 C=C-1 !000
630 CALL HCHAR(R+2,C,120)!08
2
640 FOR D=1 TO 50 !104
650 CALL KEY(0,K,S)!187
660 IF S<>0 THEN 710 !144
670 NEXT D !218
680 CALL HCHAR(R+2,C,35)!038
690 OCC=OCC+1 !035
700 GOTO 550 !119
710 CALL HCHAR(R+2,C,46)!040
720 FOR W=1 TO 200 !169
730 NEXT W !237740 GOTO
50 !119750 CALL COLOR
2,2,1)!1727
2 CALL COLOR
1,2,1)!171755 CALL SCREEN
6)!151758 R=5
!014760 C=7 !
01770 CALL HCHAR(R,C,92)!
09780 R=R+2 !
30790 C=C+2 !
00800 IF C<>17 THEN 830
050810 R=R+
 !029820 C=C
1 !255830 IF R<25 THEN 77
 !068
840 R=5 !014
850 C=26 !051
860 CALL HCHAR(R,C,47)!109
870 R=R+2 !030
880 C=C-2 !001
890 IF C<>16 THEN 920 !139
900 R=R+1 !029
910 C=C-1 !000
920 IF R<25 THEN 860 !158
930 A$="OCCL" !015
940 P=14 !061
950 Y=3 !019
960 GOSUB 1580 !130
970 A$=STR$(OCC)!064
980 P=15 !062
990 Y=3 !019
1000 GOSUB 1580 !130
1010 RETURN !136
1020 CALL HCHAR(14,17,111)!0
98
1030 CALL HCHAR(15,16,111)!0
98
1040 RETURN !136
1050 GOSUB 750 !064
1060 IF (Q$="N")THEN 1210 !1
13
1070 OPEN #1:"PIO" !253
1075 PRINT #1:CHR$(27);CHR$(
33)!229
1080 PRINT #1:"AMSLER GRID T
EST":NAME$:DATE$ !122
1090 PRINT #1:CHR$(27);CHR$(
84);"16" !207
1096 PRINT #1:CHR$(27);CHR$(
88);!163
1100 FOR R=5 TO 24 !123
1110 P$="" !249
1120 FOR C=7 TO 26 !112
```


AMSLERTEST

```
1130 CALL GCHAR(R,C,X)!143
1132 IF X=45 THEN 1138 !188
1134 IF X=46 THEN 1138 !189
1136 GOTO 1140 !199
1138 X=32 !069
1140 P$=P$&CHR$(X)!249
1150 NEXT C !217
1160 PRINT #1:P$ !188
1170 NEXT R !232
1180 PRINT #1:CHR$(27);CHR$(
65)!234
1182 PRINT #1:CHR$(27);CHR$(
89);!164
1184 VS=TL*100-OCC !189
1186 VP=VS/TL !113
1190 PRINT #1:"#-OCCLUDED-";
OCC !099
1192 PRINT #1:"VISIBLE";VS;V
P;"%" !124
1195 PRINT #1:CHR$(27);CHR$(
34)!230
1200 CLOSE #1 !151
1210 GOTO 1210 !013
1220 DATA 19,13,22,20,20,14,
16,14,14,26,15,21,14,14,17,9
,12,22,19,17,16,22,21,9,20,1
8,10,24,6,10 !212
1230 DATA 18,12,10,18,16,12,
8,8,18,14,20,8,6,24,13,17,16
,20,14,20,13,15,12,12,9,25,2
2,10,15,13,1,1 !202
1240 DATA 18,12,10,18,16,12,
8,8,18,14,20,8,6,24,13,17,16
,20,14,20,13,15,12,12,9,25,2
2,10,15,13 !112
1250 DATA 19,13,22,20,20,14,
16,14,14,26,15,21,14,14,17,9
,12,22,19,17,16,22,21,9,20,1
8,10,24,6,10 !212
1260 DATA 15,17,4,16,10,14,8
,14,5,21,13,21,14,12,12,16,2
1,25,15,11,6,22,12,18,19,21,
18,26,5,25 !112
1270 DATA 11,21,17,13,4,18,2
2,14,18,16,17,23,10,12,17,11
,4,8,21,17,5,19,15,15,3,15,1
6,18,6,8 !026
1280 DATA 19,7,14,18,20,24,5
,9,8,26,11,15,14,8,8,16,11,7
,22,12,4,12,18,10,9,21,15,9,
7,19 !097
1290 DATA 16,16,7,13,17,15,5
,7,20,12,9,11,21,23,13,25,11
,19,6,26,22,16,5,23,3,21,22,
22,5,17 !222
1300 DATA 11,25,7,15,21,11,1
6,26,4,10,9,9,22,24,4,14,3,1
7,19,23,21,15,6,18,19,9,14,1
6,3,25 !188
1310 DATA 9,17,17,19,10,20,2
1,7,14,24,12,10,21,13,21,21,
4,24,8,22,13,23,8,12,17,7,15
,23,5,11 !006
1320 DATA 9,15,17,21,5,13,9,
13,14,22,6,20,15,7,7,17,12,2
0,16,8,13,19,18,20,9,7,19,11
,11,11 !183
1330 DATA 22,8,15,19,22,18,1
7,25,18,22,16,10,15,25,12,14
,10,8,10,16,18,18,5,15,11,9,
21,19,12,8 !132
1340 DATA 19,15,11,23,20,26,
3,19,20,20,13,11,8,18,3,23,2
0,22,7,23,18,8,10,22,4,20,10
,10,6,16 !253
```

Continued on page 10

AMSLERTEST

Continued from page 9

```
1350 DATA 8,20,3,13,6,12,18,
24,8,10,4,22,13,7,6,14,13,13
,7,7,10,26,3,7,7,9,9,19,8,24
 !145
1360 DATA 14,10,4,26,22,26,3
,11,12,26,17,17,11,17,19,19,
3,9,16,24,7,21,11,13,19,25,7
,25,9,23 !033
1370 DATA 13,9,7,11,20,16,12
,24,20,10,0,0 !163
1380 IF TL=2 THEN 1050 !116
1390 IF F=1 THEN 1490 !210
1400 CALL GCHAR(14,16,UC)!15
4
1410 CALL GCHAR(15,17,VC)!15
7
1420 CALL HCHAR(14,16,111)!0
97
1430 CALL HCHAR(15,17,111)!0
99
1440 CALL HCHAR(14,17,45)!05
5
1450 CALL HCHAR(15,16,45)!05
5
1460 F=1 !254
1470 RESTORE !148
1480 GOTO 450 !018
1490 CALL HCHAR(14,16,UC)!15
5
1500 CALL HCHAR(15,17,VC)!15
8
1510 GOTO 1050 !109
1520 RESTORE 1220 !037
1530 GOTO 550 !119
1540 RESTORE 1240 !057
1550 GOTO 550 !119
1560 RESTORE 1260 !077
1570 GOTO 550 !119
1580 FOR Z=1 TO LEN(A$)!246
1590 CALL HCHAR(P,Y,ASC(SEG$
(A$,Z,1)))!249
1600 Y=Y+1 !043
1610 NEXT Z !240
1620 RETURN !136
```

THE ART OF ASSEMBLY PART 72

Still Afloat

BY BRUCE HARRISON

This month's column is a continuation of last month's. You'll need to get the sidebar from last month in front of you to understand what we're writing about. Got it? Okay, ready or not, here we go.

Sidebar 71 is a complete program to demo some typical floating point math operations. You'll notice that the source starts off with more than the usual EQUates. These are of course not absolutely necessary, but are used so that there will be mnemonics available to make the bulk of the source code a bit easier to grasp.

FAC stands for the *Floating point ACcumulator*. This is a very important memory location for any of the floating point operations. It refers to the eight

THE ART OF ASSEMBLY PART 72

bytes starting at address >834A in the RAM Pad. That's where floating point numbers get placed by the ROM and GROM floating point math routines that we'll use in this program. ARG, which stands for *floating point ARGument*, is another eight-byte portion of RAM Pad, starting at >835C. This is used for a second floating point number. For example, the addition routine adds the number in FAC to the one in ARG, and puts the result at FAC. The subtraction routine subtracts the number at FAC from the one at ARG, and puts the difference at FAC. Some operations, such as the computation of the sine, use only the number at FAC, but also use a stack area in VDP RAM to store intermediate results. To accommodate those operations, we've set up a stack address (VSTACK) in VDP RAM using >1000 as the start of the stack.

Starting up

The code section, which begins at label START, first sets the workspace pointer to our own workspace at label WS. Next it clears the word at >8374 (KEYADR) to insure that we're using key-unit zero. Next we set R0 to the value of our VDP Stack (>1000), then stash that number in RAM Pad at >836E, which serves as the pointer for stack use by some routines. Thus those routines will put stuff into VDP RAM at a place which won't interfere with our screen displays or character definitions.

Getting the Numbers

At label RESTR, set up for getting our first number input from the user. First we "point" R0 at row 1, column 4, then put a prompt on the screen. Now we BL @ACCEPT to allow user input.

ACCEPT uses four data lines following the BL to determine its parameters. The first word after the BL determines the screen position for accepting keyboard input. In this case, that's row 2, column 3. The next data word is the field length, in this case 28 characters, which is actually more than enough for numeric inputs. The third data word is a signal to the routine that determines whether or not to clear out the input field before accepting input. In this case, it's 1, so the field will be cleared by the ACCEPT routine. Any number here other than zero will make the field clear. Zero will allow any previous content to remain in the field. The fourth parameter is of no importance in this case, but is the address of a block of memory to store the user's input as a string. Here, we've set that to the address of TEMSTR, a block of 30 bytes in our data section. The string placed there won't actually be used in this program, but there has to be a block of at least one byte more than the field length. This way, we can use the same ACCEPT routine for either strings or numbers.

After the four data words, we take the content of R0, and place it at location FAC12 (>8356). This happens just after we've exited from the ACCEPT routine. R0 at this point contains the VDP address of the first byte of the input field.

Continued on page 12

THE ART OF ASSEMBLY PART 72

Continued from page 11

Placing that address at >8356 is necessary to allow us to use the Convert String to Number (CSN) routine via XMLLNK. That routine examines the contents of VDP RAM starting at the address that we've put in FAC12 (>8356), and converts what's there into a floating point number at FAC. If what's there does not represent a number, FAC will contain zero in its first two bytes, meaning the input is regarded as zero.

The routine CSN keeps reading the string until either it runs out of digits or it finds a character that's not part of the numeric set. For this routine, the numeric set consists of the numbers 0-9, the plus or minus sign, and the capital E (for exponent). Any other character found in the input string will be regarded as non-numeric, and will terminate the conversion routine. Thus if we put a number, like 2.135, into the 28 byte input field, the conversion routine will stop when it finds the space just after the 5. It will correctly report the number 2.135 into FAC in floating point format. (We explained that format last month.) The way we've written this program, the number input must be left-justified in the input field. Any leading spaces before the number starts will cause the conversion to yield zero.

Once this first number has been accepted, we want to save it to our own data memory, so that accepting another number can be done without losing this one. For that, we use a special little subroutine called MOVNUM. To use that, we load R9 with the address FAC, and R10 with the address NUM1, which is a block of eight bytes set aside in our Data Section. MOVNUM then copies the eight bytes from FAC to the block at NUM1.

All of this now repeats for the second number input, except that number gets copied into the block of bytes at label NUM2.

Now the first math operation we want to perform is just to add these two floating point numbers. First we put a string on the screen to label this as the sum of the numbers, then we use MOVNUM to take the number we placed at NUM1 into the ARG block. The number at NUM2 is still present in FAC, so we don't need to put it there for the addition. Thus we have two floating point numbers at ARG and FAC, these being NUM1 and NUM2, respectively.

We now use the ROM routine FADD through XMLLNK to add these two numbers. XMLLNK finds and executes the FADD routine, which places the result at location FAC as eight bytes. Now we clear an 18-character area just after the label on the screen, and then use our subroutine DISNUM to display the number taken from FAC. The subroutine DISNUM uses a GPL routine called Convert Number to String (CNS) through GPLLNK. That routine takes the floating point number at FAC and converts it to a string located in RAM Pad. When that routine exits, the byte at FAC12 (>8356) contains the length of the string, and the byte at FAC11 (>8355) contains the low order part of the

THE ART OF ASSEMBLY PART 72

address at which the string is to be found.

To get this string on the screen, we take the byte at FAC12 into R2, right justify it, and then take the byte from FAC11 into R1, right justify that, then add >8300 to R1 so it points at the string's location in RAM Pad. The desired address on the screen is already in R0, so a simple VMBW operation puts the number string on the screen at the correct location. The string will always have a length of at least one, so we needn't check for a zero length string. If the number was zero or positive, there will be a space in the first character of this string. If the number is negative, the first character will be a minus sign.

Other Math Operations

We proceed now to perform other math operations on the two numbers we accepted. They have been left unsullied at the locations NUM1 and NUM2 in memory, so we can reuse them at will. In all cases from here on, we have to assume that whatever was left at ARG and FAC have been corrupted, so our first order of business before any more math operations is to use MOVNUM to place NUM1 at ARG and NUM2 at FAC. Remember that for subtraction, the number at FAC gets subtracted from the number at ARG. Thus, when our FSUB finishes (via XMLLNK), FAC will contain ARG-FAC. We go through this process a couple more times, putting the product of NUM1 * NUM2 on the screen, and the result of NUM1 / NUM2 on the screen.

The next to last operation we perform is to take the sine of the number at NUM1. This computation uses a routine in GROM, so we have to use GPLLNK instead of XMLLNK. In our source, we've included the Warren/Miller GPLLNK, mainly to avoid the well known problems that TI's GPLLNK presents. We don't have to put anything into ARG in this case, but just put the number from NUM1 into FAC. This routine uses that VDP stack we mentioned at the beginning, putting "God-knows-what" into the stack as intermediate results. When it's finished, there's a floating point number at FAC that equals the Sine of whatever number (in radians) was at FAC when we called the GPLLNK routine. This result is always a number between -1 and 1, inclusive.

The final operation is a comparison of the two numbers. For this, we place NUM1 at ARG and NUM2 at FAC as usual. After the XMLLNK performs the FCOM routine, we have to examine the GPL status byte (>837C) to see the result.

We check first to see if the numbers are equal, since that's the easiest test. We just do CB @STATUS,@ANYKEY, and if those two bytes are equal, so are the numbers at FAC and ARG. If they're not equal, we put the STATUS byte in a register (e.g. R3), then strip off all but the >4000 bit. If that result is not zero, then the number at ARG was greater than the one at FAC. If the result is zero, given that the numbers are not equal, then the one at ARG must be less than the

Continued on page 14

THE ART OF ASSEMBLY PART 72

Continued from page 13

one at FAC. The code in last month's sidebar performs just this way, and puts one of three messages on the screen to indicate the relationship between NUM1 and NUM2. One can also test for a "logical high" relationship using the >8000 bit, but we can't see any sense in doing that for floating point numbers.

Variations You Can Try

In your own work, you might want to try out the idea of allowing leading spaces to be present in the input field. This might come in handy if, for example, a default positive number string were in the input field to start with. Such a string starts with a space, which must then be skipped over after an ACCEPT operation.

The following discussion assumes that our own version of ACCEPT, as included in last month's sidebar, is being used. Among other things, that means that the length of the string typed in the field, excluding trailing spaces, is in R2 upon return from the routine. We'll show here the code that would allow your routine to skip over any leading spaces.

```
        BL    @ACCEPT    Use Accept subroutine
        DATA  32*5+2     Row 6, Col 3
        DATA  28         Field Length 28
        DATA  0          Don't Clear field
        DATA  TEMSTR     String Buffer
READ1   BLWP  @VSBR      Read byte from field
        CB    R1,@ANYKEY Is that a space?
        JNE   GNUM1      If Not, jump
        INC   R0         Next spot on screen
        DEC   R2         Dec string length count
        JGT   READ1      If positive, repeat read
GNUM1   MOV   R0,@FAC12  Put R0 at >8356
        BLWP  @XMLLNK    Use XML linkage vector
        DATA  CSN        Convert string to number
```

This method will work even if the field was left blank. In such a case, after the DEC R2, R2 will become a negative number, and the JGT test will fail, so the CSN routine will be used right away, and will report zero at FAC. No doubt some of our readers will find a more efficient way to do this, but the way we've shown is certain to work. Each time a leading space is found in a non-null entry, the pointer in R0 advances one spot and another read is done until a non-space character is found.

The "Caret" Case

We've fooled around with the other operations allowed through GPLLNK,

THE ART OF ASSEMBLY PART 72

and found that other functions work as given in the Editor/Assembler manual, with one notable exception. That exception is the "Raise number to power" routine (in BASIC or Extended BASIC). The E/A manual says that you can use this routine, which we call the "caret" routine, by placing the first number at ARG, the power to which it's to be raised at FAC, then using GPLLNK. This doesn't seem to work! The routine seems to lose its way somewhere along the line, returning to our code with a meaningless number placed in FAC. Here then is another plea to our readers. If any of you has discovered some trick to make the "caret" routine work from Assembly code, please let us know, and we'll pass that on to others.

We hope these two articles will satisfy your hunger on the subject of floating point numbers for a while. Next month's topic is undecided at present, but since we're now writing more than a year ahead of publication, we've got plenty of time to decide on that topic.

READER-TO-READER

Rich Gilbertson, 1901 H St., Vancouver, WA 98663-3352, writes:

Recently I purchased an Iomega Zip drive from a PC Mall. My first problem was the connectors for the SCSI card to the 25 pin on the Zip the internal 50-pin cable on the SCSI card from Western Horizon needs to be run through two conversion to be useful. I purchased part MCS-FM506 that is a SCSI Centronics 50/F w/bracket to internal IDC 50/M w/4-inch ribbon cable, and MCS-MM2556 SCSI 6-feet cable DB25M to Centronics 50/M. This combo can be purchased from almost any local parts supply house for about $39 compared to $64 for SCSI2 connectors.

The Zip drive works very well with the TI and is faster than the two 42-Meg. Teac SCSI2 drives in my system. My 200-Meg. Rodine is SCSI1 and is a little slower than the Teac's. I had to set the Zip to drive 7 as my Rodine responds to 1 and 6. For $100 the Zip is a good buy.

I suggest that if anyone wants a copy of the 42 Meg of my entire library, send me a Zip disk and a self-addressed stamped envelope. The copy will only work with a Western Horizon SCSI card, but will be everything I have. Most of my library is source of GPL and assembly.

Bruce Harrison has new e-mail address

Bruce Harrison, MICROpendium columnist and assembly language expert, has a new e-mail address. You can reach him at rottencat1@aol.com.

HARDWARE PROJECT

Have a card in need of an LED?

Here's one method of adding one

BY RALPH GOODWIN

The following article originally appeared in the newsletter of the 9T9 Users Group. Readers who undertake this project do so at their own risk.—Ed.

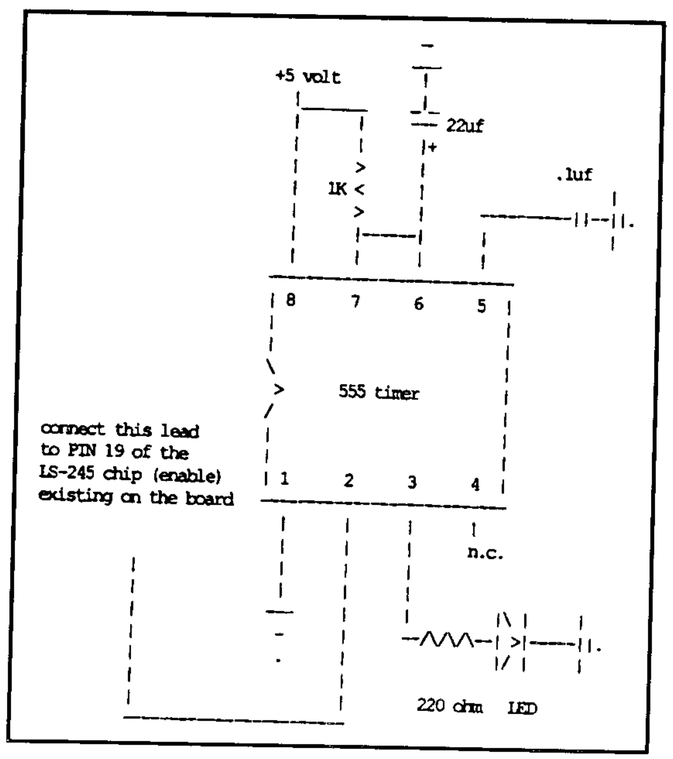
Do you have a PEB card that doesn't include an LED? Want to add one? Here's what you what need to do.

Parts placement on the card are not too critical but everything should be kept as close as possible. The 555 can be piggy-backed on top of another chip but take notice that the 555 does not have power pins assigned the same as TTL chips. The other way involves a little epoxy, cement the chip down and solder pin 1 directly to the ground plane. Placing the LED on the board about 1-inch up from the bottom, and 1/2-inch in from the front edge should place the LED in position to be in line with the window. Solder the cathode to the ground plane and use a dab of epoxy to hold it in place. The cathode is the lead that is usually identified by a flat or notch in the case and/or the shorter lead.

There are no doubt other case designs but I think these are the most common. The rest of the parts are easy enough, just follow the diagram and light up another window in your PEB (*See diagram).*

Required Parts

1 555 timer
1 220 ohm resistor (any wattage)
I 1000 ohm resistor (any wattage)
I 0.1 uf capacitor (ceramic disc or similar) any voltage
1 22 uf capacitor (electrolytic or tantilum) 8 volt or higher
1 LED any color (TI specs are amber or yellow) some hook-up wire



MODEMS

Protecting computer modems from surge damage

BY ROSS MUDIE

Computers with modems, just like fax machines, are prone to damage due to surges from lightning or high voltage electrical faults. When a high voltage power line develops an earth fault, or lightning strikes an earthed object near the customer's premises or the telephone exchange, the momentary high current into the earth causes a potential gradient across the earth's crust which can result in a large earth potential difference between the earth at the telephone exchange and the earth at the customer's premises (earth potential rise).

The telephone line from the exchange serves to deliver the telephone exchange earth potential to the line side of the computer modem or fax machine. If the momentary difference in earth potential between the line side of the modem and the mains earth of the modem sufficiently exceeds the break down voltage of the line isolation gap in the modem, it is possible for a "flash over" to occur in the modem and for very large currents to flow. This usually has the effect of causing serious damage in the modem and sometimes the attached computer.

There are a number of products in the marketplace which can reduce the incidence of such damage. These devices must be connected in the telephone line and the power to the modem/computer or fax machine to be effective. Just protecting the power or the phone line independently will not be effective.

Modem protection devices operate by limiting the amount of voltage difference between the telephone line and the mains power earth of the modem in addition to the voltage across the line and on the mains connection, by using a combination of Gas Arresters and MOV type surge suppressors.

The simpler (and usually cheaper) surge suppression products just provide a "shunt" type of over-voltage protection (gas arrester). This clamps the phone line relative to the mains earth. The more complex suppression products usually include phone line to mains earth clamping, in addition to series and shunt-surge suppression in both the phone line connection and the active-neutral, active-earth, and neutral-earth of the power.

The secret of successful protection of the modem or fax machine is to prevent the voltage difference between any two parts of the device being protected from exceeding the flash-over ratings.

Aspects to avoid are:

Continued on page 18

MODEMS

Continued from page 17

- Wiring away from the fax or modem after the protection which can provide a "back door" for a surge to enter the device.
- Separated protection in the power and phone line where a large surge pulse potential could instantaneously occur between the earths.

There is nothing that can protect from a direct or very close lightning strike. The best policy is to unplug from both the phone line and the power in periods of thunderstorm activity or when the modem is not being used.

The best surge protection known for modems and faxes is a FaxGuard manufactured by Critec.

It is in your own interest to provide surge protection for your modem and computer since telephone companies do not accept liability for lightning damage to customer-owned or rented equipment associated with the phone service.

MIDI-MASTER

The New MIDI-Master

A Musical Adventure

BY BRUCE HARRISON

We start with a sincere thank you to Mike Maksimik, who graciously permitted use of his original source code for the creation of a new generation of MIDI-Master. Our original exposure to this fine program was a revelation. Being able to control all the power of a modern electronic instrument from our faithful TI was a terrific experience. While we're about making thank yous, here's one for Jim Krych, who encouraged and supported our efforts on MIDI-Master.

The Ultimate Goal

From the outset of our "messing around" with MIDI-Master, the goal was to create a version for TI owners who have both the original program (Version 2.3) and the new Super AMS Card. Along the way, however, we decided that certain improved performance features could be added that would enhance the capability of the program even for those without the AMS Card. Thus was born MIDI-Master Version 2.5Z, with numerous added and improved features. There are two sub-versions of 2.5, with the one called 2.5Z being for those without AMS, and 2.5B for those who have AMS. Both have all of the improved performance features, but the AMS version has the added one of being able to handle much larger pieces of music. The AMS version has now (June 1998) been updated so that it will work with BOTH the SW99ers SAMS card and the SGCPU card's AMS emulation. That's why what was 2.5A is now 2.5B.

MIDI-MASTER

The "Biggie"

Dolores P. Werths, our resident musician, found one thing sorely needed in Version 2.3. That was the ability to control the volume of individual voices. That capability, which she has in Cakewalk on our PC, was just not possible in MIDI-Master, because there was only one volume control, and that applied to all voices simultaneously. In real music, one sometimes wants one voice or another (melody, chords, or bass) to be louder than the others or softer than the others. Lack of this ability led to her doing all of her MIDI work from the PC and putting our copy of MIDI-Master "on the shelf."

Now, in our new version 2.5Z, we at last have the capability to control the volume of each voice individually. Thus the capability nears that of PC-based MIDI software. Each voice has its own "volume control byte", and that can be changed at any time in the music without affecting the volume of any other voice. For the MIDI musician, this should be the one most important new feature.

Many other features have been added. In the old MIDI-Master's input fields, there was no delete, insert, or erase capability, as we're used to finding in ACCEPT AT situations. In the new version, FCTN-1 (delete), FCTN-2 (insert), and FCTN-3 (erase) are all incorporated. This makes correcting mistakes in the input fields much easier. While we were at it, we made the cursor blink rate tied to the vertical interval timer, so the blink will be the same on either TI or Geneve. In the old version, the rate was fine on the Geneve but very slow on the TI.

Even small things that were a minor annoyance have been fixed. In the original, for example, the on-screen volume control could be run past the legal limits, with sometimes strange results. In the new version, the on-screen volume won't go lower than 0 nor higher than 127. The delay factor, which controls speed of playing, could in the original be run down to zero, which led to a stoppage of play after a delay. That's been fixed so that delay won't ever go to zero from the keyboard.

The Program Image Save

In the old MIDI-Master, even a 16-bar song, when saved in "Program" (aka memory image) format would create three files on disk that added up to 100 sectors! This always struck us as a real problem, because disk space costs. In the new version, we have the program determine just how much of the 24K memory is actually filled by the music, and save only that much in the Program file(s). Now, for example, a short song may occupy only 10 or so sectors in just one file, instead of 100 sectors. In other words, the program has been given more smarts about what needs to be put out to disk. When loading memory image files, the new version will give an error report if the first file in the series is not found, but won't

Continued on page 20

Continued from page 19

bother reporting when it can't find the second or third, as it will assume you know what you're doing. It was done this way in preparation for the AMS version, in which case we don't know when loading how many files there are in the series. Also in preparation for the AMS version, we fixed it so that when the program changes to load the next file in the series, the screen display updates to let you know what's happening.

The Out of Memory Case

In the old versions of MIDI-Master there was no indication when the program had used up all of the 24K "high" memory while compiling a source file. Everything would appear normal until you played the piece, and then somewhere toward the end voices would drop out. This was particularly annoying since there had been no warning given. In the new versions, both the AMS and non-AMS, the program tells you in plain English on the screen when it runs out of room in a 24K block.

The Improved Error Traps

The "memory filled" error trap is just one example of many where we've modified the error trapping to be a bit more "user friendly". For one thing, there are no more "Fatal File Error" reports. We don't think a program should kill anyone over a lousy file error. Those cases will now report as simply "File Error" without implications of mortality. There were two error traps that could be really annoying. The most annoying was the case where one forgot to put the CHARA1 file on the working copy of the disk. The old program would report FATAL FILE ERROR, then ask you to put a disk with CHARA1 in Drive 1. That was fine for some cases, but suppose you'd put the program into a Ramdisk or on a hard drive. The program would put you into a repeating error process that you couldn't fix without exiting the program, and there was no exit at that point except the on-off switch. In the new versions, there's an "escape hatch" in this error trap so that pressing FCTN-9 will get you out of the program.

The next most annoying thing was the case of Line 1 of the source file. Line 1 in the old program had to have three fields, two of which the program ignored. The first field was for the name of the piece (not used), the second for the number of voices to be assigned (essential), and the third was for the "version number" (not used). If all three fields were not there, this line would be rejected. In the new versions, we've fixed it so that only the name and the number need be there, with version number optional. But that wasn't the only annoying thing about the Line 1 error. If a syntax error were detected in any line after 1, the program would show you the errant line on screen, then continue compiling with the next line after a keypress. This in itself could be a problem, as any blank lines or "tabs" lines in the file would become syntax errors, and in the case of blank lines, showing the line on screen would be no help. If, however, line 1 had a syntax error, no indication would be given at all, but the program would just exit to its main menu immediately.

Several changes have been made. If Line 1 is incorrect, it will produce a Syntax Error report and show that line on the screen. Since the program can't continue compiling without getting the number of voices, it will exit back to the menu, but at least you'll know why it did this. Those mysterious syntax errors on blank lines won't happen any more, because the program now ignores blank lines except for incrementing the line count. The same goes for any "tabs" records in the file. They're counted, but not scanned, so they won't cause syntax errors.

In old versions, if the program found the end of the input file without finding a record that contained END it would issue an error report. Since the end is the end with or without END, we've had the program simply ignore the end of file error and go back to the menu. Thus if you forgot that END line, the program will forgive you without a hiccup.

Last but not least, if you're compiling a really badly mangled file, and getting lots of syntax errors, you can escape from completing the compilation by just pressing FCTN-9 when a syntax error report is on the screen.

The "SHUT UP" Problem

MIDI-Master allows you to "stop the music" in three ways, two documented and one undocumented. While in play mode, you can press P for Pause or FCTN-9 to exit play. The undocumented way is to press the 1 key, which also exits play. In most cases, however, the keyboard would simply go on playing the last notes it had been sent, so it would drone on all day unless you either turned its power off or took it out of MIDI mode. This was very annoying, even to the non-musicians in the house. In our new versions, there's a little subroutine called SHUTUP, and the program cycles through that when you press FCTN-9, P, or 1. This routine looks to see what notes are currently playing, and sends out each of those notes with a zero "velocity" byte. This tells the keyboard to stop playing those notes immediately. Our testers thought this a wonderful feature.

The "What Did I LOAD?" Problem

Once the old version loaded either a source or memory image file, there was no way to tell what had been loaded. Now, in the Play mode, there's a "Now Playing" indication which tells you on the screen the name of the file whose contents are being played. This can be especially handy when using the Album, as you may have missed the name when album was loading it. If a series of memory image files was loaded, this indication will show the first name in

Continued on page 22

MIDI-MASTER

Continued from page 21

that series, which is of course the "master" file for the series.

The Random Play Problem

In MIDI Album's original form, after selecting a few files to play, you had the choice of playing them in the order shown on the file directory or in random order. Like many people, we would assume that random would mean each selected file would play once and only once, but in random order. NOT SO! In that original MIDI Album, the random play would contunue "forever," with some selections playing three or four times while others in the selected list might not play until a half hour later. This meant that most users avoided the Random Play option.

In our new versions, Random Play means what you probably thought it should. Each selected file gets played once and only once, but in random order. When all have been played, you return gracefully to the menu. When testing Album, by the way, we found the use for the keypress 1 to stop the current music instead of FCTN-9. If you're playing from Album, pressing 1 stops the current selection and makes Album go on to the next one, if any more are waiting. FCTN-9, on the other hand, stops play and returns to the menu. This came in handy for us during testing, so we could test Album without actually having to listen to each selection all the way through. That's probably what the 1 keypress was put there for.

Larger Musical Works

Okay, now we're into the subject of the AMS version, 2.5B. If you don't have AMS, you can skip reading this part, but beware that if you do read this part and own MIDI-Master, you may be sorely tempted to go get yourself an AMS just to take advantage of this "breakthrough" in software for the AMS card.

First, as you probably know, there are different sizes of AMS cards in terms of total memory capacity. So far, there are cards of 128K, 256K, 512K, and 1024K (aka 1 Meg). In all software that your author has written for the AMS, there's a section of code at the start of the program which measures the size of the card in use and tailors the program's operation to use the capacity found. MIDI-Master 2.5B is no exception. It "knows" what size card you have, and won't let you exceed that capacity.

In the case of MIDI-Master, the program resides entirely in the Low memory at pages 2 and 3 on the card. It maps in pages to act as the high memory, where the music data is stored, starting with pages 4 thru 9 as the first "group", then 10 thru 15 as the second, and so on. Because they have to be mapped six at a time, the musical capacity is measured by how many times six divides into the number of pages from 4 onwards. A small chart will illustrate:

AMS Size	Music Capacity
128K	4 groups of 24K
256K	0 groups of 24K
512K	20 groups of 24K
1024K	42 groups of 24K

This means simply that the amount of music data that can be loaded when compared to the non-AMS MIDI Master is multiplied by the number of "groups" listed above. Thus a 256K card can handle works 10 times the size that can be handled by the normal MIDI-Master. An example will perhaps make clearer what this means.

Harold Timmons, of Columbus, Ohio, used MIDI-Master to program Gershwin's Rhapsody in Blue. That's a large piece of music. In the standard MIDI-Master, it comprised four separate source files, so that the way to play it was to load in a section, play that portion, then stop and re-load for the next section. This is a difficult way to listen to music, when there are three stoppages to load new material into memory. Mr. Timmons very kindly provided us a copy of his SNF source files so we could use them in testing the AMS version of MIDI-Master. Without Mr. Timmons' generous act, we might never have finished this version, simply for lack of a test subject. Once we had his source files in hand, the pressure was on to get the AMS version finished.

No revision of his files was required. In each section, he'd thoughtfully used the same number of voices, and assigned them to the same MIDI channels. It works like this: One uses the compile function (option 4) and types in the name of the first source file. (e.g. DSK2.RAPSODY1) Compiling takes a while, as each section fills more than 2/3 of the 24K memory group. When that finishes, a prompt saying "MORE? (Y/N)" appears on the screen. If you answer Y for Yes, the program puts you back in the input file field, with the previous entry still there, so you can change it to the next file, in this case DSK2.RAPSODY2. Unbeknownst to you, but knownst to me, when you pressed Y, the AMS card was paged so that this new file will compile into pages 10 through 15 of the AMS, leaving the first part still there in pages 4 thru 9. Again you wait, then repeat the process, answering Y two more times, compiling into pages 16 thru 21, then 22 thru 27. The program keeps track of what's the highest "group" that's been used, and you don't even have to think about that. Finally, after compiling RAPSODY4, you press N for No at the MORE prompt. You're back on the menu, but now ALL of Gershwin's Rhapsody in Blue is sitting there in pages 4 through 27 of your AMS card. Hook up the keyboard and press 2.

The entire Rhapsody will play from beginning to end. The program "knows" when a section has finished, and very quickly sets the AMS to start playing what's in the next group of six pages. It knows what the highest group loaded is, so at the end of the fourth section, you're back to the menu. Now if you've got an initialized disk handy, press 3 to save all this in memory image format. We use short file names, so we started with

Continued on page 24

MIDI-MASTER

MIDI-MASTER

Continued from page 23

RBA for the first file name. Saving to floppy takes a while, as there are twelve sequential files in this case, but it all works, updating the last letter of the file name on the screen each time it writes a new file.

Now we could make much shorter work of getting the Rhapsody into our AMS by using these memory image files. Just select 1 from the menu, type in DSKx.RBA, and all twelve files will be piled into your AMS card, ready to play. As an experiment, we made room on one of our Ramdisk drives for these twelve files, and that made the loading process very quick indeed.

Works of even longer duration could be handled by our 256K card, since the Rhapsody uses only four of its ten groups. Just imagine the possibilities! Perhaps a whole symphony played through MIDI-Master!

You Gotta Have Cables!

To use either new version, you must have the special cable supplied by either Crystal Software or Cecure Electronics. That's one thing we can't supply. Through special arrangement with Cecure, Richard Bell of Staten Island, New York, can now also supply the original MIDI-Master package, including cables. Contact him at 38 Bement Ave. Staten Island, NY 10310, or via e-mail swim2shore@email.msn.com. Richard will also include a copy of the updated version 2.5Z or 2.5B, as appropriate to your needs. Both versions are still copyrighted by Michael J. Maksimik, and are being made available as such. Make a backup for your own use, but "sharing" is not permitted.

Before We Leave, More Thanks

Throughout the development of these new versions, two people contributed their time and efforts in testing the many pre-release editions. Thanks then to Harold Timmons and Richard Bell for all their help and encouragement.

RAM USAGE

How 32K of RAM memory stacks up

BY PETER HUTCHISON

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

Just what is a 32K memory and how does it work and what is it used for? I hope this article will answer these questions. If you run Extended BASIC with expansion RAM and type the command SIZE, you'll see:

13928 BYTES OF STACK FREE
24511 BYTES OF PROGRAM SPACE FREE

RAM USAGE

Note that differing values will be given if you have a disk system, the state of CALL FILES, and the version of Extended BASIC you are using.

The 13K part is the RAM that comes in the console (VDP RAM) and the 24K part is the expansion RAM. Where has the other 8K gone to? The 8K part is for machine code routines and is not used by Extended Basic. Extended BASIC can LINK to machine code routines in this area. Pure machine code programs can fill all 32K if required.

Another question is — why am I limited to 12K programs if I use cassette for storage? The answer is the way the TI saves code. If you have one, look at the Editor/Assembler manual, page 297-SAVE. It says "the SAVE operation writes a file from VDP RAM to a peripheral." How much VDP RAM do we have? Thirteen kilobytes. In fact, the TI copies the program from expansion RAM to VDP RAM and then to the peripherals. This explains the short delay when you type SAVE CS1. Longer programs are allowed on disk as the console switches to an alternate format that saves programs as shorter records instead of dumping them all at once. This format is "Internal Variable 254," but you needn't worry about that.

The 32K RAM area does not have continuous addresses (Editor Assembler manual page 400):

>0000	console ROM (2x4K ROM chips)
>2000	low memory expansion (8K)
>4000	peripheral ROMS for Device Service Routines
>6000	reserved for modules (8K) — ROM or RAM
>8000	memory mapped devices, VDP, GROM, SOUND, SPEECH
>AOOO	High memory expansion (24K)

If you use Extended BASIC, High Mem will contain your BASIC program and numeric data. Console RAM will contain strings, and data for the screen display, sounds, sprites, and pattern definitions.

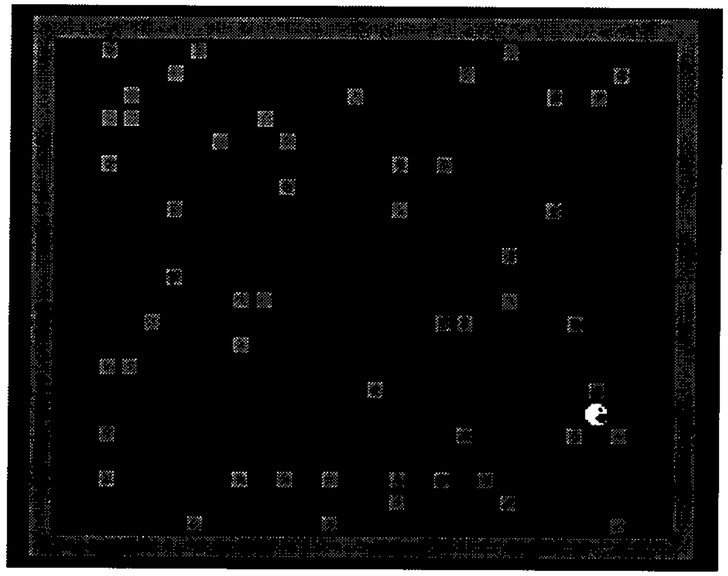
Low memory, on the other hand, will be used (if at all) by CALL LOAD and CALL LINK commands, for loading and linking to machine code routines. Expansion memory can be used by other modules, such as Editor/Assembler (32K to store machine code programs), Mini-Memory (as Editor/Assembler, or as two RAMdisks), TI-LOGO (32K required), etc.

With the disk system, 32K is essential as the disk operation takes up some memory from VDP RAM.

The 32K RAM cannot be used by TI BASIC or modules not designed for its use. RAMdisks may incorporate rather than replace the 32K standard expansion. For example, the Myarc 512K card uses 32K for normal purposes, and the remainder for RAMdisk or printer buffer uses.

CLAUSTROPHOBIA

The object is to keep the critter contained



The object of Claustrophobia, by W. Van Santvliet, is to contain an onscreen critter that wants to be free. You do this by having a spider-like creature push blocks around the screen using the arrow keys. Just to make it a little more complicated, some of the blocks are stationary.

The game includes a number of difficulty levels so that once you've accomplished your task at one level you can look forward to doing it again at a more difficult level. Of course, to survive you have to avoid collisions between your spider and the critter you're trying to contain. Collisions end the game.

CLAUSTROPHOBIA

```
0 CALL BXB !068
100 CALL CLEAR !209
110 CALL SCREEN(5)!150
120 CALL CHAR(128,"007E7E7E7
E7E7E00")!140
130 CALL COLOR(13,10,5)!019
140 CALL SOUND(500,262,0,523
,0)!072
150 CALL HCHAR(3,3,128,3)!17
6
160 CALL VCHAR(4,3,128,4)!19
2
170 CALL HCHAR(7,4,128,2)!18
0
180 CALL SOUND(500,294,0,587
,0)!087
190 CALL VCHAR(9,6,128,5)!20
1
200 CALL HCHAR(13,7,128,2)!2
29
210 CALL SOUND(500,330,0,659
,0)!078
220 CALL HCHAR(1,9,128,3)!18
0
230 CALL VCHAR(2,9,128,4)!19
6
```


CLAUSTROPHOBIA

```
240 CALL VCHAR(2,11,128,4)!2
38
250 CALL HCHAR(3,10,128)!048
260 CALL SOUND(500,349,0,698
,0)!091
270 CALL VCHAR(2,15,128,5)!2
43
280 CALL VCHAR(2,17,128,5)!2
45
290 CALL HCHAR(6,16,128)!057
300 CALL SOUND(500,392,0,784
,0)!085
310 CALL HCHAR(8,13,128,3)!2
31
320 CALL VCHAR(9,13,128,2)!2
45
330 CALL HCHAR(10,14,128,2)!
017
340 CALL VCHAR(11,15,128,2)!
033
350 CALL HCHAR(12,13,128,2)!
018
360 CALL SOUND(500,440,0,880
,0)!076
370 CALL HCHAR(6,21,128,3)!2
28
380 CALL VCHAR(7,22,128,4)!2
45
390 CALL SOUND(500,494,0,988
,0)!094
400 CALL HCHAR(2,26,128,3)!2
29
410 CALL VCHAR(3,26,128,4)!2
45
420 CALL VCHAR(3,28,128,2)!2
45
430 CALL VCHAR(4,27,128,2)!2
45
440 CALL HCHAR(6,28,128)!060
450 CALL SOUND(500,523,0,131
,0)!067
460 CALL HCHAR(10,27,128,3)!
022
470 CALL VCHAR(11,27,128,4)!
038
480 CALL VCHAR(11,29,128,4)!
040
490 CALL HCHAR(14,28,128)!10
8
500 CALL SOUND(500,587,0,147
,0)!084
510 CALL HCHAR(17,3,128,3)!2
30
520 CALL VCHAR(18,3,128,4)!2
46
530 CALL HCHAR(19,4,128)!058
540 CALL SOUND(500,659,0,165
,0)!084
550 CALL HCHAR(15,9,128,3)!2
34
560 CALL VCHAR(16,9,128,4)!2
50
570 CALL VCHAR(16,11,128,4)!
036
580 CALL HCHAR(19,10,128)!10
4
590 CALL SOUND(500,698,0,175
,0)!088
600 CALL HCHAR(16,16,128,3)!
026
610 CALL VCHAR(17,16,128,4)!
042
620 CALL HCHAR(17,18,128)!11
```

Continued on page 28

CLAUSTROPHOBIA

Continued from page 27

```
0
630 CALL HCHAR(18,17,128,3)!
029
640 CALL HCHAR(19,19,128)!11
3
650 CALL HCHAR(20,17,128,3)!
022
660 CALL SOUND(500,784,0,196
,0)!087
670 CALL VCHAR(14,22,128,5)!
037
680 CALL SOUND(500,880,0,220
,0)!072
690 CALL HCHAR(17,26,128,3)!
028
700 CALL VCHAR(18,26,128,4)!
044
710 CALL VCHAR(18,28,128,4)!
046
720 CALL HCHAR(19,27,128)!11
2
730 CALL SOUND(500,988,0,247
,0)!090
740 CALL HCHAR(24,7,66)!009
750 CALL HCHAR(24,8,89)!015
760 CALL HCHAR(24,10,87)!055
770 CALL HCHAR(24,11,46)!051
780 CALL SOUND(500,1047,0,26
2,0)!123
790 CALL HCHAR(24,13,86)!057
800 CALL HCHAR(24,14,65)!055
810 CALL HCHAR(24,15,78)!060
820 CALL HCHAR(24,17,83)!058
830 CALL HCHAR(24,18,65)!059
840 CALL HCHAR(24,19,78)!064
850 CALL HCHAR(24,20,84)!053
860 CALL HCHAR(24,21,86)!056
870 CALL HCHAR(24,22,76)!056
880 CALL HCHAR(24,23,73)!054
890 CALL HCHAR(24,24,69)!060
900 CALL HCHAR(24,25,84)!058
910 FOR A=1 TO 1000 !195
920 NEXT A !215
930 CALL CLEAR !209
940 PRINT TAB(7);"CLAUSTROFO
BIA": :!197
950 PRINT TAB(6);"==========
=====": :!139
960 PRINT "TRY TO CAPTURE TH
E ENEMY BY ENCIRCLING HIM WI
TH BUFFERS.": :!008
970 PRINT "YOU CAN PUSH ONE
OR MORE    BUFFERS HORIZONTA
LLY OR     VERTICALLY.": :!0
95
980 PRINT "BUT WATCH OUT !!!
": :!251
990 PRINT "THE GREEN BUFFERS
 CAN'T BE  MOVED AND THE ENE
MY CAN     SUDDENLY CHANGE P
LACES.": :!226
1000 PRINT "USE THE ARROW-KE
YS TO MOVE  AND PRESS 'C' WH
EN YOU HAVE CAPTURED THE ENE
MY (DO IT   FAST).": :!251
1010 PRINT "PRESS ANY KEY TO
 CONTINUE." !145
1020 CALL KEY(0,K,S)!187
1030 IF S=0 THEN 1020 !006
1040 CALL CLEAR !209
1050 PRINT "TEN LEVELS ARE P
ROVIDED.": :!090
1060 PRINT "THE HIGHER THE L
```


CLAUSTROPHOBIA

```
EVEL:": :!109
1070 PRINT "  A) THE MORE FI
XED BUFFERS.": :!210
1080 PRINT "  B) THE QUICKER
 THE ENEMY       CAN CHANGE
PLACES.": :!253
1090 PRINT "THERE IS ALSO A
LEVEL FOR   CHILDREN (LEVEL
0).         HERE POINTS A AN
D B DON'T   APPLY.": : :!246
1100 PRINT "      GOOD LUCK
!!!": : :!080
1110 PRINT "PRESS ANY KEY TO
 START.": : : : :!064
1120 CALL KEY(0,K,S)!187
1130 IF S=0 THEN 1120 !107
1140 CALL CLEAR !209
1150 PRINT "LEVEL OF DIFFICU
LTY" !182
1160 PRINT !156
1170 INPUT "(0-10)
    ":DIF !196
1180 PRINT !156
1190 PRINT !156
1200 IF (DIF>10)+(DIF<0)THEN
 1150 !099
1210 PRINT "NUMBER OF BUFFER
S:" !090
1220 PRINT !156
1230 INPUT "(MAX.400)
    ":QUA !098
1240 IF (QUA>400)+(QUA<4)THE
N 1180 !225
1250 CALL CLEAR !209
1260 HYPER=0 !063
1270 TIME=0 !230
1280 IF DIF=0 THEN 1310 !170
1290 TEST=(11-DIF)*10 !026
1300 GOTO 1320 !124
1310 TEST=1000 !139
1320 CALL CHAR(136,"FFFFFFFF
FFFFFFFF")!067
1330 CALL CHAR(137,"007E7E7E
7E7E7E00")!140
1340 CALL CHAR(144,"99997E3C
3C7E9981")!133
1350 CALL CHAR(145,"C3243C7F
7F3C24C3")!129
1360 CALL CHAR(146,"81997E3C
3C7E9999")!135
1370 CALL CHAR(147,"C3243CFE
FE3C24C3")!159
1380 CALL CHAR(152,"3C7EFBFF
F8F0793E")!195
1390 CALL CHAR(96,"18187E7E1
8181818")!037
1400 CALL COLOR(14,3,5)!229
1410 CALL COLOR(15,2,5)!229
1420 CALL COLOR(16,12,5)!024
1430 CALL CLEAR !209
1440 CALL HCHAR(1,2,136,30)!
221
1450 CALL VCHAR(2,2,136,22)!
237
1460 CALL VCHAR(2,31,136,22)
!032
1470 CALL HCHAR(24,2,136,30)
!019
1480 RANDOMIZE !149
1490 IF DIF=0 THEN 1570 !175
1500 FOR A=1 TO 2*DIF !199
1510 X=INT(RND*22)+2 !214
1520 Y=INT(RND*24)+5 !220
```

Continued on page 30

CLAUSTROPHOBIA

Continued from page 29

```
1530 CALL GCHAR(X,Y,C)!150
1540 IF C=137 THEN 1510 !079
1550 CALL HCHAR(X,Y,137)!186
1560 NEXT A !215
1570 FOR A=1 TO QUA !029
1580 X=INT(RND*22)+2 !214
1590 Y=INT(RND*24)+5 !220
1600 CALL GCHAR(X,Y,C)!150
1610 IF C<>32 THEN 1580 !032
1620 CALL HCHAR(X,Y,128)!186
1630 NEXT A !215
1640 RICH=4 !225
1650 MX=INT(RND*22)+2 !035
1660 MY=30 !145
1670 CALL HCHAR(MX,MY,152)!0
81
1680 X=INT(RND*22)+2 !214
1690 Y=3 !019
1700 CALL HCHAR(X,Y,145)!185
1710 FOR A=1 TO 3 !050
1720 CALL SOUND(100,523,0)!1
28
1730 CALL SOUND(100,659,0)!1
38
1740 NEXT A !215
1750 CALL SOUND(400,523,0)!1
31
1760 CALL KEY(0,KEY,STATUS)!
234
1770 TIME=TIME+1 !215
1780 IF STATUS=0 THEN 2700 !
046
1790 IF KEY=69 THEN 1840 !02
0
1800 IF KEY=68 THEN 2090 !01
4
1810 IF KEY=88 THEN 1860 !04
1
1820 IF KEY=83 THEN 2070 !24
7
1830 GOTO 2700 !229
1840 DIR=-1 !089
1850 GOTO 1870 !164
1860 DIR=1 !151
1870 CALL SOUND(30,500,0)!07
6
1880 TEL=0 !156
1890 CALL GCHAR(X+DIR,Y,A)!0
52
1900 IF A=32 THEN 1950 !208
1910 IF (A=136)+(A=137)+(A=1
52)THEN 2030 !230
1920 X=X+DIR !014
1930 TEL=TEL+DIR !040
1940 GOTO 1890 !184
1950 IF TEL=0 THEN 1980 !092
1960 IF A=136 THEN 2030 !086
1970 CALL HCHAR(X+DIR,Y,128)
!090
1980 X=X-TEL+DIR !181
1990 CALL SOUND(50,-1,0)!170
2000 CALL HCHAR(X-DIR,Y,32)!
036
2010 CALL HCHAR(X,Y,145+DIR)
!089
2020 GOTO 2700 !229
2030 X=X-TEL !021
2040 IF A=152 THEN 2300 !099
2050 CALL SOUND(100,110,0)!1
20
2060 GOTO 2700 !229
2070 DIR=-1 !089
2080 GOTO 2100 !139
```


CLAUSTROPHOBIA

```
2090 DIR=1 !151
2100 CALL SOUND(30,500,0)!07
6
2110 TEL=0 !156
2120 CALL GCHAR(X,Y+DIR,A)!0
52
2130 IF A=32 THEN 2180 !183
2140 IF (A=136)+(A=137)+(A=1
52)THEN 2260 !205
2150 Y=Y+DIR !016
2160 TEL=TEL+DIR !040
2170 GOTO 2120 !159
2180 IF TEL=0 THEN 2210 !067
2190 IF A=136 THEN 2030 !086
2200 CALL HCHAR(X,Y+DIR,128)
!090
2210 Y=Y-TEL+DIR !183
2220 CALL SOUND(50,-1,0)!170
2230 CALL HCHAR(X,Y-DIR,32)!
036
2240 CALL HCHAR(X,Y,146-DIR)
!091
2250 GOTO 2700 !229
2260 Y=Y-TEL !023
2270 IF A=152 THEN 2390 !190
2280 CALL SOUND(100,110,0)!1
20
2290 GOTO 2700 !229
2300 X=X+DIR !014
2310 IF TEL=0 THEN 2370 !228
2320 CALL SOUND(10,500,0)!07
4
2330 CALL SOUND(10,500,0)!07
4
2340 CALL HCHAR(X-DIR,Y,32)!
036
2350 CALL HCHAR(X,Y,145+DIR)
!089
2360 GOTO 2700 !229
2370 CALL HCHAR(X-DIR,Y,32)!
036
2380 GOTO 2480 !008
2390 Y=Y+DIR !016
2400 IF TEL=0 THEN 2460 !062
2410 CALL SOUND(10,500,0)!07
4
2420 CALL SOUND(10,500,0)!07
4
2430 CALL HCHAR(X,Y-DIR,32)!
036
2440 CALL HCHAR(X,Y,146-DIR)
!091
2450 GOTO 2700 !229
2460 CALL HCHAR(X,Y-DIR,32)!
036
2470 GOTO 2480 !008
2480 CALL HCHAR(X,Y,96)!141
2490 DATA 500,262,375,262,12
5,262,500,262,375,311,125,29
4,250,294,250,262,250,262,25
0,233,500,262 !222
2500 RESTORE 2490 !032
2510 FOR A=1 TO 11 !098
2520 READ T !235
2530 READ F !221
2540 CALL SOUND(T,F,0)!089
2550 NEXT A !215
2560 CALL CLEAR !209
2570 PRINT TAB(11);"YOU LOSE
" !004
2580 FOR A=1 TO 12 !099
2590 PRINT !156
2600 NEXT A !215
```

Continued on page 32

CLAUSTROPHOBIA

Continued from page 31

```
2610 PRINT TAB(4);"PRESS ANY
 KEY TO PLAY AGAIN" !207
2620 CALL KEY(0,K,S)!187
2630 IF S=0 THEN 2620 !077
2640 GOTO 1140 !199
2650 CALL KEY(0,K,S)!187
2660 IF K=67 THEN 2970 !226
2670 HYPER=HYPER+1 !137
2680 IF HYPER>TEST THEN 3210
 !027
2690 RICH=INT(RND*4)+1 !114
2700 ON RICH GOTO 2710,2840,
2710,2840 !010
2710 ZIN=INT(RICH-1.5)!050
2720 CALL GCHAR(MX+ZIN,MY,CH
)!042
2730 IF CH<>32 THEN 2790 !03
8
2740 CALL HCHAR(MX,MY,32)!02
9
2750 CALL SOUND(30,-7,0)!174
2760 MX=MX+ZIN !186
2770 CALL HCHAR(MX,MY,152)!0
81
2780 GOTO 1760 !053
2790 IF (CH<144)+(CH>147)THE
N 2650 !092
2800 CALL HCHAR(MX,MY,32)!02
9
2810 CALL HCHAR(MX+ZIN,MY,15
2)!003
2820 CALL SOUND(1000,-7,0)!0
14
2830 GOTO 2500 !028
2840 ZIN=-(INT(RICH-2.5))!09
8
2850 CALL GCHAR(MX,MY+ZIN,CH
)!042
2860 IF CH<>32 THEN 2920 !16
9
2870 CALL HCHAR(MX,MY,32)!02
9
2880 CALL SOUND(30,-7,0)!174
2890 MY=MY+ZIN !188
2900 CALL HCHAR(MX,MY,152)!0
81
2910 GOTO 1760 !053
2920 IF (CH<144)+(CH>147)THE
N 2650 !092
2930 CALL HCHAR(MX,MY,32)!02
9
2940 CALL HCHAR(MX,MY+ZIN,15
2)!003
2950 CALL SOUND(1000,-7,0)!0
14
2960 GOTO 2500 !028
2970 CALL GCHAR(MX-1,MY,N)!2
47
2980 CALL GCHAR(MX,MY+1,E)!2
37
2990 CALL GCHAR(MX+1,MY,S)!2
51
3000 CALL GCHAR(MX,MY-1,W)!0
00
3010 IF (N=128)*(E=128)*(S=1
28)*(W=128)THEN 3070 !208
3020 N=0 !005
3030 E=0 !252
3040 S=0 !010
3050 W=0 !014
3060 GOTO 1760 !053
```


CLAUSTROPHOBIA

```
3070 CALL HCHAR(MX,MY,96)!03
9
3080 FOR F=1000 TO 110 STEP
-40 !205
3090 CALL SOUND(-200,F,0)!03
6
3100 NEXT F !220
3110 FOR A=1 TO 1000 !195
3120 NEXT A !215
3130 CALL CLEAR !209
3140 PRINT TAB(12);"YOU WIN"
 !191
3150 PRINT !156
3160 PRINT TAB(11);"TIME =";
TIME !033
3170 FOR A=1 TO 10 !097
3180 PRINT !156
3190 NEXT A !215
3200 GOTO 2610 !139
3210 CALL HCHAR(MX,MY,32)!02
9
3220 MX=INT(RND*22)+2 !035
3230 MY=INT(RND*30)+2 !035
3240 CALL GCHAR(MX-1,MY,N)!2
47
3250 CALL GCHAR(MX,MY-1,W)!0
00
3260 CALL GCHAR(MX,MY,CH)!12
0
3270 CALL GCHAR(MX,MY+1,E)!2
37
3280 CALL GCHAR(MX+1,MY,S)!2
51
3290 IF (CH<>32)+((N<>32)*(E
<>32)*(S<>32)*(W<>32))THEN 3
220 !099
3300 CALL HCHAR(MX,MY,152)!0
81
3310 FOR F=110 TO 1000 STEP
70 !014
3320 CALL SOUND(-100,F,0)!03
5
3330 NEXT F !220
3340 HYPER=0 !063
3350 GOTO 1760 !053
30000 SUB BXB :: CALL INIT :
: CALL LOAD(8194,37,194,63,2
40)!126
30001 CALL LOAD(16368,80,79,
67,72,65,82,37,58,80,79,75,6
9,86,32,37,168)!133
30002 ][\[]$=" %        00
%     %          %
      P     %
    a     q       q  F
  %      $    %
 |      ` p" !024
30003 FOR J=1 TO 136 :: CALL
 LOAD(9529+J,ASC(SEG$(][\[]$
,J,1))):: NEXT J :: SUBEND !
051
30004 SUB CHAR(A,A$):: CALL
LOAD(9500,A):: CALL LINK("PO
CHAR",A$):: SUBEND !169
30005 SUB COLOR(A,B,C):: CAL
L LOAD(9492,8,15+A,(B-1)*16+
C-1)!013
30006 CALL LINK("POKEV"):: S
UBEND !127
```

EXTENDED BASIC

Figuring sales taxes depends on percentages and a little help from the TI

BY LEONARD TAFFS

This "journal" column never abandons the thought that there may be people interested in the TI who are not TI veterans immersed in SCSI and AMS cards, using Geneves and Rave keyboards, devoting reams to how to patch TI's to PC's, etc.... For those who go "on-line," just to read the messages on The River TI server list to see....

Rather, it is the intent of this column to strive to contribute to what is perceived to be a growing void of less complicated material for the TI "newcomer." If the TI is a dying species, its end will only be hastened by the dominating elite being oblivious to the possible needs of newcomers, however unintentional this might be.

Most of this column's program listings contributions are of programs utilizing Extended BASIC (some of which require memory expansion). I would be glad to include more materials in TI BASIC format if you inform me of such need.

Finding Percentage

Consider the situation where a phone bill is shared with others — each paying his fair proportion of the total bill and proportion of the tax and other charges. For phone bills which do not tell the customer the percentage rate of these extra charges, this means using a calculator unless you are gifted in math. If the calculator is not used very much there is always the question as to how worn its batteries may be and did the last person to use it drop it — in which case it may not be reliable. The TI can come to the rescue.

Using command mode with the TI you need to know what to divide by what to calculate percentage. The simple formula: TOTAL AMOUNT OF TAXES AND OTHER CHARGES divided by TOTAL OF PHONE CALL charges will equal the TAX PERCENTAGE. If one has no problem with remembering this formula, one does not need a program. But some people still do. There are many calculator utilities which have a menu option for calculating percentage but if you don't have such a utility, FINDPERCNT, the Extended BASIC program listing with this article may be useful.

FINDPERCNT

```
1 REM [FINDPERCNT] 4-19-97
By W.Leonard Taffs, SW99ers
!018
2 !!131
3 ! call key commands:
    1 CALCULATE TAX percent
    2 CALCULATE TAX amount !
210
4 ! 3 ADD-SUB adder
    4 DISPLAY COMMAND KEYS !
123
5 ! 5 DEF DISPLAY ADD-SUB ME
MRY !033
6 ! 6 DISPLAY ARRAY
    7 CLEAR SCREEN !236
7 ! 8 SEND ARRAY TO PRINTER
    9 QUIT PROGRAM !063
8 !!131
100 DIM AR(10),AR$(10):: CAL
L CLEAR :: CALL BLUE :: DSP$
="1=% 2=TX 3=AS 4=K 5=M 6=A
7C8=P
9Q" :: OPEN #1:"PIO" !048
110 AR$(1)="AMT OF TAX" :: A
R$(2)="TOT TAXED" :: AR$(3)=
"TAX %" :: AR$(4)="CMP TAXAB
LE=" :: AR$(5)="CMP TAX RATE
" !075
120 AR$(6)="COMP TAX=" :: AR
$(7)="ITEM 1" :: AR$(8)="ITE
M 2" :: AR$(9)="IT1 + IT2" !
204
130 DISPLAY AT(1,1):DSP$ ::
DISPLAY AT(3,4):"TAX PERCENT
/AMT FINDER" :: GOTO 320 !06
2
140 DISPLAY AT(5,1):"ENTER T
AX AMT $" :: ACCEPT AT(5,17)
:TAX :: IF TAX=999 THEN GOSU
B 270 :: GOTO 140 ELSE AR(1)
=TAX !199
150 DISPLAY AT(7,1):"ENTER $
 AMT " :: ACCEPT AT(7,14):AM
T :: AR(2)=AMT !049
160 AR(3)=AR(1)/AR(2)!115
```


EXTENDED BASIC

```
170 DISPLAY AT(9,1):"TAX";AR
(1);"/ AMT";AR(2);"=": :"
 [";AR(3);"% ]" :: MEM=AR(1)
/AR(2)!154
180 GOTO 310 !134
190 DISPLAY AT(14,1):"CALCUL
ATE TAX NOW:" :: DISPLAY AT(
16,1):"ENTER $ AMT: " :: ACC
EPT AT(16,15):AMT :: IF AMT=
-99 THEN 230 ELSE IF AMT=999
99 THEN GOSUB 270 ELSE AR(4)
=AMT :: DISPLAY AT(1,1):DSP$
 !045
200 DISPLAY AT(18,1):"ENTER
FOUND % " :: ACCEPT AT(18,16
):FP :: AR(5)=FP !161
210 DISPLAY AT(20,1):"TAX IS
: ";AMT*FP :: AR(6)=AMT*FP !
200
220 GOTO 310 !134
230 REM ** CALCULATE ** !048
240 DISPLAY AT(22,1):"#1= "
:: ACCEPT AT(22,10):IT1 :: D
ISPLAY AT(23,1):"#2= " :: AC
CEPT AT(23,10):IT2 :: DISPLA
Y AT(24,1):"DIFF= ";IT1+IT2;
" P.E.T.C." :: AR(7)=IT1 ::
AR(8)=IT2 :: AR(9)=IT1+IT2
250 CALL KEY(0,K,S):: IF S<1
 THEN 250 :: DISPLAY AT(20,1
):RPT$(" ",140):: GOTO 310 !
160
260 DISPLAY AT(20,1):RPT$("
",140):: GOTO 190 !101
270 REM ** DISPLAY CALCS **
!062
280 DISPLAY AT(1,1):"IT1";AR
```

Continued on page 36

EXTENDED BASIC

Continued from page 35

```
(5);"IT2";AR(6);"DF";AR(7)!0
96
290 CALL KEY(0,K,S):: IF S<1
 THEN 290 !105
300 RETURN !136
310 REM ** CALL KEY ** !199
320 CALL KEY(0,K,S):: IF S<1
 THEN 320 :: IF (K-48<1)+(K-
48>9)THEN 320 !227
330 ON K-48 GOTO 140,190,230
,340,350,430,420,360,480 !14
3
340 DISPLAY AT(1,1):DSP$ ::
GOTO 320 !120
350 DISPLAY AT(1,1):"IT1";AR
(7);"IT2";AR(8);"DF";AR(9):R
PT$(" ",28):: GOTO 320 !074
360 FOR I=1 TO 9 !064
370 PRINT #1:TAB(10);AR$(I);
" ";AR(I)!150
380 IF I=3 THEN PRINT #1:TAB
(10);RPT$("-",20)ELSE IF I=6
 THEN PRINT #1:TAB(10);RPT$(
"-",20)!086
390 NEXT I !223
400 PRINT #1:TAB(10);RPT$("=
",20)!194
410 GOTO 320 !144
420 CALL CLEAR :: GOTO 130 !
036
430 CALL CLEAR :: J=2 !086
440 FOR I=1 TO 9 !064
450 DISPLAY AT(I+J,1):AR$(I)
,AR(I):: J=J+1 :: IF I=2 THE
N DISPLAY AT(I+J+2,1):RPT$("
=",28)!016
455 IF I=4 THEN DISPLAY AT(I
+J+4,1):RPT$("=",28)!229
460 NEXT I !223
465 DISPLAY AT(24,1):"Use""
7"" to Clear This Screen" !2
31
470 GOTO 320 !144
480 DISPLAY AT(22,1):"SURE Y
OU WANT TO QUIT? Y/N" :: ACC
EPT AT(22,28)SIZE(-1)VALIDAT
E("NYny"):Y$ !035
490 IF Y$<>"Y" AND Y$<>"y" T
HEN DISPLAY AT(21,1):RPT$("
",84):: GOTO 130 ELSE STOP !
098
500 REM [CALL/BLUE] !229
510 SUB BLUE !149
520 CALL SCREEN(5)!150
530 FOR L=0 TO 14 !111
540 CALL COLOR(L,16,1)!051
550 NEXT L !226
560 SUBEND !168
```

The Extended BASIC program listing above does this calculation for you when you enter needed factors. FINDPERCNT consists of three sections:

1. A section to calculate the TAX PERCENTAGE
2. A section to find TAX AMOUNT for specified phone charges you input
3. A mini add-subtractor section

All data you enter is stored in arrays. This information is not changed until you make new entries.

```
1 REM [FINUDERCNT] 4-19-97
By W.Leonard Taffs, SW99ers
2 !
3 ! call key commands:
  1 CALCULATE TAX percent
  2 CALCULATE TAX amount
4 !
  3 AM-SUB adder
  4 DISPLAY COMMAND KEYS
5 !
  5 DISPLAY ADD-SUB MEMRY
```


EXTENDED BASIC

The opening screen consists of a three-line display at the top of your screen:

```
1=% 2=TX 3=CA 4=D 5=M 6=A 7=C
8=P                        9=Q
  TAX PERCENT/AMT FINDER
```

Note that there is no blinking cursor. This is because this program uses CALL KEY action for all its commands, of which there are nine. Keys 1 through 3 are used to enter amounts to be calculated. Keys 4 through 9 are user conveniences for displays or sending results to your printer.

The CALL KEY commands, in abbreviated form, appear in the first two lines — this constitutes the "MENU" of this program. The key-presses and functions are:

1. CALCULATE TAX percent (%)
2. CALCULATE TAX amount (TX)
3. Mini ADD-SUBTRACTOR (CA)
4. DISPLAY COMMAND Call Keys (D) (top 2 lines)
5. DISPLAY mini ADD-SUB-TRACTOR entries and product (M) (top line of screen)
6. DISPLAY ARRAY (A) (show all data entered)
7. CLEAR SCREEN (C)
8. PRINT ARRAY (P) (send all data to PRINTER)
9. QUIT PROGRAM (Q)

To view this list on your screen, use FCTN-4 to break out of the program and:

1. ENTER: CALL CLEAR
2. ENTER: PRINT DSP$ (won't appear if you have not used FCTN-4 to BREAK)
3. ENTER: LIST 3-7

PRINTER

(8) Note: when you send your entries to your printer it will print the summary of any data resident in arrays. This is the same information

Continued on page 38

```
AMT OF TAX      1.23
TOT TAXED       21.69
TAX %           .0567081604
============================
CMP TAXABLE=    21.69
CMP TAX RATE    .0567
COMP TAX=       1.229823
============================
ITEM 1          0
ITEM 2          0
IT1 + IT2       0


 Use "7" to Clear This Screen
```

Screen 1

Continued from page 37

that is displayed to your screen if you you press "6." When using "8," the printer must be on online. If it isn't, the program cannot continue until it is. The summary will appear as shown in Screen 1.

This display summary summarizes whatever your last entries were and computer calculations prior to your pressing "6" or "8." (In this example the Add-Sub was not used). With "8," the full summary will be printed regardless if any entries have been made. The above figures are for:

AMT OF TAX what you entered for phone company bill tax total

TOT TAXED what you entered as bill total without tax

TAX percentage calculated using these 2 figures.

Below the double line appear your entries for "CALCULATE TAX NOW (AMT)" and "ENTER FOUND %" with calculated tax amount result.

The last three summary items show the two Add-Subtract entries made along with their product.

Screen 2 shows your screen display when entries have been made using the three input sections of the program:

Stepping through the program from a cold start:

Press "1" for the first prompt:

ENTER TAX AMT $

Enter the total of taxes/excise charges, etc., from your bill and press Enter. Now appears;

ENTER $ AMT

Enter the phone bill total (without taxes) and press Enter. The screen displays the figures you just entered and shows the tax percentage rate in brackets and the program waits for your next keypress.

Press "2." You will see:

CALCULATE TAX NOW:

ENTER $ AMT:

Enter whatever amount you wish to figure tax for and press Enter. Next appears:

ENTER FOUND %

Here you enter either the rate displayed in brackets in step 1 (be sure to use a decimal point), or any other rate if desired. Press Enter again

```
1=% 2=TX 3=AS 4=K 5=M 6=A 7C
8=P                        9Q
   TAX PERCENT/AMT FINDER
ENTER TAX AMT $ 1.23
ENTER $ AMT   21.69
TAX 1.23 / AMT 21.69 =
    [ .0567081604 % ]

CALCULATE TAX NOW:
ENTER $ AMT:   128.43
ENTER FOUND %   .0567
TAX IS:   7.281981

#1=        400
#2=       -18.99
DIFF=   381.01   P.E.T.C.
```

Screen 2

and the screen displays:

TAX IS: (shows tax amount).

Press "3" (to use the mini Add-Subtractor) and you will see the prompt "#1=" near the bottom left of the screen. This allows the first of two figure entry inputs. Entering a number and pressing Enter brings the second prompt "#2=". Enter your second figure (using a minus sign if you wish to subtract) and press Enter and the sum appears as:

DIFF= (sum) P.E.T.C.

"P.E.T.C." means Press Enter To Continue. Pressing Enter (to continue) will erase display of your Add-Subtract figures. They are retained in memory until you use the Add-Subtractor again. At any time you wish to see these Add-Subtractor figures, press "5" and your figures will be shown at the top line of your screen as "IT1," "IT2," and "DF." You can toggle between the "4" and "5" keys to switch between top-screen displays.

The remaining keypress options are:

Pressing "6" will clear the screen and display all information stored in the program array.

Pressing "7" will clear the screen at any time (variables are not cleared from the array) and return you to your opening screen.

Pressing "8" will send data stored in your array to your printer.

Pressing "9" will end your program but not without a chance for you to get back in at "SURE YOU WANT TO QUIT? Y/N" prompt.

Being able to clear your screen with "7" is convenient when your screen gets messed up, which will happen if you enter zero for both inputs of step 1. (You would get numeric overflow error warnings — if you do, just press "7.") Another case where your screen will get messed up is if you accidentally press a comma, instead of period for a decimal point, or attempt to use multiplication or division in the Add-Subtract portion, in which case you get a String-Number Mismatch Warning — again, press "7" when the figure has been correctly entered.

When you clear the screen by using "7," the previous entries will no longer appear. However, you do not have to re-enter these if they were correct. Simply press "6" and your figures will appear in the array listing.

The CALL KEY action of this program facilitates your toggling between any keypress selection. Options 1 through 3 require completing entries before you can toggle other keypresses. The blinking cursor tells you an entry is needed. When the cursor is not blinking you are at the CALL KEY choice. You can repeat any keypress as often as you wish.

CALL KEY What?

Here's something for those curious about the TI CALL KEY subprogram. According to my Extended BASIC manual, it implies the only useful values in the CALL KEY "unit" (besides 0) are 1 and 2, stating that 3, 4, and 5 are "reserved for future use."

Continued on page 40

EXTENDED BASIC

Continued from page 39

Even though the manual implies the units 3, 4, and 5 were for "future" use, I wrote the following program with 6 choices, being the calls for 0 through 5. For reference, the manual says that use of 0 allows use of the full keyboard, 1 allows use of left side of keyboard, and 2 allows use of the right.

The following program allows you to first enter any number between 1 and 6. Entering one of these numbers and pressing Enter will display a line on the screen. A CALL KEY action has been initiated by this entry. Now there is no blinking cursor as the program waits for you to press another key. When a key is pressed you are returned to the choose input line again.

After selecting a Call Key number the program is programmed to respond, either positively or negatively to your entry of the uppercase letter "A." A positive response is indicated in the line directly below the CALL KEY line and a negative response is indicated one line further down.

CALKEY

```
1 REM [CALKEY:0-5] 4-21-97
 By W.Leonard Taffs, SW99ers
 !195
2 !!131
3 ! Use "C" to clear screen
  ! Use "Q" to Quit !138
4 !!131
5 ! CALL KEY (0-5) STUDY....
    How Do Different Values
    of First Call Parameter
    affect CALL KEY results?
 !189
6 !!131
7 ! ENTER UPPERCASE "A"
    AFTER CHOOSING AND ENTER
    ING A NUMBER BETWEEN 1
    AND 6, THEN TRY OTHER
    LETTERS INSTEAD OF "A" !
248
8 !!131
100 CALL CLEAR !209
110 C1$="C.K.(0,K,S) " :: C2
$="C.K.(1,K,S) " :: C3$="C.K
.(2,K,S) " :: C4$="C.K.(4,K,
S) " :: C6$="C.K.(5,K,S) " !
113
120 J=0 :: DISPLAY AT(1,1):"
CALL KEY CHOOSE: (1-6) ";CHR
$(K)!099
130 ACCEPT AT(1,24)VALIDATE(
"1234567CcQq"):K$ :: IF K$="
" THEN 130 ELSE DISPLAY AT(1
,24):K$ :: DISPLAY AT(24,1):
"Last ";K$ :: K=ASC(K$):: IF
 (K$="Q")+(K$="q")THEN STOP
!023
140 IF (K-48=19)+(K-48=51)TH
EN GOSUB 360 :: GOTO 120 ELS
E IF (K-48<1)+(K-48)>6 THEN
130 !208
150 ! ON ERROR 130 :: ON K-4
8 GOTO 170,200,230,260,290,3
20 !207
160 ON K-48 GOTO 170,200,230
,260,290,320 ! Test line !10
1
170 DISPLAY AT(4,1):CHR$(95)
;C1$;K$ :: CALL KEY(0,K,S)::
 DISPLAY AT(1,23):K :: IF S<
1 THEN 170 :: C1=C1+1 !250
180 IF K=65 THEN DISPLAY AT(
5,2):"1 ""A"" K=";K;" x=";C1
:RPT$(" ",28):: GOTO 350 !15
5
190 IF K<>65 THEN DISPLAY AT
(5,1):RPT$(" ",28):" ";K$;"
REFUSED (0)";C1 :: GOTO 350
!029
200 DISPLAY AT(7,1):CHR$(95)
;C2$;K$:RPT$(" ",56):: CALL
KEY(1,K,S):: DISPLAY AT(1,23
):K :: IF S<1 THEN 200 :: C2
=C2+1 !242
210 IF K=65 THEN DISPLAY AT(
8,2):"2 ""A"" K=";K;" x=";C2
:RPT$(" ",28):: GOTO 350 !16
0
220 IF K<>65 THEN DISPLAY AT
(8,1):RPT$(" ",28):"*";K$;"
REFUSED (1)";C2 :: GOTO 350
!044
230 DISPLAY AT(10,1):CHR$(95
);C3$;K$:RPT$(" ",56):: CALL
 KEY(2,K,S):: DISPLAY AT(1,2
3):K :: IF S<1 THEN 230 :: C
3=C3+1 !063
240 IF K=65 THEN DISPLAY AT(
11,2):"3 ""A"" K=";K;" x=";C
3:RPT$(" ",28):: GOTO 350 !2
05
250 IF K<>65 THEN DISPLAY AT
(11,1):RPT$(" ",28):"*";K$;"
 REFUSED (2)";C3 :: GOTO 350
 !089
260 DISPLAY AT(13,1):CHR$(95
);C4$;K$ :: CALL KEY(3,K,S):
: DISPLAY AT(1,23):K :: IF S
<1 THEN 260 :: C4=C4+1 !146
270 IF K=65 THEN DISPLAY AT(
14,2):"4 ""A"" K=";K;" x=";C
4:RPT$(" ",28):: GOTO 350 !2
10
280 IF K<>65 THEN DISPLAY AT
(14,1):RPT$(" ",28):" ";K$;"
 REFUSED (3)";C4 :: GOTO 350
 !084
290 DISPLAY AT(16,1):CHR$(95
);C5$;K$ :: CALL KEY(4,K,S):
: DISPLAY AT(1,23):K :: IF S
<1 THEN 290 :: C5=C5+1 !183
300 IF K=65 THEN DISPLAY AT(
17,2):"5 ""A"" K=";K;" x=";C
5:RPT$(" ",28):: GOTO 350 !2
15
310 IF K<>65 THEN DISPLAY AT
(17,1):RPT$(" ",28):" ";K$;"
 REFUSED (4)";C5 :: GOTO 350
 !089
320 DISPLAY AT(19,1):CHR$(95
);C6$;K$ :: CALL KEY(5,K,S):
: DISPLAY AT(1,23):K :: IF S
<1 THEN 320 :: C6=C6+1 !220
330 IF K=65 THEN DISPLAY AT(
20,2):"6 ""A"" K=";K;" x=";C
6:RPT$(" ",28):: GOTO 350 !2
11
340 IF K<>65 THEN DISPLAY AT
(20,1):RPT$(" ",28):" ";K$;"
 REFUSED (5)";C6 :: GOTO 350
 !085
350 DISPLAY AT(24,12):"K$=";
CHR$(K);" K=";K;"CHR$=";CHR$
```

Continued on page 42

EXTENDED BASIC

EXTENDED BASIC

Continued from page 41

```
(K):: K$="" :: K=0 :: GOTO 1
20 !228
360 REM ** CLEAR SCREEN ** !
233
370 DISPLAY AT(1,1):"C L E A
 R I N G  S C R E E N" !003
380 FOR I=2 TO 24 :: DISPLAY
 AT(I,1):RPT$(" ",28);CHR$(3
0):: NEXT I :: RETURN !071
390 REM ** END OF PROGRAM **
 !102
```

FILE COMPRESSION

Using PFC, Archiver, and ARCHIE to increase disk space

BY JEFF WHITE

The following article has appeared in several venues, including user group newsletters and online services.—Ed.

I have found two programs to be very useful to me while restoring my hard disk to reduce wear and tear. These are Program File Compressor by Koen Holtman, and ARCHIE by Jim Reiss.

I proceeded to use PFC on Disk Utilities by John Birdwell, Archiver III by Barry Boone, and ARCHIE All went fine until I ran the compressed ARCHIE file which I had named ARCIRE-PFC. The ARCHIE title screen was glitched at the bottom, with garbage characters following:

* Press any key to proc

I checked the original ARCHIE file, and sure enough it runs fine. The missing characters were "eed *". That is five missing characters, so I had a pretty good idea what had gone wrong. So I loaded Disk Utilities and started looking at ARCHIE with the sector editor.

What I found was "eed */" in the last six bytes of the file. Thus, PFC was somehow not finding those bytes. Looking at the six-byte loader header in the first sector of ARCHIE I found the value >OC08 (3080) in the second word, which is the actual length of the program, and the proper value to store there. At the 17th byte (byte 0 is the first byte, so I am talking about byte >10 or 16) in the ARCHIE file descriptor record I found the value >0E, which is the number of bytes used in the last sector of the file.

Then I started looking at Archiver III V3.03 which I had patched with the correction implemented in version 3.03g. At the 17th byte in its FDR was the value >B2, and in the second word of the six-byte loader header was the value >1FB2. Going to the last sector of the file, the absolutely correct values should be >62 and >1F5C, since only >62 (98 bytes) are used in that last sector. But that does not really matter.

FILE COMPRESSION

I checked the Disk Utilities files, and sure enough the values at the second word in the six-byte loader headers were six higher than they needed to be to properly load. For DSKU1 and DSKU2, the value was >2000, and each of the values in their respective FDR's at the 17th byte was >00 (signifying all 256, or >100, bytes were used in the last sector of the files). The values should have been >1FFA and >00, but this time it was fortunate they weren't. For the DSKU3 file, the value in the FDR was >F2, and the value in the six-byte loader header was >1BF2. If you are following closely, you will realize that the value in the six-byte loader header need only be >1BEC.

However, for PFC to work properly, the second byte in the second word of the six-byte loader header in the first sector of each program image file must be equal to the 17th byte (byte >10 or 16) in the FDR of that file. If that byte is >00, the first byte in the second word of the six-byte loader header must be incremented by 1. In other words, if the DSKU1 file had as its six-byte loader header 0000 1FFA AOOO, before using PFC on it you would look at the FDR for DSKU1, find that byte >10 has the value >00, and change the value >1FFA in the loader header to >2000.

Anyway, what started this was the glitch PFC had compressing ARCHIE. The value in its six-byte header that needed changing was >OC08. So I changed the value to >OCOE, and ran PFC on ARCHIE. This time when I ran the compressed version of ARCHIE, the title screen looked right.

I recommend that ARCHIE executables not be archived with docs, as an archive of the separate program image files, support files (such as CHARA1), and doc files will normally be smaller.

Program File Compressor has the bug, in my opinion. But now you know how to work around it. Had the last few bytes of ARCHIE been program code rather than title screen data, unpredictable results might have occurred while running ARCHIE.

Now for the rest of the story. Program image files such as PFC1 and PFC2 can be packed with

Continued on page 44

FILE COMPRESSION

Continued from page 43

Archiver III as described in the ARCHIE docs, and then you can run them with ARCHIE. I created an uncompressed archive of PFC1 and PFC2, called it PFC, and saved disk space and a filename in the directory. I used PFC to change DSKU1 to DSKU2 into DSKUT to DSKUV, then packed the latter three files into DSKU, and now I can run DSKU with ARCHIE.

Of course, single files such as ARCHIE and Archiver III need not be packed, but compressing them with PFC works fine. ARCHIE compressed to 89 percent of its original size, Archiver to 79 percent, and Disk Utilities to 77 percent.

You may be wondering if you should compress your program files with PFC and follow that by archiving them with docs with Archiver III. I recommend that you check both ways. To support my position, consider the case of ARCHIE. The file I downloaded was 19 sectors archived with ARCHIE and ARCHIE/DOC. I used PFC to make an executable compressed file named ARCFUE-PFC, and when archived with the ARCHIE/DOC file I had a 22-sector file.

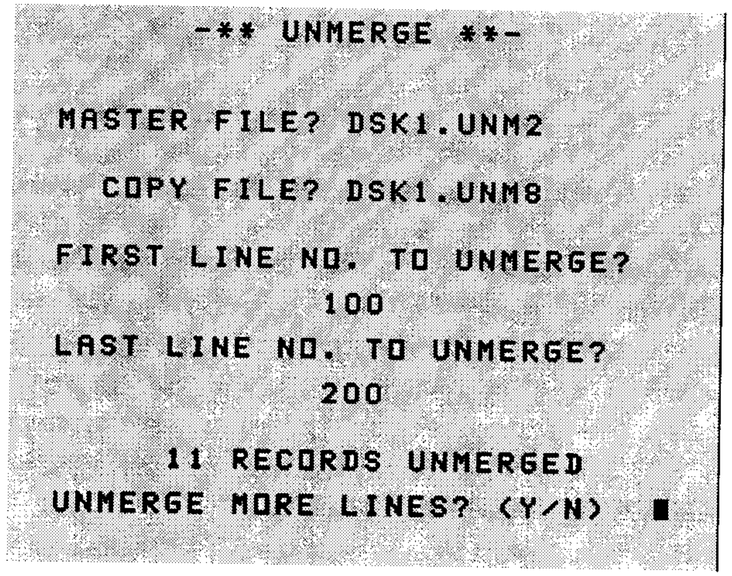
Then I created a non-executable file with PFC called ARCHIE-HELP, and archived it with ARCHIE/DOC and got a 20-sector file. Obviously, for file transfers it pays to have the smallest possible file size. But that was only one case. I then used Archiver III to archive itself, and it created a 27-sector file that is not runnable until it is unarc'd — a catch-22. However, the version of Archiver III compressed with PFC that I named ARC-PFC was 26 sectors. That is a reasonable way to distribute Archiver III.

I recommend that ARCHIE executables not be archived with docs, as an archive of the separate program image files, support files (such as CHARA1), and doc files will normally be smaller. Nevertheless, it is possible to get a small archive of ARCHIE executables by using Archiver III twice. Case in point: a compressed archive of DSKU/REF, DSKU1, DSKU2, and DSKU3 was 94 sectors. I compressed DSKU1 through DSKU3 into DSKUT through DSKUV, then packed them into an ARCHIE-runnable file called DSKU. I compressed DSKU/REF with Archiver III, then packed it with DSKU, getting a file of 93 sectors.

That is a savings of only 1 sector, but a big savings in time. Unpacking the DSKU file is a much quicker process than decompressing and unpacking the DSKU1 through DSKU3 files. With ARCHIE, DSKU is runnable. By the way, the packed DSKU file of PFC- compressed files is only 74 sectors, while a compressed archive of DSKU1 through DSKU3 is 76 sectors. The DSKU/REF file compressed to 19 sectors with Archiver III (wonder what Clint Pulley's Text compressor would do).

EXTENDED BASIC

UNMERGE lets users move, renumber program segments



UNMERGE, by Ed Neu, does exactly what it's name proclaims — it extracts (UNMERGES) program segments from programs on disk and saves them to a second disk file. It can be used to renumber sections of a program or to move subroutines from one program to another. UNMERGE is written in Extended BASIC and requires an expansion memory and disk system.

To use UNMERGE, save the master program in MERGE format (DSKx.FILENAME,MERGE). The program then is used to save the desired lines in a second file in MERGE format.

Files are automatically checked to make certain the master file exists. If the copy file already exists, a warning is given and the user is given the choice to use the name or not. The program allows the user to extract as many program segments as desired.

UNMERGE

```
100 REM-----------*
    |             |
    |   ** UNMERGE **    |
    |                    |
    *--------------------*
!182
110 !!131
120 !              by  Ed Neu
!015
130 !                  7/9/83
!188
140 !                   XB.DD
!243
150 !!131
160 !!131
170 CALL CLEAR !209
180 CALL SOUND(100,880,2)!13
6
190 CALL SOUND(100,880,2)!13
6
```

Continued on page 46

EXTENDED BASIC

Continued from page 45

```
200 CALL SOUND(100,880,2)!13
6
210 PRINT "A TI99/4A PROGRAM
 IN:": : : : : : : : : :!1
24
220 PRINT "       EXTENDED B
ASIC": : : :!180
230 PRINT "              by":
 : :"          Ed Neu": : :
 :!009
240 FOR T=1 TO 750 !176
250 NEXT T !234
260 CALL CLEAR !209
270 !-------------
!097
280 ! MAINLINE PROGRAM !040
290 !!131
300 DEF LNO(L$)=ASC(SEG$(L$,
1,1))*256+ASC(SEG$(L$,2,1))!
170
310 DISPLAY AT(12,3)BEEP ERA
SE ALL:"Want instructions? (
Y/N)" !173
320 CALL KEY(3,K,S):: IF S=0
 OR(K<>78 AND K<>89)THEN 320
 !114
330 IF K=89 THEN GOSUB 1120
!044
340 DISPLAY AT(1,6)ERASE ALL
 BEEP:" -** UNMERGE **-" !11
7
350 DISPLAY AT(5,1):USING "M
ASTER FILE? DSK############"
:FS$ !146
360 !-------------
!097
370 ! INPUT AND CHECK
      SOURCE FILE !085
380 !!131
390 ACCEPT AT(5,17)BEEP SIZE
(-12):FS$ !139
400 ON ERROR 430 !184
410 OPEN #1:"DSK"&FS$,DISPLA
Y ,INPUT ,VARIABLE 163 !017
420 GOTO 560 !129
430 DISPLAY AT(7,1)BEEP:"MER
GED MASTER FILE:" !067
440 DISPLAY AT(8,(26-LEN(FS$
))/2):"""DSK"&FS$&""""" !109
450 DISPLAY AT(9,8):"NOT AVA
ILABLE" !208
460 DISPLAY AT(15,3):"press
any key to re-enter" !182
470 CALL KEY(3,K,S):: IF S=0
 THEN 470 !030
480 DISPLAY AT(7,1):"" !040
490 DISPLAY AT(8,1):"" !041
500 DISPLAY AT(9,1):"" !042
510 DISPLAY AT(15,1):"" !088
520 GOTO 390 !214
530 !-------------
!097
540 ! INPUT AND CHECK
      TARGET FILE !075
550 !!131
560 DISPLAY AT(8,1)BEEP:"  C
OPY FILE? DSK" !035
570 ACCEPT AT(8,17)SIZE(-12)
:FT$ !161
580 OPEN #2:"DSK"&SEG$(FT$,1
,2),INTERNAL,INPUT ,RELATIVE
 !162
590 INPUT #2:A$,I,I,I !152
600 INPUT #2:A$,I,I,I !152
610 IF A$="" THEN 700 !165
```


EXTENDED BASIC

```
620 IF A$=SEG$(FT$,3,12)THEN
 630 ELSE 600 !206
630 CLOSE #2 !152
640 DISPLAY AT(10,1)BEEP:"
COPY FILE ""DSK"&FT$&""""" !0
72
650 DISPLAY AT(11,6):"ALREAD
Y EXISTS!" !139
660 DISPLAY AT(13,1):"USE IT
? (Y/N)" !147
670 ACCEPT AT(13,15)VALIDATE
("YN")SIZE(1):USE$ !005
680 FOR I=10 TO 13 :: DISPLA
Y AT(I,1):"" :: NEXT I !241
690 IF USE$="Y" THEN 710 ELS
E 570 !059
700 CLOSE #2 !152
710 OPEN #2:"DSK"&FT$,DISPLA
Y ,OUTPUT,VARIABLE 163 !120
720 !-------------
!097
730 ! INPUT LINE NUMBER
      RANGE TO UNMERGE !039
740 !!131
750 DISPLAY AT(11,1)BEEP:"FI
RST LINE NO. TO UNMERGE?" !0
76
760 ACCEPT AT(13,13)VALIDATE
(DIGIT)SIZE(5):L1 !236
770 DISPLAY AT(15,1)BEEP:"LA
ST LINE NO. TO UNMERGE?" !25
1
780 ACCEPT AT(17,13)VALIDATE
(DIGIT)SIZE(5):L2 !241
790 !-------------
!097
800 ! READ SOURCE FILE & !05
4
810 ! WRITE TO TARGET FILE !
024
820 ! IF WITHIN UNMERGE !088
830 ! RANGE !016
840 !!131
850 DISPLAY AT(20,7):"-> UM
ERGING <-" !054
860 NREC=NREC+1 !201
870 LINPUT #1:T$ !206
880 CHK=LNO(T$)!098
890 IF CHK<L1 THEN 870 !120
900 IF CHK>L2 THEN 940 !192
910 PRINT #2:T$ !193
920 IF EOF(1)<>0 THEN 940 !0
85
930 GOTO 860 !174
940 PRINT #2:CHR$(255)&CHR$(
255)!085
950 DISPLAY AT(20,4)BEEP:USI
NG "#### RECORDS UNMERGED":N
REC-1 !021
960 FOR TD=1 TO 100 :: NEXT
TD !153
970 CLOSE #2 !152
980 !-------------
!097
990 ! REPEAT UNMERGE !151
1000 !!131
1010 DISPLAY AT(22,1)BEEP:"U
NMERGE MORE LINES? (Y/N)" !2
28
1020 ACCEPT AT(22,28)VALIDAT
E("YN"):GO$ !096
1030 IF GO$="N" THEN CLOSE #
1 :: END !102
1040 RESTORE #1 !139
1050 NREC=0 !223
```

Continued on page 48

EXTENDED BASIC

Continued from page 47

```
1060 FOR I=9 TO 22 :: DISPLA
Y AT(I,1):"" :: NEXT I !200
1070 GOTO 570 !139
1080 END !139
1090 !————————
 !097
1100 ! PRINTS INSTRUCTIONS !
088
1110 !!131
1120 DISPLAY AT(1,8)BEEP ERA
SE ALL:"** UNMERGE **" !250
1130 DISPLAY AT(3,1):"This p
rogram allows you to  extrac
t portions of a       progra
m stored on disk and  save t
he extracted portion" !198
1140 DISPLAY AT(7,1):"on ano
ther disk file.  It isgood f
or using subroutines  in oth
er programs or for    renumb
ering sections of a" !214
1150 DISPLAY AT(11,1):"progr
am." !130
1160 DISPLAY AT(13,1):"First
, the ""master"" program mus
t be saved using the    ""M
ERGE"" option.  The       d
esired lines are saved in" !
068
1170 DISPLAY AT(17,1):"a sec
ond ""copy"" file in    mer
ged format." !066
1180 DISPLAY AT(24,2):"press
 any key to continue" !248
1190 CALL KEY(0,K,S):: IF S=
0 THEN 1190 !238
1200 CALL CLEAR !209
1210 DISPLAY AT(1,1):"Files
are automatically    checke
d to make certain the master
 file exists. If the  copy f
ile already exists a" !139
1220 DISPLAY AT(5,1):"warnin
g message is given    and th
e user is given the   choice
 to use it or not." !007
1230 DISPLAY AT(9,1):"The pr
ogram allows the user to exr
act as many program   sectio
ns as he/she desires  with a
 prompt at the end of each"
!013
1240 DISPLAY AT(13,1):"each
""UNMERGING""." !054
1250 DISPLAY AT(24,4):"press
 any key to begin" !151
1260 CALL KEY(0,K,S):: IF S=
0 THEN 1260 !052
1270 RETURN !136
```

MDOS 6.0

MDOS 6.0 ready for Y2K

BY GAMBIT

The following article originally appeared in The Computer Voice, the newsletter of the Southern California Computer Group.—Ed.

It's here... M-DOS 6.0 has arrived!

MDOS 6.0

I am particularly excited about this version, since it has some important changes in it.

Unless you have been hiding under a rock, you've probably heard of the problems that we are about to face regarding the year 2000. Many computers are not equipped to handle the year 2000. This has caused great concern regarding the stock market, bank accounts, phone service, or whether the power will be on when we wake up on Jan. 1, 2000.

I started a discussion on the TI listserver about this Y2K problem and how it affected us Geneve users. We are a little more fortunate, as the clock chip in the Geneve uses only two digits for the year, leaving the rest of it up to M-DOS. Tim Tesch has made the necessary changes to M-DOS, so the year 2000 and beyond (until 2086) will show the correct date. I doubt any of us will be around past 2087, so I am not concerned about the date being correct 100 years after the Geneve was made. If I am around then and the date is wrong, I'll be sure to raise a ruckus!

A few years ago, Tim wrote a program called, "IBMGRAPH." This made the IBM character set (characters 128-255) available to M-DOS, so you could use it with programs that would display them, including the "type" command in DOS. Now that character set is part of M-DOS 6.0, and can be initialized by the IBMGRF [on/off] command.

This works rather nicely, except for one program, Directory Manager, by Clint Pulley. As Tim explains it, Clint defines his character set before having M-DOS define the remaining characters. As it is, you get a bunch of strange-looking characters, instead of the nice lines that were defined by Clint. Since FED and a large number of other programs don't have a problem with the IBMGRF command, the problem definitely is with DM. Perhaps we can get Clint to do an update, or release the source code, so another ambitious programmer can make the corrections to DM.

There are a number of other commands that have been added or removed. The "VIDEO" command has returned. This is invoked differently than before. Now you type VIDEO [fast/slow]. If you type VIDEO FAST, the wait-states are turned off, making screen displays faster. VIDEO SLOW will return the wait-state to its default setting.

This command is not compatible with all programs, in which case you should make sure VIDEO is slow. If you used to use the VIDEO ON command in the earlier versions of M-DOS with some of your programs, and you now have the Turbo-Video chip installed on your 9640, the two may not work together. At most, you might have strange display results with the two combined. With the "TV" chip on the 9640, you shouldn't have a need for the VIDEO command, but it is there again.

A new SCSI mapping command has been added SCSMAP [nn], which

Continued on page 50

MDOS 6.0

Continued from page 49

should help those having problems with SCSI devices not responding to ID 0, 1, or 2. This command will make more sense to those of you already using a SCSI card.

The MIRROR command has been removed. It has been replaced by an external command (program) called "SAVEIMAGE" This is compatible with all hard drives, including those (MFM) formatted at 34 sectors per track with CFORM. The image is saved to a file on a floppy, or a directory on another hard drive. (If sector zero has been corrupted on your C: drive, it won't do much good to store the image of it on the same drive, which is now inaccessible.)

The TI [on/off] command has also been removed. I have never used this, but I do know it has something to do with the way you access hard drives, whether as WDS or HDS.

I've run out of room, so I'll close for now.

MICROREVIEWS

Download File Converter and Notepad80

By CHARLES GOOD

DOWNLOAD FILE CONVERTER by Bruce Harrison

When you download text files or the source code of Web pages from the Internet to a 99/4A the text files often end up as a file in DF128 format. The same thing occurs if you download text from a BBS. Almost all TI word processors ("PRESS" is the exception) cannot handle this file format and require their text to be in DV80 or sometimes DF80. Bruce Harrison has written a pair of public domain assembly language programs that convert DF128 text to DV80 text and vice versa.

These programs are easy to use. Just enter the path of the input file and the path of the output file. Long path names will work so those with SCSI or HFDC hard drives will have no problems. The DV80 output is nicely formatted with word wrap. You see the conversion on screen as it progresses.

In addition to Harrison's software, two other older software products accomplish DF128-to-DV80 conversion and one well-known product does DV80-to-DF128. Richard Phillips' CONVERTIT v1.1 will do DF128-to-DV80 conversion. It is written in Extended BASIC and is thus very slow. It was written specifically to convert text downloaded from a Macintosh computer and is supposed to recognize special symbol codes on the Mac and translate them properly on a TI. These symbols include copyright, trade mark, pound sterling, and all the common mathematics symbols. I haven't tried CONVERTIT with Mactext but I have tried it with "regular" DF128 text. It works, but very slowly.

Ben Yates wrote PRINT128 in the "c" language. This does essentially the same DF128-to-DV80 job as Harrison's software, and like the Harrison product PRINT128 is hard drive and Geneve compatible and does word wrap. An unusual aspect of PRINT128 is that it will do DF128-to-anything, such as DF24 or DV30. I don't know why anyone would want to do this with text, but the ability is there.

Another thing I don't know why anyone would want to do is create DF128 text from DV80 on the TI. Harrison's software will do this and so will the Funnelweb word processor. Using either the 40- or 80-column versions of the Funnelweb editor you can use PF (print file) to save a disk file in DF128 format compatible with UNIX or MS/DOS, your choice. For example, in the Funnelweb editor after typing PF and <enter> type M, a space and a path name to save the text in the edit buffer as a DF128 file in DOS format with cr and lf at the end of each line and ^Z at the end of the text.

Send me $1 and I will send you all the software described above on a TI DSSD disk. Or e-mail me and I will e-mail you the software as an attached file in PC99 format. This software includes Bruce Harrison's new Download File Converter as well as the older Convertit, and Print128.

NOTEPAD80 by Walid Maalouli

This is 80-column word processing on the cheap! If you can find an old 80-column terminal then this software lets you do 80-column word processing without an 80-column card. You hook the 80-column terminal to the RS-232 port of your TI, and it acts as a second monitor for 80-column work.

Yes, you do need two video displays for this software, your normal TI monitor and the 80-column terminal. The software is written in Extended BASIC and is thus kinda slow. It is, however, one of the most full-featured Extended BASIC word processors I have seen. You start out on your TI monitor selecting RS-232 port, size of left/right and top/bottom margins, double or single space, paragraph indentation and the number of text lines per page. There is also, on the TI monitor, an indicator that tells you which page of your document is being displayed on the terminal. You get a 24-line 80-column display, about 1/3 of a 60-line document "page."

When typing a new document keyboard response is somewhat slow, but adequate for most people. Newly entered text is word wrapped.

Continued on page 52

MICROREVIEWS

MICROREVIEWS

Continued from page 51

Documents are saved to disk as DV80 files. While typing or editing existing text the following operations are possible: line and character delete, blank line and new character insertion, paragraph indent, destructive backspace, move cursor left/right one character of up/down one line, quit program, write document to disk, print document to printer, load new document, go to end of document, go to beginning of document, next page or screen and previous page or screen.

Besides its slow speed, which is not unbearable, Notepad80 suffers from the same problem that affects all word processors written in Extended BASIC, namely that you can only modify one line of existing text at a time. Any changes you make to a line do not affect text on adjacent lines. This makes it difficult to insert lots of text within a line and have the whole document look good. You can, of course, insert blank lines within existing text and then add additional text to these blank lines.

Terminals that hook to the RS-232 port are not easy to find. I am not sure if they are still manufactured. When you can find one they are either free or almost free.

Notepad80 is public domain. The author asks no money for his efforts, but welcomes your suggestions and comments. Send me $1 and I will mail it to you on a SSSD TI disk.

ACCESS

Walid Maalouli (author of Notepad80), 757 Main St., Olean, NY 14760; e-mail wmalouli@eznet.net

Bruce Harrison (author of Download File Converter),5705 40th Place, Hyattsville, MD 29781; e-mail rottencat13@hotmail.com; phone (331) 277-3467

Charles Good (source of software described in this article), P.O. Box 647, Venedocia, OH 45894; e-mail good.6@osu.edu; phone (419) 667-3131

USER NOTES

Program checks for FOR/NEXT errors

The following program, by Jim Peterson, checks for FOR/NEXT nesting errors in BASIC and Extended BASIC programs. To use it save the program you want to check in MERGE format and follow the onscreen prompts.

F/NCHECKER

```
90 JIM PETERSON DISK program
100 DISPLAY AT(3,1)ERASE ALL
:"FOR/NEXT CHECKER": : :" To
 edit a program, SAVEd in":"
MERGE format, for FOR/NEXT":
"nesting errors."
110 DISPLAY AT(12,1):"FILENA
ME? DSK" :: ACCEPT AT(12,14)
:F$ :: OPEN #1:"DSK"&F$,VARI
ABLE 163,INPUT :: R=1 :: CAL
L CLEAR :: DIM W$(75)
120 IF EOF(1)=1 THEN 230 ::
P=3 :: LINPUT #1:M$
130 A=POS(M$,CHR$(140),P)::
B=POS(M$,CHR$(150),P):: IF A
+B=0 THEN 120 :: IF B=0 THEN
 C=A ELSE IF A=0 THEN C=B EL
```

Continued on page 54

USER NOTES

Continued from page 53

```
SE C=MIN(A,B)
140 LN=ASC(SEG$(M$,1,1))*256
+ASC(SEG$(M$,2,1))
150 IF C=B THEN 180 :: V$=SE
G$(M$,A+1,POS(M$,CHR$(190),A
)-A-1):: DISPLAY AT(R,1):LN;
"FOR ";V$ :: W$(R)=V$ :: GOS
UB 250 :: P=C+1 :: Z=0
160 FOR J=1 TO R-2 :: IF V$=
W$(J)THEN DISPLAY AT(R-1,23)
:"ERROR!"
170 NEXT J :: GOTO 130
180 X=POS(M$,CHR$(130),B)::
IF X=0 THEN X=POS(M$,CHR$(0)
,B)
190 V$=SEG$(M$,B+1,X-B-1)
200 FOR J=1 TO R-1 :: IF V$<
>W$(J)THEN 220 :: DISPLAY AT
(J,18):LN;:: IF V$<>W$(R-1-
Z)THEN DISPLAY AT(J,23):"ERRO
R?"
210 P=C+1 :: W$(J)="" :: Z=Z
+1 :: GOTO 130
220 NEXT J :: DISPLAY AT(R,5
):LN;"NEXT ";V$ :: DISPLAY A
T(R,23):"ERROR!" :: GOSUB 25
0 :: P=C+1 :: GOTO 130
230 CLOSE #1 :: DISPLAY AT(R
,5):"ANY KEY TO QUIT"
240 CALL KEY(0,K,S):: IF S=0
 THEN 240 :: END
250 R=R+1 :: IF R=24 OR R=48
 THEN DISPLAY AT(24,1):"PRES
S ANY KEY TO CONTINUE" ELSE
RETURN
260 CALL KEY(0,K,S):: IF S=0
 THEN 260 ELSE CALL CLEAR ::
 RETURN
```

Running A/L programs from XB

The following was written by John Bull and comes from an online TI FAQ.

To run from Extended BASIC, an assembly language program must be written for that purpose or modified. Extended BASIC cannot use files created by the C (compressed) option of the assembler. Extended BASIC does not handle DEFs but the address of utilities must be EQUated. For instance, a program written for the Editor/Assembler loaders might begin:

```
        DEF   START
        REF   STRASG,STRREF
```

For XB, this should be:

```
        REF  START
STRASG EQU   >2010
STRREF EQU   >2014
```

The list of EQUates for XB is on pages 415 and 416 of the Editor/Assembler manual. Modifying an E/A program to run from XBASIC is sometimes as simple as the above. Note that the E/A manual has a typo. Page 416 should read NUMREF EQU >200C

One way you can run an assembly language program from XBASIC is to put the following at the beginning of your XBASIC program:

```
10 CALL INIT
20 CALL LOAD("DSKn.FILENA
ME")
```

"FILENAME" is the name of the assembly language program.

Now the program is in memory and can be run with:

```
CALL LINK("START")
```

"START" is the entry point as defined in the program.

Another way, involving more work but often more convenient, is to embed the assembly language program in your XBASIC program where it will remain, ready to run, whenever you load the XBASIC program. There are two programs available to do the embedding. One is Harry Wilhelm's HML (High Memory Loader) and the other is Scott Kaplan's ALSAVE, which loads into low memory. HML is easier to use but will not work with large XBASIC programs.

Whichever way you load the assembly language programs, it remains in memory until you do CALL INIT, or BYE. Thus, it is available in other XBASIC programs that you RUN from the first one. Your assembly language program will remain in memory even after issuing a "NEW" from XBASIC.

Have a subscription problem?
Didn't receive a disk?

You can call us at 512-255-1512
between 9 a.m. and noon Saturdays CT;
or fax us anytime at 512-255-1512;
or e-mail us at
micropendium@yahoo.com.

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 PUBLIC DOMAIN DISKS
(SSSD unless specified)

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